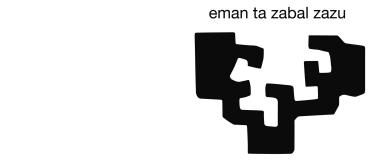


Universidad del País Vasco
Euskal Herriko Unibertsitatea

Bachelor Degree in Computer Engineering

Software Engineering

Thesis

Multi-objective algorithms for the optimization of a sequence of generator matrices in topological quantum computing

---

Author

Guillermo Lapuente Valea

2015

Abstract

Quantum Computing is a relatively modern field which simulates quantum computation conditions. Moreover, it can be used to estimate which quasiparticles would endure better in a quantum environment. Topological Quantum Computing (TQC) is an approximation for reducing the quantum decoherence problem[1], which is responsible for error appearance in the representation of information.

This project tackles specific instances of TQC problems using MOEAs (Multi-objective Optimization Evolutionary Algorithms). A MOEA is a type of algorithm which will optimize two or more objectives of a problem simultaneously, using a population based approach.
We have implemented MOEAs that use probabilistic procedures found in EDAs (Estimation of Distribution Algorithms), since in general, EDAs have found better solutions than ordinary EAs (Evolutionary Algorithms), even though they are more costly. Both, EDAs and MOEAs are population-based algorithms.
The objective of this project was to use a multi-objective approach in order to find good solutions for several instances of a TQC problem. In particular, the objectives considered in the project were the error approximation and the length of a solution. The tool we used to solve the instances of the problem was the multi-objective framework PISA. Because PISA has not too much documentation available, we had to go through a process of reverse-engineering of the framework to understand its modules and the way they communicate with each other. Once its functioning was understood, we began working on a module dedicated to the braid problem. Finally, we submitted this module to an exhaustive experimentation phase and collected results.

Key words: MOEA, Pareto front, quantum braid, TQC, PISA.

[1] In the thesis we present a brief definition of quatum decoherence and other concepts used in it.

Contents

List of Figures

Table index

List of Algorithms

CHAPTER 1

Introduction

In the early years of the 20th Century, physicists started to work on a model called "quantum mechanics". This model is used to describe the behavior of matter at a subatomic scale. In classic physics, some effects are not strong enough to change the matter's behavior and are considered insignificant. However, at subatomic levels these have to be taken into account making quantum systems very fragile. Likewise, research on quantum computing appeared as a result of the limit in the ever-decreasing size of computers. Even though some quantum computers have been built, it is not mainstream. For now, only theoretical solutions have been presented.

This problem is reduced to finding an optimal ordering for the quasiparticles that compose the system. This ordering can be formulated as an optimization problem in which the objective is to find a product of matrices which minimizes an error function. In this case, the problem is multi-objective because the error and the length of the resulting solution are evaluated.
The problem is often called "braid problem" too, since the matrices can be represented as topological braids following the solution order.

The problem has been tackled using several one-objective optimization algorithms, but no previous papers were found using multi-objective optimization for this problem. Although, multi-objective optimization has been widely used in many other fields such as finance, engineering or logistics.

This objective of this work is to evaluate the convenience of using multi-objective evolu-

tionary algorithms (MOEAs) to solve the braid problem. In order to do this, we needed to implement a program to optimize the braid problem for two or more objectives. In our case, the length and error approximation of the braid are the objectives, therefore it is a two-objective problem. The program was implemented using the platform PISA which is a framework for multi-objective optimization, written in C. To implement the program, reverse-engineering was used at first to fully understand the components and working procedure of PISA. We then implemented new operators and modules. A previous C++ implementation of the braid problem was adapted and assembled to the PISA framework. Since probabilistic models, such as EDAs, were also implemented, they will also be covered in this document. In order to test and compare solutions, three MOEAs are introduced and latterly put to solve the problem. Finally statistical tests and metrics are used for estimating each MOEA's performance and determine which one solves the problem with better accuracy. These tests are driven using the PISA tool as well.

CHAPTER 2

Project Management

An initial planning was made based on the required number of hours to be dedicated to the project. Throughout the development of the project, there have been tweaks in this planning.

2.1 Project Breakdown Structure

The PBS is a managerial resource which helps in determining all the different tasks, based on what particular field they belong in. Figure 2.1 illustrates what has the project consisted on according to the identified tasks.

2.2 Planning

To plan the work for each week the project lasted, a Gantt table was used. In a Gantt table there are presented identified tasks and when are these tasks going to be executed. Figure 2.2 is the Gantt table of the project.

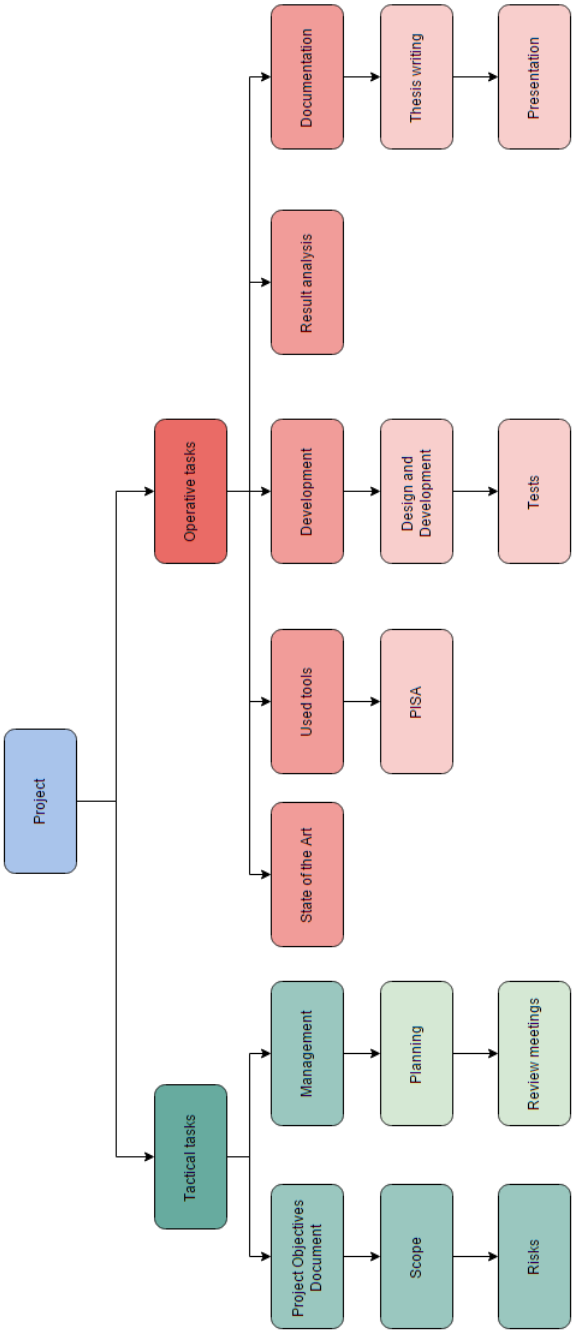
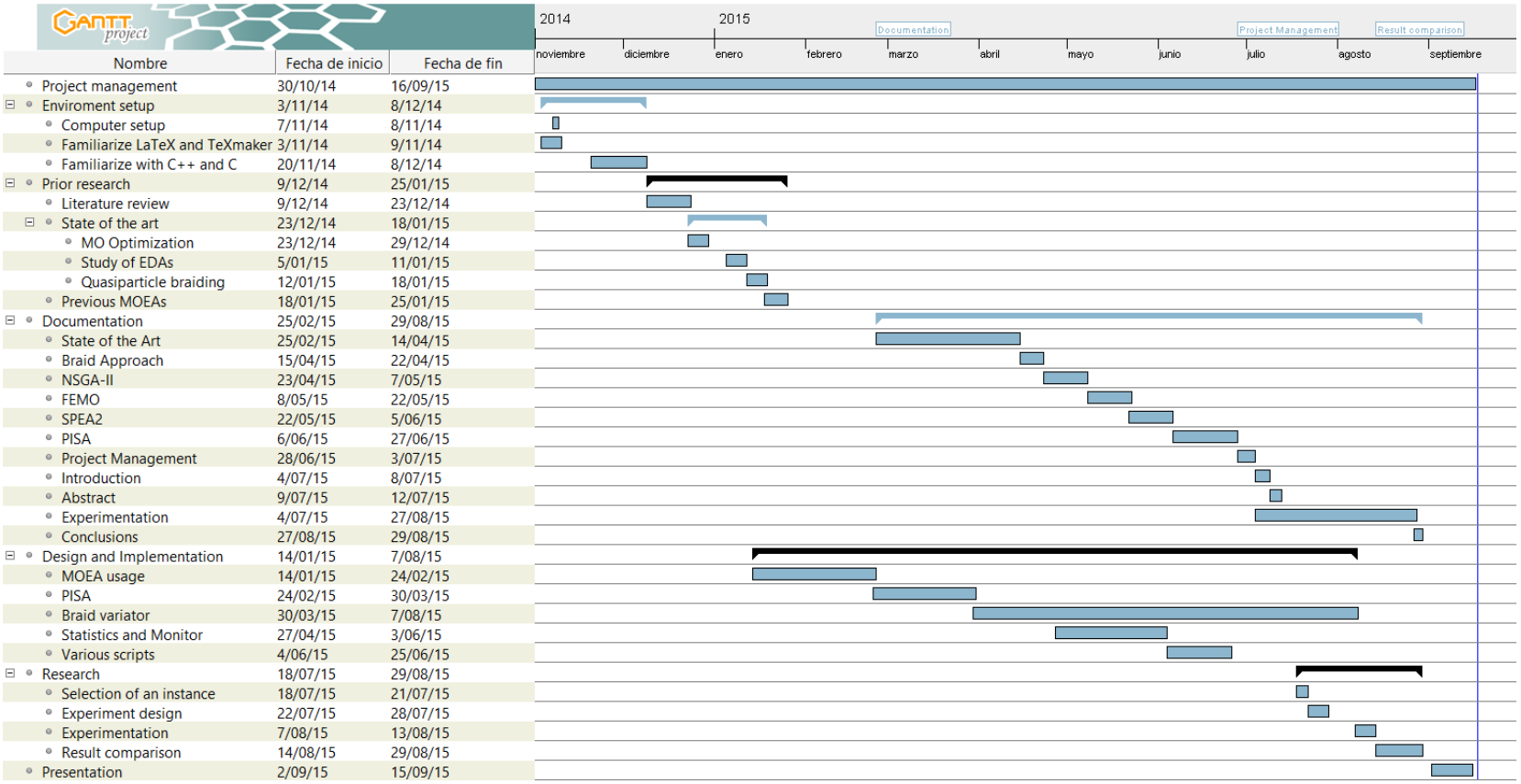


Figure 2.1: Project Breakdown Structure.



Nombre	Fecha de inicio	Fecha de fin
Project management	30/10/14	16/09/15
Enviroment setup	3/11/14	8/12/14
Computer setup	7/11/14	8/11/14
Familiarize LaTeX and TeXmaker	3/11/14	9/11/14
Familiarize with C++ and C	20/11/14	8/12/14
Prior research	9/12/14	25/01/15
Literature review	9/12/14	23/12/14
State of the art	23/12/14	18/01/15
MO Optimization	23/12/14	29/12/14
Study of EDAs	5/01/15	11/01/15
Quasiparticle braiding	12/01/15	18/01/15
Previous MOEAs	18/01/15	25/01/15
Documentation	25/02/15	29/08/15
State of the Art	25/02/15	14/04/15
Braid Approach	15/04/15	22/04/15
NSGA-II	23/04/15	7/05/15
FEMO	8/05/15	22/05/15
SPEA2	22/05/15	5/06/15
PISA	6/06/15	27/06/15
Project Management	28/06/15	3/07/15
Introduction	4/07/15	8/07/15
Abstract	9/07/15	12/07/15
Experimentation	4/07/15	27/08/15
Conclusions	27/08/15	29/08/15
Design and Implementation	14/01/15	7/08/15
MOEA usage	14/01/15	24/02/15
PISA	24/02/15	30/03/15
Braid variator	30/03/15	7/08/15
Statistics and Monitor	27/04/15	3/06/15
Various scripts	4/06/15	25/06/15
Research	18/07/15	29/08/15
Selection of an instance	18/07/15	21/07/15
Experiment design	22/07/15	28/07/15
Experimentation	7/08/15	13/08/15
Result comparison	14/08/15	29/08/15
Presentation	2/09/15	15/09/15

Figure 2.2: Project's Gantt diagram.

Task	Hour estimate
Project management	10h
Environment setup	5h
Prior research	40h
Documentation	120h
Development and design	100h
Experimentation	20h
Presentation	5h
Total	300h

Table 2.1: Estimated hour dedication for each task.

Task	Hour estimate	Real hour dedication
Project management	10h	10h
Environment setup	5h	5h
Prior research	40h	40h
Documentation	120h	140h
Development and design	100h	120h
Experimentation	20h	20h
Presentation	5h	5h
Total	300h	340h

Table 2.2: Final hour dedication.

2.3 Hour estimate

To estimate how many hours the project could last, the ECTS weight of the project has been considered. An ECTS represents 25 hours of work and the project has a 12 ECTS value. Therefore, the minimum amount of time that should be put into the project is 300 hours. The initial hour planning corresponds to the estimation present in table 2.1.

However, due to complications in the schedule, continuous reviews of the documentation and error correction in the software, the hour estimate could not be completed, resulting in the final hour dedication present in table 2.2.

2.4 Working methodology

The working methodology has stayed roughly the same throughout the project. I worked at home with my computer and if I had any problem, I wrote an e-mail the supervisor.

If the problem was more demanding, I would wait until we had a meeting so we could discuss possible answers.
With reference to work management, the supervisor and I decided to use a shared Dropbox folder so we could share information (*e.g.* bibliograpy, scanned reviews by the supervisor, the versions of the document, etc.). This way it was easier to check updates of the project. Moreover, it is much safer to keep the work in the cloud, rather than in local. Additionally, regular backups of the software and the documentation were done.

2.5 Communication

The communication with the supervisor has changed during the project. In the first half of the project I was on exchange in Oulu, Finland; thanks to the Erasmus+ program. Because of that, it was impossible to do face-to-face meetings with the supervisor of the project. I also had to take more courses than in the second semester. In this manner, the only possible communication was e-mail or Skype. Meetings were held regularly using Skype to exchange information.

When I returned, there were regular meetings each Tuesday in the faculty. These meetings were used as a control session and to state what next steps should be taken. Although, if any immediate question arose, communication via e-mail was used.

2.6 Risk management

In order to avoid possible risks, we have to identify them and declare a possible solution in case they happen. In this way, the damage to the project could be minimized.

- **Data loss**
  - **Description**: the loss of data could be a critical aspect for the project. The data could be lost because of many reasons.
  - **Contingency plan**: as we said in the working methodology section, the loss of data risk was countered using a shared Dropbox folder and by regularly making backups of both documentation and software assets.
- **Unexpected formats**

  - **Description**: because I usually work with two operating systems (Windows 8.1 Pro and Ubuntu 14.04 LTS), it is common to try to re-size a partition and unintentionally erase the whole content.
  - **Contingency plan**: to hasten the process of re-installation, I had USB drives with the Operating Systems prepared.

- **Delays**

  - **Description**: a delay in the project could happen for many reasons, and it means that the planned schedule for a task cannot be fulfilled.
  - **Contingency plan**: more effort had to be put in the rest of the project's length. We also considered the option of delaying the project's presentation to September in case we were very late.

- **Understanding difficulties**

  - **Description**: due to the difficult nature of the problem, some concepts may require more researching. Moreover, understanding PISA can also be difficult.
  - **Contingency plan**: invest more time on research and looking carefully at the code.

- **Poor results**

  - **Description**: when driving the experiments, it could turn out that the results were not satisfying.
  - **Contingency plan**: revise and correct the code if there was any mistake. If it was not the code's fault, try other instances.

CHAPTER 3

State of the art

The state of the art chapter is focused on explaining which are the latest findings on the field that it is being covered in the project. In this case, we will review topics such as quantum computing, multi-objective optimization and evolutionary algorithms. To write this chapter and present the concepts we have reviewed several publications which are cited throughout the chapter.

First of all, some general concepts are explained and later we will look more into detail at some specific challenges that this work will try to overtake.

3.1 Quantum Computing

3.1.1 What is the idea behind quantum computing?

Quantum computing is a new computing paradigm which tries to apply quantum physics theories to the computer science context. Instead of using bits to represent information, qubits are used. Likewise, logic gates are not used either, but quantum gates instead [1].

Quantum physics studies the behavior of the very small objects. According to the Moore law [2], every time computers are getting smaller, and we are approaching computers so small that sometimes quantum properties, like tunnel effect, can appear. The tunnel effect will let a particle go through an object instead of bouncing because of its small size. The

small size of the particles can also cause disturbances on whether they behave like a classical or quantum particles; this effect is often called quantum decoherence.
To avoid such things from happening, quantum computers started to be designed [3]. However we must differentiate between mechanical and theoretical quantum computation. On the one hand, mechanical quantum computers are computers whose mechanisms rely on the quantum information-theory constraints. On the other hand, theoretical quantum computing tries to simulate events that may take place in a quantum environment.

Applications to quantum computing have appeared as real alternatives to existing classical computing procedures like cryptography, communications or error correction [4].

3.1.2 The qubit

The qubit, as the analog in the classic systems, is the representation of the minimum unit of information for quantum systems. It is a vector in a two-dimensional complex vector space, represented as:

$$|\psi\rangle \equiv \begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \alpha\begin{bmatrix}1\\ 0\end{bmatrix} + \beta\begin{bmatrix}0\\ 1\end{bmatrix} \equiv \alpha\,|0\rangle + \beta\,|1\rangle \text{ for any } \alpha,\beta \text{ with } |\alpha|^2 + |\beta|^2 = 1 \quad (3.1)$$

In equation 3.1, for $\alpha = 0$ and $\beta = 1$ we get as result qubit 1 and for $\alpha = 1$ and $\beta = 0$, we get qubit 0. The qubit obtained is a superposition of states $|0\rangle$ and $|1\rangle$. Because α and β are usually complex numbers, these have phase information useful to represent information in a quantum environment.

Quantum information has to be processed using quantum gates, which are unitary transformation matrices analog to logic gates in classic computing. However, it is not possible to realize a gate precisely because simply by observing the system, this would interact with the qubits, corrupting it [5].

3.1.3 Topological Quantum Computing

Topology is the field of mathematics that studies the properties of geometric shapes and how these properties can remain identical after several transformations. After a series of transformations there must be a biunivocal relationship between the points of the original figure and the points of the transformed figure [6].

Topological quantum computing is an alternative framework that avoids quantum decoherence produced by local noise and other physical level problems. The problem of executing a gate in this framework can be posed as the problem of braiding quasiparticles. Because these are non-abelian, the problem is reduced to find an optimal product of generator matrices. A quasiparticle is non-abelian if it is non-commutative with other quasiparticles, meaning that the order in which they interact is important since it is not the same if $quasiparticle_a$ operates with $quasiparticle_b$, as if $quasiparticle_b$ operates with $quasiparticle_a$. The product of generators must approximate a quantum gate with the smallest error and as short as possible braid length to prevent loss [7].

Quantum systems are very fragile. So fragile that any interaction with its outer world can disrupt the whole system. Here is where topology partakes. Instead of manipulating quantum information itself, the topological states of the system are used so that the quantum information does not suffer from local disturbances.

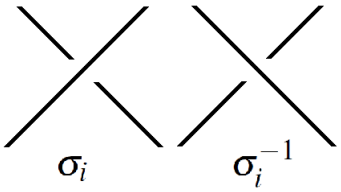


Figure 3.1: Two basic braids

At this point the anyons are theoretical constructs that simulate particles described by physicist as those that do not behave as fermions or bosons. These are elementary particles, also called quasiparticles, which group any subatomic particle in the 3-dimensional space that follows the laws of quantum mechanics depending on the particles' spin attribute. They can be abelian or non-abelian [8]. These are usually used to simulate the behavior of quantum gates. The manipulation of anyons will lead to the creation of braids, which is the topological representation of generator matrices. The concept of braids is thoroughly covered in the next section. In figure 3.1 we can see how a braid will look like when two anyons interact. Usually we will line up anyons and swap them around to recreate approximations to quantum gates as depicted in figure 3.2 in the form of braids. Because anyons are represented as matrices, these operations will have the properties of matrix algebra [5]. Fibonacci anyon braids are the simplest anyon with non-abelian braiding statistics and only encompasses one qubit gates.

3.1.4 Braids

When it comes to quantum computing, braid topology is often used to approach the problems of decoherence and imprecise quantum gates where the error approximation to a target gate is quite big. A braid is collection of strands between two parallel discs. Emil Artin stated that a braid group $B_n/n = 1,2,3,...$ with n strands has a representation of $n-1$ generators [9]. In this thesis, these generators will be called braid matrices. Two braids can be joined to obtain a new one. This operation is called composition and gives a group structure to the set of braids with a fixed number of strands. Other topology operations like knotting can be applied too [10].

Figure 3.2 represents a braid. The crossing of strands is determined by the order of the solution's variables. In figure 3.2, the order is the following: $\sigma_2^{-2}\sigma_1\sigma_1^{-1}\sigma_2 \equiv \sigma_2^{-1}$

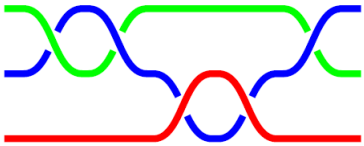


Figure 3.2: Braids swapping.

Let σ_1 and σ_2 represent two generators and σ_1^{-1} and σ_2^{-1} their inverses. Given a braid B, *len()* is a function that returns the braid's length l. Another function, $elen()$ returns the effective length of a braid. This is the braid's length after all simplifications have been done. A braid can be simplified when a generator is directly followed by its inverse representation.

$$\sigma_1\sigma_1^{-1} = I \tag{3.2}$$

Let T represent the target matrix (gate to be emulated), the braid error approximation, ε, is calculated with:

$$\varepsilon = |B - T| \tag{3.3}$$

where B is the product of matrices in the braid. The problem of simulating a quantum gate is reduced to finding a product of simplified generators and their inverses that approximates the matrix representing the quantum gate. The quality of the braid is given by: length of the braid l and error ε [7]. Although in previous works the length and the error approximation has been simultaneously optimized combining them in a single function, in this project we deal with the problem for the first time using a different approach: multi-objective optimization.

3.2 Braid optimization

3.2.1 Simplified problem representation

The goal of the braid problem is to find a product of generators where either the length of the braid is minimized or the error is minimized in the approximation to the quantum gate [7].
Two Fibonacci anyon braid generators are shown in equations 3.4 and 3.5:

$$\sigma_1 = \begin{pmatrix} e^{\frac{-i7\pi}{10}} & 0 \\ 0 & -e^{\frac{-i3\pi}{10}} \end{pmatrix} \tag{3.4}$$

$$\sigma_2 = \begin{pmatrix} -\tau e^{\frac{-i\pi}{10}} & -i\sqrt{\tau} \\ -i\sqrt{\tau} & -\tau e^{\frac{-i\pi}{10}} \end{pmatrix} \tag{3.5}$$

where $\tau = \frac{\sqrt{5}-1}{2}$.
Two different braids with effective length 3 using those generators are:

$$\sigma_1\sigma_1\sigma_1\sigma_2\sigma_2^{-1}\sigma_1 = \sigma_1\sigma_1\sigma_1\sigma_1 = \sigma_1^4 \tag{3.6}$$

$$\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_2\sigma_1^{-1} = \sigma_2^{-1}\sigma_1^{-1}\sigma_1^{-1} = \sigma_2^{-1}\sigma_1^{-2} \tag{3.7}$$

The error is computed as the difference between the braid and the target gate as seen on equation 3.3 from the previous section. The target gate will be the one at which we will try to approximate with our generators.

3.2.2 Traditional approaches

In some of the early approaches described in the literature, the braiding problem was solved using brute-force search where a target error was set [11]. The set of all braids is searched from shortest to longest until finding a braid whose error is smaller or equal to the target error. However, as the braid length grew, the search space grew exponentially, making it unsustainable; but it can still be used if the problem is small enough. Bidirectional search would improve the time of execution [12]. Braid hashing is much faster, but it does not solve the problem of increasing the length of the braid with accuracy. Braid hashing only works for Fibonacci anyons which makes difficult generalization for other

solutions like Majorana fermions [13].
Nevertheless, finding related work in the field of quantum braid computation using evolutionary optimization was very hard.

3.3 Evolutionary optimization for single-objective problems

Evolutionary algorithms are population-based optimization algorithms, which involve the reproduction, random variation, competition and selection of contending individuals in a population [14]. The main idea is that, applying all these techniques natural selection will happen (survival of the fittest) in the population through several generations. Moreover, the fitness of each member will be taken into account by the time a breeding needs to happen; those members with higher fitness will breed. Later, variation will take place thanks to mutation and recombination of solutions. In this manner, the randomization of solutions is guaranteed.
In single-objective problems, the population will try to either maximize or minimize this one objective, depending on the problem description. However, it may seem redundant to use a population of solutions when trying to find one optimal solution [15]. In this manner, various solutions are found instead of just one optimal solution.

3.3.1 Differences from classical optimization methodologies

Evolutionary optimization differs from classical optimization methodologies in:

- Evolutionary optimization does not use gradient information in the searching process. Instead direct search procedures are used.
- Evolutionary optimization methodologies produce more than one solution (*population of solutions*). Using a population-based approach, instead of a single solution approach, is beneficial in many ways: the computation generates many solutions to choose from depending on the researcher's interests, and it normalizes decision variables with the limits set to the maximum or minimum values of a solution.
- Evolutionary optimization uses stochastic operators. This means that the recombination, initialization, breeding, mutation, etc. operators are randomly applied to solutions.

3.3.2 Evolutionary optimization general form

The population where we start from is usually generated randomly (between bounds for each variable). Then it uses 4 main operators iteratively to update the population: selection, crossover, mutation and elite preservation [15]. Algorithm 1 shows the general form every evolutionary optimization algorithm has in a pseudocode representation.

Algorithm 1 General evolutionary optimization procedure

1: $population \leftarrow$ new population of size m
2: **for** $generations \leftarrow i = 1$ **do**
3: Select a subset of individuals according to the fitness function.
4: Apply crossover between selected individuals to produce offspring.
5: Mutate the offspring according to a given probability.
6: Combine current population and offspring to create a new population maintaining the elite.
7: $i \leftarrow i + 1$
8: **end for**

3.3.3 Main operators used in Evolutionary algorithms

The main operators used by Evolutionary Algorithms are the following:

1. **Initialization** is usually done using some random solutions. If there are good solutions available, it might be better to use them.

2. Then **selection** takes place, where better solutions are picked. The chosen solutions are those with a larger probability to fill a mating pool, those who have a better fitness. As said earlier, evolutionary algorithms will derive in natural selection maintaining members with the best fitness.

3. A **variation** is a collection of operators (crossover, mutation, etc) which will be used to modify the population. These are usually applied randomly.

   (a) **Crossover** will pick two or more solutions randomly from the mating pool and create one or more solutions by exchanging information among parent solutions. In this step the offspring will be created.

   (b) Then, each offspring produced by crossover is perturbed in its vicinity by a **mutation** operator which applied according to a given probability will allow an EO to search around a solution and it is independent of other solutions.

4. **Elitism** will combine the old and new populations and keep the best solutions from the combination. To do this, an evaluation function considers all members of a population, choosing the fittest.

5. Finally, a **termination** criterion must be fulfilled. Usually the number of generations is used. Additionally, goal-based criteria can be taken into account when choosing when will the algorithm stop. This goal will most likely be to find a solution with sufficient quality.

3.4 Evolutionary multi-objective optimization

3.4.1 Multi-objective optimization

Multi-objective Optimization techniques are those applied when there is a need for optimizing two or more objectives simultaneously [15]. The biggest problem appears when these objectives are in conflict so that the optimal function for an objective is different for the optimal function of another objective. In order to explain optimal solutions, the concepts of domination have to be introduced first. The symbol $\prec$ is usually found in the literature as a domination symbol [16] [15]. For instance, $x \prec y$, means that a solution x dominates another solution y.

A solution $x^{(1)}$ dominates another solution $x^{(2)}$, if both following conditions are true:

1. $x^{(1)}$ is not worse than $x^{(2)}$ in all objectives. The comparison is done based on their objective function values (location of points $z^{(1)}$ and $z^{(2)}$ on the objective space).

2. $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective.

When dealing with multi-objective optimization, it is common to find in the literature references to multi-objective optimization problems (MOP) [17] [18]. These problems are difficult and costly to solve. Rather than solving a MOP, it is usually said that a good solution has been found, since a MOP could be optimized forever if there are no limits to the solution generation.
Not only do MOPs exist in research environments (Knapsack problem [19], LOTZ [20], quantum braid computation [7], etc.); but in the real world as well (Aircraft construction [21], economics [22], crops distribution [23], etc.).

All the solutions that apply to the constraints of the problem are in a set called Pareto set which represents all the solutions found throughout a tradeoff of optimal solutions. Furthermore, all the points in the set of non-dominated solutions are known as Pareto-optimal points, which as a group conform the Pareto-optimal front. This front gathers the best solutions among the optimal ones found in the Pareto set [15].

In figure 3.3 we could draw an imaginary curve through the green points which would give us the Pareto-optimal front. In this MOP example, both "Objective 1" and "Objective 2" are to be minimized.

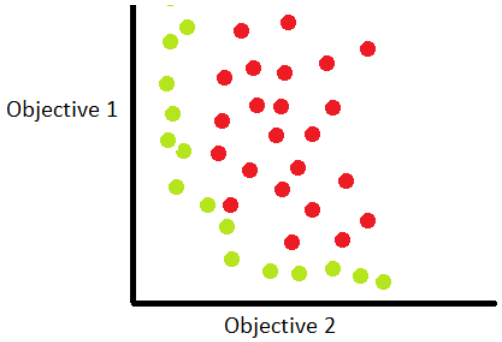


Figure 3.3: Dominated and non-dominated solutions. Dominated solutions are red colored and non-dominated solutions are green colored.

The graph of the Pareto set would change depending on the problem's objectives, minimization or maximization.

3.4.2 Principles of Evolutionary Multi-objective Optimization (EMO) Search

The two main objectives of EMO are:

1. Find solutions that belong to the Pareto-optimal front.
2. These solutions have to be disperse enough in the set.

In a MOP, a user will finally get a set of optimal solutions. The particular solution that a user will select normally depends on qualitative and non-technical comparison between solutions which best fit the user needs [15].

Figure 3.4 represents which are the main steps and guidelines for EMO search. Step 2 is done using qualitative and non-technical comparisons based on the researcher needs.

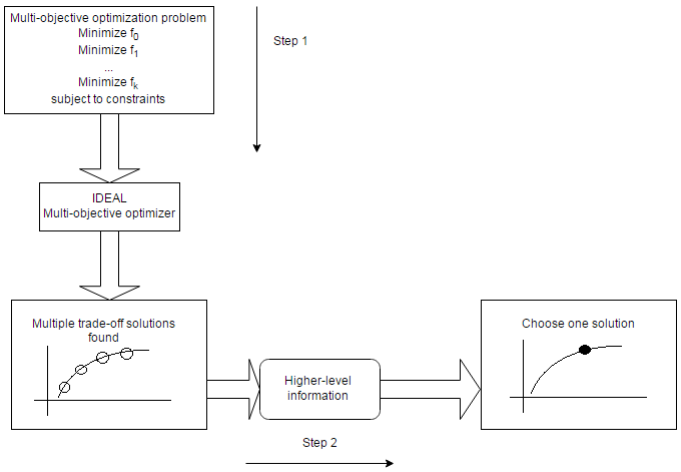


Figure 3.4: Main steps of EMO search.

Classical Methods compared to EMO

In classical multi-objective optimization methods, multiple Pareto-optimal solutions are found executing many independent single-objective optimizations, finding each time a single Pareto-optimal solution [24]; whereas in EMO multiple Pareto-optimal solutions are attempted to be found in a single simulation by looking at non-dominated solutions. Yet, there are classical methods that attempt to find multiple Pareto-optimal solutions in a single run (*e.g.* Schäffler's Stochastic Method or Timmel's Population Based Method). In EMO, when a population member encounters problems (unfeasible regions, local fronts, etc.) and tries to go towards the Pareto-optimal front, its variable values and their combination reflect the fact that EMO is designed to search in parallel. Later this information is shared to other variables through exchanges and blending [15].
In the research paper [24], both classical methods (SSM, TPM and NBI) and MOEAs (NSGA-II) are compared while solving several MOPs. In general, the MOEA performs well on all MOPs, whereas not every classical method performs equally on the tested MOPs. Given all these advantages and disadvantages that both classic methods and MOEAs present, it has been suggested that it is a good idea to design hybrid methods involving EA and classical procedures [25].

3.4.3 Performance Measures Used in EMO

In the early years of EMO many sets of performance measures were used [15] [26] to compare different MOEAs solving a set of MOPs

- Metrics evaluating convergence to the known Pareto-optimal front.
- Metrics evaluating spread of solutions on the Pareto-optimal front.
- Metrics evaluating both convergence and spread of solutions in a Pareto-optimal front (hypervolume, coverage, R-metrics, etc.).
- Metrics involving the overall non-dominated solutions.
- Metrics that measure the computational complexity a MOEA has.

A study [27] concluded that binary performance metrics (ε-indicator, binary hypervolume indicator, utility indicator R1 to R3, etc.) are better measures for multi-objective optimization. In this project we use as a performance measure the hypervolume which is more thoroughly explained in subsection 5.1.4.

3.4.4 EMO and Decision-Making

A multi-objective problem is considered solved once we manage to find a set of Pareto-optimal solutions. Another equally important task is to decide on a single preferred solution. There is a need for decision making since many problems in the real world are multi-objective. Furthermore, we might sometimes want to scale the problem at a cost of making a decision [28]. There are 3 main types of decision making strategies we could follow according to [15] and [29] presented below.

1. A-priori approach: preliminary information is used to focus the search effort on a part of the Pareto-optimal front, instead of on the entire frontier, with data we already know, such as patterns or distribution probabilities.
2. A-posteriori approach: preference information is used after a set of representative Pareto-optimal solutions are found by an EMO procedure. This approach is valid for few-objective problems [30].

3. Interactive approach: the preference information of a decision maker (DM) is integrated into an EMO algorithm during the optimization run. The DM will be called after every t generations and it is presented with a few well-diversified solutions chosen from the current non-dominated front [31]. Then DM will rank the solutions according to preference.

There is also software available for data visualization and decision making guidelines such as VIDEO [32]. This framework graphs the data from the output of an EMO and gives metric information which would help the researcher with the decision making procedure.

3.5 Evolutionary approach to the braid problem

The algorithms proposed in this thesis belong to the group of MOEAs (Multi-objective Optimization Evolutionary Algorithms). Their objective is to find an optimal braid, and to do so, the error approximation to a determined target gate has to be small enough and the length of the computed braid too.
Like any other evolutionary algorithm, an initial population will be generated, and it will evolve through several generations until a maximum generation has passed or until a good solution has been found. In each generation, a parent solution will create an offspring whose individuals represent braids. Mutation and recombination operators are applied to the solutions which conform the braids.
In chapter 5, the evolutionary computation approach to the braid problem is more thoroughly covered.

The pseudocode shown in algorithm 2 is a simulation of the mono-objective approach conducted by McDonald and Katzgraber [7], where m is the size of the population, *generations* is the number of generations to evolve the population, and *best* is the current best braid. The culling operator will remove the 10% of the solutions, whereas breeding refills the mating pool up to a 100%. The algorithm will run until the generations counter reaches the maximum, and will finally deliver the best found braid. The best braid is considered the braid with the highest fitness value.

Algorithm 2 Braid evolutionary optimization

```
1: population ← new population of size m
2: for generations ← i = 1 do
3:     Sort population ascending by fitness
4:     if fitness(best) < fitness(population[m]) then
5:         best ← population[m]
6:     end if
7:     Perform culling (least fit 10% removed)
8:     Repopulate missing 10% with breeding
9:     i ← i + 1
10: end for
```

3.6 Estimation of Distribution Algorithms (EDAs)

EDAs are evolutionary algorithms that apply learning and sampling of distributions instead of crossover and mutation operators like other evolutionary algorithms would do. The search for optimal solutions is done building and sampling probabilistic models of candidate solutions. A probabilistic model is any form of modeling which uses known probability distributions to calculate the probability distribution for the output. In other words, it is a tool to try to predict what the output of a problem would be, based on probability distribution. The continuous optimization is achieved thanks to the incremental update of the probabilistic model in each evolved generation until a termination criterion is met [33]. In other words, EDAs are algorithms which search the solution space by building and sampling probabilistic models of promising solutions [34].

Population-based EDAs start with a candidate initial population. This first population is generated according to a uniform distribution over all admissible solutions. Then, like in a EA would happen, the population is given a fitness value. Once this is done, the algorithm will construct a probabilistic model trying to estimate the probability distribution of the selected solutions. Coming new solutions will be generated following the probabilistic model. Finally, these new solutions are introduced in the population replacing old members, or even the whole population [35]. Some population-based EDAs will have a probability vector to store the probabilities of candidate solutions. However, in some cases this probability vector is not sufficient. It really depends on problem the EDA is trying to solve, but an example where the probability vector is insufficient would be in a MOP where several probabilities should be stored [34].
In incremental EDAs, the population of solutions is fully replaced each generation by the probabilistic model.

3.6.1 EDA categories

EDAs can be categorized in two big groups: discrete and real variables. We will focus in discrete EDAs which work on a finite cardinality (binary or integer representations). In this group, there are categories too, depending on if the problem's variables are related with each other (multivariate EDAs) or not (univariate EDAs). The categories are related to the type of distribution the EDA will be able to encode [34] [36].

- **Univariate models:** in this type of EDA model, the variables are considered independent. The probability of distribution of a variable does not depend on another variable. The model biases the search to improve itself for future iterations. Because the model is improved each generation, there is no need to store each generation or even an archive.
- **Multivariate models:** unlike univariate models, this type of EDA considers that the variables are related with each other. The way they are related may adopt different representations:
  - **Tree EDAs:** like the tree structure, these EDAs will be able to capture pair interactions between their variables.
  - **Bayesian networks:** a Bayesian network is a directed graph with no cycles. Each node is dedicated to a variable, where an edge between two nodes represents a dependency of two variables. The difference between these and Tree EDAs, is that in these each variable can depend on more than one other variable.
  - **Markov chain-shaped EDA:** this is another type of multivariate EDA based on Markov networks. Markov networks are represented by undirected graphs and therefore, the connections between variables in a Markov chain are undirected.

3.6.2 Main differences between EDAs and EAs

The main difference between EDAs and other evolutionary algorithms is that the last generate new candidate solutions using an implicit distribution defined by one or more variation operators, whereas EDAs use an explicit probability distribution encoded by a model (like Bayesian networks or Markov chains) [33].
The following pseudocode represents what any EDA must basically do.

Algorithm 3 General EDA procedure

1: $t = 0$
2: Initialize the model M_0 to represent a uniform distribution of feasible solutions.
3: **while** Termination criterion is not met **do**
4: $P =$ generate $N > 0$ candidate solutions by sampling M_t
5: $F =$ evaluate all candidate solutions in P
6: $M_{t+1} =$ adjust model(P, F, M_t)
7: $t = t + 1$
8: **end while**

3.6.3 Benefits and disadvantages of EDAs

EDAs have shown to outperform other classic methods in many different fields of study. However, there are downsides too. Thus, it is usually more complex to design the probabilistic model instead of using an algorithm which will eventually solve the problem. Additionally, it is more difficult to implement EDA procedures, rather than simple search operators. Nevertheless, the time of learning a model is worth the effort [33] [34].
As for the benefits, the already mentioned benefits of using EAs are present in EDAs too. EDAs also require less memory compared to other methodologies [37]. They also are more scalable to other more complex problems. Although, choosing an initial probabilistic model can constrain this scalability; in this case, adaptive EDAs could be used [38].

3.6.4 EDA applications to real-world problems

Applications of EDAs to real world problems vary widely in topics: military antenna designs [39], protein folding [40] [41] or chemotherapy researches [42]. A previous EDA approach to the braid problem included the use of a local optimizer, partial sampling and recoding [43].

CHAPTER 4

Evolutionary Multi-Objective optimization approach to the braid problem

This chapter presents the main steps a MOEA has to take in order to solve a braid optimization problem. Then, the three MOEAs which are used for experimentation are explained. Generally speaking, the length of the braid and the error have to be minimized through several generations. Each time a new generation is produced the set of solutions, this is, the approximation to the Pareto set will improve. Lastly, a Pareto front will be drawn containing the braids with the best possible quality. Even though, different approaches have tried to solve the braid problem, it has never been done using a MOEA approach.

4.1 The braid and its variables

In our domain, a braid represents a solution. Furthermore, each variable i composing the braid, represents which of the available generators is multiplied in position i.
A braid can also be seen as a collection of generators, which are the braid's variables. When we started to work on the project, the first approach used a binary representation for each variable. After a binary initialization, a function translated this binary representation into an integer representation in order for the solutions to be evaluated. Depending on the variables an individual had, the translated integer value was bounded between 0 and $2^n - 1$, where n is the number of variables.

However, using binary based variables is not a very good procedure. In the Fibonacci anyon braid case, which uses a cardinality of 4, representing each generator in binary is easy since $2^2 = 4$. This means that, with two bits we could represent all the solutions. But, what if the cardinality was not a power of 2?. Majorana fermions, for instance, have cardinality 10. Since there is no way for a power of 2 to produce 10, we would need the power which produces a result close to 10, which is 4, $2^4 = 16$. This way we would be simulating a problem with cardinality 16 and no 10.
This is the main reason we started to work with solutions containing discrete integer variables bounded between 0 and $n-1$ where n is the cardinality of the MOP. Notice that using an integer representation means that we would have to adapt some of the variation operators included in the software, since they have mean mainly implemented for binary problems.

4.2 Length of the braid

The length of a braid is given by the number of generator matrices multiplied with each other to obtain it [43]. For example, for $B = \sigma_1\sigma_2$, $len(B) = 2$.

This is one of the objectives that needs to be minimized in order to achieve an optimized braid. Because the problem is multi-objective, both length and error approximation have to be optimized simultaneously.

4.3 Quality assessment of the braid

Saying quality of the braid is alike to saying the error approximation to a target gate from a multiplication of generator matrices. In the application of the MOEAs to the braid problem, one of the objectives is to minimize this error approximation, or archiving better quality braids. An indicator for the quality of a braid is the distance between this braid and the target gate; the less the distance to the target gate is, the better the braid is. Although, the computation of quality depends on the way the evaluation of the solutions is done by the MOEA.

The calculation of the quality of the braid in the MOEAs used for this thesis, is done in two steps:

- First let us subtract each cell of the target gate to the corresponding cell of the braid matrix.
- The quality of the braid is given then by the result of multiplying each cell of the resulting matrix by itself and summing all these values. However, not everybody uses the same quality representation; for instance $quality = \frac{1}{1+length}$ is used in [7].

The following equations are an example of how the distance to a target gate is computed:

$$\text{target gate} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \text{braid matrix} = \begin{pmatrix} -1 & 7 \\ 0 & 13 \end{pmatrix} \tag{4.1}$$

$$distance(\text{target gate, braid matrix}) \equiv \begin{pmatrix} 1-(-1) & 0-7 \\ 0-0 & 1-13 \end{pmatrix} \tag{4.2}$$

$$\begin{pmatrix} 2 & -7 \\ 0 & -12 \end{pmatrix} \equiv 2^2 + (-7)^2 + 0^2 + (-12)^2 = 197 \tag{4.3}$$

4.4 Non-dominated Sorting Genetic Algorithm (NSGA-II)

NSGA-II was presented as a better alternative to other existing MOEAs (SPEA [16] and PAES[44]). It was developed by Kalyanmov Deb *et al* in 2002 and it looked forward to achieving better results than its predecessor, NSGA.
Algorithm 4 shows the main parts of the algorithm in pseudocode format. Notice that in line 8 of Algorithm 4, the program calls the non-domination sorter presented in Algorithm 5.

Part of the software analysis of this project included the study of the original implementation of NSGA-II and of the PISA version. Therefore, NSGA-II will be used in the experimentation phase. The decision of choosing this algorithm over others was mainly because of its popularity and good performance in other MOP solving [19] [45].

In the Algorithm 4, P represents the parent population, Q the offspring population, R the archived population, and t the generation counter. At first, all of them are put ready. Then the parent population and all its members are randomly initialized. Now a loop is entered until the generation counter reaches the maximum. The parent and offspring populations are archived in R_t. Then the non-dominated sorting takes place. The parent population of the next generation is now prepared. A new loop is entered while fulfilling

the condition that the parent population plus the non-dominated front sizes are less than N, which represents the maximum number of individuals a population can have. Once the loop is left, the non-dominated front is sorted according to the crowding distance [46] [45].

Algorithm 4 Pseudocode of NSGA-II

```
1:  P ← ∅
2:  Q ← ∅
3:  R ← ∅
4:  t ← 0
5:  Randomly initialize P
6:  while t < maximum generation number do
7:      R_t ← P_t ∪ Q_t
8:      Sort all solutions of R_t using the fast non-dominated sort procedure 5
9:      P_{t+1} ← ∅ and i ← 1
10:     while the parent population size | P_{t+1} | + | F_i | < N do
11:         Calculate crowding-distance of F_i
12:         P_{t+1} ← F_i
13:         i ← i + 1
14:     end while
15:     Sort F_i accordingly to the crowding distance
16:     Fill P_{t+1} with the first N − | P_{t+1} | elements of F_i
17:     Generate the next offspring Q_{t+1}
18:     t ← t + 1
19: end while
```

Algorithm 5 shows a fast non-domination sorting method as proposed in [45]. In the algorithm, n_p represents a domination count, which has the value of the number of solutions which dominate the solution p. Additionally, a set S_p is presented. This set contains the solutions p dominates. All the solutions will have their domination counter set to zero in the first non-dominated front. For each solution p with $n_p = 0$, each member q of the set S_p will be visited and reduced its domination count n_p by one. If any member q has its domination counter set to zero, it will be put in a separate list Q. Q will contain the second non-dominated front. The described process is repeated until all fronts are identified and included in Q. Finally, for each solution p with a domination count equals to zero, it will be assigned a non-domination level and will never be visited again [45].
The objective of the non-dominated sorting procedure is to ease and hasten the search for the fittest solutions. Also, the non-dominated sorting mechanism present in NSGA-II outperforms the non-dominated sorting procedure NSGA had.

Algorithm 5 Pseudocode of the non-domination sorting of NSGA-II

```
1:  for p ∈ P do
2:      S_p ← ∅
3:      n_p ← 0
4:      for q ∈ P do
5:          if p ≺ q then                          // If p dominates q
6:              S_p ← S_p ∪ {q}                    // Add q to the set of solutions dominated by p
7:          else
8:              if q ≺ p then
9:                  n_p ← n_p + 1                  // Increment the domination counter of p
10:             end if
11:         end if
12:     end for
13:     if n_p = 0 then                            // If p belongs to the first front
14:         p_rank ← 1
15:         F_1 ← F_1 ∪ {p}
16:     end if
17: end for
18: i ← 1                                          // Initialize the front counter
19: while F_i ≠ ∅ do
20:     Q ← ∅                                      // Used to store the members of the next front
21:     for p ∈ F_i do
22:         for q ∈ S_p do
23:             n_q ← n_q − 1
24:             if n_q = 0 then                    // if q belongs to the next front
25:                 q_rank ← i + 1
26:                 Q ← Q ∪ {q}
27:             end if
28:         end for
29:     end for
30:     i ← i + 1
31:     F_i ← Q
32: end while
```

4.4.1 NSGA-II's procedure

Overall functioning

The algorithm is presented in 4 and the sorting mechanism in 5. The initialization is done generating a random parent population which is firstly evaluated. Then an offspring population is created based on binary tournament, recombination and mutation operators. The algorithm will need parameters to do these operations, such as recombination and mutation factors. Because elitism is used, the procedure will change in upcoming generations. We will archive the best solutions of each generation.

When the first generation is done, the MOEA will go through a loop until a finalization criterion is met. The parent and offspring populations are merged in a new population which is sorted according to non-domination. Then, the best non-dominated solutions are chosen until we reach N members, where N represents the initial population's size. The sets of individuals are sorted in decreasing order according to the crowded comparison operator.

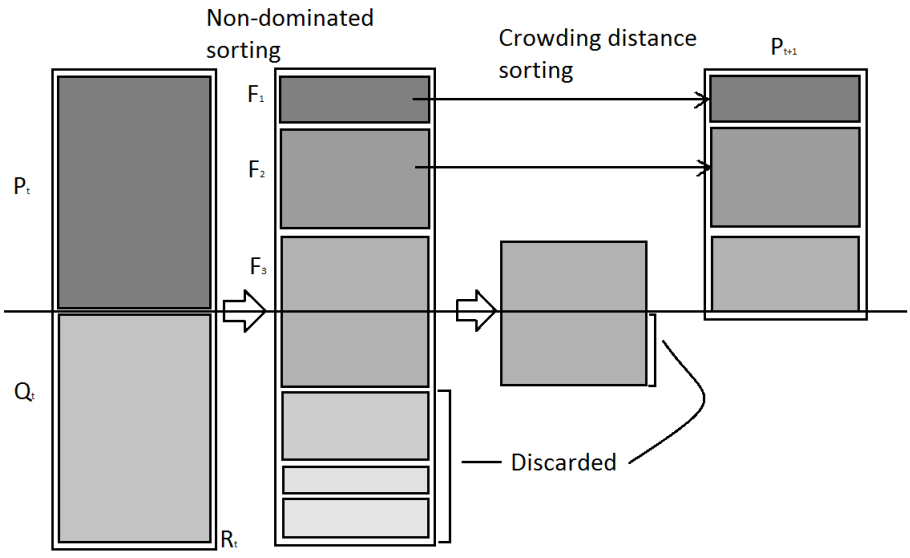


Figure 4.1: P_t represents the parent population of a generation and Q_t the offspring population. The populations are considered as a single set of individuals that are sorted based on their fitness, F_N. Half of those individuals are chosen and a new parent population, P_{t+1} is created.

Finally, the result population of size N is considered the new generation's parent population and therefore yielded to selection, crossover and mutation, to create a new offspring population.

The selection phase takes into account the best solutions, this is, elitism. It will select the non-dominated solutions firstly preserving the elite.

Crossover and mutation requires mutation and crossover probabilities which are parameters set by the user.

4.4.2 Improvements over other MOEAs

Throughout the evolution of NSGA to NSGA-II some changes have been introduced in the algorithm in order to outperform other MOEAs. Below, these mechanisms are briefly explained [45].

Elitism

In the first version of the MOEA, NSGA; the elite preservation was not used. However, an empirical study [19] has shown that maintaining the elite can improve the algorithm's performance greatly. The elitism approach involves selecting non-dominated solutions in each generation of the algorithm.

Diversity preservation

Previously, NSGA used a sharing function approach that relied on a sharing parameter σ_{share}. The parameter, chosen by the researcher, denotes the largest distance between any two members of a population. This way, the given Pareto front members, would not be very grouped. However, the complexity of this task was $\Theta(N^2)$, and therefore too complex.
In NSGA-II, instead of population distance between members, a density estimation function is used. This density estimation is computed using the distance between a solution and its two neighbors. This distance, also called crowding distance, represents an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices.

This computation requires the population to be sorted ascendant according to the objective functions. Finally, the overall density is calculated summing the average distances between adjacent individuals of the population.

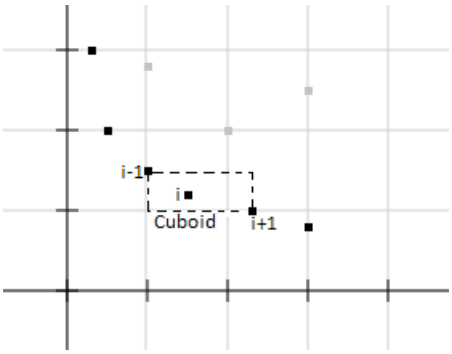


Figure 4.2: Crowding distance of solution i with its neighbors represented as the cuboid surrounding it.

4.4.3 Decreasing computational complexity

Usually, MOEAs have shown a computational complexity of $\Theta(MN^3)$, where M is the number of objectives and N the population size. Nevertheless, NSGA-II provides a procedure capable of achieving $\Theta(MN^2)$ complexity value. This value is obtained by sorting non-dominated solutions rapidly.

Each individual can be member of at most one non-dominated front, having to execute this front calculation N times at most. Also, each individual dominates $N-1$ individuals at maximum and each domination check requires M comparisons at most. Thus, adding the results leads to achieve an overall $\Theta(MN^2)$ computational complexity, although the storage requirement increases to $\Theta(N^2)$ [45].

4.5 Strength Pareto Evolutionary Algorithm (SPEA2)

This chapter introduces the MOEA SPEA2 which will be used in the experiment phase. SPEA2 is the improved version of the original SPEA developed by Zitzler and Thiele [47]. SPEA2 tries to correct the errors and weaknesses of SPEA and it is an up-to-date solution. Like NSGA-II, SPEA2 will also be used to do the experiments. Firstly, some research was done on the original SPEA2 implementation and then we worked with the PISA implementation.

4.5.1 SPEA

SPEA uses a population and an archive (external set). Starting with an initial population and an empty archive, the following steps are done per generation. **First**, all non-dominated individuals are copied to the archive. Dominated individuals or duplications are removed from the archive during this update operation. If the size of the archive is bigger than a limit after copying the non-dominated solutions, the least fit members of the archived population are removed in order to preserve the characteristics of a non-domination front. Then, fitness values are assigned to both archive and population members [47]. To compute the fitness, first each member is given a strength value which represents how many other members dominates or are equal to it, divided by the population size plus one. Then, the fitness is calculated by summing the strength values of all archive members which dominate or are equal to a solution, and adding one at the end. Because this fitness value is assigned based on how many members dominate a solution, the less, the higher fitness will have.

The next step is the **mating selection** where individuals from the union of population and archive are selected by means of binary tournament. This tournament is held comparing the fitness values of adjacent solutions two-by-two. **Finally**, after **recombination** and **mutation**, the old population is replaced by the resulting offspring population [47].

4.5.2 SPEA2

The pseudocode of SPEA2 is presented in Algorithm 6; where P represents the parent population, A the archive population and Q the offspring population. Note that after each generation has passed, Q will be considered as the new parent population.
Firstly, an initial parent population is created and initialized randomly. The archive population is also created and set ready to be filled. Once the loop is entered various operations happen. First, all the solutions in P and A are evaluated and given a fitness value based on the objective function or objective functions if it is a MOP. Then, all the non-dominated individuals from P and A are copied to A. The capacity of A is evaluated in order to truncate or fill it. From the archive A, the best solutions are selected and given to an offspring population Q. Finally, crossover and mutation is applied to Q to ensure diversity of solutions [48] [16].

Algorithm 6 Pseudocode of SPEA2

```
P ← ∅
P is now initialized                                   // P is initialized at random
A ← ∅
while !Stop condition do
    Compute fitness of each individual in P and A
    Copy all non-dominated individuals from P and A to A
    if The capacity of A has been exceeded then
        Truncate A to remove individuals from A
    else
        Include some of the dominated individuals from P into A
    end if
    Perform selection to fill Q
    Apply crossover and mutation to Q
end while
```

4.5.3 Differences between SPEA and SPEA2

The main improvements of SPEA2 over SPEA are [16]:

- Improved fitness assignment scheme. For each individual in the population, it computes how many other individuals it dominates and how many individuals dominate it.
- A nearest neighbor density estimation technique that allows to search more precisely.
- New archive truncation methods which preserve the boundary solutions.

4.5.4 Results

Mention that SPEA2 has shown to have a computational complexity of $\Theta(M2logM)$, due to the truncation procedure it uses. In that *big O* notation, $M = N + N'$, where N is the population size and N' the archive population size [16].
SPEA2 outperforms its predecessor SPEA in many known MOPs like the Kursawe function or the Quagliarella and Vicini function [16]. It is slower, in terms of computational complexity, than NSGA-II, but it is because the density estimation SPEA2 makes. SPEA2 shows a better distribution of solutions, but NSGA-II is able to find solutions closer to the edges of the Pareto front. The techniques of density estimation could be beneficial with a

growth of the objective space. However the archiving and truncation methods would have to be updated in order to deal with more objectives [49] [16].

SPEA2 has been used to solve a wide range of problems, MOPs and real world problems which vary from neural network training [50] to web development [51]. Through the years, improvements to SPEA2 have appeared such as SPEA2+ [52] and ISPEA [53].

4.6 Fair Evolutionary Multi-objective Optimizer (FEMO)

FEMO is a MOEA proposed by Laumann *et al* in [20]. It is an alternative to their previously introduced SEMO (Simple Evolutionary Multi-objective Optimizer). It was presented in 2002 as a new paradigm of black box multi-objective optimization.
This chapter summarizes the behavior of the FEMO algorithm and explains its main parts. Finally, some results found in previous investigations are presented.

4.6.1 SEMO

In order to explain FEMO, SEMO should be introduced first, since FEMO is the improved version of SEMO.

SEMO is an algorithm which comprises a population of variable size that stores all non-dominated individuals; introducing the concept of archive. The mutation of SEMO is done by flipping a randomly chosen bit (if binary decision variables are considered) of an individual, turning it into a different individual. Moreover, the parent population is selected using a probability factor of uniform distribution.

In algorithm 7 the SEMO pseudocode is presented. P represents the parent population and x each individual of the population. The criterion for the algorithm to stop is usually a generation counter, but like in earlier explained MOEAs (NSGA-II and SPEA2), the stopping criterion can be the discovery of a sufficiently good solution.

The step 7 of the algorithm 7 explains the way new solutions are added to the archive. After a new solution x' has been generated, it will be added to the archive. If there is not any other solution $z \in P$ that dominates x' or has the same function values as x', means that the solution x' will be part of the parent population for the next generation.

Algorithm 7 SEMO pseudocode

1: Choose an initial individual x uniformly from $X = \{0,1\}^n$
2: $P \leftarrow \{x\}$
3: **while** A generation counter is not reached or until a desired solution is not reached **do**
4: Select one element x out of P uniformly
5: Create offspring x' by flipping a randomly chosen bit
6: $P \leftarrow P\{z \in P \mid x' \prec z\}$
7: **if** $\nexists z \in P$ such that $(z \prec x' \vee f(z) = f(x'))$ **then**
8: $P \leftarrow P \cup \{x'\}$
9: **end if**
10: **end while**

4.6.2 FEMO

FEMO was born as the improved version of SEMO. SEMO's biggest weakness was the fact that a large number of mutations happen to parents whose neighborhood had already been visited enough. Also, because uniform sampling is used, the populations are sampled unevenly depending on the time an individual entered the population [20]. A fair sample would solve this problem, guaranteeing that all individuals would receive, approximately, the same amount of samples.

4.6.3 Fairness mechanism

In terms of MOEAs, fairness relies on the fact that every individual of a population should have the same possibility of being mutated [20]. Compared to other MOEAs, FEMO does not prefer an individual over another one by how close it is to others, like NSGA-II; or any other preference mechanism. This way, the descendants are generated by fairly chosen individuals. This fairness could be good when all the individuals of a population have created approximately the same amount of descendants [54].

FEMO works with a local mutation operator and uses a counter for each individual in the population to measure the number of descendants it has created. As seen in line 10 of algorithm 8, the counter of an individual, $w(x)$, would be set to 0 whenever a new individual is produced.

Algorithm 8 Pseudocode of FEMO

Choose an initial individual x uniformly from $X = \{0,1\}^n$
$w(x) \leftarrow 0$
$P \leftarrow \{x\}$
while A generation counter is not reached or until a desired solution is not reached **do**
  Select one element x out of $\{y \in P \mid w(y) \leq w(z) \forall z \in P\}$ uniformly
  $w(x) \leftarrow w(x) + 1$
  Create offspring x' by flipping a randomly chosen bit
  $P \leftarrow P\{z \in P \mid x' \prec z\}$
  if $\nexists z \in P$ such that $(z \prec x' \vee f(z) = f(x'))$ **then**
    if $x' \notin P$ **then**
      $w(x) \leftarrow 0$
    end if
    $P \leftarrow P \cup \{x'\}$
  end if
end while

4.6.4 Results

In [54], Friedrich *et al* make a distinction between the computation of the decision space and the objective space. They also notice that a FEMO which searches the decision space performs much better than a FEMO which uses the objective space while trying to solve the *PL* problem they present in [54]. The *PL* problem is a similar problem to the single objective function SPC that analyzes which solutions choose from a set of solutions of same fitness [55].

FEMO has also served as the first MOEA showing that the concept of population and generations was better than the traditional multi-objective approaches which involved function scaling [17].

However, Giel [56] suggests than FEMO and SEMO should only be used if we knew that the MOP we are trying to solve will not be trapped in a local optimal solution and be running forever. Because of this, Giel proposes a variant of SEMO called global SEMO. FEMO also had the version of global FEMO [54]. Another alternative would be a greedy approach named GEMO (Greedy Evolutionary Multiobjective Optimizer) in which a greedy selection takes place, selecting among the most recently successful mutant from the offspring [57].

Thus, FEMO reaches a computational complexity of $\Theta(n^2 logn)$ [54] [20].

CHAPTER 5

Design and development

This chapter is focused on explaining how the design and coding of the tools used to solve the braid problem have been done.
There are three distinguishable phases where the coding has been an important task. These phases are:

- Usage and adaptation of the PISA framework to the braid problem.
- Braid problem design and coding.
- Scripting for other tasks such as data visualization or graphs generation.

Using the phases as a roadmap, the chapter firstly introduces how the PISA framework works and the parts it has. Later, it presents how the braid problem representation design and development has been done, and finally, the scripting phase is described.

5.1 PISA framework

This section aims to explain how the PISA interface works. It is based on the publication [58] and on the PISA framework itself.

5.1.1 Description of the framework

PISA is a platform and programming language independent framework for search algorithms. It is mainly focused on multi-objective optimization algorithms and it also has some MOEAs implemented. Moreover, it provides metrics analysis and statistical test modules. Figure 5.1 shows the modules that intercede in the PISA framework.

It was developed by Zitzler *et al* in the Computer Engineering and Networks Laboratory (TIK) in ETH Zürich. Like any other multi-objective optimization platform, it is intended to both solve real-life optimization problems and help researchers develop other optimization algorithms.

It mainly consists on two parts:

- A text-file-based interface which works with search algorithms. It is organized in two major modules: the variator and the selector. These two modules will communicate with each other through text files containing common parameters.
- A library containing problems (variators), solving algorithms (selectors) and performance assessment modules.

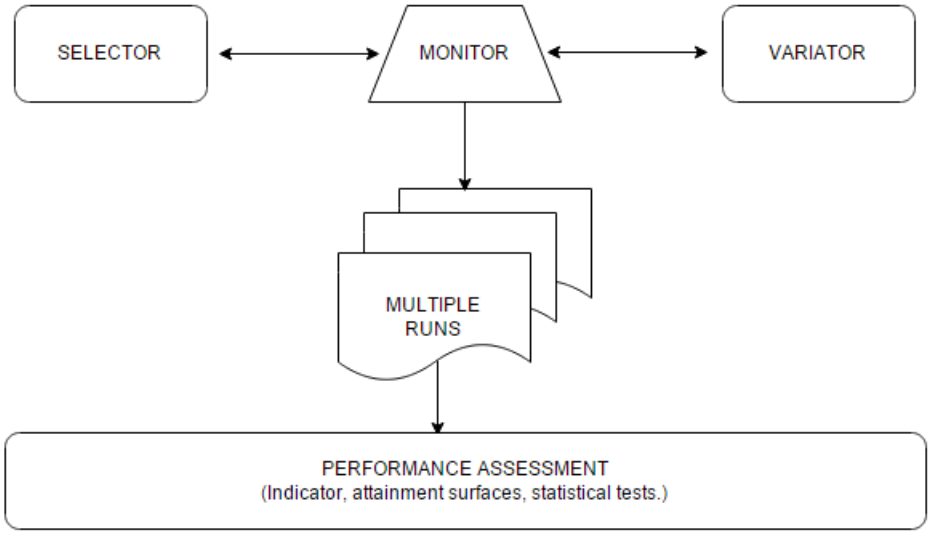


Figure 5.1: Modules involved in PISA.

As a consequence of its modular design, PISA can be extended with own implementations of variators and selectors. In our case, after understanding the organization of the

software, we had to develop a variator capable of representing the braid problem. Next, some selectors were used to solve it.

5.1.2 Organization of variator and selector modules

These two modules are implemented in a similar manner. Each module has: the executable file, which holds the variator or the selector and a parameter file.

The variator parameter file will contain data such as the mutation and crossover probabilities, maximum number of generations or which problem is going to be executed. The selector parameter file contains information like which type of tournament selection is going to take place and the archive size of the population.

Moreover, the following text files are allocated in a common directory for the selector and the variator to work. These are the files containing information about the population and the ones that will be updated in each generation:

- **Initial file**: the initial and freshly created population will be represented in this file. It will only be used to keep the initial population.
- **Archive file**: this file will keep track of the solutions of a generation that are feasible.
- **Selection file**: once a selection of a population is done, its representation is reported here.
- **Variation file**: after doing mutation and crossover, a new variated population is created, which is captured in this file.
- **State file**: probably the most important file of all, since it is the one that holds the state of the execution. Both variator and selector are continuously reading this file checking in their implementations if there is something to do in a certain state. If a module is working with the previous files, the other module will not do so.

There is also a configuration file which contains information such as the population maximum size, the offspring size, etc.
Even though many selectors are available, we will put special effort on NSGA-II due to have been working with it firstly. Also, SPEA2 will be used since it has achieved similar results to NSGA-II in previous studies [19] [45]. In addition, FEMO [20] will be used too,

hence its fairness properties. In the initial experiments we saw that FEMO behaved better than NSGA-II and SPEA2 when dealing with the braid problem.

5.1.3 Monitor module

There is a module named 'Monitor' [59], which is also part of the PISA framework. The main role of this module is the automation of runs of algorithms. This way, a MOEA will try to solve a problem a certain number of times, namely runs. It is implemented using a bash script. It has been modified to output the time consumption each MOEA takes to solve a given problem. Additionally, more scripts can be added to extend its functionality.

The monitor will initiate the selector and then the variator. Once the maximum generation is reached, monitor will reset the common files (state, initial, etc.) and execute another run. It will do this until all the runs have been executed.

Finally, two types of files are generated. One will have general information of all the runs, like the date of execution or the parameter files used. The other kind of file contains information about the populations generated in each generation of each run.

As a part of the PISA project, this module will also work with text-file communications. As shown in figure 5.1, the monitor module is located between the variator and the selector, and acts like a supervisor for both.

5.1.4 Performance assessment

The performance assessment module is delivered in a package under the name of 'Distribution'. This module tests the performance of MOEAs solving certain problems, through statistical tests and metrics. To achieve this, the module monitor is executed a number of runs for different seeds in order to reach randomness.
The package works with system calls. It includes Solaris, Windows and Linux versions.

Metrics

Three indicators are recommended [60]:

- **Unary epsilon indicator**: in order to explain the unary epsilon indicator, the binary indicator has to be presented first. The binary epsilon indicator, $I_\varepsilon(A,B)$, gives the

minimum factor ε by which objective vector associated with B can be multiplied, such that the resulting transformed Pareto front approximation is weakly dominated by the Pareto front approximation represented by A

$$I_\varepsilon(A,B) = \inf_{\varepsilon \in \mathbf{R}} \{\forall x^2 \in B \exists x^1 \in A : x^1 \preceq_\varepsilon x^2\} \tag{5.1}$$

This indicator relies on the ε-dominance relation, $\preceq_\varepsilon$, defined as :

$$x^1 \preceq_\varepsilon x^2 \Leftrightarrow \forall i \in 1..n : f_i(x^1) \leq \varepsilon \cdot f_i(x^2) \tag{5.2}$$

Similarly, the unary epsilon indicator, $I^1_\varepsilon(A)$ can be defined as :

$$I^1_\varepsilon(A) = I_\varepsilon(A,R) \tag{5.3}$$

- R **(Representative) indicators**: these indicators were used to asses the relative quality of sets of non-dominated vectors with respect to a reference set R [62].

- **Hypervolume indicator**: this indicator is used under the constraints of attainment functions which show dominance relations. These attainment functions give, for each objective vector, the probability that it is weakly dominated by the outcome of a MOEA. The attainment function is given by the following equation.

$$\alpha_A(z) := \begin{cases} 1 & \text{if } A \succeq \{z\} \\ 0 & \text{else} \end{cases} \tag{5.4}$$

For $z \in Z$, the weakly dominated objective vectors will be given value 1. Else, value 0 will be given. This means that all weakly dominated objective vectors have the same weight and contribute equally to the indicator value. The main idea is to give different weights to different regions in the objective space according to the dominance of solutions. This way, the indicator will tell us which region is more or less dominated [63]. This metric's complexity increases as the number of objectives increase.

Therefore, the hypervolume indicator is represented by the following formula [64].

$$I^*_H(A) := \int_{(0,\ldots,0)}^{(1,\ldots,1)} \alpha_A(z)dz \tag{5.5}$$

Where A is any objective vector set. Thus, the hypervolume calculates the dominated volume of the optimal solution sets, with respect to the reference sets.
In other words, the hypervolume measures the area that the Pareto set covers intersecting with some reference points. These reference points are the edges of a so-called hypercube, which is an area enclosed by said reference points. It will also give us information about the closeness of solutions in the Pareto front and the spread of solutions of the objective space [65] [66].

For the experimentation phase, the hypervolume is the metric which we will use. This is because many articles propose to apply it when the MOEAs we selected are used [67] [68].

Statistical tests

where R is any reference set of points. An indicator value less than or equal to 1, implies that A weakly dominates the reference set R [61] [27]. Statistical tests are used in this project to determine if there are statistical differences in the behavior of 2 or more MOEAs when applied to the problem.

These statistical tests will compare the indicator value samples given by the chosen indicator metric. The PISA package is distributed along these five statistical tests, which are categorized in how many independent samples they can work with. The tests try to do resampling:

- Two independent samples
  - **Fisher's permutation test**: the output of the test, is the p-value of the one-tailed test for rejecting the null hypothesis of no difference between each pair of sample populations.
  - **Mann-Whitney U test**: the output of the test is the p-value of the two-tailed test for rejecting the null hypothesis of no difference between each pair of sample populations.
- Two matched samples
  - **Wilcoxon signed-rank test**: the output is the one-tailed p-value that the null hypothesis is true for each pair.

  - **Fisher's matched samples test**: the output of the test, is the one-tailed p-value that the null hypothesis is true for each pair.

- Three or more independent samples
  - **Kruskal-Wallis test**: if and only if a first test for significance of any differences between the samples is passed, at the given alpha value, then the output will be the one-tailed p-values for rejecting a null hypothesis of no significant difference between two samples, for each pair-wise combination.

The statistical test which we will use in the experimentation phase, will be the Mann-Whitney U-test.

5.1.5 Applications of PISA

The framework has been tested and used in several fields of study involving MOEAs. Some studies focus on the solutions given by a MOEA to a certain optimization problem. Although most use the metrics available in the performance assessment module in order to better understand the results.
A couple of examples where PISA has been used are a chemical neutralization plant's PID controllers [69] or a bus network scheduling problem proposal [70]. Nevertheless, many more projects have used PISA and can be found in the literature. It has also served as predecessor of the Java MOP solving framework jMetal [71].

5.2 Development of the braid problem

Even though the framework PISA has some popular optimization problems implemented like Knapsack or the DTLZ group problems, it does not have the braid problem implemented. This is why, basing on the already implemented problems, the thesis' supervisor and I implemented the braid problem. Specifically, the Knapsack problem was chosen as a starting point since it is too a minimization multiobjective problem.

Initially, I was given the code used for a single-objective EDA approach to the braid problem [7] [43] written in C++. Firstly, I tried to understand how the code worked and what parts it had. Finally, I took a data structure implementation to work with matrices called "Quaternion" from their work and adapted it to the population representation PISA used.

5.2.1 Quaternion structure

The quaternion was created as a mathematical structure by Hamilton [72]. The quaternion appeared as an alternative to be used in non-commutative algebra. In our context, quaternions come in handy since Fibonacci anyon braids are also non-commutative, non-abelian.
MOEAs are population based algorithms. The population will contain individuals which are possible solutions to the problem as well. In our case, a solution or individual will be a braid. A braid is a collection of generator matrices, also called variables, which follow a determined order. To represent each variable of a braid, quaternions are used. Similarly, there will be a quaternion symbolizing the target gate.

The quaternion structure also has basic operations to solve the braid problem. These operations include the multiplication of quaternions, the copy of a quaternion to another one and the distance between two quaternions, among others. This distance operation is the operation used to compute the error approximation, ε, between a braid matrix and the target gate.
When designing and developing the programs, I had two programming paradigms in mind: object oriented programming and functional programming. The first because in some cases, like the quaternion example 5.2, there were clear object-like data structures. The functional programming approach was followed elsewhere since there are many functions which do simple operations. Nevertheless, the "divide and conquer" idea was present throughout the development of the project, dividing big functions into many small ones.

5.2.2 Braid variator

The sequence diagram 5.3 represents the natural one-run-execution behavior of the braid variator interacting with the selector. When the monitor module is used, the diagram restarts whenever it has finished executing, until a number of runs is reached. The numbers on the sequence changes represent the state that the program is currently on. Therefore, it is the state written in the PISA_ sta file.

To build a variator representing the braid problem, we took some parts of already implemented problems such as Knapsack and made changes to fit our problem. At first, the variator had a collection of individuals which we would later use to store our own individuals which were the braids. Each braid was composed of the previously described quaternions. Also, the evaluation function had to be changed.

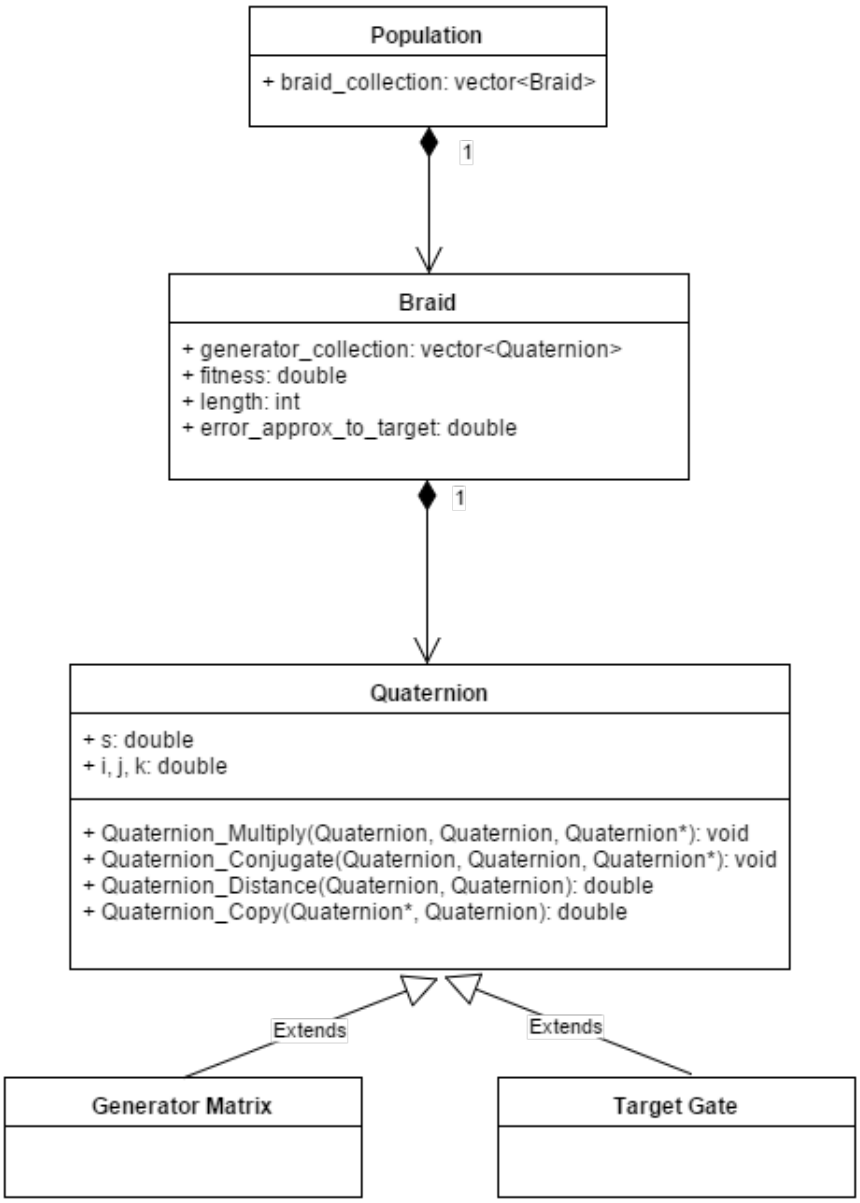


Figure 5.2: Class diagram of the concepts of population and its individuals, braids.

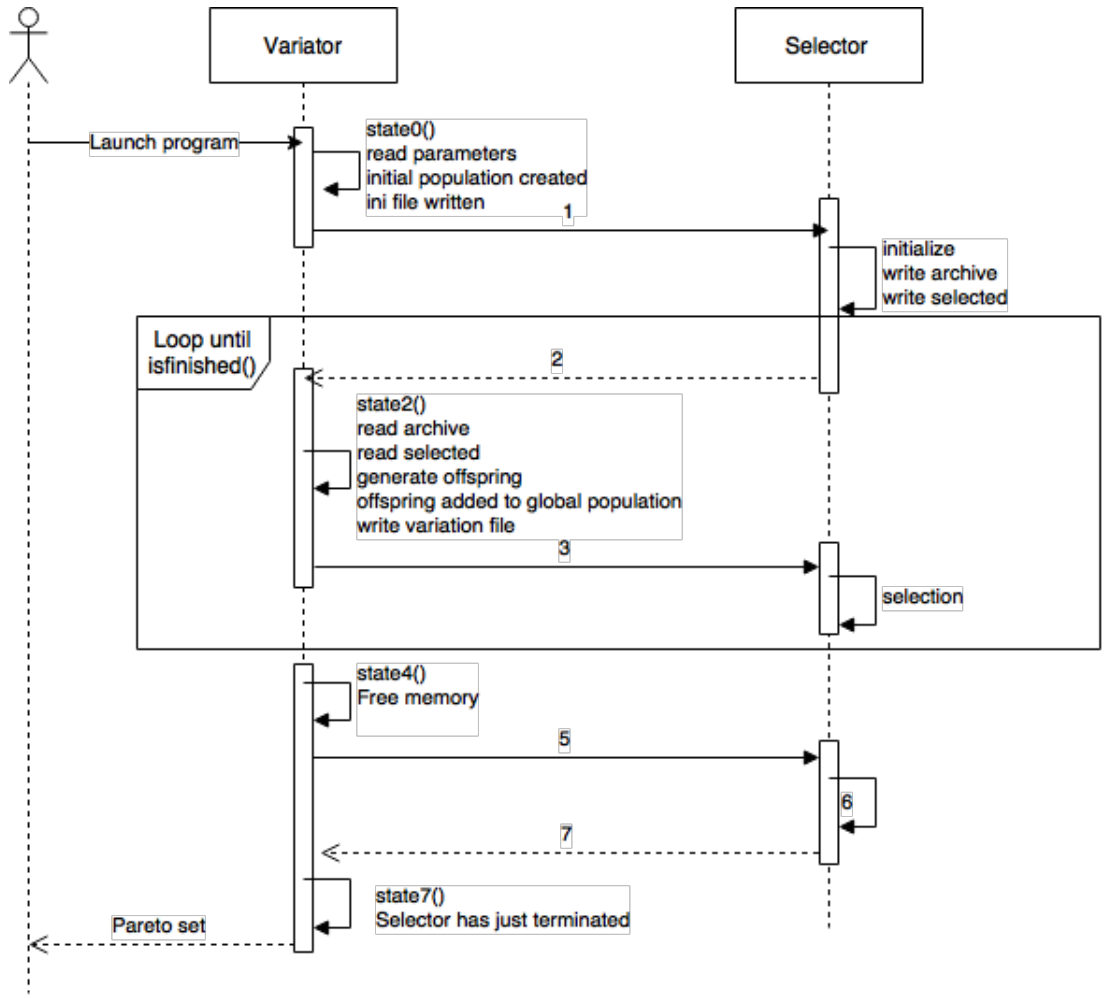


Figure 5.3: Sequence diagram of the program.

The evaluation function is the function which gives the fitness values to a population depending on how fit its individuals are. We consider our individuals to be fitter if the error approximation, ε, to a target gate is smaller and if the length of the braid is smaller. Therefore, the non-dominated solutions will be those which have the smallest error and the shortest length among other solutions. In the implemented evaluation function there is also a simplification step where the braid is simplified if possible. Later, we developed a greedy evaluation function which evaluates the solutions roughly the same way the original function does. The main difference is that this greedy evaluation tries all the possible generators in every variable of the braid until the previous generator is reached; instead of taking the last best found braid. For example, if the last best found braid was 012, this greedy evaluation will try 002, 012, 010, 011 and 012. The braid's variables will be changing until the braid can no longer be improved. It is worth mentioning, that this greedy evaluation method consumes much more time than the initial evaluation function. The variators available in the PISA webpage are written in C language, nonetheless because compiled C++ can work fine with C, C++ was used.

5.2.3 Multi-instance variator

In order to ensure that the MOEAs worked fine, different instances of the quasiparticle problem were used. We consider an instance to be a set of generators and a target gate. These instances have already been tested in their own and have their own publications. In those papers we can find information about the quality of obtained braids and the length too, even though most of the papers are not multi-objective. In this manner, we would have available published results to compare with our own results.
Because the project was going to use a single instance at first, we decided to begin with Fibonacci anyon braids [7][43]. However, due to a programming bug, this gate was not finally tested. Instead, gate 5.8 was used. Then, we included the instance from [13], but the results obtained were not enough satisfying, obtaining errors in the order of $2e-2$, instead of the approximation they obtained, $3.1e-3$, when testing the problem in similar conditions that those proposed in their paper.
Finally, we maintained the generators for the Fibonacci anyon braid instance, but changing the target gate. Therefore, we implemented two more instances:

$$T_1 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \tag{5.6}$$

$$T_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{5.7}$$

$$T_i = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} \tag{5.8}$$

5.2.4 Braid variator parameter file

Like any other variator in PISA has, our newly created variator module has a parameter file too. This file includes information on the random seed, the recombination and mutation type and their probabilities. It also contains common information with other variators like the maximum number of generations or the size of the population. A parameter for the instance selection is also present.

We have defined six different types of recombination methods, three classic recombination methods and three EDA-based recombination procedures. The EDA procedures are explained in the EDA section of the State of the Art. A recombination is the procedure by which an offspring is generated based on the variables its parents have.

- No recombination
- One point crossover: a position in the braid is chosen randomly. Until that position is reached, all the variables before that position will be swapped.

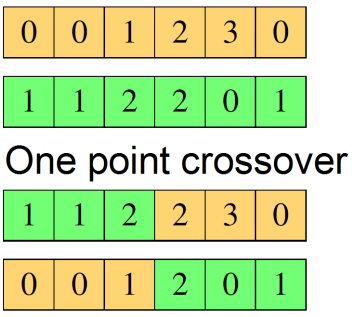


Figure 5.4: One point crossover example where the crossover point is 3.

- Uniform crossover: the variables will be recombined only if the position's variable differ in both parents. Then, the variables will swap given a recombination probability.

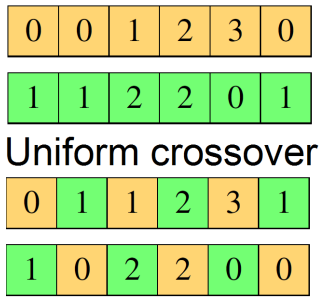


Figure 5.5: Uniform crossover with a crossover probability of 0.5.

- Univariate EDA
- Markov chain-shaped EDA
- Tree EDA

5.2.5 Data visualization

Additionally, I implemented a visualization function which plotted all the solutions each generation the MOEA progressed. In this manner, we could see how the Pareto front evolves and improves each generation. This was done using the visualization program gnuplot [73]. Each time a population is created, all the individuals of the population are drawn in a two-dimensional space as points, where the x axis represents the length of the braid and the y axis the error approximation to the target gate. Figure 5.6 is an example of what is drawn.

5.3 General purpose scripting

The outputs generated by the PISA framework are text files with the results that the MOEA has discovered. However, in order to prepare figures for this thesis and to help me see better what that data meant, I had to do some bash and Python scripting. There are two important bash scripts which do box-plot diagrams and histograms. As for Python, I did a script which generated a heat map.

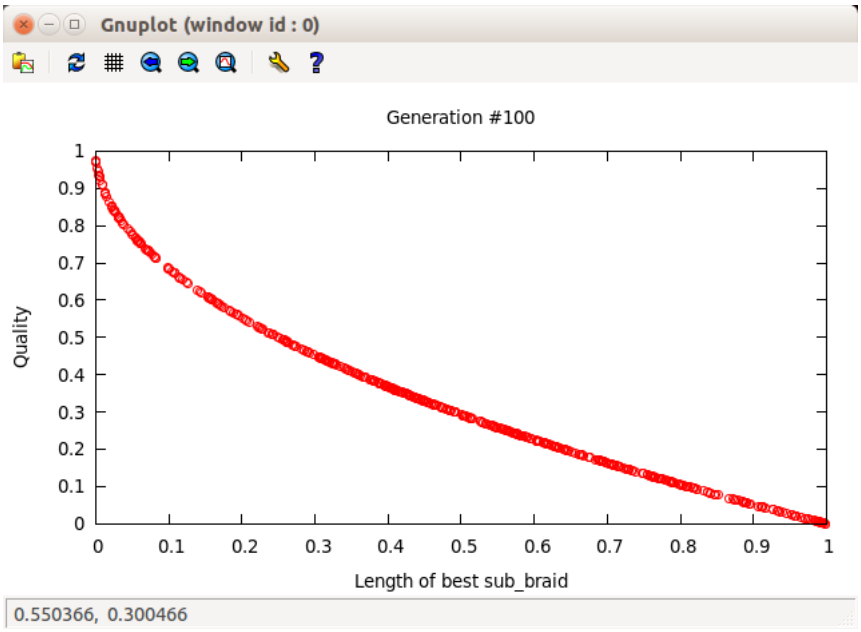


Figure 5.6: Population drawn using gnuplot.

5.3.1 Bash scripts

A bash script is a program which runs on the shell command interpreter. It will contain system calls and other programming procedures. I have used Ubuntu 14.04 LTS to develop the project and therefore the commands used in the scripts are UNIX commands. I have developed three bash scripts, two will plot the data obtained from PISA in different diagrams, and the third bash script will automatize the process of changing variables in the braid variator and collect the data from each run of the monitor module. Because two of the bash scripts are meant to plot data in diagrams, the "gnuplot" program is used. In this manner, the bash scripting will prepare data in a text file so that gnuplot can read and plot it. I decided to use bash scripts due to their easy integration to the monitor module from PISA.

The automation script will change the recombination method in the braid variator, as well as the instance which is being tested. Thus, the monitor module will be called with its respective runs for each instance and for each recombination method. Additionally, the results from each run are copied into folders in order to process it later.

One of the plotting scripts I developed plots a histogram representing how much time, in seconds, each MOEA takes to solve our braid problem for each recombination type. Histogram 5.7 represents what the program outputs.

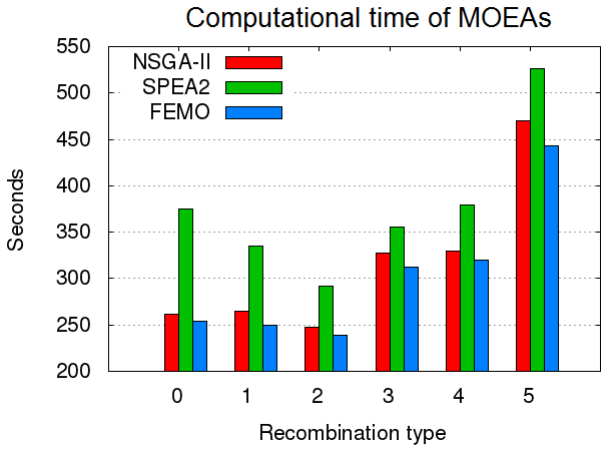


Figure 5.7: Histogram generated with bash scripting.

The other plotting bash script generates a box plot diagram using data from the hypervolume values given by PISA. The lower the diagram is, the better. Figure 5.8 is an example of the drawn box-plots.

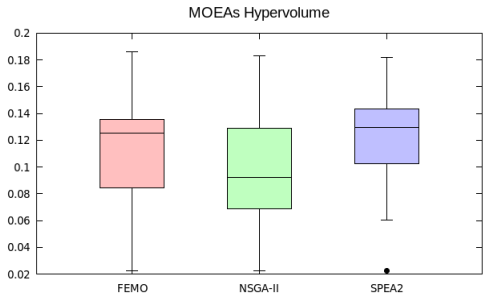


Figure 5.8: Box plot with hypervolume values generated with bash scripting.

5.3.2 Python script

Python [74] [75] is a multi-paradigm programming language used in many projects around the world for all kind of purposes. I chose to use Python for some scripts because I had already worked with it and because it is easy to use. Additionally, there are many modules available for any field of study. Because the script was meant to be used for plotting and for manipulating data, I used the "NumPy" and the "matplotlib" modules [76]. NumPy will work with matrices and matplotlib will plot the data. As mentioned earlier, the Python script generates a heat map. This heat map will help to visualize and search patterns in the

output Pareto front solutions, braids. For each braid, it will plot each variable which composes the braid with a distinctive color depending on the value the variable has, bounded between 1 and 4. Figure 5.9 is an example of what the script will plot. In the example, there are 3 arbitrary braids with length 10.

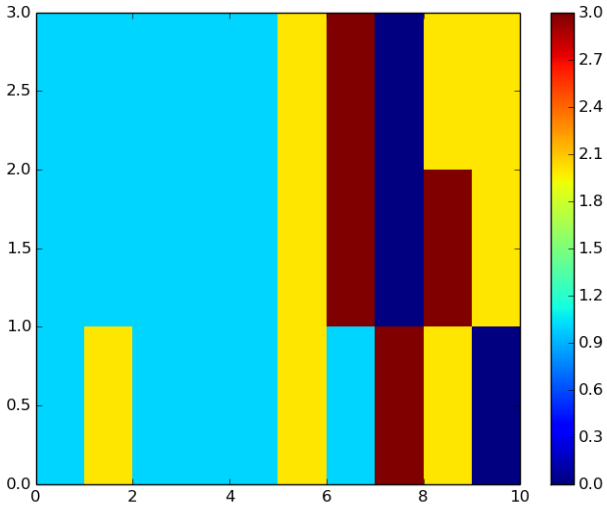


Figure 5.9: Heat map diagram plotted using Python.

5.4 Programming challenges

As a general advice, it is very important to prepare the environment you are going to be working on and choosing which tools suit best for the project. This task requires time and planning ahead in order to not lose too much time in the future. I have found challenging programming techniques as well as some problems with the programs themselves.
Even though the framework provided by PISA is quite easy to use, it takes time learning how everything works. Furthermore, because PISA works with file writing and reading, it is important to grant the appropriate permissions to the program it uses.
Even though PISA framework works correctly, the data it collects from the tests is difficult to analyze. Because it is raw data, it is sometimes hard to find a meaning to those numbers. That is why the framework needs of a careful reading of the 'README' file to fully understand what files are being created and what is their role in the execution. For a good comprehension on what PISA outputs, it is recommendable to have an adequate

understanding on metrics and statistical tests. In this manner, the data plotting scripts help a lot in understanding the obtained results.
With regard to MOEAs, they have been quite difficult for me to understand since I had not worked with them previously and having to modify or implement MOEA parts was difficult at first. However, the more papers I read about MOEAs, the better I understood the concepts that usually appear (Pareto front, recombination, generations, etc.). Gradually, it became an easy task to understand how MOEAs worked.
Nevertheless, the most difficult task was to understand the nature of the braid problem and how could it be implemented. Finding MOEA approaches to the problem was very hard; not only in the literature review for writing the thesis, but also for reusing available code. It has also been hard to adapt some paper's approaches to the programming and I had many doubts on whether I was programming them right or wrong. This is mainly due to the problem not having been solved using a MOEA approach.

5.5 Further use of the software

In case the developed software wants to be used in the future, some considerations must be taken into account.
All the software I have worked with can be divided into three parts: the applied EDA procedures, the PISA framework and the quaternion library.
The EDA procedures were developed by Roberto Santana Hermida and are free to use.
The PISA framework can be used for academic purposes without fee. Additionally, each module has its own usage policy. However, all the modules I have worked with have the same policy since they have been developed by Laumanns, M. In the following link there is available the policy applicable for each module I used http://www.tik.ee.ethz.ch/sop/pisa/variators/knapsack/knapsack_license.txt. Even though, this is the knapsack specific license, it is the same for all the other software components.
The quaternion library was developed by McDonald, R. and Katzgraber, H. The code is available upon request to the authors.
The braid variator is not available in the PISA webpage. However, it is free to use, copy, modify and distribute upon request.

In order to ease the maintenance and the future use of the braid variator, it will be delivered in a package containing the software assets, this thesis and a 'README' file with the instructions to operate the module.

CHAPTER 6

Experimentation phase

In the previous sections we have explained the different modules implemented and added to the PISA framework. This chapter presents the experiments done to validate the software and to evaluate the behavior of the introduced MOEAs for the braid problem. Firstly, some previous information and parameters are introduced and later the results are presented.

6.1 Representation of the results of the algorithm

As we said in previous chapters, the MOEAs will try to solve the problem of quasiparticle braid computation using a multi-objective approach. Therefore, the goal is to minimize both the length and the error approximation to a target gate.
In order to see the results better, a graphical representation of the results of the algorithm was implemented. In this representation, the objective (function values) of every solution in the population are plotted. This way, the Pareto front is drawn in each generation. Figure 6.1 shows an example of a final Pareto front. As seen in the generation counter, the algorithm stopped when it reached the maximum generations possible, 100 in this case.

Thanks to this real time graphical representation, we are able to see how the MOEAs improve the solution through several generations. This information can help the user to modify the parameters of the algorithm to improve the search. In figure 6.6 we can see

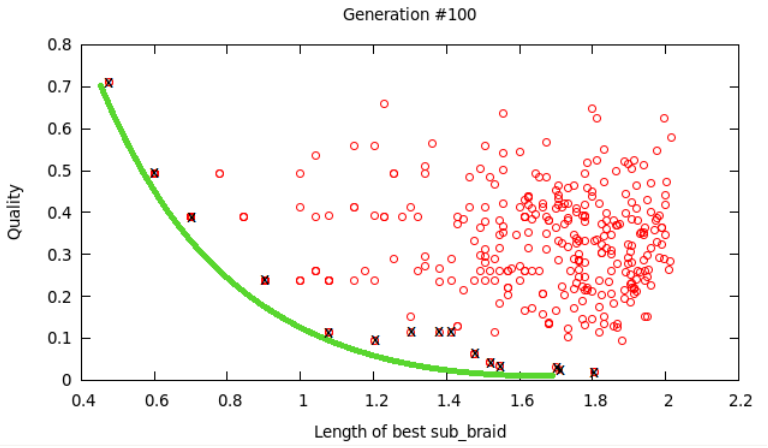


Figure 6.1: Obtained Pareto set. The green line symbolizes the imaginary Pareto front. The Pareto front is composed of the dots marked with an 'x'.

how, with the advance of the evolution the error approximation gets smaller and the Pareto front is more clearly drawn to the left of the graphs; glimpsing a curve.

6.2 Initial considerations

In order to run the experiments under similar conditions, the chosen parameters have been the same through this phase. Such parameters are displayed below, and can be taken into account in case the experiments need reproduction.

- **Runs**: the more runs the MOEAs complete, the more data we will be able to collect and compare. Also, a number between 15 and 30 was recommended by the supervisor. This number was, finally, 20 runs.
- **Generations**: each MOEA evolved until it reached 50 generations.
- **Population sizes**: the size of the population was also selected. It is big enough to ensure homogeneity in the Pareto set and small enough so the system can provide the computation power. These values can be found in the PISA configuration file (PISA_cfg), for each MOEA and for each problem representation.
  - α Alpha: it represents the size of the whole population. The value used was 200 individuals.

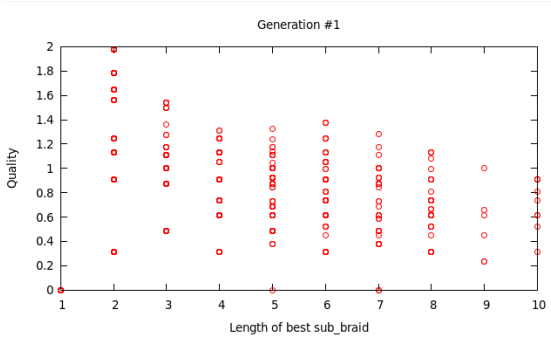


Figure 6.2: Generation 1

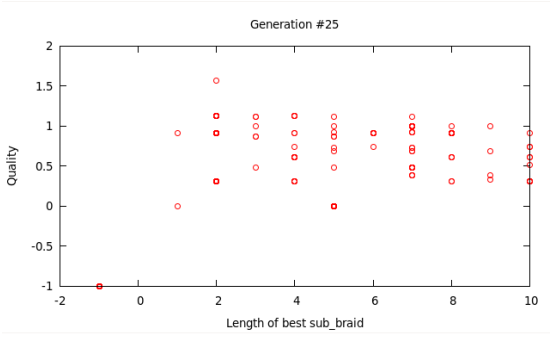


Figure 6.3: Generation 25

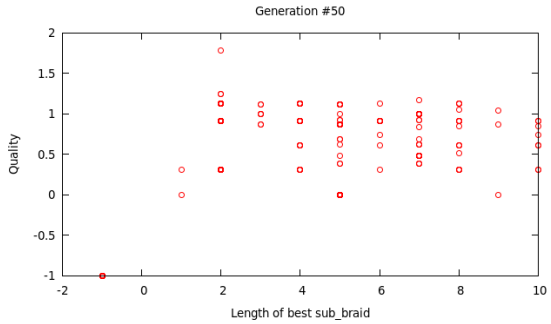


Figure 6.4: Generation 50

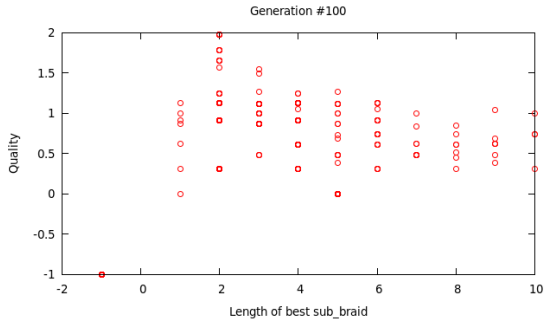


Figure 6.5: Generation 100

Figure 6.6: Pareto front evolution through several generations.

  - λ Lambda: it represents the size of the offspring. The value used was 200 individuals.
  - μ Mu: it represents the size of the parent population. The value used was 200 individuals.

- **Mutation type**: even though three types of mutation were available (no mutation, one integer mutation and independent integer mutation), the last one was used to ensure the randomness of the solutions. The integer mutation was not available in the PISA framework, therefore it had to be implemented as part of the thesis.

- **Recombination type**: the experiments were done, so that we could compare different MOEAs using different recombination types. These types are six: no recombination at all (Only mutation), one point crossover (1-point-CX), uniform crossover (u-CX), univariate EDA (UMDA), Markov chain-shaped EDA (Mk-EDA) and Tree EDA (Tree-EDA). The last four recombination methods were implemented as part of this project.

- **Generators**: the chosen generator matrices which are initially given, are the ones present in the article by R. B. McDonald and H. G. Katzgraber [7] described by equations 6.1, 6.2, 6.3 and 6.4.

$$\sigma_0 = \begin{pmatrix} e^{\frac{-i7\pi}{10}} & 0 \\ 0 & -e^{\frac{-i3\pi}{10}} \end{pmatrix} \tag{6.1}$$

$$\sigma_1 = \begin{pmatrix} -\tau e^{\frac{-i\pi}{10}} & -i\sqrt{\tau} \\ -i\sqrt{\tau} & -\tau e^{\frac{-i\pi}{10}} \end{pmatrix} \tag{6.2}$$

  where $\tau = \frac{\sqrt{5}-1}{2}$.

$$\sigma_2 = \sigma_0^{-1} \tag{6.3}$$

$$\sigma_3 = \sigma_1^{-1} \tag{6.4}$$

- **Target gate**: the target gate is the gate at which we will try to approach with the chosen generators. Each target gate defines a problem with different characteristics. Therefore, the chosen target gates are shown in equations 6.5 and 6.6. Throughout this chapter, the nomenclature for each target gate will be T_1 and T_2, accordingly.

$$T_1 = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} \tag{6.5}$$

$$T_2 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \tag{6.6}$$

As previously mentioned in chapter 5, the hypervolume would be used as the metric, and THE Mann-Whitney as the statistical test; in order to compare the performance of different MOEAs solving the same problem.

6.3 Instance 1: using T_1

This section will put to test the initial instance we had, this is, using T_1 as the target gate.

Operators	FEMO		NSGA-II		SPEA2	
	μ	σ^2	μ	σ^2	μ	σ^2
Only mutation	0.06946	0.01189	0.01253	0.00194	0.01276	0.00203
1-point-CX	0.06946	0.01189	0.01276	0.00203	0.00941	0.00095
u-CX	0.04363	0.00661	0.01379	0.00223	0.00298	$8.01960e-15$
UMDA	0.03529	0.00567	0.00588	0.00065	0.01764	0.00176
Mk-EDA	0.04117	0.00591	0	0	0.01764	0.00176
Tree-EDA	0.03646	0.00519	0	0	0	0

Table 6.1: Hypervolume values for T_1 for each recombination method and each MOEA.

6.3.1 Hypervolume values

As explained earlier in the design chapter, the hypervolume is a metric used to asses the performance of MOEAs. Because the braid problem is a minimization problem, the lower the hypervolume values are, the better the results of the algorithm are.
Table 6.1 represents the hypervolume's mean and variance for each MOEA and for each recombination type. For nomenclature purposes μ is used to represent the mean and σ^2 to represent the variance. In some cases the computed hypervolume is 0. This is because PISA first normalizes the objective values in a range $[1,2]$, and when these objectives values are very small, its normalization gives the lowest value, treating it as zero.
Figure 6.7 contains the box plot diagrams generated by a bash script according to the hypervolume data. The figures in which there is not a visible box, is due to reaching very low hypervolume values.
To calculate the mean and variance of the hypervolume, PISA computes the hypervolume of every run. In this case, the hypervolume mean values decrease when applying more advanced recombination methods like crossover or the EDAs. Also, the variance gets smaller too, meaning that the differences between the hypervolume values of the 20 runs get smaller, giving each time more even solutions.

6.3.2 Mann-Whitney statistic test

Tables 6.2 and 6.3 show how much better a MOEA with respect to another MOEA is, in terms of evaluating the Mann-Whitney statistical test. The values presented are $p-values$. For each instance, we will subject two recombination types to the statistical test. These two recombination types are uniform crossover and univariate EDA. These were the two recombination types which found the best solutions in our experiments.

The statistical analysis reveals that for uniform crossover there are not statistical differ-

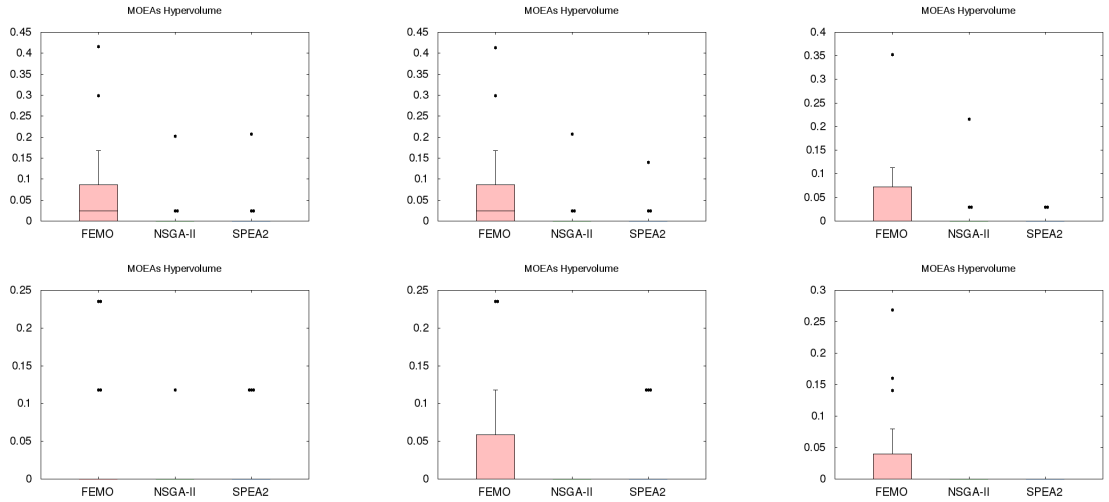


Figure 6.7: Hypervolume box plots for each recombination type for T_1. The order for each recombination type is read from left to bottom and top to bottom.

	NSGA-II	SPEA2	FEMO
NSGA-II	–	0.92789814	0.702687564
SPEA2	0.0721018603	–	0.148976531
FEMO	0.297312436	0.851023469	–

Table 6.2: Mann-Whitney test results for target gate T_1 and using uniform crossover as the recombination type.

ences ($p - value < 0.05$), while for univariate EDA there are significant statistical differences between SPEA2 and NSGA-II ($p - value < 0.05$) and between SPEA2 and FEMO ($p - value < 0.05$).

6.3.3 Error approximation and length

Table 6.4 contains the smallest error approximation with its correspondent length, found by each MOEA and for each recombination type. Notice that the data belongs to non-

	NSGA-II	SPEA2	FEMO
NSGA-II	–	0.990747257	0.814215907
SPEA2	0.00925274274	–	0.0376816537
FEMO	0.185784093	0.962318346	–

Table 6.3: Mann-Whitney test results for target gate T_1 and using univariate EDA as the recombination type.

Operators	FEMO		NSGA-II		SPEA2	
	ε	l	ε	l	ε	l
Only mutation	$1.576214e-16$	15	$1.576214e-16$	15	$1.576214e-16$	15
1-point-CX	$1.576214e-16$	15	$1.576214e-16$	15	$1.3947e-16$	7
u-CX	$1.576214e-16$	15	$1.35974e-16$	17	$1.576214e-16$	15
UMDA	$7.850462e-17$	21	$1.177569e-16$	11	$6.938894e-17$	13
Mk-EDA	$9.614813e-17$	17	$1.35974e-16$	11	$1.3947e-16$	7
Tree-EDA	$1.35974e-16$	17	$1.110223e-16$	19	$6.938894e-17$	17

Table 6.4: Minimal error approximation and length values for each recombination type for T_1.

dominated solutions.

The results present in table 6.4 show solutions of minimal length. The MOEAs also found other solutions but they were longer or had bigger error approximations. Therefore, in the table, there are only present those solutions which were found to have the smallest error approximation possible.
The errors are in the order of $e-16$, which are very small errors showing that the target gate T_1 can be approximated with a high accuracy using a relatively small number of generators.
The length is also quite small. The length values are bounded between 5 and 45, whereas for other gates analyzed in previous publications ([7][12][43], the length grows greatly in order to maintain small error approximations. In this manner, we could say that the multi-objective approach is good to find solutions for this instance, since the error approximation and the length are low.

Considering the performance of tested MOEAs to the problem, all the MOEAs seem to find quite good solutions for any recombination type. But in general, NSGA-II has found more solutions than FEMO and SPEA2. Additionally, FEMO has also found more solutions of shorter length than SPEA2.

6.3.4 Computational time

Figure 6.8 contains information about how much time each MOEAs takes in order to run the 50 generations. The time is represented in seconds. It is also represented for each MOEA, all the possible variants of recombination.

In figures 6.8 and 6.13, each number corresponds to a recombination type, being:

- 0 - No recombination
- 1 - One point crossover

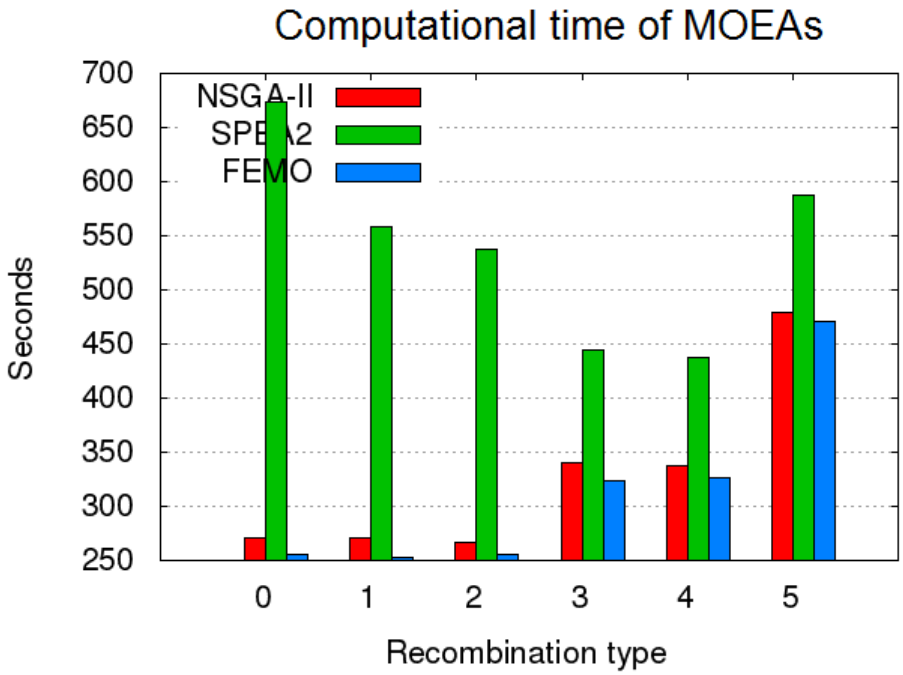


Figure 6.8: Time bar graph for T_1.

- 2 - Uniform crossover
- 3 - Univariate EDA
- 4 - Markov chain-shaped EDA
- 5 - Tree EDA

As seen on the bar graph, the fastest MOEA happens to be FEMO for all the recombination types, closely followed by NSGA-II. On the contrary, SPEA2 is the slowest MOEA for the braid problem. Additionally, the fastest recombination type is the uniform crossover and the methods that implement the probabilistic models are computationally more costly.

Thus, the simplicity of the fairness mechanisms of FEMO make it the fastest algorithm. NSGA-II is faster than SPEA2 because instead of comparing each member with its neighbors, like SPEA2 does, it makes a density computation, and according to that it sorts the solutions in a non-dominated ranking.

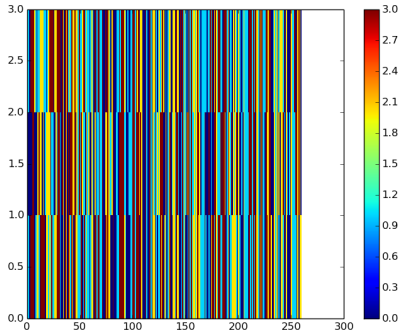


Figure 6.9: T_1 and u-CX.

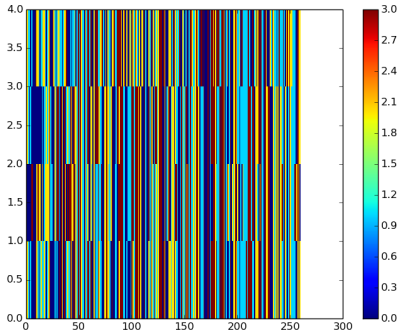


Figure 6.10: T_1 and UMDA.

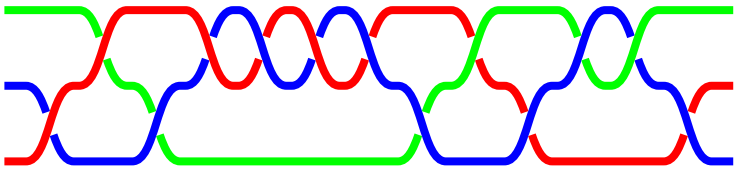


Figure 6.11: Partial diagram of a minimal braid.

6.3.5 Numerical representation

Figures 6.9 and 6.10 represent the Pareto set's numerical representation. Each number represents a generator from the Fibonacci anyon braid group defined in [7]. Because this type of braid topology uses four possible generator matrices, the numbers are in the range $[0,3]$. Accordingly, the numbers correspond to the subindex in the equations 6.1, 6.2, 6.3 and 6.4.

The heatmap diagrams shown in figures 6.9 and 6.10 will allow us to appreciate in more detail if there are any repetitive patterns in the best found braids. The horizontal axis represents the variables of the problem. It was set to 260. The vertical axis contains each braid. For instance, figure 6.9 contains 3 solutions and figure 6.10 contains 4 solutions.

It is common to see in research papers that braids are also drawn in its topological form. For that, each variable of the braid is supposed to be σ_i, and the order in which the variables are set determines the crossings the braid will have. Figure 6.11 shows an example of a drawn braid. This particular representation corresponds to a braid with an error approximation $\varepsilon = 6.938894000e-17$ and length 13, obtained for target T_1.

Operators	FEMO		NSGA-II		SPEA2	
	μ	σ^2	μ	σ^2	μ	σ^2
Only mutation	0.002	$1.6e-5$	0.002	$1.6e-5$	0.002	$1.6e-5$
1-point-CX	0.002	$1.6e-5$	0.002	$1.6e-5$	0.002	$1.6e-5$
u-CX	0.0015	$1.275e-5$	0.002	$1.6e-5$	0.002	$1.6e-5$
UMDA	0.00083	$1.31944e-5$	0	0	0	0
Mk-EDA	0.001	$9.0e-6$	0	0	0	0
Tree-EDA	0.00083	$1.31944e-5$	0.00041	$3.29861e-6$	0	0

Table 6.5: Hypervolume values for T_2 for each recombination method and each MOEA.

6.4 Instance 2: using T_2

In this section we will test the other instance of the braid problem, that is, the instance using target gate T_2 (Equation 6.6). The experimentation phase for this instance will be driven like the experimentation phase for the previous instance T_1.

6.4.1 Hypervolume values

Table 6.5 represents the hypervolume's mean and variance for each MOEA and for each recombination type in the case of the second instance using T_2. For nomenclature purposes μ is used to represent the mean and σ^2 to represent the variance.
Figure 6.12 contains the box plot diagrams generated by a bash script according to the hypervolume data gathered by PISA.

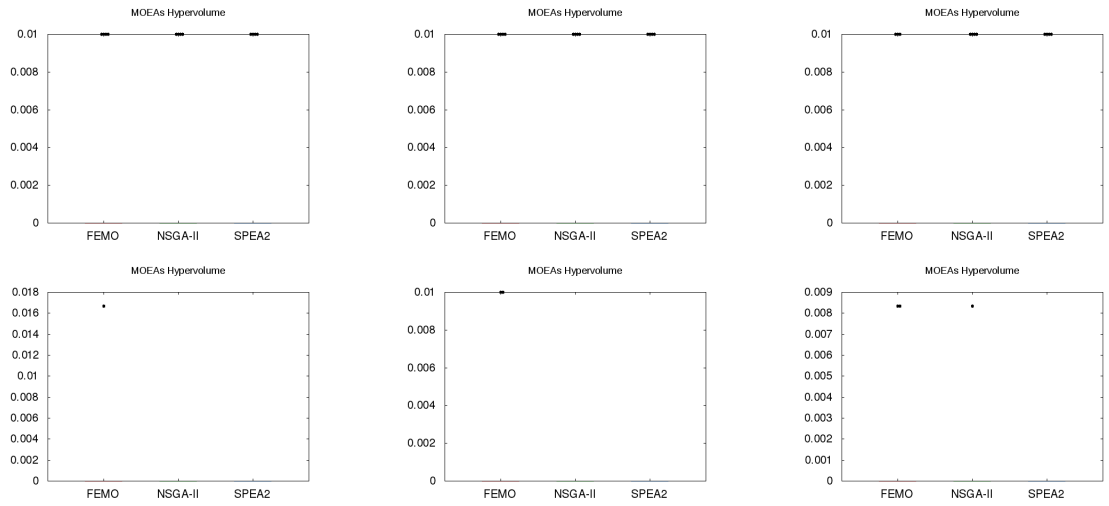


Figure 6.12: Hypervolume box plots for each recombination type for T_2. The order for each recombination type is read from left to right and top to bottom.

	NSGA-II	SPEA2	FEMO
NSGA-II	–	0.841344746	0.841344746
SPEA2	0.158655254	–	0.5
FEMO	0.158655254	0.5	–

Table 6.6: Mann-Whitney test results for target gate T_2 and using uniform crossover as the recombination type.

	NSGA-II	SPEA2	FEMO
NSGA-II	–	0.924028319	0.924028319
SPEA2	0.0759716806	–	0.5
FEMO	0.0759716806	0.5	–

Table 6.7: Mann-Whitney test results for target gate T_2 and using univariate EDA as the recombination type.

6.4.2 Mann-Whitney statistic test

Tables 6.6 and 6.7 show how much better a MOEA with respect to another MOEA is, in terms of evaluating the Mann-Whitney statistical test. The values presented are $p-values$. As it had been done for the previous instance, this statistical tests will be shown for only two recombination types. Namely, uniform crossover and univariate EDA. Also, for this instance, these were too the recombination types which found the best solutions.

The statistical analysis reveals that there are not significant statistical differences the algorithms for this instance.

6.4.3 Error approximation and length

Table 6.8 presents the minimal found solutions for each recombination type and for each MOEA we tested. Both error approximation and length of the braid are shown.

The results present in table 6.8 show solutions of minimal length. The MOEAs also found other solutions but they were longer. Therefore, in the table, there are only present those solutions which were found to have the smallest error approximation possible.
The errors are in the order of $e-17$, which are very small errors showing that the target gate T_2 can be approximated with a high accuracy using a relatively small number of generators.
The length is also quite small. The length values are bounded between 8 and 12, whereas

Operators	FEMO		NSGA-II		SPEA2	
	ε	l	ε	l	ε	l
Only mutation	$4.509251000e-17$	8	$4.509251000e-17$	8	$4.509251000e-17$	8
1-point-CX	$4.509251000e-17$	8	$4.509251000e-17$	8	$4.509251000e-17$	8
u-CX	$4.509251000e-17$	8	$4.509251000e-17$	8	$4.509251000e-17$	8
UMDA	$4.509251000e-17$	8	$4.509251000e-17$	8	$4.509251000e-17$	8
Mk-EDA	$4.509251000e-17$	8	$2.775558e-17$	12	$4.509251000e-17$	8
Tree-EDA	$4.509251000e-17$	8	$2.775558e-17$	12	$4.509251000e-17$	8

Table 6.8: Minimal error approximation and length values for each recombination type for T_2.

for other gates analyzed in previous publications([7][12][43], the length grows greatly in order to maintain small error approximations. Thus, we could say that the multi-objective approach is good to find solutions for this instance, since the error approximation and the length are low. For T_2 is seems that all the MOEAs using any type of recombination come up to a very good solution. However, NSGA-II reaches an even better solution when Markov-chain shaped EDA or Tree EDA is applied.

Considering the performance of tested MOEAs to the problem, all the MOEAs seem to find quite good solutions for any recombination type. But in general, NSGA-II has found more solutions than FEMO and SPEA2. Additionally, FEMO has also found more solutions of shorter length than SPEA2.

6.4.4 Computational time

Figure 6.13 shows information about how much time each MOEAs takes in order to run the 50 generations for instance T_2. The time is represented in seconds. It is also represented for each MOEA and all the possible variants of recombination.

As seen on the bar graph, the fastest MOEA happens to be FEMO for all the recombination types, closely followed by NSGA-II. On the contrary, SPEA2 is the slowest MOEA for the braid problem. Additionally, the fastest recombination type is the uniform crossover and the methods that implement the probabilistic models are computationally more costly.

Thus, the simplicity of the fairness mechanisms of FEMO makes it the fastest algorithm. NSGA-II is faster than SPEA2 because instead of comparing each member with its neighbors, like SPEA2 does, it makes a density computation, and according to that it sorts the solutions in a non-dominated ranking.

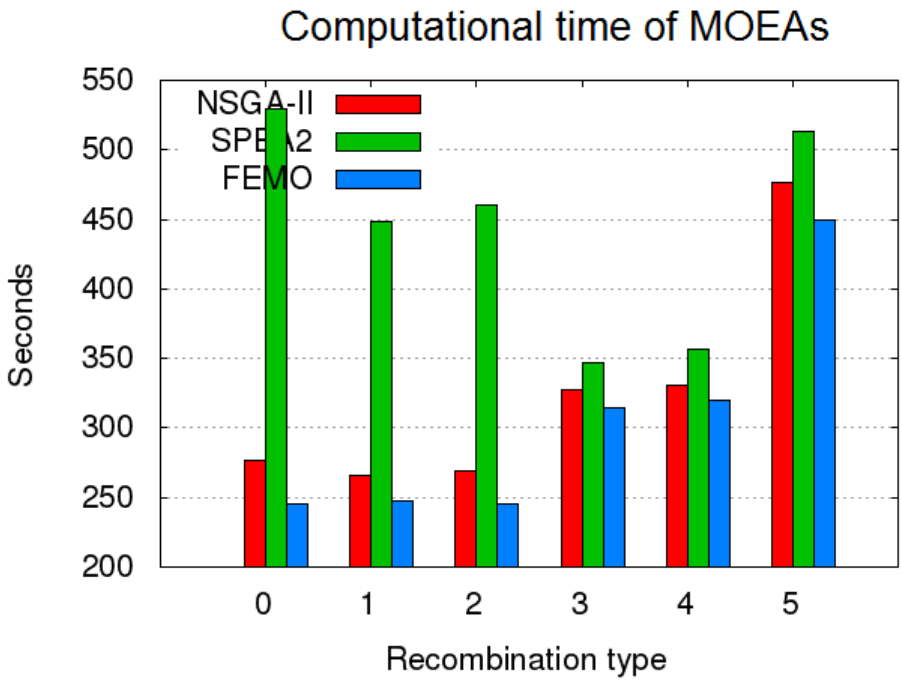


Figure 6.13: Time bar graph for T_2.

6.4.5 Numerical representation

Figures 6.14 and 6.15 represent the Pareto set's numerical representation. Each number represents a generator from the Fibonacci anyon braid group defined in [7]. Because this type of braid topology uses four possible generator matrices, the numbers are in the range $[0,3]$.

The heatmap diagrams shown in figures 6.14 and 6.15 will allow us to appreciate in more detail if there are any repetitive patterns in the best found braids. The horizontal axis represents the variables each braid had during the experiments. It was set to 260. The vertical axis contains each braid. In this case, figure 6.15 contains 2 solutions and figure 6.14 contain 4 solutions.

Figure 6.16 shows an example of a drawn braid for T_2. This particular representation corresponds to a braid with an error approximation $\varepsilon = 4.509251000e-17$ and length 8 obtained for target T_2.

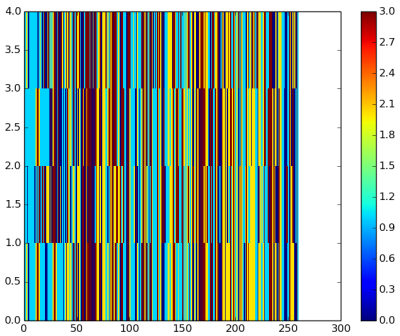


Figure 6.14: T_2 and u-CX.

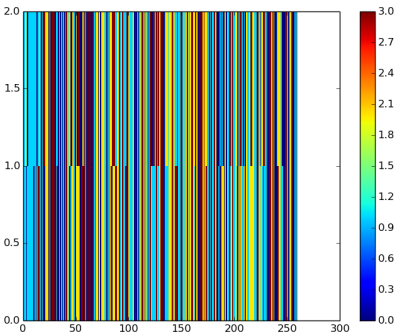


Figure 6.15: T_2 and UMDA.

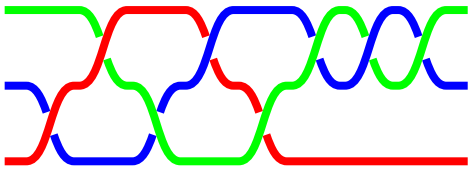


Figure 6.16: Partial diagram of a minimal braid.

CHAPTER 7

Conclusions

7.1 Final results

In this project we have analyzed the PISA software from a reverse-engineering perspective that allowed us to incorporate new routines for the solution of the braid problem. We have implemented new optimization operators, developed new evaluation functions for the braid problem and linked all the developed software and validated it, using real instances of the braid problem and applying statistical and performance tests.
The software we have developed has served to study how multi-objective algorithms behaved in the resolution of the braid problem. To do that, existing MOEAs have been combined with new implemented techniques such as EDA procedures. In particular NSGA-II, SPEA2 and FEMO are the MOEAs used in the project. Additionally, 6 recombination methods are tested, of which 3 are the mentioned EDA procedures: univariate EDA, Markov-chain shaped EDA, and Tree-EDA. In pursuance of combining all these concepts, the PISA framework was used. We defined an experimental benchmark to evaluate the algorithms and ran exhaustive experiments on them. During the experiments we did not identify any bugs in the code implemented. The best solutions for the tested instances were found using the univariate EDA and a uniform crossover recombination operator. NSGA-II and FEMO seemed to find the best solutions in a small period of time, whereas SPEA2 consumed much more time than the others and did not find greatly improved solutions compared to NSGA-II and FEMO. Unfortunately, the comparison to other published results has not been successful, since we did not find results as good as those obtained for

the tested instances.

7.1.1 Strengths of the project

This project is the first to evaluate instances of the braid problem using a multi-objective approach. There have been other attempts to solve the problem, but not treating the objectives of the problem simultaneously.
A new module with easy integration into the PISA framework, with instances of the braid problem has been developed. In addition, the automation module of PISA has been customized to automatize even more the process of testing. Also, the data treatment of PISA counts now with an automatic diagram plotting for easy data analysis.
The project is designed to work with EDAs and traditional evolutionary operators. In this manner, we could evaluate which strategies work best for the instance being tested.

7.2 Further studies

Even though I have tried to achieve the best possible results for the problem, there are a few lines that could be followed in order to give this project continuity.

- It would be an important achievement to develop a MOEA specifically designed for finding good solutions for every instance of the braid problem. However, this goal might prove to be difficult since each instance has its own characteristics.
- Improving the statistical models EDAs used would lead to a faster and better solution discovery. This project has proven that for the tested instances, the solutions found using EDA procedures are good, but greatly increase the computational time.
- In this project, only braid topology has been used, but maybe other topologies, such as knots, achieve better solutions.
- The application of hybrid methods are proposed in the state of the art.
- Pattern recognition techniques in the variables of a braid could contribute to identify good solutions.
- The use of more recent MOEAs would be an interesting line of work. Even the comparison of updated MOEAs to their earlier designs would represent a valuable contribution.

- The software implementations could be better written. This way, it would be easier to understand what the algorithms are doing, or even to optimize the code.

7.3 Personal conclusions

Throughout the duration of the project, the complexity of the tasks to solve has been very high. Not only due to the complex nature of the problem, but also because there is not abundance of information about it yet.
Conceptually, the braid problem is quite hard to understand since it involves concepts taken from the advanced fields of physics and mathematics, such as quantum mechanics and topology.
From a software engineering point of view, it is also very complex to implement the problem in a sufficiently good model. In this project, MOEAs have been used. I had never worked with this kind of algorithms before, hence it was hard for me to understand the way they performed. Once I managed to understand the basic MOEA procedures, more specific techniques of population-based algorithms were presented to me. Regarding EDAs, the basic functioning was easy to understand, but the different types of EDAs and different probabilistic models make them more difficult to understand.
Because the amount of techniques used for this project and because the nature of the problem is hard to understand, it was required to do an exhaustive state of the art phase.
Even though PISA is a flexible platform, it is not so well documented. This is the main reason why understanding PISA is so much time consuming. It is hard to understand how all the modules interact. The files it outputs often lack explanation. Additionally, the software assets are not regularly updated and some researchers have left the project, stopping its support.
The project has many tools which are not all implemented in the same programming language. Python, C, C++ and Bash scripting was used. I had previous experience with most of these programming languages, but not so much in applying them in this type of problems. The documentation was done using LaTeXwhich I had never worked with before.
The programs developed were difficult to debug since they implement stochastic algorithms whose output change in every run. In this manner, doing black-box predictions is very hard.

Appendixes

Glossary

anyon two-dimensional space quasiparticle that can take on any phase. When two anyons exchange, the joint wave function is multiplied by a factor between 1 and -1 [5]. 9, 24, 75

braid construct obtained when anyons move in circular motion. It also represents how anyons are swapped in adjacent pairs [5]. 9, 75

braid hashing a function that maps a unitary matrix (qubit gate) to a number of Fibonacci anyon braids [13]. 12, 75

crowding distance a function that maps a unitary matrix (qubit gate) to a number of Fibonacci anyon braids [13]. 27, 31, 75

generator matrix mathematical representation of a braid operation. 9, 75

gnuplot Open source software capable of plotting different functions.. 75

Majorana fermion another possible instance for the braid problem. This instance is composed of two qubit gates and operates with 10 different matrices. 12, 75

MOP initials which stand for "Multi-objective Optimization Problem". 17, 34, 35, 40, 75

mutation change in the internal structure of a being due to the modification of the inherited fenotype. In the topological quantum computing context, the change of the numerical representation of a gate. 75

non-abelian category of a mathematical group which is noncommutative. It also represents the importance of the order in which particles are swapped. 9, 75

null hypothesis also represented as H_0, is a hypothesis that must be considered true unless sufficient opposite evidence is found. If this happens, the alternative hypothesis, H_a, is accepted. 47, 75

p-value also called critical value, it is the minimum value for which we can refuse the null hypothesis plural. 46, 75

quantum decoherence this happens to a system under certain conditions when it losses its quantum properties and starts behaving like a classic system. It can be temporarily solved with quantum error correction. Low temperatures can protect the system too [77]. 8, 75

quantum gate product of generator matrices that encode single braid operations [43]. 7, 75

quasiparticle it is generally the name given in physics to particles that are related to fermions. In our study anyons are considered quasiparticles. 9, 75

qubit minimum unit of information for quantum systems. 7, 75

recombination process which produces a new association of the fenotype of a descendant individual, from parents which had it separately. 75

selector PISA module responsible for the selection algorithms which are applied to the variators. 42–44, 75

tunnel effect quantum property in which a quantum particle can sometimes get through an obstacle instead of bouncing because of its small size. This is the reason why a component could malfunction, if a particle gets where it should not. 8, 75

variator PISA module responsible for the optimization problem's specific parts (problem representation, evaluation of solutions, etc.). 42–44, 75

Bibliography

[1] N. D. Mermin, *Quantum Computer Science: An Introduction*. Cambridge University Press, 2007.

[2] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[3] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 400, pp. 97–117, The Royal Society, 1985.

[4] A. Steane, "Quantum computing," *Reports on Progress in Physics*, vol. 61, no. 2, p. 117, 1998.

[5] J. S. Demski, S. A. FitzGerald, Y. Ijiri, Y. Ijiri, and H. Lin, "Quantum information and accounting information: Exploring conceptual applications of topology," *Journal of Accounting and Public Policy*, vol. 28, no. 2, pp. 133–147, 2009.

[6] M. M. Stadler, "¿ Qué es la topología?," *Sigma: revista de matemáticas= matematika aldizkaria*, no. 20, pp. 63–77, 2002.

[7] R. B. McDonald and H. G. Katzgraber, "Genetic braid optimization: A heuristic approach to compute quasiparticle braids," *Physical Review B*, vol. 87, no. 5, p. 054414, 2013.

[8] J. Preskill, "Lecture Notes for Physics 219: Quantum computation," 2004.

[9] E. Artin, "Theory of braids," *Annals of Mathematics*, pp. 101–126, 1947.

[10] E. Dalvit *et al.*, "Braids: A mathematics documentary," in *Proceedings of Bridges 2013: Mathematics, Music, Art, Architecture, Culture*, pp. 415–418, Tessellations Publishing, 2013.

[11] N. E. Bonesteel, L. Hormozi, G. Zikos, and S. H. Simon, "Braid topologies for quantum computation," *Physical review letters*, vol. 95, no. 14, p. 140503, 2005.

[12] H. Xu and X. Wan, "Constructing functional braids for low-leakage topological quantum computing," *Physical Review A*, vol. 78, no. 4, p. 042325, 2008.

[13] M. Burrello, H. Xu, G. Mussardo, and X. Wan, "Topological quantum hashing with the icosahedral group," *Physical review letters*, vol. 104, no. 16, p. 160502, 2010.

[14] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*, vol. 1. CRC Press, 2000.

[15] K. Deb, "Multi-objective optimisation using evolutionary algorithms: an introduction," in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pp. 3–34, Springer, 2011.

[16] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," 2001.

[17] C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary algorithms for solving multi-objective problems*, vol. 242. Springer, 2002.

[18] Breu, Guggenbichler, and Wollmann, "Evolutionary Multiobjective Optimization," *Vasa*, vol. 37, pp. 3–29, 2008. PMID: 18690602.

[19] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.

[20] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb, *Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem*. Springer, 2002.

[21] M. B. Anderson, "The potential of genetic algorithms for subsonic wing design," in *AIAA, Aircraft Engineering, Technology, and Operations Congress, 1st, Los Angeles, CA*, 1995.

[22] A. C. Briza and P. C. Naval, "Stock trading system based on the multi-objective particle swarm optimization of technical indicators on end-of-day market data," *Applied Soft Computing*, vol. 11, no. 1, pp. 1191–1201, 2011.

[23] R. Sarker and T. Ray, "An improved evolutionary algorithm for solving multi-objective crop planning models," *Computers and Electronics in Agriculture*, vol. 68, no. 2, pp. 191–199, 2009.

[24] P. K. Shukla, K. Deb, and S. Tiwari, "Comparing classical generating methods with an evolutionary multi-objective optimization method," *Lecture Notes in Computer Science*, vol. 3410, pp. 311–325, 2005.

[25] K. Deb and S. Chaudhuri, "I-EMO: An interactive evolutionary multi-objective optimization tool," *Pattern Recognition and Machine Intelligence*, no. Dm, pp. 690–695, 2005.

[26] D. A. Van Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algorithm performance," *Proceedings of the Congress on Evolutionary Computation, 2000.*, vol. 1, pp. 204–211 vol.1, 2000.

[27] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, 2003.

[28] S. Whiteson and D. M. Roijers, "Multi-objective decision making," *Journal of Artificial Intelligence Research*, vol. 48, no. 67113, 2013.

[29] X. Blasco, J. Herrero, J. Sanchis, and M. Martínez, "A new graphical visualization of n-dimensional Pareto front for decision-making in multiobjective optimization," *Information Sciences*, vol. 178, no. 20, pp. 3908–3924, 2008.

[30] R. O. Parreiras, J. H. Maciel, J. Vasconcelos, *et al.*, "The a posteriori decision in multiobjective optimization problems with smarts, Promethee II, and a fuzzy algorithm," *Magnetics, IEEE Transactions on*, vol. 42, no. 4, pp. 1139–1142, 2006.

[31] K. Deb, A. Sinha, and S. Kukkonen, "Multi-objective test problems, linkages, and evolutionary methodologies," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 1141–1148, ACM, 2006.

[32] J. B. Kollat and P. Reed, "A framework for visually interactive decision-making and design using evolutionary multi-objective optimization (video)," *Environmental Modelling & Software*, vol. 22, no. 12, pp. 1691–1704, 2007.

[33] P. Larrañaga and J. A. Lozano, *Estimation of distribution algorithms: A new tool for evolutionary computation*, vol. 2. Springer Science & Business Media, 2002.

[34] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011.

[35] M. Pelikan, M. W. Hauschild, and F. G. Lobo, "Introduction to estimation of distribution algorithms," *MEDAL Report*, no. 2012003, 2012.

[36] S. Shakya and R. Santana, "A review of estimation of distribution algorithms and Markov networks," in *Markov Networks in Evolutionary Computation*, pp. 21–37, Springer, 2012.

[37] K. Sastry, D. E. Goldberg, and X. Llora, "Towards billion-bit optimization via a parallel estimation of distribution algorithm," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 577–584, ACM, 2007.

[38] R. Santana, P. Larrañaga, and J. A. Lozano, "Adaptive estimation of distribution algorithms," in *Adaptive and Multilevel Metaheuristics*, pp. 177–197, Springer, 2008.

[39] T.-L. Yu, S. Santarelli, and D. E. Goldberg, "Military antenna design using a simple genetic algorithm and hBOA," in *Scalable Optimization via Probabilistic Modeling*, pp. 275–289, Springer, 2006.

[40] R. Armañanzas, I. Inza, R. Santana, Y. Saeys, J. L. Flores, J. A. Lozano, Y. Van de Peer, R. Blanco, V. Robles, C. Bielza, and P. Larrañaga, "A review of estimation of distribution algorithms in bioinformatics.," *BioData mining*, vol. 1, no. 1, p. 6, 2008.

[41] J. Bacardit, M. Stout, J. D. Hirst, K. Sastry, X. Llorà, and N. Krasnogor, "Automated alphabet reduction method with evolutionary algorithms for protein structure prediction," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 346–353, ACM, 2007.

[42] A. Petrovski, S. Shakya, and J. McCall, "Optimising cancer chemotherapy using an estimation of distribution algorithm and genetic algorithms," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 413–418, ACM, 2006.

[43] R. Santana, R. B. McDonald, and H. G. Katzgraber, "A probabilistic evolutionary optimization approach to compute quasiparticle braids," in *Simulated Evolution and Learning*, pp. 13–24, Springer, 2014.

[44] J. Knowles and D. Corne, "The pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1, IEEE, 1999.

[45] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[46] Z. Wang, C. Tianshi, K. Tang, and X. Yao, "A multi-objective approach to redundancy allocation problem in parallel-series systems," *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, pp. 582–589, 2009.

[47] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *Evolutionary Computation, IEEE transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

[48] L. Thu Bui, *Multi-Objective Optimization in Computational Intelligence: Theory and Practice: Theory and Practice*. IGI Global, 2008.

[49] D. Kunkle, "A summary and comparison of MOEA algorithms," *Northeastern University in Boston, Massachusetts*, vol. 2, no. 2003, pp. 1–19, 2005.

[50] M. Delgado, M. P. Cuéllar, and M. C. Pegalajar, "Multiobjective hybrid optimization and training of recurrent neural networks," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 38, no. 2, pp. 381–403, 2008.

[51] G. N. Demir, A. Ş. Uyar, and Ş. Gündüz-Öğüdücü, "Multiobjective evolutionary clustering of web user sessions: a case study in web page recommendation," *Soft Computing*, vol. 14, no. 6, pp. 579–597, 2010.

[52] M. Kim, T. Hiroyasu, M. Miki, and S. Watanabe, "SPEA2+: improving the performance of the strength pareto evolutionary algorithm 2," in *Parallel problem solving from nature-PPSN VIII*, pp. 742–751, Springer, 2004.

[53] M. Hongyun and L. Sanyang, "ISPEA: improvement for the strength Pareto evolutionary algorithm for multiobjective optimization with immunity," in *Computational Intelligence and Multimedia Applications, 2003. ICCIMA 2003. Proceedings. Fifth International Conference on*, pp. 368–372, IEEE, 2003.

[54] T. Friedrich, C. Horoba, and F. Neumann, *Runtime analyses for using fairness in evolutionary multi-objective optimization*. Springer, 2008.

[55] T. Jansen and I. Wegener, "Evolutionary algorithms-how to cope with plateaus of constant fitness and when to reject strings of the same fitness," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 6, pp. 589–599, 2001.

[56] O. Giel, "Expected runtimes of a simple multi-objective evolutionary algorithm," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 3, pp. 1918–1925, IEEE, 2003.

[57] M. Laumanns, L. Thiele, and E. Zitzler, "Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 2, pp. 170–182, 2004.

[58] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA—a platform and programming language independent interface for search algorithms," in *Evolutionary multi-criterion optimization*, pp. 494–508, Springer, 2003.

[59] L. Thiele, "Using the monitor in PISA," *Computer Engineering and Networks Lab (TIK), ETH Zurich*.

[60] C. M. Fonseca, J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," in *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, vol. 216, p. 240, 2005.

[61] E. Zitzler, J. Knowles, and L. Thiele, "Quality assessment of Pareto set approximations," in *Multiobjective Optimization*, pp. 373–404, Springer, 2008.

[62] M. P. Hansen and A. Jaszkiewicz, *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical University of Denmark, 1998.

[63] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration," in *Evolutionary multi-criterion optimization*, pp. 862–876, Springer, 2007.

[64] S. Jiang, Y.-S. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *Cybernetics, IEEE Transactions on*, vol. 44, no. 12, pp. 2391–2404, 2014.

[65] E. J. Hughes, "Evolutionary many-objective optimisation: many once or one many?," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 1, pp. 222–227, IEEE, 2005.

[66] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 1, pp. 29–38, 2006.

[67] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[68] D. Brockhoff and E. Zitzler, "Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 2086–2093, IEEE, 2007.

[69] A. Popov, A. Farag, and H. Werner, "Tuning of a PID controller using a multiobjective optimization technique applied to a neutralization plant," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pp. 7139–7143, IEEE, 2005.

[70] A. C. Olivera, M. Frutos, J. A. Carballido, I. Ponzoni, and N. B. Brignole, "Bus Network Scheduling Problem: GRASP+ EAs with PISA* Simulation," in *Bio-Inspired Systems: Computational and Ambient Intelligence*, pp. 1272–1279, Springer, 2009.

[71] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[72] W. R. Hamilton, "LXXVIII. on quaternions; or on a new system of imaginaries in algebra: To the editors of the philosophical magazine and journal," 1844.

[73] P. K. Janert, *Gnuplot in action*. Manning, 2010.

[74] J. R. Briggs, *Python for kids: A playful introduction to programming*. no starch press, 2013.

[75] M. Lutz, *Python pocket reference*. " O'Reilly Media, Inc.", 2014.

[76] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science and engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[77] M. Freedman, A. Kitaev, M. Larsen, and Z. Wang, "Topological quantum computation," *Bulletin of the American Mathematical Society*, vol. 40, no. 1, pp. 31–38, 2003.