eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

ZIENTZIA ETA TEKNOLOGIA FAKULTATEA
FACULTAD DE CIENCIA Y TECNOLOGÍA

# Computational intelligence techniques for maximum energy efficiency of cogeneration processes based on internal combustion engines

DISSERTATION

to obtain the degree of doctor at the University of the

Basque Country by

**Sandra Seijo Fernández**

Thesis Advisors

**Prof. Inés Juliana del Campo Hagelstrom**

**Prof. Francisco Javier Echanove Arias**

Department of Electricity and Electronics

Elektrizitatea eta Elektronika Saila

Leioa, 2017

# Acknowledgments

Este trabajo ha sido posible primeramente gracias al apoyo económico otorgado por la Universidad del País Vasco mediante la concesión de la beca Zabalduz para realizar esta tesis doctoral y de la empresa Optimitive en la que he realizado parte del trabajo de esta tesis.

Muchísimas gracias de corazón a mis directores, Inés y Javi, habéis sido un gran apoyo en todo este proceso, siempre implicados, trabajadores, atentos, disponibles, amables,...sois muy buenos docentes, grandes investigadores, y mejores personas. Es innegable que esto no hubiese sido posible sin vosotros. Incluso desde que empecé mi actual trabajo hace más de un año y sacar tiempo para terminar de escribir la tesis ha sido cada vez más complicado, vosotros habéis estado siempre "al pie del cañón".

Un abrazo muy especial a la gente de Optimitive que me ha ayudado y que me habéis enseñado tantísimo: Imanol (gracias por tu Matlab support ), Ramón, Álvaro, Mikel, Enrique, Borja, Leire... y todos los demás con los que he coincidido y he pasado tan buenos ratos. Os deseo la mejor de las suertes en vuestras andaduras profesionales.

Aprovecho también agradecer a los compañeros del Grupo de Investigación de Diseño en Electrónica Digital: Mariví, Martínez, José Tarela, Estibaliz Asua, Raúl Finker,... por resolver dudas y dar apoyo en este apasionante periodo de mi vida.

Por último quiero también saludar a toda la gente del Departamento de Electricidad y Electrónica con la que he coincidido, aunque iba menos que el resto por el departamento las visitas siempre eran agradables y siempre había alguien con quien tomar un café, compartir nervios, dudas, o simplemente hablar.

# Contents

# List of Figures

# List of Tables

# Abstract

Industrial processes, such as those in cogeneration plants, involve a large number of pieces of equipment, machinery, and instruments. Consequently, there are a huge number of variables involved in them. In addition, they imply very complex non-linear dynamics. Data from these process variables is often captured and stored, resulting in a large quantity of data that contains hidden information about the process behaviour.

The aim of this thesis, is to model and optimise a complex real-life combined heat and power plant (cogeneration) and a slurry drying process by applying computational intelligence techniques and using real data from the process. The objective of cogeneration processes is to extract the maximum possible energy contained in the fuel by using flue gases produced by engines, gas turbines, or other machines to generate more electricity, or for use in other processes. This implies economic and environmental advantages. Combined heat and power processes are a true example of complex industrial processes. The modelling and dynamic optimisation of complex industrial processes is a hard task due to many factors, including the degree of complexity, uncertainties, high dimensionality, non-linearity and time delays.

Computational intelligence techniques have proved very successful in dealing with cogeneration processes, for both modelling and optimisation purposes. Thanks to this powerful ability to solve and understand numerous complex problems, in recent decades, computational intelligence techniques have attracted growing interest from scientific research communities. The main computational intelligence techniques involved are: artificial neural networks, fuzzy systems, evolutionary systems and hybrid approaches, i.e., combinations of the previously mentioned approaches.

In this thesis, artificial neural networks and neuro-fuzzy systems have been used to obtain accurate predictive models for the different systems in the cogeneration process. A real combined heat and power plant has been modelled using a new training algorithm for neural networks: extreme learning machine. As well as this, a new exhaustive hybrid feature selection method has been applied to the plant with the aim of checking the initial variable selection. The models obtained for the plant systems have been used used to optimise the combined heat and power plant and the slurry drying process using a multi-objective function: effective electrical efficiency. The optimisation process has been carried out using two different

approaches: a single-objective optimisation based on the gradient descent method, and a multi-objective optimisation using a genetic algorithm. The optimisation approaches provide encouraging results with an average increase of 3.05% energy efficiency with the gradient descent method, and 4.16% using the non-dominated sorting genetic algorithm II.

# Chapter 1: Introduction

According to the European Association for the Promotion of Cogeneration (COGEN) [3], "cogeneration (combined heat and power) or CHP, is the simultaneous production of electricity and heat, both of which are used". On average, conventional power plants, operate at an efficiency of just 35%. This means that up to 65% of the potential energy is released as waste heat. More recently, CHP plants have been designed which can increase this efficiency up to 80% [4]. The objective of cogeneration is to use as much of the potential energy contained within a fuel as possible. In CHP, the flue gases produced by engines, gas turbines, or other machines are used to generate more electricity or for other process. This reduces costs, because the total amount of fuel needed to produce both electricity and heat in a cogeneration plant is less than the total fuel needed to produce the same amount of electricity and heat in separate technologies (e.g., an electric utility generating plant and an industrial boiler). These fuel savings also result in less pollution. In light of these economic and environmental factors cogeneration plants currently play an important role in some EU countries [3].

Optimising the behaviour of CHP processes is a hard task for many reasons, including complexity, uncertainties, high dimensionality, non-linearity and time delays. Mathematical models with a large number of assumptions are necessary [5, 6, 7] to simulate these processes, they are often either practically impossible to model or takes too much computational time and resources. Since such physical models usually use iterative methods for final solutions they take a long time to yield results, thus they are impractical for 'real-time' applications.

Computational intelligence (CI) techniques are a set of methodologies, inspired by nature, that can deal with problems which are very hard to solve with standard methods. The most commonly used CI techniques are: artificial neural networks (ANNs), fuzzy inference systems (FISs), hybrid approaches such as neuro-fuzzy systems (NFs), and evolutionary systems (ESs). ANNs, are highly interconnected structures, inspired by the human brain, capable of learning from input-output samples. They are widely used in industry because they are data driven, adaptive, respond quickly and are highly accurate if trained with the correct data. FISs, based on fuzzy set theory, provide a framework to represent imprecise information and to reason using this kind of information. Neuro-fuzzy systems comprise a synergistic combination of ANNs and FISs. ANNs enhance the performance of FISs with their

capacity for learning from input-output samples; this learning is used to adapt FIS parameters such as a membership function or rules. ESs are based on a heuristic search that mimics the process of natural selection. CI methods are widely used for modelling different industrial processes, for example in water treatment [8], steel production [9] and paper industries [10], or industrial boilers [11, 12], amongst other uses. They are also very useful to detect and predict faults in industrial processes [13, 14, 15] and engines [16, 17, 18, 19, 20].

With respect to cogeneration plants, neural/fuzzy techniques are used in many applications, for example, the design of adaptive load-shedding models [21, 22, 23] or temperature controllers [24] [25]. Fuzzy systems are used in [26] to identify failures in a boiler during the cogeneration process and GAs are used to alleviate these failures. ANNs are used to predict the baseline energy consumption in CHP plants because they are particularly robust to uncertainties that affect the measured values of input parameters [27]. Furthermore, ANNs have proven to be a useful tool when using simple models with relatively few variables to predict the power generated in a cogeneration plant [28, 29]. In [30] the power output was predicted based on ANN by studying the relationships between the electricity produced in a CHP plant and the properties of the fuel. Another study [31] predicted power by dividing the process into two submodules, each with its own ANN model. Although NF techniques are used to model cogeneration plant far less frequently than ANNs, some studies have employed them to obtain a predictive model for two pieces of equipment in a CHP plant with accurate results [32]. NF algorithms have been used by [33] to model some parameters of a heat recovery steam generator in a cogeneration and cooling plant. All the studies mentioned above model cogeneration systems with accurate prediction results, although only relatively few variables are employed.

Although BP-based ANNs have been successfully applied to solve numerous problems in cogeneration processes, as much for classification [34] as for regression applications [35], they present some drawbacks that make them unsuitable for an increasing number of cutting-edge applications. It is well known that the design of BP based ANNs is a time-consuming task that depends on the skills of the designer to obtain effective solutions. The designer has to select the most suitable network parameters, optimise the parameters to avoid overfitting, and be aware of local minima. As a consequence, applications requiring autonomy (i.e. no human intervention) are difficult to manage using this approach.

Extreme learning machine (ELM) is a novel learning algorithm for training single hidden-layer feed-forward neural networks (SLFNs) [36, 37]. It randomly chooses the input weights of the hidden-layer neurons and analytically determines the output weights through simple matrix computations, therefore featuring a much faster learning algorithm than most popular learning methods such as back-propagation [38]. ELM is based on a simple tuning-free algorithm and parameter selection is not required. Besides, learning with ELM does not present local minima or overfitting problems. All these characteristics makes ELM very useful dealing with real-life applications where autonomous control systems and fast adaptation are necessary [39].

A proper example of this kind of applications are complex cogeneration processes where the efficiency of the process is to be modelled and optimised in real-time [40].

Some researchers not only deal with CHP modelling but also with the optimisation of cogeneration processes. The process is modelled in [41] using ANNs, then the plant is statically optimised using dynamic neural networks (one optimum value per variable) in order to minimize the fuel mass flow. After ANN and thermodynamic modelling, [42] developed a static multi-objective optimisation using GAs. In [43] a mathematical model of the process is created using thermodynamic principles, then multi-objective optimisation is performed determining static optimum values for the decision variables using GAs. After thermodynamically modelling a CHP plant, [44] uses GAs to produce a static multi-objective optimisation in order to obtain the optimal operation planning for a CHP process under different scenarios while [45] uses linear programming (i.e., a mathematical method of solving optimisation problems by means of linear functions in which the variables involved are subject to linear constraints). In [46] a combined cooling, heating and power system is statically optimised using two objective functions under three different scenarios using GAs. Multi-objective statical optimisation is performed in [47, 48] using Particle swarm optimisation (PSO). PSO is a population-based optimisation technique inspired by the social behaviour of birds flocking or fish schooling. Dynamic systems, i.e. systems that change over time as is the case of the real cogeneration process used in this study, can be optimised statically with a fixed solution that corresponds to some average system state (e.g. works referred to in previous literature review), or dynamically where the solution tries to follow the system change over time. It is a normal expectation that dynamic optimisation has to give better results than a static one. Dynamic optimisation is more complex, requires more computation, more advanced methods, but is superior to static optimisation because it can always be transformed to the static case simply by neglecting change of the system in time and selecting a single state as a representative. The main limitation of static optimisation is that static values are not the best option due to possible changes in working and atmospheric conditions [49]. Therefore, the best option for the optimisation of a real cogeneration process is to obtain the optimum values for the decision variables in each time-step of the dataset (i.e., a dynamic optimisation).

In [50] the dynamic single-objective minimization through a network flow model by using linear programming was achieved. Several optimisation methods, including GAs and PSO, are used in [51] to dynamically minimize the operating cost of the CHP system. A dynamic single-objective optimisation through a mathematical model was proposed in [52] with the use of an exhaustive search method (i.e.,a mathematical method that consists in the enumeration of all possible candidates for the solution and checking whether each candidate satisfies the problem's statement). In [53] a PSO and fuzzy decision-making system was proposed to optimise a benchmark cogeneration system. The work carried out in [54] presented a thermodynamic model of the process and dynamically optimised the set points related to the economically efficient management of a combined heat, power and cooling plant

using dynamic programming (a method for solving complex problems by breaking them down into a collection of simpler sub-problems). This allows to minimize the total daily cost based on the demand for the process. However, it did not improve the efficiency of the process or any of its components.

In summary, the studies cited above deal with the modelling of only one or more components of the plant, and/or the optimisation of certain parameters of the process. In this sense, the main contributions of the thesis are: 1) CI algorithms have been applied to both modelling and optimization of a CHP process; 2) the modelling process is performed for all the components of a CHP process; 3) data from a real plant have been used; 4) the optimisation has been carried out for a multi-objective function; and 5) it is intended for a continuous on-line operation of the plant.

The present thesis performs the dynamic optimisation of a CHP process using a global multi-objective function, i.e., the effective electrical efficiency ($\varepsilon_{EE}$), by determining the optimal values for 12 decision variables for each time-step in a one-week dataset. The dynamic optimisation procedure was based on gradient descent methods (GDMs), which searches for optimum value for a minimization problem with an objective function from a given point [55], and genetic algorithms (GAs), that mimics the process of natural selection through genetic operators [56].

The CHP plant generates electricity with four internal combustion engines. The flue gases from the engines are used to generate steam that feeds a steam turbine and generates more electricity. The heat from the engines is also used in a slurry drying process. The multi-objective function considers the whole cogeneration and slurry process while taking into account three goals: to maximize the electrical energy generated by the engines and steam turbine; to minimize the use of primary energy i.e., the total fuel used to feed the engines; and to maximize the use of thermal energy from the engine's flue gases in the slurry drying process. CI techniques were employed to model the cogeneration and the slurry treatment process.

The rest of this study is structured as follows:

- Chapter 2 presents an introduction to cogeneration systems including the basic concepts and how they differ from traditional power plants. A brief history of cogeneration is developed. The different cogeneration technologies and prime movers used are described, and, finally, a description of the cogeneration plant used in this study and its main components is also explained.
- In Chapter 3 the Computational Intelligence techniques used in this study are presented, including their basics concepts a brief history. These techniques are: artificial neural networks, fuzzy inference systems, neuro-fuzzy systems, and genetic algorithms. Finally, other widely-used computational intelligence techniques and hybrid approaches are listed.
- In Chapter 4, firstly, the data cleaning process and the selection of the inputs-output variables are explained. Subsequently, the different approaches to model the cogeneration process and the slurry drying process are explained,

and the results are discussed and analysed. Finally, a new exhaustive hybrid feature selection method is applied to a system of the plant with the aim of check the initial variable selection.

- Chapter 5 introduces the basics of mathematical optimisation, its formulation and principles, and a classification of the different optimisation techniques. Then, the multi-objective function used in this study is explained and the two optimisation approaches are explained: a single-objective optimisation based on the gradient descent method, and a multi-objective optimisation using a genetic algorithm, and its results are discussed and analyzed.
- Finally, Chapter 6 provide the conclusions for the different modelling and optimisation approaches and the results for this present study.

# Chapter 2: Combined heat and power processes

## 2.1 Overview

This chapter begins by presenting an introduction to cogeneration systems including the basic concepts and how they differ from traditional power plants. Next, the main benefits of cogeneration are listed. This is followed by a brief history of how cogeneration has been developed, as well as an analysis of the main regulatory legislation in Europe and United States concerning cogeneration systems. The different cogeneration technologies are explained according to the sequence of energy used (topping and bottoming cycles). Also described are the most widely used prime movers in cogeneration systems and their fundamentals: internal combustion engines, gas turbines and steam turbines. Finally, there is a description of the cogeneration plant used in this study and its main components: four internal combustion engines, an exhaust steam boiler, and a steam turbine. A slurry drying process which uses heat from the cogeneration process is also explained.

## 2.2 Introduction

According to the European Association for the Promotion of Cogeneration (COGEN), cogeneration (combined heat and power) or CHP, is the simultaneous production of electricity and heat, both of which are used [3]. On average, conventional power plants, operate at an efficiency of just 35%. This means that up to 65% of the potential energy is released as waste heat. More recently, CHP plants have been designed which can increase this efficiency up to 80% [57], as shown in Figure 2.1.

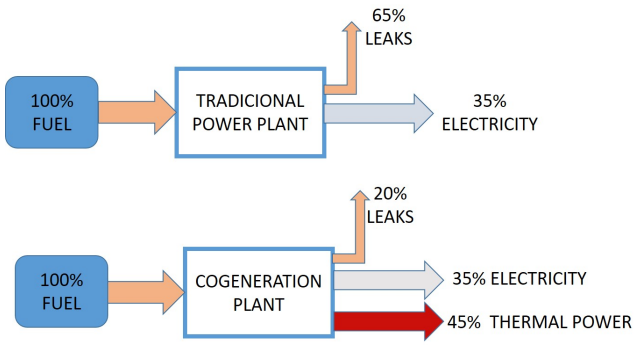



**Figure 2.1:** Efficiency of traditional power plant versus cogeneration systems.

The objective of cogeneration is to use as much of the potential energy contained within a fuel as possible. In CHP, the flue gases produced by engines, gas turbines, or other machines are used to generate more electricity, or are used in other processes. This reduces costs, because the total amount of fuel needed to produce both electricity and heat in a cogeneration plant is less than the total fuel needed to produce the same amount of electricity and heat through separate technologies (e.g., an electricity generating plant and an industrial boiler). This fuel saving also result in less pollution. In light of these economic and environmental advantages, cogeneration plants currently play an important role in certain EU countries [3].

### 2.2.1 Advantages and disadvantages of cogeneration

According to the United States Environmental Protection Agency (EPA) and taking into account the above-mentioned facts, CHP plays an important role in meeting energy needs, as well as in reducing the environmental impact of power generation. Some advantages of CHP systems including [58]:

- Efficiency benefits: CHP requires less fuel to produce a given energy output. It also avoids the transmission and distribution losses that occur when electricity travels along power lines, and is more efficient than separate heat and power generators. Higher efficiency translates into lower operating costs, reduced emissions of all pollutants, increased reliability and power quality, and reduced grid congestion and distribution losses.
- Reliability benefits: CHP can be designed to provide high-quality electricity and thermal energy to a site regardless of what might occur on the power grid, decreasing the impact of outages and improving power quality for sensitive equipment.
- Environmental benefits: because less fuel is burned to produce each unit of energy, CHP reduces air pollution and greenhouse gas emissions.

- Economic benefits: CHP can save facilities considerable money on their energy bills due to its high efficiency, and can provide a hedge against unstable energy costs.

There are of course some drawbacks to CHP systems:

- In order for CHP to be viable it is essential that the demand for heat and electricity are simultaneous.
- CHP is a local installation therefore high maintenance costs must be paid.
- Building CHP plants typically requires a greater initial investment. Energy savings eventually give returns, but more money still has to be spent upfront.

### 2.2.2 Brief history of cogeneration and its current stat

At the beginning of the twentieth century, steam was the main source of mechanical power. However, as electricity became more controllable, many small "power houses" that produced steam realised they could also produce and use electricity, and they adapted their systems to cogenerate both steam and electricity. From 1940–1970, the concept developed of a centralised electric utility that delivered power to the surrounding area. Large utility companies quickly became reliable, relatively inexpensive sources of electricity, so the small power houses stopped cogenerating and bought their electricity from the utilities [59].

During the late 1960s and early 1970s, interest in cogeneration began to revive, and by the late 1970s the need to conserve energy resources became clear. In the United States, legislation was passed to encourage the development of cogeneration facilities [60]. This legislation was the Public Utilities Regulatory Policies Act (PURPA) in 1978, and following its coming into force, cogeneration projects multiplied in the United States [61]. PURPA required utilities to interconnect with and purchase electricity from "qualified facilities" like cogeneration systems, thus giving industrial and institutional users access to the grid and the ability to sell back excess electricity. Shortly after the enactment of PURPA, Congress also created federal tax credits for CHP investments. Following the enactment of PURPA and CHP tax credits, cogeneration grew dramatically, with its capacity increasing significatively in two decades (from about 12 gigawatts in 1978 to 66 gigawatts in 2000).

The years 2006 to 2009 saw much lower levels of cogeneration deployment than historical growth rates owing, in part, to higher natural gas prices and economic uncertainty. One factor affecting the growth of CHP was the change to PURPA regulations that resulted from the Energy Policy Act of 2005 [62]. As instructed by the act, the federal energy regulatory commission (FERC) issued new rules that no longer required utilities to buy electricity from larger "qualified facilities" when those facilities have access to competitive electricity markets, and FERC issued rules to ensure that new CHP "qualified facilities" were not principally electricity-

generating facilities taking advantage of the incentives offered to CHP facilities (so-called "PURPA machines")[63].

In Europe, there was little government support in the 1990s because cogeneration was not seen as new technology and therefore was not covered under "Thermie," the European Community's (EC) Energy Programme. Under Thermie, 40% of the cost of projects was covered by the EC government [64].

Cogeneration was first promoted at European Union (EU) level in the commission green paper of 29 November, 2000, "Towards a European strategy for the security of energy supply" [65]. The paper argued that if the EU's share of cogeneration, which only accounted for 11% of total electricity production in the EU in 1998, were to be increased to 18% by 2010, the ensuing savings could amount to 3-4% of total gross consumption in the EU 15.

In February 2004, the EU adopted the CHP 2004/8/EC Directive to promote cogeneration in the EU by addressing several problems, including a lack of awareness, unclear provisions relating to electricity network access, inadequate support from local and regional authorities, and disparate rules for qualifying CHP as highly efficient [66]. Most notably, this directive, repealed since the entry into force of the Energy Efficiency Directive 2012/27/EU (EED) [67], established a common and harmonised method for calculating the efficiency of cogeneration plants, and required member states to carry out an analysis of their national cogeneration potentials. According to EU 2004/8/EC directive, support can only be granted to cogeneration plants that save at least 10% of primary energy fuel compared to separated means of heat and electricity production, those cogeneration plants are then labelled high efficiency cogeneration plants.

A few years later, the EU's 2006 action plan on 'Energy Efficiency for 2007-2012' proposed further measures to promote cogeneration in the future, acknowledging that it accounted for only 13% of EU electricity consumption in 2006 [68].

The technology was once again put on the table on 13 November, 2008, when the Commission launched its second strategic energy review [69]. As implementation of the cogeneration directive had progressed more slowly than expected, the EU executive asked member states to further work on removing barriers and facilitating electricity grid access.

In 2009, EU leaders enacted in legislation the climate strategies and targets set in 2007, in the Renewable Energy Directive 2009/28/EC [70]. The package set three key targets: a 20% cut in greenhouse gas emissions from 1990 levels, 20% of EU energy to come from renewables, and a 20% improvement in energy efficiency. Thes are also the headline targets of the Europe 2020 strategy for smart, sustainable and inclusive growth.

In 2011, the EC adopted the 'Energy Efficiency Plan 2011' aimed at exploring the most effective measures for achieving the 20% energy efficiency target by 2020 [71]. CHP is mentioned as one of the sectors that could deliver up to 15-20 million tons

of oil equivalent per year of primary energy savings and a reduction of 35-50 million tons per year of carbon dioxide ($CO_2$) emissions, based on an additional economic potential of around 350 terawatt-hours of electricity output from cogeneration.

To ensure that this potential is met, the EED adopted in 2012 addresses CHP in Article 14 'Promotion of efficiency in heating and cooling' and Article 15 'Energy transformation, transmission and distribution'. The EED can be viewed as a small step forward compared to the first CHP directive, nevertheless member states are required to take clear action to promoting CHP wherever it makes economic sense for developers. However, each country holds the key to appropiate and progressive implementation of the CHP-related provisions contained in the EED.

In Spain, from 2013 onwards, various legislation on CHP plants (Law 24/2013, RD 413/2014 and Order IET 1045/2014) raised taxes on all power generation by 7%, as well as on natural gas (which is the primary fuel used by about 90% of Spain's CHP plants), while also reducing subsidies for renewable energy and CHP plants [72, 73, 74]. As a consequence of this, several plants have shut down since the middle of 2013. The electricity generated by CHP plants has decreased by around 20% from 2012 to 2015 according to ACOGEN (Spanish Association for the Promotion of Cogeneration).

To promote cogeneration, Spanish CHP associations propose the following essential policy actions: the tariffs that were in force in July 2013 must be maintained, and actions must be taken in the proposed new regime for facilitating the refurbishment of older cogeneration plants. Both these actions are in line with the need for further measures on cogeneration introduced in the new EDD and to achieve the Europe 2020 strategy targets.

## 2.3 Types of CHP systems

Cogeneration systems are normally classified according to their sequence of energy use and operating schemes [75]. Based on this, cogeneration systems can be classified as being either topping or bottoming cycle:

- In a topping cycle, the fuel is used first to produce power and then thermal energy, which is the by-product of the cycle and is used to satisfy process heat or other thermal requirements, as shown in Figure 2.2. Topping cycle cogeneration is widely used and is the most popular cogeneration method.

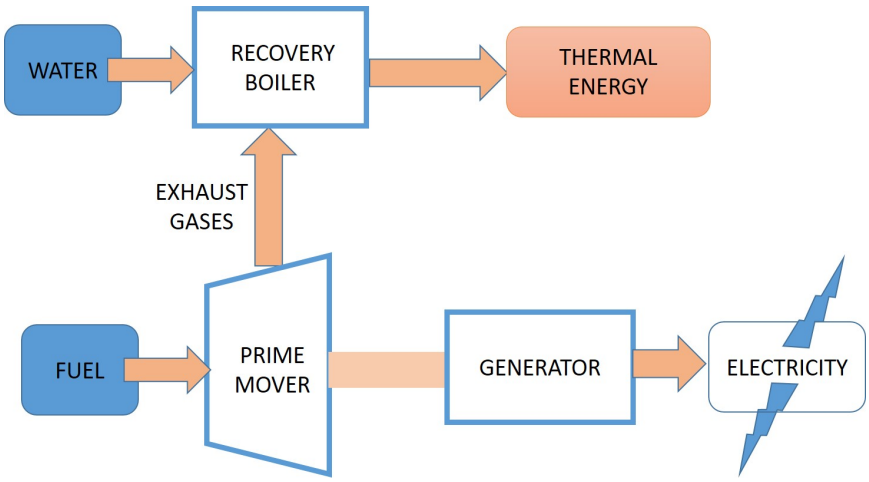



**Figure 2.2:** Diagram of a topping cycle.

- In a bottoming cycle, the primary fuel produces high temperature thermal energy, and the heat rejected from the process is used to generate power through a recovery boiler and a turbine generator, as shown in Figure 2.3. Bottoming cycles are suitable for manufacturing processes that require heat at high temperature in furnaces and kilns, and reject heat at significantly high temperatures. Typical areas of application include the cement, steel, ceramic, gas and petrochemical industries. Bottoming cycle plants are much less common than topping cycle plants.

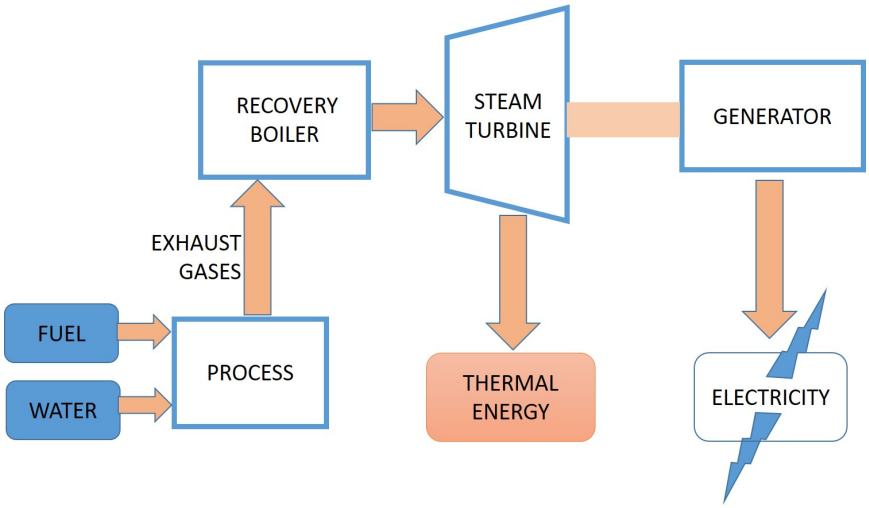



**Figure 2.3:** Diagram of a bottoming cycle.

### 2.3.1 Prime movers in cogeneration

Prime movers are machines capable of burning a wide variety of fuels, including natural gas, coal, oil, and alternative fuels, to produce shaft power or mechanical

energy. Although mechanical energy from the prime mover is most often used to drive a generator for producing electricity, it can also be used to drive rotating equipment such as compressors, pumps, and fans. The most widely used prime movers in cogeneration systems are internal combustion engines, gas turbines, and steam turbines. They are explained briefly below.

### Internal combustion engine

The internal combustion engine is a heat engine that converts the chemical energy in a fuel into mechanical energy, usually made available on a rotating output shaft [76]. The fuel's chemical energy is first converted into thermal energy by means of combustion or oxidation with air inside the engine. This thermal energy raises the temperature and pressure of the gases within the engine, and the high-pressure gas then expands against the mechanical mechanisms of the engine. Mechanical linkages apply this expansion to a rotating crankshaft, which is the engine's output. The crankshaft, in turn, is connected to a transmission and/or power train to transmit the rotating mechanical energy to the desired final use. For engines this will often be the propulsion of a vehicle (i.e., automobile, truck, locomotive, marine vessel, or airplane). Other applications include stationary engines to drive generators or pumps, and portable engines for things like chain saws and lawn mowers.

Most internal combustion engines are reciprocating engines, having pistons that reciprocate back and forth in cylinders within the engine (Figure 2.4). The main four steps in the engine are:

- Firstly, the piston is drawn backwards into the cylinder by the crankshaft. As the piston moves backwards, the volume of the cylinder increases, which results in a drop in pressure within the cylinder (i.e., a vacuum is generated). The pressure soon reaches a critical point, and causes the intake valve to be "sucked" open. Air-fuel mixture is then "sucked" from the turbocharger into the cylinder. As soon as the crankshaft starts to revolve the piston forwards into the cylinder, the volume of the cylinder decreases and the intake valve is forced closed. This is known as the intake stroke.
- Secondly, the piston continues to move forward and compress the air-fuel mixture that is trapped inside the cylinder. The piston will then keep moving forward until the crankshaft just about starts to pull back on the piston (at top dead centre, or the top of the stroke). At this point, the fuel and air in the cylinder becomes highly compressed. This is the compression stroke.
- Once the piston has compressed the air-fuel mixture, a battery or another power source causes a single spark to jump the gap of the spark plug. Since the spark plug is screwed directly into the cylinder of the engine and is in direct contact with the air and fuel, it causes the air and fuel to explode and throw the piston backward. This stroke is known as the ignition or power stroke.

- Finally, the crankshaft spins 180° and cause the piston to move forwards again. The push rod and rocker arm then open the exhaust valve so that the remaining fumes can be forced out by the piston. This is known as the exhaust stroke.

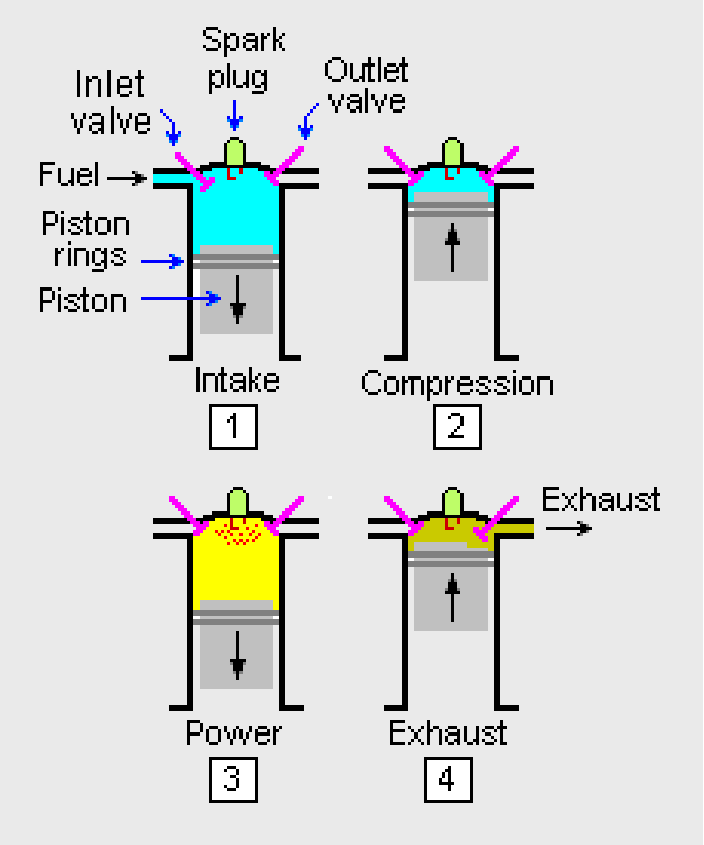



**Figure 2.4:** Four strokes for an internal combustion engine [1].

### Gas turbines

The gas turbine is an internal combustion engine that uses air as the working fluid in continuous combustion. In the case of the reciprocating internal combustion engine explained previously, the intake, compression, ignition, and exhaust steps occur in the same place (cylinder head) at different times as the piston goes up and down. In a gas turbine, however, these same four steps occur simultaneously but in different places, and in continuous combustion. As a result of this fundamental difference, the gas turbine engines have sections called the inlet section, the compressor section, the combustion section and the exhaust section as shown in Figure 2.5.

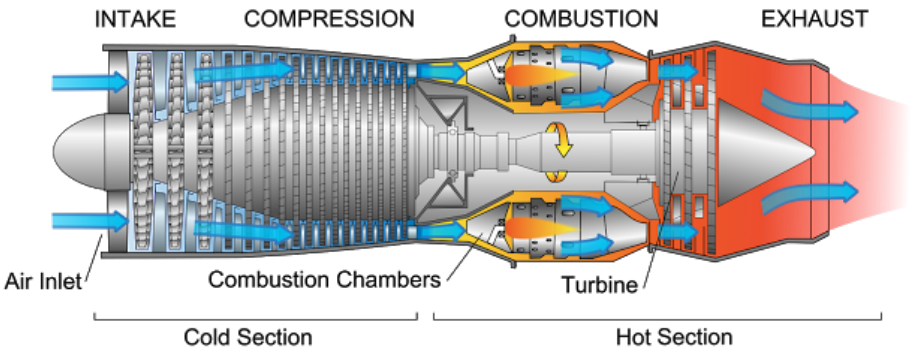



**Figure 2.5:** Diagram of the sections in a gas turbine [1].

Gas turbines comprise a compressor, combustor (or combustion chamber), and turbine. These components work together to produce power or thrust, depending on the application. To begin the cycle, the compressor rotates and draws in ambient air. As it is taken in by the compressor, the air is pressurised, in some cases to 40 times atmospheric pressure. The pressurised air then moves into the combustion chamber where a fuel mixture is ignited, heating it and causing it to expand into the turbine. As the heated air expands through the turbine it pushes against the turbine blades which then rotate the turbine shaft. The rotational energy is used to spin a generator and create electricity. Because they are attached to the same shaft, the rotation of the turbine also rotates the compressor, keeping the system operating.

### Steam turbines

Steam turbines generate electricity from high pressure steam produced in a boiler. Thermal energy is extracted from the steam, used to spin the turbine's blades and drive a rotating shaft and generator, as shown in Figure 2.6. Since steam is produced in a separate boiler, the steam turbine can be powered by a variety of fuels: natural gas, solid waste, coal, woody biomass or agricultural by-products. Steam turbines work well for industrial facilities where the turbines can generate electricity as a by-product of excess or waste heat, or where solid or waste fuels are readily available to use in the boiler.

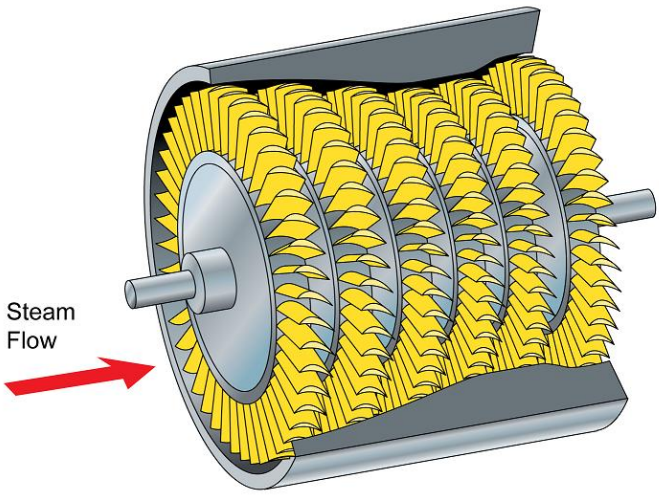



**Figure 2.6:** Diagram of a steam turbine [1].

## 2.4 CHP plant being evaluated

The CHP plant being evaluated in this thesis is located in Monzón (Huesca), in the northern Spain [77]. The plant generates electricity via four internal combustion engines and a steam turbine. Heat from the engines is used to generate steam which is then used in the turbine and a slurry drying process. The slurry drying process utilises slurry obtained from nearby farms. The results of this process are organic fertilizer and water suitable for irrigation. A generalised diagram of the process is shown in Figure 2.7.

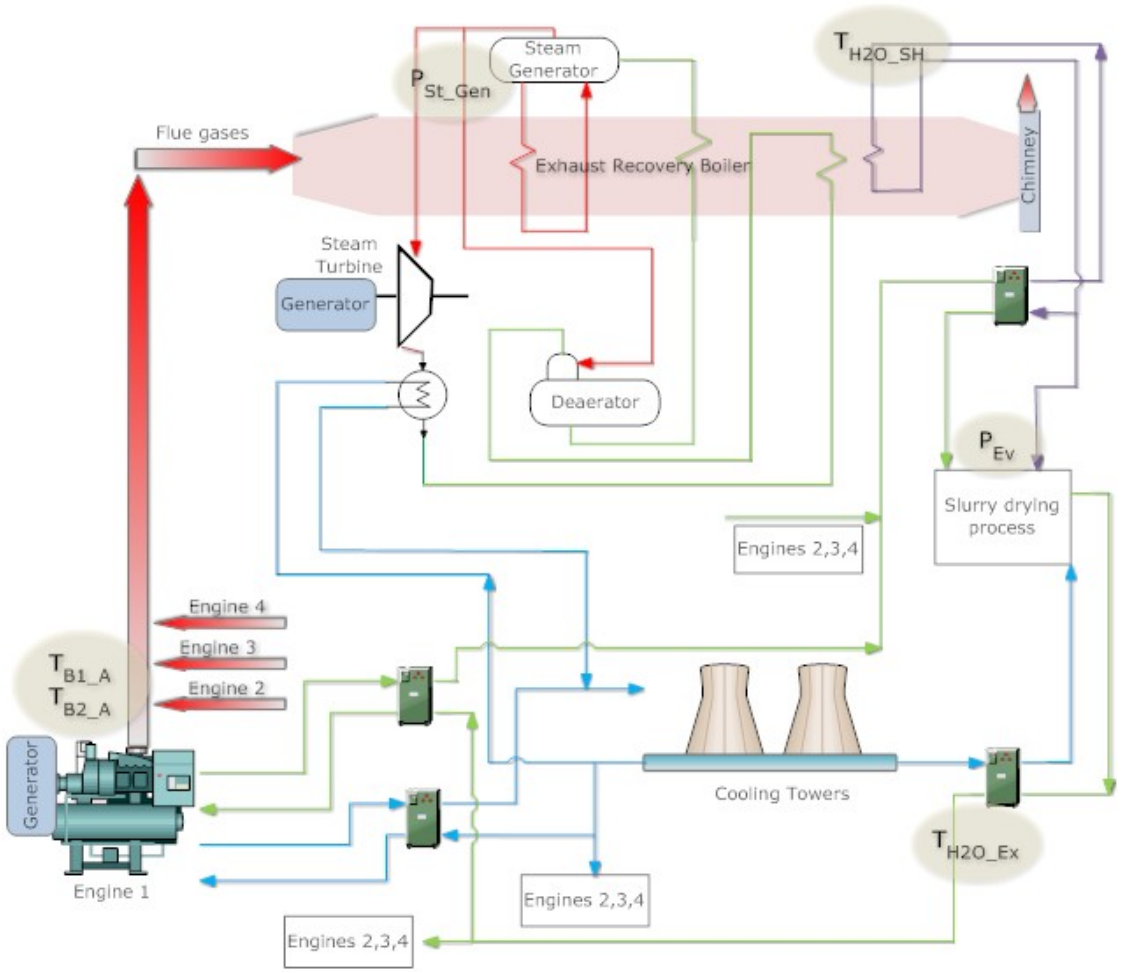



**Figure 2.7:** Diagram of a combined heat and power plant with its related equipment. The red circuit represents steam, the purple circuit is the superheated water circuit, and the green and blue circuits are water circuits.

### 2.4.1 Engines

The CHP plant has four internal combustion engines. These engines are identical, each having the same characteristics and a nominal power of 3700 kW. They have two banks of eight cylinders (see Figure 2.8) and operate with natural gas. The engines exchange heat with two circuits that take water from cooling towers, as shown in Figure 2.7. One cooling circuit preheats the air intake to around 35 ºC. The air intake is then mixed with the natural gas, creating, the required mixture. Another cooling circuit maintains the temperature of the air-fuel mixture at around 50 ºC. Finally, the mixture is fed into the engines, which generate electricity and flue gases.

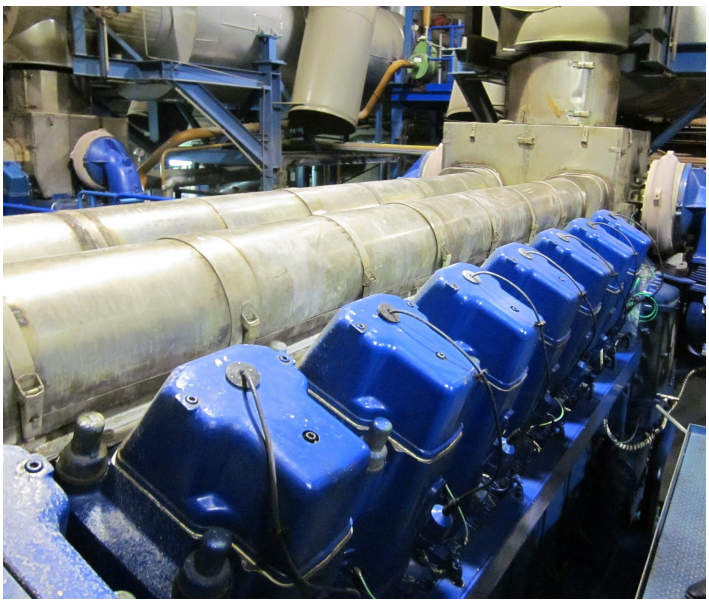


**Figure 2.8:** Photograph of one of the four engines at the CHP plant.

The electricity that is generated is sold. The plant owner has determined, as an economic constraint, that the four engines must operate at at least 94 % of rated power.

Each engine has a diverter which sends the flue gases to an exhaust steam boiler, whenever the given engine is operating at greater than 50 % of its rated power, or to the chimney, if operating at less than 50 % rated power. The engines usually run at levels above this threshold and so the flue gases are generally sent to the exhaust steam boiler.

### 2.4.2 Exhaust steam boiler

The flue gases from the engines are generally sent to the exhaust steam boiler at a temperature of around 500 ºC. The product of the exhaust steam boiler are (see Figure 2.9):

- Superheated steam at around 22.5 bar and 400 ºC to feed the steam turbine.
- Superheated water at around 120-125 ºC, which is used in the slurry drying process.
- Exhaust gases that exit through the chimney at around 120-125 ºC.

The exhaust steam boiler has three different stages. In the first stage the gases from the engines have the highest temperature, meaning they can transfer more heat than when the gases are close to the chimney because the gases gradually transfer heat as they pass through the exhaust steam boiler.

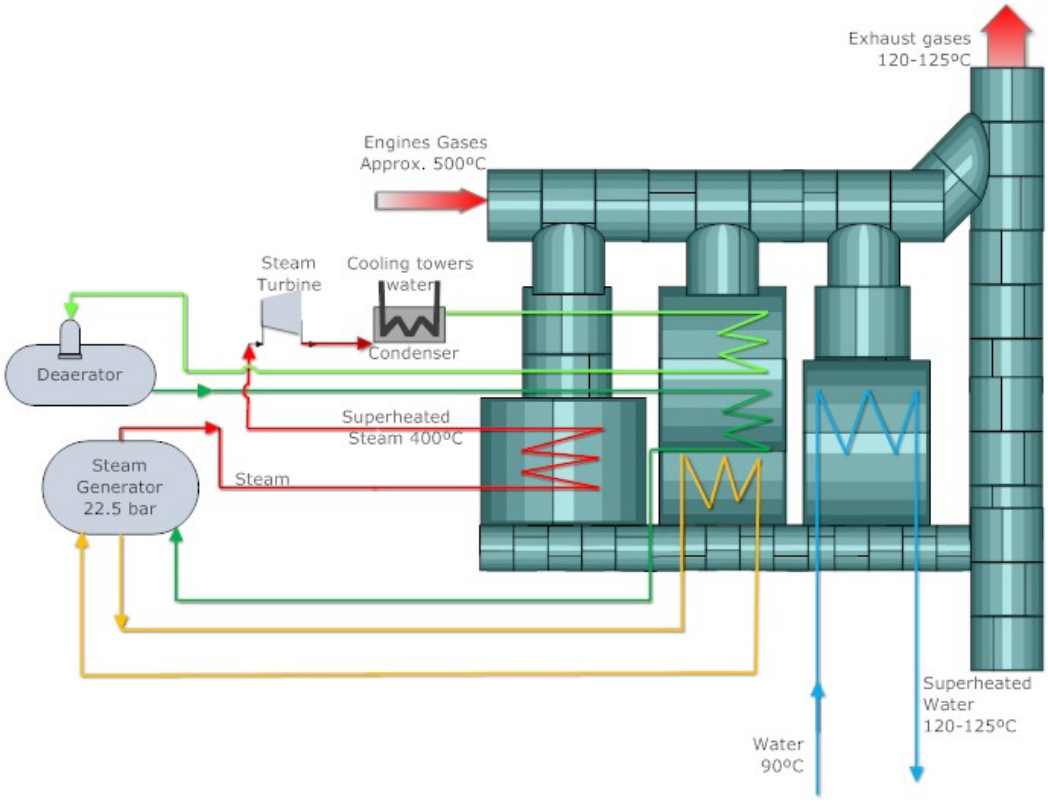



**Figure 2.9:** Diagram of an exhaust steam boiler and its related circuits.

In the steam generator tank steam for the steam turbine is generated. For this purpose, the steam generator tank uses water from the deaerator. The level of the steam generator is regulated with this water from the deaerator. The water is preheated in the second stage of the exhaust steam boiler. The steam generator has a heat circuit which also uses the heat from the second stage. The steam generated in the steam generator tank goes to the first stage of the exhaust steam boiler to superheated the steam to around 400 ºC. This is regulated with a diverter which varies the quantity of gases sent to the first stage. Subsequently, the superheated steam is sent to the steam turbine.

In the third stage in the exhaust steam boiler superheated water is generated. The water reaches a temperature of around 90 ºC and leaves the exhaust steam boiler at around 120-125 ºC. This is regulated with a diverter which either sends more gases to the third stage, when the superheated water needs more heat, or, alternatively, to the chimney.

### 2.4.3 Steam turbine

The superheated steam from the exhaust steam boiler feeds the steam turbine to generate more electricity, with a nominal power of 1200 kW. The steam turbine

condenser uses water from the cooling towers to force the steam from the turbine to condense before recirculating it to the system, and after being preheated again, in the second stage in the exhaust steam boiler, the water is recirculated to the deaerator (Figure 2.9).

As with the engines, the power generated from the steam turbine is sold.

### 2.4.4 Slurry drying process

The slurry from surrounding farms comprises approximately 6% solids. It is first subjected to a mechanical treatment using rotatory equipment to remove the solid fraction from the liquid portion (Figure 2.10). The liquid portion is then chemically treated to reduce the chemical load. The product of the chemical treatment is then thermally treated to separate the condensable from non-condensable matter in an evaporator which uses superheated water generated in the exhaust steam boiler (water at a temperature of around 120 $^{o}$C). A tubular heater recirculates the effluent to the evaporator and preheats it. The tubular heater takes water from the cooling circuit that preheats the engine's air intake. The non-condensable portion is combined with the solid fraction produced by the mechanical treatment, this is then sold as fertiliser. The condensable effluent is condensed again using water from the cooling towers. Finally, the steriliser uses superheated water to purify the condensed effluent and produces water suitable for irrigation.

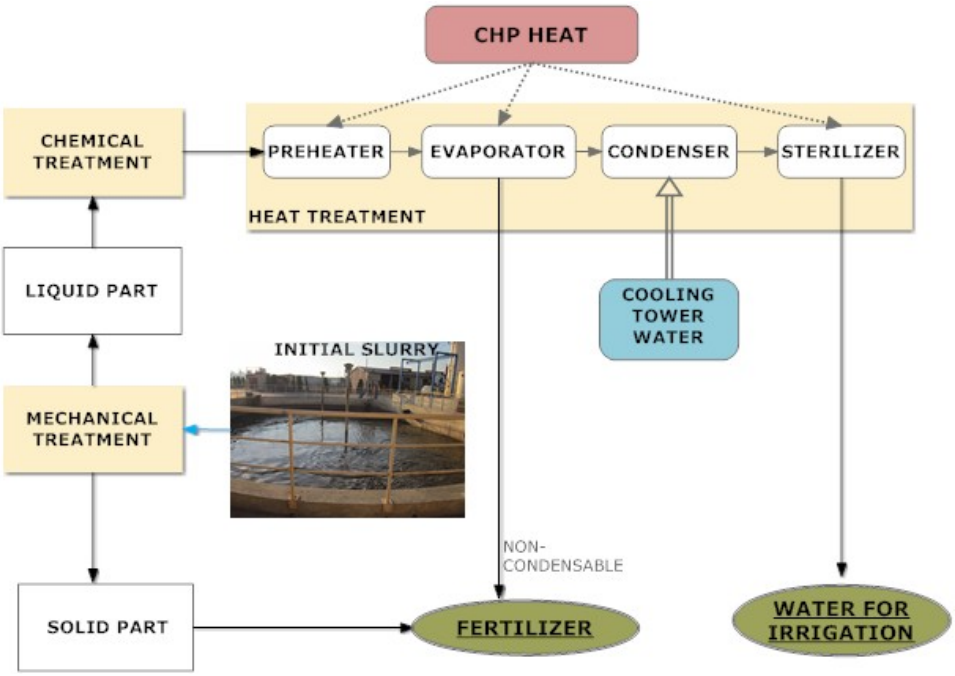



**Figure 2.10:** Diagram of the slurry drying process.

# Chapter 3: Computational intelligence algorithms

## 3.1 Overview

This chapter presents the computational intelligence (CI) techniques that have been used in this study, including their basics concepts, and a brief history of each. The CI techniques are: artificial neural networks (ANNs), fuzzy inference systems (FISs), neuro-fuzzy systems (NFs) (which are hybrid approaches), and genetic algorithms (GAs). ANNs try to mimic how the human brain learns, and consist of a highly interconnected system of processing elements called neurons. ANNs are capable of learning from input-output samples. FISs, based on fuzzy set theory, are inspired by how the human brain can reason with imprecision, vagueness, and incomplete information. Neuro-fuzzy systems comprise a synergistic combination of ANNs and FISs. NFs find the parameters of a fuzzy system by exploiting the learning capability of ANNs. The idea behind GAs is the natural evolutionary process of biological organisms. Finally, this chapter lists other widely-used computational intelligence techniques and hybrid approaches.

## 3.2 Computational intelligence

Computational intelligence (also called soft computing) comprises a collection of alternative techniques and methods able to deal with problems that are difficult to solve with conventional techniques due to their complexity, high dimensionality, hard non-linearities, lack of analytical models and vague or imprecise knowledge [78].

Some of the most important CI techniques are the following:

- Fuzzy logic: fuzzy sets theory was established by Zadeh [79] in the 60s and it was the base for the development of fuzzy reasoning. Fuzzy logic's main feature is its ability to work with imprecise information and as a universal function approximator [80], [81], [82]. In recent years, fuzzy inference systems have been

used in a wide area of applications, especially in control mechanisms. A FIS is formed by an inference mechanism, a set of IF-THEN type linguistic rules and information converters: a fuzzifier [83] and a defuziffier [84], [85]. These converters are needed between the environment, which uses crisp information, and the fuzzy system in which the information is presented by fuzzy sets.

- Artificial neural networks: these are networks formed by processing elements (i.e. neurons) as proposed by McCulloch and Pitts in 1943. They are interconnected to emulate living beings' brains [86]. Their main advantage is their ability to learn and adapt from data samples [87]. Neural networks are efficient in pattern recognition and classification [88], [89], function approximations [90], data clustering [91] and vector quantization [92].
- Evolutionary algorithms: these are searching or learning algorithms based on the mechanics of natural selection, genetics and evolution. The most used paradigm is perhaps the genetic algorithms whose principles were first published by Holland in 1962 [93] with the mathematical framework being presented in 1975 [94]. In genetic algorithms, every chromosome is shown as a string of binary numbers codifying a possible solution to a problem. The learning is based on the process of selection and reproduction (where the chromosomes are re- combined simulating sexual reproduction) and mutation, in which a gene of the chromosome is altered randomly. They are used in optimization, parameter learning (e.g. the parameters of a neural network [95]), path planning [96], or system control [97].

Many of these techniques exhibit complementary aspects and hence, they very often provide better performance when combined in a cooperative way rather than acting exclusively (e.g. neuro- fuzzy systems (NFs), evolutionary-fuzzy systems (EFs), or neuro-evolutionary systems (NEs)). For example, neuro-fuzzy systems [98], [99] have the advantage of showing information in a linguistic way, like the fuzzy systems, and also have the learning abilities of neural networks.

## 3.3 Artificial neural networks

### 3.3.1 Introduction to and biological background of artificial neurons

Artificial neural networks were originally designed to, in some way, model the functionality of the biological neural networks that are part of animal brains. In a biological brain, a typical neuron collects signals from other neurons through a host of fine structures known as dendrites (see Figure 3.1). The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical impulses that inhibit or excite

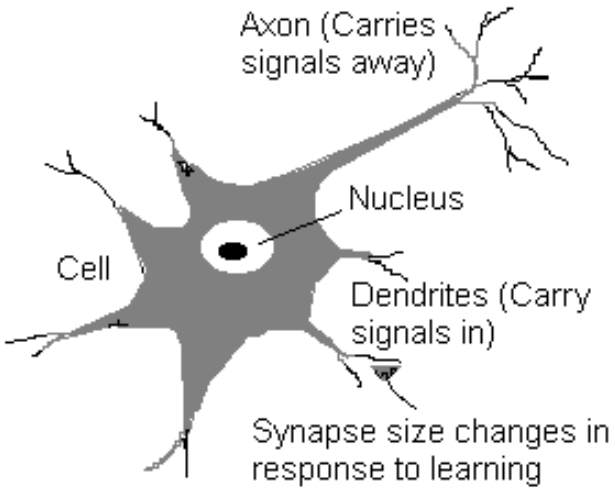



**Figure 3.1:** Diagram showing the components of a biological neuron.

activity from the axon into electrical impulses that inhibit or excite activity in the connected neurons. When a neuron receives excitation input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another is altered.

### 3.3.2 Brief history of neural networks

The earliest work on artificial neural networks dates back to 1943 when McCulloch and Pitts introduced the first artificial neural network computing model [86]. Their neural network was based on simple neurons which were considered binary devices with fixed thresholds. Their model resulted in simple logic functions such as "a OR b" and "a AND b". Reinforcing this concept of neurons and how they work, in 1949 Donald Hebb postulated a new learning paradigm where he pointed out that neural pathways are strengthened each time they are used [87].

In the 1950s, as the computers advanced into their infancy, it became possible to model the rudiments of these theories on human thought. Nathalien Rochester at the IBM research laboratories led the first effort to simulate an artificial neural network. It was during this time that traditional computing began to increase in popularity, and, the emphasis on computing side-lined neural research. The 1956 Dartmouth Summer Research project on Artificial Intelligence significantly boosted interest in neural networks [100].

In 1958, Rosenblatt developed the perceptron [101], an algorithm for pattern recognition based on a single layer learning computer network using addition and subtraction. The original perceptron received two inputs, gave a single output, and used

threshold function as activation function. The perceptron learned from its mistakes, i.e., when a sample is not correctly classified, the weights are updated in order to move the output closer to the target. This is done by adding the input vector to the weight vector when the error is positive, or subtracting the input vector from the weight vector when the error is negative.

In 1959, Bernard Widrow and Marcian Hoff at Stanford developed models they called ADALINE and MADALINE. These models were named for their use of Multiple ADAptive LINear Elements [102]. The method used for learning was different to that of the perceptron, they employed the least mean squares (LMS) learning rule, instead of the perceptron learning method, as well as a linear activation function. MADALINE was the first neural network to be applied to a real-world problem. It is an adaptive filter that eliminates echoes on phone lines, and this neural network is still in commercial use.

In 1969, Marvin Minsky and Seymour Papert demonstrated the limitations of single layer perceptrons [103], and, as a result, activated considerable prejudice against this field.

Paul Werbos, in 1974, developed and used the back-propagation learning mbioethod for multiple layer neural networks [104]. In essence, the back propagation algorithm is used to train neural networks with multiple layers, a different threshold in the artificial neuron, and a more robust and capable learning rule. The back-propagation algorithm effectively solved the single layer perceptron problem.

In 1982, interest in the field was renewed. John Hopfield presented a paper to the National Academy of Sciences. His approach was not only to simply model brains but also to create useful devices [105, 106]. That same year, Fukushima proposed "Neocognitron", the first multi-layered network [107]. Reilly and Cooper used "Hybrid Network" with multiple layers, each of the layers using a different problem solving strategy [108]. The self-organising map (SOM) was originally developed by Kohonen to explore and organise data in an unsupervised way [109, 110]. At the same time, a joint US-Japan Conference was held on Cooperative /Competitive Neural Networks [111], which led to more funding and thus further research in the field.

In 1986, three independent groups of researches worked on how to extend the Widrow-Hoff rule to multiple layers, obtaining one of the most common ANNs, the multi-layer perceptron (MLP) [112, 113, 114].

Adaptive resonance theory (ART) was first introduced by Grossberg in 1976 [115]. The development of ART continued into the 1980s and 1990s at the hands of Grossberg, Carpenter, and other co-workers, resulting in more advanced network models [116, 117, 118]. The term "resonance" refers to the state of a neural network in which a category prototype vector matches closely enough to the current input vector; ART matching leads to this resonant state, which permits learning. The network learns only in its resonant state.

Radial basis function (RBF) networks were first introduced by Broomhead and Lowe in 1988 [119]. Although the basic idea of RBF was first developed 30 years ago under the name method of potential function, the work by Broomhead and Lowe opened up a new frontier in the neural network community.

Nowadays, ANNs have two new trends: extreme learning machines (ELMs) and deep learning (DL). The extreme learning machine is a novel learning algorithm for training single hidden-layer feed-forward neural networks (SLFNs), as proposed in 2004 by Huang et al. [120]. This randomly chooses the input weights of the hidden-layer neurons and analytically determines the output weights through simple matrix computations, therefore enabling a much faster learning algorithm than most popular learning methods such as back-propagation [38]. In 2006, the concept of deep learning was first proposed by Hinton and Salakhutdinov [121]. Deep learning is based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers with complex structures or, otherwise, composed of multiple non-linear transformations. A deep neural network (DNN) is an ANN with multiple hidden layers of units between the input and output layers [122, 123].

### 3.3.3 Artificial neurons

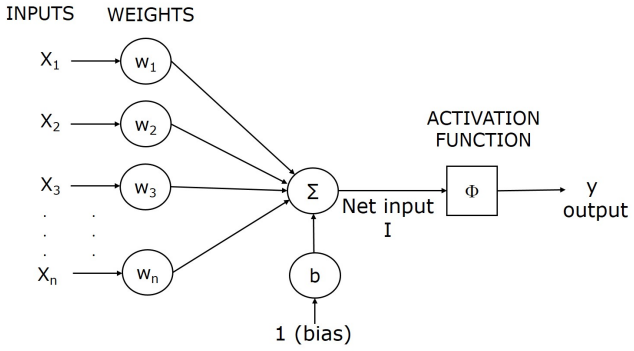



**Figure 3.2:** Diagram of a simplified model of an artificial neuron.

An artificial neuron is a model whose components have a direct counterpart in the components of a biological neuron. Figure 3.2 shows the schematic representation of an artificial neuron. A neuron receives one or more inputs ($x_i$). Each of these inputs is multiplied by weights ($\omega_i$), which are similar to a synapse in a biological neuron. These weights can be either positive or negative, corresponding to strengthening or inhibiting the flow of electrical signals. Subsequently, the weighted inputs are aggregated, resulting in a quantity:

$$I = b + \sum_{i=1}^{n} x_i \omega_i, \tag{3.1}$$

with $n$ being the number of inputs and weights, and $b$ a bias input. The bias term, $b$, can be seen as a weight operating on an extra constant input equal to one.

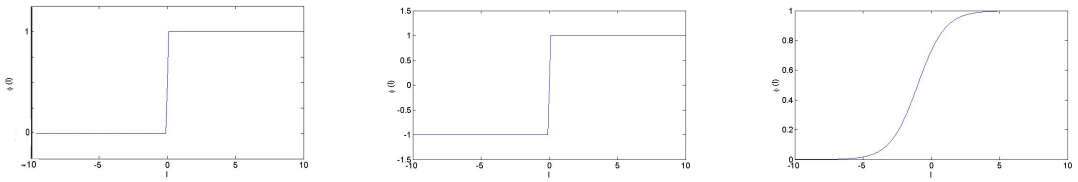
Then, the result of this aggregation is passed through an activation function, $\Phi$, making the activation of the neuron continuously value, according to Eq. 3.2.

$$y = \Phi(I). \tag{3.2}$$

Artificial neurones use a number of common activation functions. Figure 3.3 shows several widely-used activation functions. These include threshold functions like in Fig. 3.3a, that pass the information (usually a +1 signal) only when the net input $I$ exceeds the threshold . There are also sign functions, as shown in Fig. 3.3b, that pass negative information (-1) when the net input is less than the threshold, $\theta$, and positive information (+1) when the net input is greater than the threshold. One of the most commonly used activation functions is the sigmoid function, which is a continuous function that varies gradually between two asymptotic values, typically between 0 and 1.This function is shown in Fig. 3.3c and is defined by the equation:

$$\Phi(I) = \frac{1}{1 + e^{-\alpha \cdot I}}, \tag{3.3}$$

with $\alpha$ being a coefficient that adjusts the abruptness of this function as it changes between the two asymptotic values.



**(a)** **(b)** **(c)**

**Figure 3.3:** Activation functions for neurons. a) Step activation function, b) Sign activation function, and c) Sigmoid activation function.

When a neuron fires, its output has a non-zero value and the output of the neuron is transmitted to the neurons in the next layer. When a neuron is situated in the output layer, its output is (along with the other outputs from the neurons in the same layer) the output of the network. By analysing the structure of a neural network, it can be stated that the information is stored in the weights and biases.

### 3.3.4 Artificial neural networks

An artificial neural network can be defined as a data processing system consisting of a large number of simple, highly interconnected processing elements (artificial neurons) which work in parallel. These neurons communicate by sending information in the form of activation signals to each other along directed connections. A commonly used synonym for "neural network" is "connectionist model". Furthermore, the expression "parallel distributed processing (PDP)" can often be found relating to artificial neural networks [124, 125].

The different kinds of ANNs can be grouped into three generic architectures according to their graphs, with neurons situated at the vertices, and weights on the edges:

1. Single layer feed-forward network: consists of a single layer of weights, where the input signals are directly connected to the output signals, via the weights. The synaptic links carrying the weights connect every input to every output, but not the other way round. This feed-forward network type is shown in Figure 3.4.

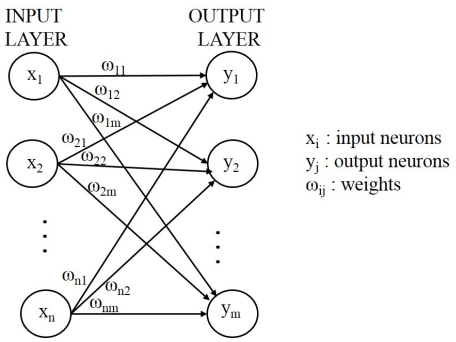



**Figure 3.4:** Diagram of a Single Layer Feed-forward Network.

2. Multi-layer feed-forward network: consists of multiple layers. This architecture has an input layer, an output layer, and one or more intermediate layers, the hidden layers. The computational units of hidden layers are called the hidden nodes. Hidden layers carry out intermediate computation between the input layer and the output layer. An example is shown in Figure 3.5.

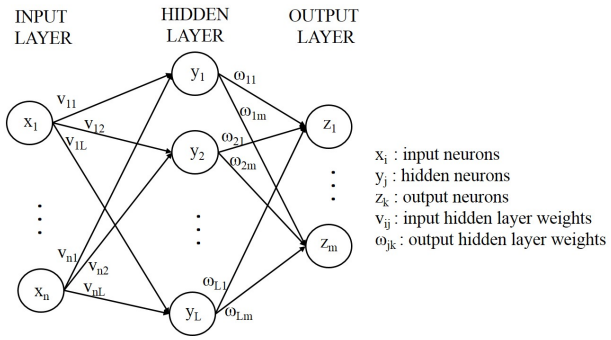



**Figure 3.5:** Diagram of a Multi-Layer Feed-forward Network.

3. Recurrent network: recurrent networks differ from feed-forward architecture as they consist of any network with at least one feed-back connection. They may, or may not, have hidden units. An example is shown in Figure 3.6.

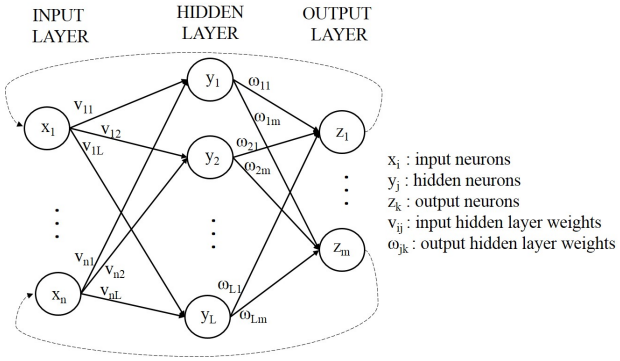



**Figure 3.6:** Diagram of a Recurrent Neural Network.

One of the most important aspects of ANNs is the learning process, because it can be stated that ANNs store the information in the weights. The learning process in ANNs is a result of altering the network' weights, with some kind of learning algorithms. The learning algorithms can be classified into three methods:

1. In supervised learning, both the inputs and the outputs are provided. The network then processes the inputs and compares the resulting outputs against the desired outputs. Errors are then calculated, causing the system to adjust the weights which control the network. This process occurs over again and over as the weights are continuously tweaked.

   Back propagation (BP) is a systematic method for training (learning) multi-layer neural networks, widely used to solve numerous problems. BP was first proposed in 1974 by Werbos [104]. Subsequently, it was developed in 1982 by Parker [112], and in 1986 by Rumelhart, Hinton, and Williams [113]. BP is based on gradient descent methods (GDMs). During the training process the weights are updated in order to minimize the sum of the squared difference

between the real output (desired output) and the network output. Assuming that the given training data set has $K$ entries, an error for the $i$th ($1 \leq i \leq K$) entry of the training set data is defined as the sum of the squared errors:

$$E = \frac{1}{2}\sum_{i=1}^{K}(y_i - y_i^{'})^2, \tag{3.4}$$

where $E$ represents the error, and $y_i$ and $y_i^{'}$ are respectively the actual and the desired $i$-th outputs. Thus the task here is to minimise the overall error measurement. The connection weight $w_{ij}$ is updated according to the Eq. (3.5). The weight is updated in every $k-$th iteration of the GDM algorithm:

$$w_{ij}^{k} = w_{ij}^{k-1} - \eta\frac{dE}{dw_{ij}}, \tag{3.5}$$

where $\eta$ is the learning rate which is a parameter that affects the convergence speed and stability of weights during learning. $\eta$ applies a greater or lesser portion of the respective adjustment to the old weight. If the factor is set to a large value, then the neural network may learn more quickly, but if there is a large variability in the input set then the network may not learn very well or at all [126, 127, 128].

The main strengths and limitations of ANNs trained with BP are shown in Table 3.1.

| **Advantages** | **Disadvantages** |
|---|---|
| Capability of learning from data | Overfitting of training data |
| Ability to learn nonlinear functions | Local minima |
| Robustness and fault tolerance | Difficult to adjust network parameters and network structure. |

**Table 3.1:** Strengths and limitations of ANNs trained with BP.

2. In unsupervised learning, the network is provided with inputs but not with desired outputs. The system must then decide itself which features it will use to group the input data. This is often referred to as self-organisation or adaption. The Hebbian algorithm [87, 129], ART networks, and self-organising networks are examples of ANNs that use unsupervised learning [130].
3. Reinforcement learning is a special case of supervised learning, where the exact desired output is unknown. The teacher only supplies feedback on the success

or failure of an answer. Reinforcement learning is a learning procedure that rewards the neural network for its good output result and punishes it for the bad output result [131, 132].

### 3.3.5 Extreme learning machine

Extreme learning machines (ELMs) were originally proposed by Huang et al. [38] for single hidden-layer feed-forward networks (SLFN) and then extended to "generalised" single hidden-layer feed-forward networks where the parameters of the hidden layer are randomly generated [133]. Extreme learning machines have attracted increasing attention recently because they outperform conventional artificial neural networks in some aspects [36, 37]. ELMs provide a robust learning algorithm, free of local minima, without overfitting problems and are less dependent on human intervention than ANNs. ELM is appropriate for implementing intelligent autonomous systems with real-time learning capabilities.

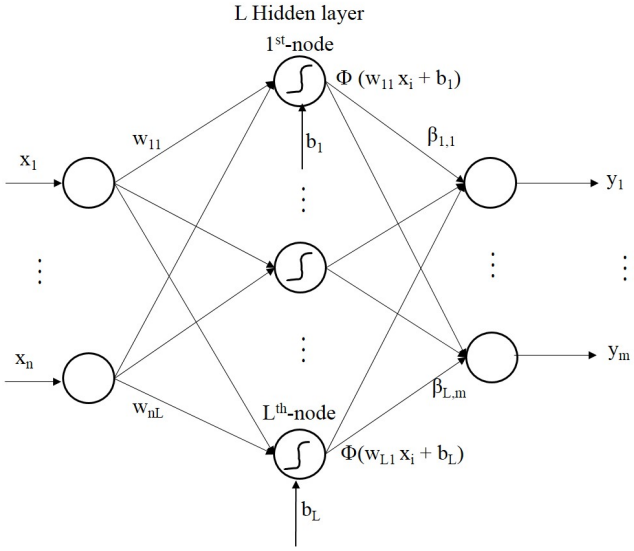



**Figure 3.7:** Topological structure of a SLFN.

Supposing a SLFN with $n$ inputs, $m$ outputs and $L$ nodes in the hidden layer (see Figure 3.7), the output $y$ of the SLFNs can be written as:

$$y_j = \beta_j \mathbf{h}(x), \tag{3.6}$$

where $\beta_\mathbf{j} = [\beta_{j1}, ..., \beta_{jl}]^T$ is the weight vector connecting the hidden layer and the $j-$th output node, $\mathbf{h} = [h_1, h_i..., h_L]$ is the vector formed by the values $h_i = \Phi(\mathbf{w_i x} + b_i)$ with $\Phi()$ being the activation function, $\mathbf{w_i} = [w_{i1}, \ldots, w_{in}]^T$ the vector connecting

the input $\mathbf{x} = [x_1, \ldots, x_n]$ with the $i-$th hidden node, and $b_i$ the bias of the $i$th hidden node.

The main difference between ELM and traditional learning approaches is that the hidden layer does not have to be tuned; i.e., it is a randomised layer. This means that the set of parameters in the hidden nodes $(\mathbf{w}_i, b_i)$, $1 \leq i \leq L$, are randomly generated. They are therefore independent of the application and the training samples. Learning in ELM is a straightforward procedure that aims to compute the vector of output weights, $\beta_j$ in (Eq. 3.6), for each output node.

For $K$ arbitrary distinct samples $(\mathbf{x^k}, \mathbf{t^k})$, where $\mathbf{x}^k = [x_1^k, \ldots, x_n^k]^T \in R^n$ are the input data and $\mathbf{t}^k = [t_1^k, t_2^k, \ldots, t_m^k]^T \in \mathbf{R^m}$ are the target data, the above linear equations can be written in matrix form:

$$\mathbf{H}\beta = \mathbf{T}, \tag{3.7}$$

where $\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_n, b_1, \ldots, b_L, \mathbf{x}^1, \ldots, \mathbf{x}^k)$ is given by:

$$\mathbf{H} = \begin{bmatrix} \Phi(\mathbf{w}_1\mathbf{x}^1 + b_1) & \ldots & \Phi(\mathbf{w}_L\mathbf{x}^1 + b_L) \\ \vdots & \ldots & \vdots \\ \Phi(\mathbf{w}_1\mathbf{x}^k + b_1) & \ldots & \Phi(\mathbf{w}_L\mathbf{x}^k + b_L) \end{bmatrix}_{K \times L}. \tag{3.8}$$

$\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_K]^T_{K \times m}$ is a target label vector and $\beta = [\beta_1, ..., \beta_m]^T_{L \times m}$. The solution of the above equation is given as: $\beta = \mathbf{H}^\dagger\mathbf{T}$, where $\mathbf{H}^\dagger$ is the Moore-Penrose generalised inverse of matrix $\mathbf{H}$ [134].

## 3.4 Fuzzy systems

Fuzzy systems are based on the fuzzy set theory proposed by Zadeh in the 1960s [79]. These are systems that can deal with imprecision, vagueness or incomplete information. Fuzzy logic is used as a means of representing and manipulating imprecise data. Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems [135]. The theory of fuzzy logic provides a mathematical framework for capturing the uncertainties associated with human cognitive processes, such as thinking and reasoning. Conventional approaches to knowledge representation lack the means for representing the meaning of fuzzy concepts. As a consequence, the approaches based on first order logic and classical probability theory do not provide an appropriate conceptual framework for dealing with the representation of common sense knowledge, as such knowledge is, by its nature, both lexically imprecise and non-categorical.

### 3.4.1 Brief history of fuzzy logic

Fuzzy sets theory was first proposed in 1965 by Lofti A. Zadeh in his paper "Fuzzy Sets" [136]. Most of the fundamental concepts in fuzzy theory were proposed by Zadeh in the late 1960s and early 1970s [137, 138, 139, 140, 141, 142]. At the beginning of this brainstorm, the papers published by Zadeh were not well received in the West, and in many cases were bitterly dismissed by the more conservative researchers in the scientific community. However, over time the theory began to gain supporters, leading to these theories being extended again and again, becoming firmly rooted amongst the most innovative scientists, particularly top professionals in Japan, and then South Korea, China and India. Europe and the United States eventually incorporated this new mathematics, but more slowly.

A big event in the 1970s was the birth of fuzzy controllers for real systems. In 1975, Mamdani and Assilian established the basic framework for fuzzy controllers and applied fuzzy inference systems to control a steam engine [143, 144]. Later in 1978, Holmblad and Ostergaard developed the first fuzzy controller for a full-scale industrial process: the fuzzy cement kiln controller [145].

In the 1980s, from a theoretical point of view, this field progressed very slowly. It was saved by the application of fuzzy control. Japanese engineers, quickly found that fuzzy controllers were very easy to design and worked very well for many problems. In 1980, Sugeno began to create Japan's first fuzzy application-control for a Fuji Electric water purification plant [146]. In 1983, he began pioneering work on a fuzzy robot [147]. In the early 1980s, Yasunobu and Miyamoto from Hitachi began to develop a fuzzy control system for the Sendai subway [148]. They finished the project in 1987 and created the most advanced subway system on earth.

The success of fuzzy systems in Japan surprised the mainstream researchers in the United States and Europe. Some still criticise fuzzy theory, but many others have changed their minds, giving fuzzy theory a chance to be taken seriously.

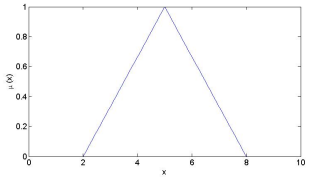
### 3.4.2 Basic definitions and terminology

**Fuzzy sets and membership functions** If $X$ is a collection of objects denoted generally by $x$, then fuzzy set $A$ in $X$ is defined as a set of ordered pairs:

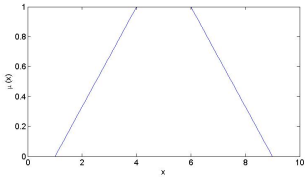
$$A = \{(x, \mu_A(x)) \,|x \,\epsilon\, X\}, \tag{3.9}$$

where $\mu_A(x)$ is the membership function (MF) for the fuzzy set $A$. The membership function maps each element of $X$ to a membership grade between 0 and 1.

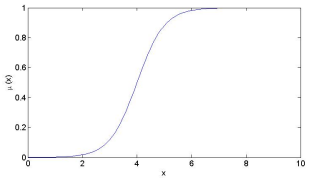
There are a number of common MFs as shown in Figure 3.8:



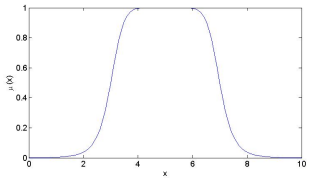
**(a)** Triangular MF.



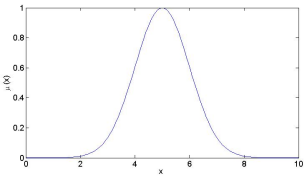
**(b)** Trapezoidal MF.



**(c)** Sigmoidal MF.



**(d)** Generalized bell MF.



**(e)** Gaussian MF.

**Figure 3.8:** Examples of the most widely-used membership functions.

- Triangular MFs are defined by three parameters$\{a, b, c\}$, the corners of the underlying MF (see Fig. 3.8a):

$$triangle\,(x; a, b, c) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases}. \tag{3.10}$$

- Trapezoidal MFs are defined by four parameters $\{a, b, c, d\}$, the corners of the underlying MF (see Fig. 3.8b):

$$trapezoid\,(x; a, b, c, d) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \frac{d-x}{d-c}, & c \leq x \leq d \\ 0, & d \leq x \end{cases}. \tag{3.11}$$

- Sigmoidal MFs are defined by two parameters$\{a, c\}$, where $a$ adjusts the slope at the crossover point $x = c$. This function changes between the two asymptotic values 0 and 1 (see Fig. 3.8c):

$$sigmoid\,(x; a, c) = \frac{1}{1 + e^{-a(x-c)}}. \tag{3.12}$$

- Bell MFs are defined by three parameters $\{a, b, c\}$, $c$ being the center, $a$ the width, and $b$ the slope (see Fig. 3.8d):

$$bell\,(x; a, b, c) = \frac{1}{1 + |\frac{x-c}{a}|^{2b}}. \tag{3.13}$$

- Gaussian MFs are defined by two parameters $\{c, \sigma\}$, being $c$ the centre, and $\sigma$ the width (see Fig. 3.8e):

$$gaussian\,(x, c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}. \tag{3.14}$$

**Linguistic variables** Linguistic variables are those variables whose values are not numbers but words or sentences in a natural or artificial language [149]. The motivation for the use of words or sentences rather than numbers is that linguistic characterisations are, in general, less specific than numerical ones. For example, "speed" is interpreted as a linguistic variable, which can take the values as "slow", "fast", "very fast", and so on.

A linguistic variable is characterised by a quintuple ($x$, $T(x)$, $X$, $G$, $\mu$) where $x$ is the name of the variable; $T(x)$ is the term set of $x$ (that is, the set of its linguistic values); $X$ is the universe of discourse; $G$ is a syntactic rule that generates the terms in $T(x)$; and $M$ is a semantic rule which associates terms in $T(x)$ to fuzzy sets in $X$.

In the previous example, if "*speed*" is interpreted as a linguistic variable, then its term set is $T\,(speed)$ could be:

$$\begin{aligned} T\,(speed) = \{&slow,\ fast,\ very\,slow,\ not\,very\,fast,\ ... \\ &too\,fast,\ not\,slow,\ not\,very\,slow\,and\,not\,very\,fast,\ ...\}, \end{aligned} \tag{3.15}$$

where each term in $T\,(speed)$ is characterized by a fuzzy set of a universe of discourse $X = [0,\ 120]$, as shown in Figure 3.9. In the expression "speed is fast", the linguistic value "fast" is applied to the linguistic variable "speed". On the other hand, in the expression "speed= 85", speed is interpreted as a numerical variable, assigning a numerical value of 85. The syntactic rule refers to the way the linguistic values in the term set $T\,(speed)$ are generated. The semantic rule defines the membership function of each linguistic value of the term set, $M$. The five MFs for defining the linguistic variable *"speed"* are shown in Figure 3.9.

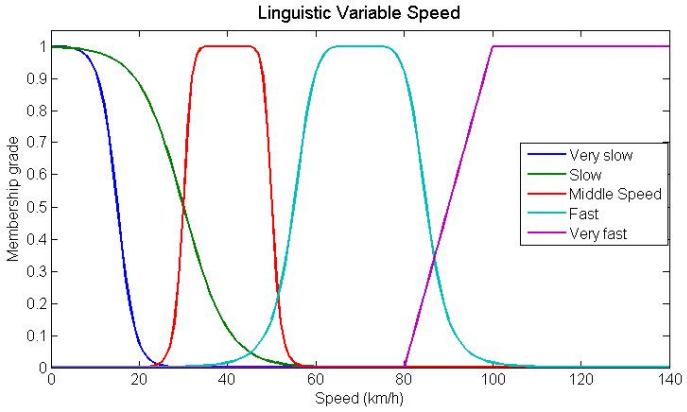



**Figure 3.9:** Typical membership functions of the term set $T\,(speed)$.

**Fuzzy rules** A linguistic "IF-THEN" fuzzy rule is an expression of the form:

$$IF\ \ x_1\ is\ A\ AND\ x_2\ is\ B\ \ THEN\ y\ is\ C, \tag{3.16}$$

where $A$ and $B$ are linguistic values defined by fuzzy sets. Typically, "$x_1\ is\ A\ AND\ x_2\ is\ B$" is known as the antecedent or premise, while "$y\ is\ C$" is the consequence or conclusion. Several example of fuzzy rules are:

- IF speed is "very fast", THEN driving is "dangerous".
- IF temperature is "cold" THEN heater is "high".
- IF temperature is "high" AND humidity is "high" THEN room is "hot".

### 3.4.3 Fuzzy inference systems

Fuzzy inference systems (FISs) are based on the concepts of fuzzy set theory. Fuzzy inference is the process of formulating mapping from a given input to an output using fuzzy logic. There are two main types of FISs [150]:

- Mamdani inference systems: These are composed of a rule base and inference mechanism, where the rules are of the IF-THEN type:

$$R_j:\ IF\ x_1\ is\ A_{1j}(x_1)\ AND\ x_2\ is\ A_{2j}(x_2)\ AND\ x_n\ is\ A_{nj}(x_n)\ THEN\ y\ is\ B_j, \tag{3.17}$$

  where $R_j$ is the $j-$th rule $(1 \leq j \leq m)$, $x_i\,(1 \leq i \leq n)$ are input variables, $y$ is the output, $A_{ij}\,(x_i)$ are linguistic labels, each associated with a membership function $\mu_{ij}\,(x_i)$, and $B_j\,(y)$ are linguistic labels, each associated with a membership function $\mu_j\,(y)$.

- Sugeno inference systems: The rules are of the type [79]:

$$R_j : \; IF \; x_1 \; is \; A_{1j}(x_1) \; AND \; x_2 \; is \; A_{2j}(x_2) \; AND \; x_n \; is \; A_{nj}(x_n)$$
$$THEN \; y = p_{1j}x_1 + ... + p_{nj}x_n + p_{0j}. \quad (3.18)$$

  In zero-order Sugeno inference systems $y = p_{0j}$, i.e., $p_{1j}, ..., p_{nj} = 0$, and in first order Sugeno inference systems, some $p_{ij}$ have non-zero values. The main difference is that the output in Sugeno inference systems is a crisp value and in Mamdani inference systems it is a fuzzy value.

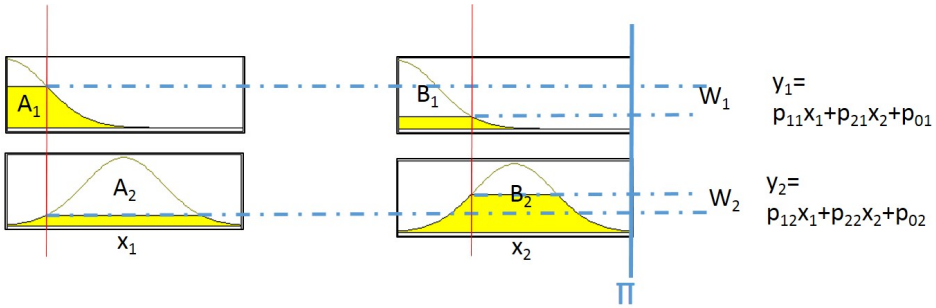



**Figure 3.10:** A two input first-order Sugeno fuzzy model with two rules.

Figure 3.10 represents the reasoning mechanism for first order Sugeno-type FIS. The inference procedure used to derive the conclusion for a specific input $x_i$ consists of two main steps. First, the firing strength or weight $w_j$ of each rule is calculated as follows:

$$w_j = \prod_{i=1}^{n} \mu_{ij}\left(x_i\right). \quad (3.19)$$

After that, the overall inference result, $y$, is obtained by means of the weighted average of the consequent:

$$y = \frac{\sum_{j=1}^{m} w_j y_j}{\sum_{j=1}^{m} w_j}. \quad (3.20)$$

Eqs. (3.19) and (3.20) provide a compact representation of the inference model.

The main strengths and limitations of fuzzy systems are shown in Table 3.2.

| Advantages | Disadvantages |
|---|---|
| Linguistic interpretation | Parameters and structures are difficult to tune |
| Capability of learning nonlinear functions | Previous knowledge from the process is necessary |
| Ability to deal with imprecision, vagueness or incomplete information. | No learning mechanism |
| Robustness and fault tolerance | |

**Table 3.2:** Strengths and limitations of fuzzy systems.

## 3.5 Neuro-fuzzy systems

The term neuro-fuzzy systems (NFs) refers to a combination of techniques from neural networks and fuzzy systems [151]. Ever since fuzzy systems were applied industrially, the community has perceived that the development of a fuzzy system with good performance is not an easy task. The problem of finding membership functions and appropriate rules is frequently a tiring process of trial and error. Therefore the idea of applying learning algorithms to fuzzy systems was considered early on. One of the first studies that proposed a combination of neural network learning methods with fuzzy system concepts was published in 1985 [152]. Several other approaches date from the beginning of the 1990s, including those of Jang [153, 154, 155], Lin and Lee in 1991 [156], Berenji in 1992 [157], and Nauck in 1993 [158, 159]. The majority of the first applications were in the field of process control. Gradually, it began to be applied to all areas of knowledge, including data analysis, data classification, imperfection detection, and support to decision-making.

The aim of NF systems is to combine the advantages of both the above-mentioned approaches (ANNs and FISs). Knowledge of the system is expressed as a linguistic fuzzy relationship while neural network learning schemes, capable of learning non-linear mappings of numerical data, are used to train the system. Furthermore, an NF system is capable of extracting fuzzy knowledge from numerical data. In this particular work, the well-known adaptive neuro-fuzzy inference system (ANFIS) algorithm proposed by Jang in 1993 has been used [154].

Figure 3.11 represents the computational ANFIS architecture equivalent to the two input first-order Sugeno-type FIS shown in Figure 3.10. In the first layer, the inputs are fuzzified using membership functions, in this case the Gaussian function. In the second layer, the firing strength is calculated using Eq. (3.19). In layer 3, the firing strengths of the fuzzy rules are normalised. The next layer produces the output for each rule. Finally, in layer 5, the overall output is computed using Eq. (3.20). The topology depicted in Figure 3.11 can be seen as a particular case of ANN.

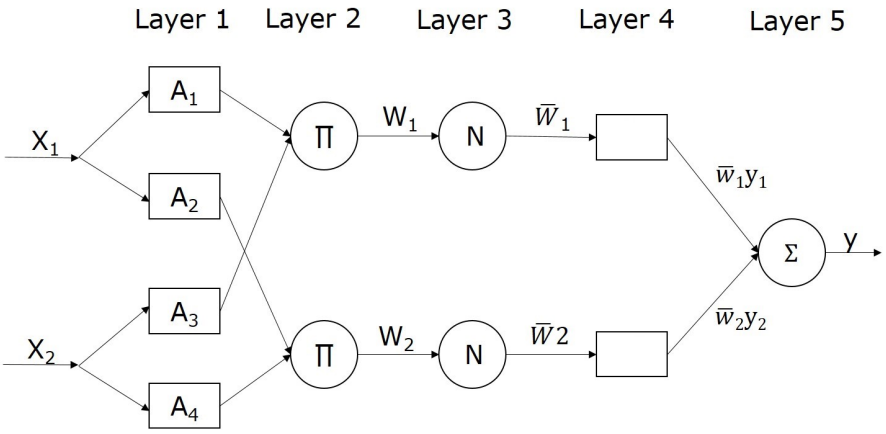



**Figure 3.11:** Scheme of the network-type structure of a fuzzy inference system similar to that of a neural network with two inputs and one output.

The main difference between a standard FIS and an ANFIS is that the parameters associated with the membership functions are fixed values in a FIS, while in the ANFIS algorithm, those parameters change throughout the learning process.

ANFIS is trained with a hybrid learning algorithm comprising a GDM process to find the optimal values for the parameters of the antecedent membership functions, and the least squares estimation (LSE) process to compute the linear consequent parameters of the fuzzy rules. The parameters associated with the membership functions are calculated so as to minimise the error between the calculated output and the target output. An enhanced ANFIS algorithm with structure learning capability was used in this study. The rule base is incremental, meanings that the number of rules increases gradually as long as the model improves its behaviour [127].

## 3.6 Genetic algorithms

Genetic algorithms (GAs) are directed random search techniques that mimic the process of natural selection and evolution. GAs are based on Darwin's theory of "survival of the fitness" as set out in The Origin of Species [160]. As in nature, the fittest species remain intact, while the unfitare are eliminated. In GAs, a population of candidate solutions available to an optimisation problem evolves towards a better solution. Each candidate solution has a set of properties which can be mutated and altered.

### 3.6.1 Brief history of genetic algorithms

In the 1950s and 60s several computer scientists independently studied computer-simulated evolution, including Nils Aall Barricelli [161, 162], Alex Fraser [163], and Hans-Joachim, among others. Bremermann published a series of papers in the 1960s that also adopted a population of solutions for optimisation problems, undergoing recombination, mutation, and selection. Bremermann's research also included the elements of modern genetic algorithms [164, 165].

Genetic algorithms became particularly popular thanks to the work of John Holland in the 1960s [93] and were further developed by Holland, his students and colleagues at the University of Michigan in the 1970s [166, 167]. In contrast with other evolution algorithms, Holland's original goal was not to design algorithms for solving specific problems, but rather to formally study the phenomenon of adaptation as it occurs in nature and develop ways in which the mechanisms of natural adaptation might be imported into computer systems [168]. In 1975, Holland published his book *Adaptation in Natural and Artificial Systems* [94], and interest in GAs in the research community grew, as did their popularity.

As academic interest grew in the early to middle 1980s, genetic algorithms were applied to a broad range of subjects, such as control systems [169], communication networks [170], and modelling [171].

### 3.6.2 Genetic algorithms: basics and genetic operators

In a genetic algorithm, a single candidate solution is called individual, and a population is a group of individuals (see Section 3.7). Each step of the simulated evolution is a generation. The word genome always denotes all the individual's genetic information. The smallest fragment of the genome that can be modified during the evolution is called a gene: a gene can also be seen as the functional unit of inherited characteristics [172]. The objective function (also known as a "fitness function" in genetic algorithms) measures the effectiveness of an individual in solving the problem. Individuals with high fitness values are more likely to propagate their characteristics to the next generation.

A simple GA algorithm, as shown in the flowchart in Figure 3.12, proceeds with an initial population (which may be generated at random or seeded by other heuristics), and selects parents from this population for mating. Crossover and mutation operators are applied on the parents to generate new offspring. And finally, in survivor selection, a replacement strategy decides if offspring will replace parents, and which parents to replace.

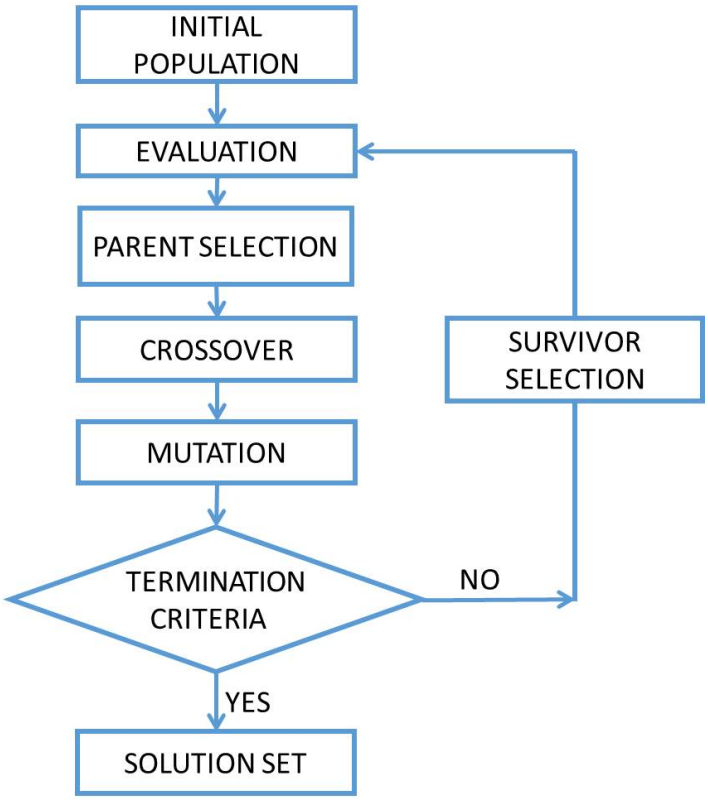



**Figure 3.12:** Flowchart of a simple genetic algorithm.

## Parent selection

Parent Selection is the process of selecting parents which mate and recombine to create offspring for the next generation. Parent selection operators give preference to individuals whose fitted values are better, allowing them to pass on their "genes" to the next generation of the algorithm. The best solutions are determined using the objective function, before being passed to the crossover operator. The selection procedure drives the search to a promising area in a short time period. Also, it is important to maintain a degree of diversity in the population to avoid premature convergence and therefore be able to obtain a globally optimal solution. There are various methods for choosing the best solutions exist, for example, fitness proportionate selection, rank based selection and tournament selection; different methods may choose different solutions as being "best". The selection operator may also simply pass the best solutions from the current generation directly to the next generation without them being mutated; this is known as elitism or elitist selection.

One of the most popular fitness proportionate selection procedures is roulette wheel proposed in 1989 by Goldberg [173]. In roulette-wheel selection, each individual in the population is assigned a roulette wheel slot sized in proportion to its fitness. Given a population of size $K$ of a variable $x$, first the fitness $f_i$ of each individual

$x_i$ in the population is evaluated. Then, the probability of selecting each member of the population $p_i$ is calculated as:

$$p_i = \frac{f_i}{\sum_{i=1}^{K} f_i}. \tag{3.21}$$

A proportion of the wheel is assigned to each of the possible selections based on their fitness value as shows Figure 3.13 for five individuals.

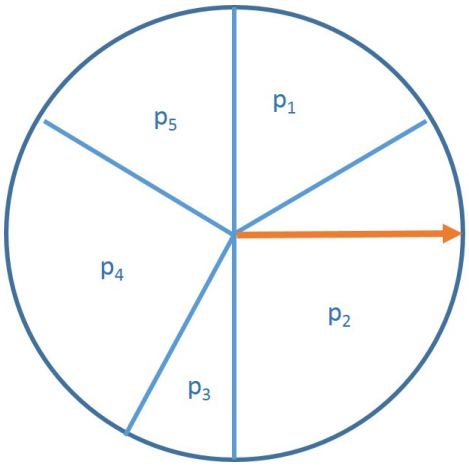
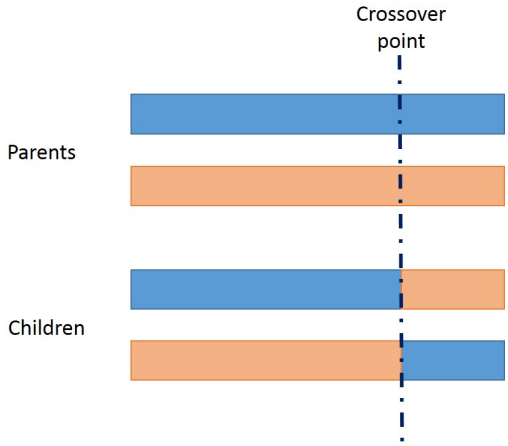


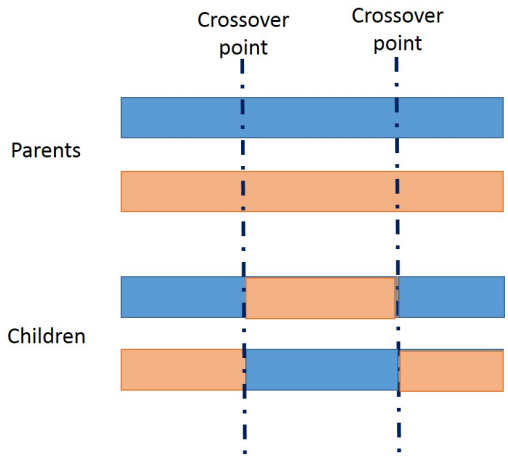

**Figure 3.13:** Roulette wheel selection for five individuals.

After that, a random number is generated $r \epsilon (0, 1]$ (similar to the rotating arrow on a roulette wheel), and selects the individual $x_i$ from this corresponding section just as is shown in Figure 3.13. Then the arrow returns to its original position. This can be done $n$ times until $n$ individual candidates have been selected.

## Crossover




**(a)** Single-point crossover method.




**(b)** Two-points crossover method.

**Figure 3.14:** Examples of crossover methods.

Crossover is the process of taking more than one existing individuals (parents) picked from the current population by the selection operator and producing a new individual solution from them (child). By recombining portions of good solutions, the genetic algorithm is more likely to create a better solution. Crossover techniques include single-point crossovers, two-points crossovers, cycle crossovers, and uniform crossovers. Single-point crossover are the simplest crossover operations. They work by first randomly picking two individuals formed by the selection procedure. The participating individuals (parents) are then split at a randomly chosen crossover point. The tails (the parts beyond the cutting point) are swapped and two further individuals (children) are generated. Two-point crossover is the same procedure,

but the parents are split using two crossover points. Single-point and two-point crossover examples are shown in Figure 3.14

## Mutation

| | |
|---|---|
| Parent | 1**1**001001 |
| Children | 1**0**001001 |

**(a)** Flipping mutation with changes in bold.

| | |
|---|---|
| Parent | 1**0**100**1**10 |
| Children | 1**1**100**0**10 |

**(b)** Interchanging mutation with changes in bold.

| | |
|---|---|
| Parent | 10100**110** |
| Children | 10100**011** |

**(c)** Reversing mutation with changes in bold.

**Figure 3.15:** Examples of mutation methods.

Mutation is a genetic operator used to maintain genetic diversity from one generation of genetic algorithm individuals to the next. It is analogous to biological mutation. Mutation alters one or more gene values in an individual from its initial state. In mutation, the solution may change entirely from the previous solution. The purpose of mutation in GAs is to preserve and introduce diversity to prevent the GA prematurely converging to a local minimum. Hence, GA can reach a better solution by using mutation. Popular mutation techniques include flipping, interchanging, and reversing. Flipping involves changing 0 to 1 and 1 to 0 based on the mutation chromosome generated. Interchanging involves two random positions of the string being chosen and the segments corresponding to those positions being interchanged. In the reversing method a random position is chosen and the section next to that position are reversed, producing the child chromosome. An example of each mutation technique mentioned is shown in Figure 3.15 .

### Survivor selection

Survivor selection is often called replacement. Survivor selection determines which individuals are to be ejected and which are to be kept in the next generation. As in parent selection, it is important keep individuals whose fitted values are better and which maintain a degree of diversity. Survivor selection can be divided into two approaches: aged-based replacement and fitness-based replacement. Aged-based replacement is based on the premise that each individual is allowed in the population for a finite generation where it is permitted to reproduce. After that, it is ejected from the population no matter how good its fitness is. In fitness based selection, the offspring tend to replace the least fit individuals in the population. The selection of the least fit individuals may be done using a variation of any of the selection policies described before – tournament selection, fitness proportionate selection, etc.

In summary, GAs begin by creating a random initial population. Each individual represents a point in a search space and a possible solution. The algorithm then creates a sequence of new populations. For each generation, the algorithm uses the individuals in the current generation to create the next population through a process of evolution via the operations of selection, crossover, and mutation. Over successive generations the GA will converge towards the global (or near global) optimum.

## 3.7 Other computational intelligence techniques

In this section, other widely-known computational intelligence techniques are listed:

- *Simulated annealing* (SA): this is a probabilistic method proposed by Kirkpatrick, Gelatt, and Vecchi in 1983 [174], and by Cerny in 1985 [175]. The method finds the global minimum of an objective function inspired in the annealing process of solids. The annealing process consist of two steps: first an increase in the temperature of the heating bath to a maximum value, at which the solid melts, and a subsequent slow decrease in the temperature until the particles arrange themselves into the ground state of the solid. The heat causes the atoms to become unstuck from their initial positions (a local minimum of internal energy) and wander randomly through states of higher energy, until they reach a configuration of absolute minimum energy (ground state). The cooling process should be slow, and enough time needs to be spent at each temperature to give the atoms sufficient chance of finding configurations of lower internal energy. If the temperature is not lowered slowly and enough time is not spent at each temperature, the process can get trapped in a state of local minimum for the internal energy.
- *Evolutionary algorithms* (EAs): evolutionary algorithms are a generic population-based optimisation set of algorithms. They are search methods inspired by

natural biological evolution. EAs operate on a population of potential solutions applying the principle of "survival of the fittest" to produce ever better approximations to a solution. The current theory is the sum of several concepts: evolution and natural selection were introduced almost concurrently and independently by Charles Darwin and Alfred Russel Wallace in 19$^{th}$ century [160]. In 1950, the great computer scientist Alan Turing was probably the first to point out the similarities between the processes of learning and evolution [176]. Near the end of the same decade, inspiring ideas in that direction began to appear. However, most researchers agree that it was truly born in the 1960s and 70s, with the appearance of three independent research lines: *genetic algorithms* [94], *evolutionary strategies* (proposed in Germany by Rechenberg and Schwefel [177]), and *evolutionary programming* (proposed by Fogel in 1960 [178]). A fourth paradigm, *genetic programming*, created by John Koza [179] and which appeared in the 1990s, must be also considered both for its novelty and its similarity to the aforementioned ideas. *Genetic algorithms*, explained above in Section 3.6 and used in this work, are probably the most popular technique in evolutionary algorithms.

Two other well-known methods in evolutionary strategies, and explained below, are *particle swarm simulation and ant colony optimisation.*

- *Particle swarm simulation* (PSO): this method was introduced in 1995 by Kennedy and Eberhart [180]. PSO mimics the social behaviour of a flock of birds or school of fish. Typically, a group of animals that has no leaders will find food randomly, following the member of the group closest to a food source (potential solution). Groups achieve the best outcome simultaneously, through communication between the members who already have a better situation. An animal which has better conditions will inform its group and the others will simultaneously move to that place. This happens repeatedly until optimal conditions are achieved or a food source discovered. The way a PSO algorithm finds the optimal values follows the example of these animal societies.

  Particle swarm optimisation consists of a swarm of particles, where each particle represent a potential solution in the search space of the objective function, and each evaluates the objective function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best (best-fitness) locations with those of one or more other members of the swarm, with random perturbations. The next iteration takes place after all the particles have been moved. PSO shares many similarities with evolutionary computation techniques such as GAs. As in GA, the PSO approach begin with a randomly generated set of solutions called the initial population. An optimum solution is then sought by updating generations.

- *Ant colony optimisation* (ACO): emulates the food searching behaviour of ants. It was developed by Dorigo in 1992 [181]. At first, the ants wander around

randomly. When an ant finds a food source, it walks back to the colony leaving "markers" (pheromones) that show the path has food. When other ants come across the markers, they are likely to follow the path with a certain probability. If they do, they then populate the path with their own markers as they bring the food back. As more ants find the path, it gets stronger until there are a couple streams of ants travelling to various food sources near the colony. Because the ants drop pheromones every time they collect food, shorter paths are more likely to be stronger, hence optimising the "solution". In the meantime, some ants still randomly scout for closer food sources. Once the food source is depleted, the route is no longer populated with pheromones and slowly decays [182].

- *Support vector machines* (SVM): These methods were first introduced by Vapnik in the 1960s and received increasing attention in the 1990s [183, 184]. A support vector machine constructs a hyperplane or set of hyperplanes in high- or infinite-dimensional space, and can be applied not only to classification problems but also to the case of regression. Intuitively, good separation is achieved by the hyperplane furthest from the nearest training-data point of any class (the so-called functional margin), since in general the larger the margin the lower the classifiers´s generalisation error, i.e., SVMs find the hyperplane that maximises the margin and minimises misclassifications.
- *Hybrid approaches*: these use more than one CI technique. The goal is to combine different approaches to benefit from the advantages of each approach, like in the case of NFs, as explained in Section 3.5. Many hybrid approaches have emerged in the literature in recent decades [185]. Genetic fuzzy systems (GFSs) hybrid approaches are widely used. A GFS is basically a Fuzzy System augmented by a learning process based on a Genetic Algorithms [186, 187]. Other well-known hybrid approaches use evolutionary algorithms to train artificial neural networks, i.e., neuro-evolutionary systems (NEs) [188, 189]. Fuzzy PSO (FPSO), uses FIS and PSO together in different ways: PSO is used to determine the optimal fuzzy parameters of FIS [190, 191], or PSO particles are selected using a fuzzy variable [192]. Some hybrid approaches use even more than two methods, such as FISs, ANN, and GAs together [193], or ANNs, SVMs and GAs [194].

# Chapter 4: CHP systems modelling

## 4.1 Overview

This chapter describes the different CI approaches used to model the whole CHP plant and the slurry drying process. To facilitate the modelling, a data cleaning process was first applied to obtain a suitable dataset. Subsequently, the plant was dealt with as a set of different systems, and the input-output variables for each system were selected.

After describing this process of obtaining a useable dataset and selecting the input-output variables for each system, the modelling of the CHP plant and the slurry drying process is explained using, firstly, artificial neural networks trained with back-propagation (ANN-BP), and then adaptive neuro-fuzzy inference systems (ANFIS). After analysing the results, ANN-BP was selected as being more appropriate than ANFIS for this modelling problem. Next, to overcome certain issues that ANN-BP has, such as overfitting or local minima, a new training algorithm for ANNs was used: extreme learning machines (ELM). After this, the modelling of the effective electrical efficiency of the plant using ELM is described.

Finally, a new hybrid feature selection method, combining a clustering filter with ELM as the wrapper, is proposed with the aim of verifying the suitability of the initial feature selection. The method was applied to a particular system: the steam turbine.

## 4.2 Data cleaning process and variable selection

Data cleaning is the process of examining data to detect potential errors, missing data, outliers or unusual values, and other inconsistencies with the aim of cleaning and transforming the errors or problems that are found [195]. Used mainly in databases, the term refers to identifying issues such as incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replacing, modifying, or deleting this dirty, or coarse data. The cleaning process was performed using Rapid Miner® and Weka® software.

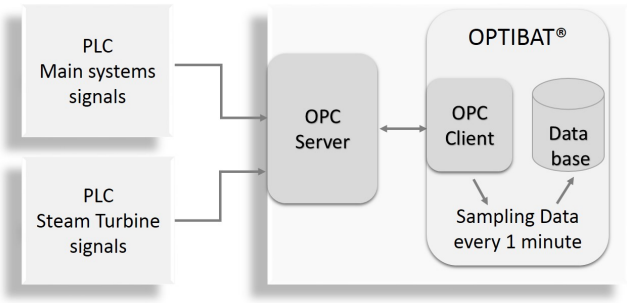



**Figure 4.1:** Diagram of the system architecture for data collection from the plant.

As explained in Chapter 2, the database used in this work contains real-life data from the CHP process and slurry drying treatment. This data was provided by two PLCs (programmable logic controllers) connected to an OPC (open platform communication) server. This server was linked to an OPTIBAT® tool with client server software that obtained data each minute and stored this in a database (see Figure 4.1). The data collected contains variables from the four engines, the refrigeration circuits, the exhaust steam boiler, the steam turbine, and the slurry drying process, with a total of 210 variables. The database obtained in this way contains information gathered over a one-year period: from December 2012 to November 2013. The lower heating value (LHV) of the natural gas used to feed the engines was added to the dataset with a sample time of one value per day. This variable was provided by the natural gas supplying company, *Enagas* [196]. In addition, the ambient temperature and humidity were also added, with sample times of one hour. These two values were provided by *Spain's State Meteorological Agency (AEMET) of Spain* [197]. Hence, the entire initial dataset contained 213 variables.

The following subsections describe the methods used in this study for data visualisation and variable smoothing. Also described is the variable selection for each system in the CHP process. The names and meanings of all the variables involved in the modelling of the CHP plant and the slurry drying process are shown in Appendix I.

### 4.2.1 Data visualisation

Data visualisation is an easy and useful method for obtaining information from the process and the variables. To do this, two different approaches can be used: single-variable graphics and multi-variable graphics.

Single-variable representations as histograms, or box plots, among other possibilities, involve only one variable. Multi-variable representations are graphic methods that involve two or more variables; the most common methods include scatter plots, time series, and multiple time series, among others. The main graphic methods used to represent variables in this thesis are explained briefly below.

## Time series

A time series can be defined as a sequence of data, typically consisting of successive measurements collected over a time interval. Initially, the time series of each variable must be represented with the objective of visually highlighting useful information, such as the way data is distributed, the manner in which one quantity varies with another, or if the variables have unusual or missing values.

Non-informative variables, i.e., those that remain constant, were removed from the original dataset (see Figure 4.2). From that original dataset including 213 variables, a total of 13 constant variables were removed, resulting in a dataset comprising 200 variables.

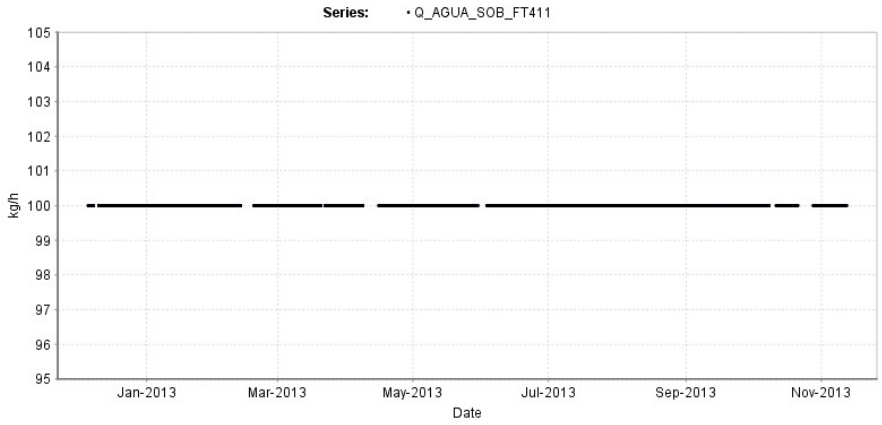



**Figure 4.2:** Example of a constant variable removed from the original dataset.

When dealing with time series, it is important to detect whether or not there are missing values. Missing values arise when no data value is stored for a variable in an observation [198]. Depending on the objective of the data cleaning process, it may be decided to leave missing data unmodified, or replace this with other values (for example the mean, median or mode because these are arithmetic measurements of central tendency). In this work missing values were discarded for the modelling and optimisation procedure, removing this time period from the original dataset. As can be seen in Figure 4.2 and Figure 4.3, both variables have time periods with missing data.

Another important issue is the detection of outliers. An outlier can be defined as a data point that is distinctly separate from the rest of the data [199]. More formally, an outlier is a data point that does not follow the trend of the other points. In Figure 4.3, for the time series of cooling water temperature for the mixture in engine-A (variable $T_{Mixt\_EngA}$ ), it can be seen that, on the 12$^{th}$ February, there are two points with values close to 60 $^{o}$C. These two points are clearly outliers because their values are very different from the usual trend of the time series, whose values range between 40 and 45 $^{o}$C. As with the missing values, any data points considered to be outliers were removed from the original dataset.

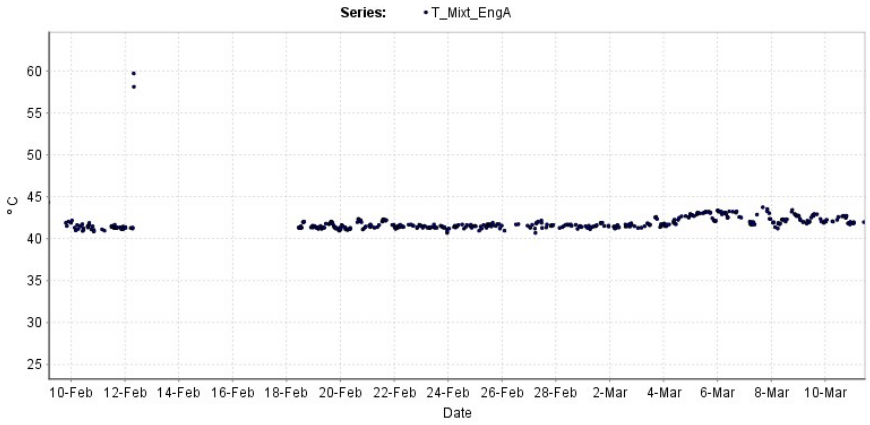



**Figure 4.3:** Time series for the variable $T_{Mixt\_EngA}$.

In summary, after an exhaustive study of all time series variables, the constant variables, missing values and outliers were all removed from the original dataset.

## Scatter plots

A scatter plot is a graphic representation of bivariate data as a set of points in a plane that has Cartesian coordinates equal to the corresponding values of the two variables. Scatter plots show the relationship between two sets of data. Figure 4.4 represents the relationship between the power generated in the steam turbine ($POW_{ST}$) and the steam flow feeding that turbine ($F_{Steam}$). As can be seen, there is a clear positive association between the two variables, which is consistent with knowledge of the steam turbine. If the turbine is fed with a higher flow of steam, that steam generates more power. This allows input variables that correlate well with the output to be selected.

### 4.2.2 Variable smoothing

In a signal or variable, noise can be defined as unpredictable variations in the measured signal from sample to sample. From the time series of the CHP plant variables, it can be concluded that most of these are affected by noise. A successful noise removal solution, or one which at least reduces its influence, is smoothing of the variables [200]. In this work two different ways to smooth signals have been used: *exponential smoothing* and *moving average*.

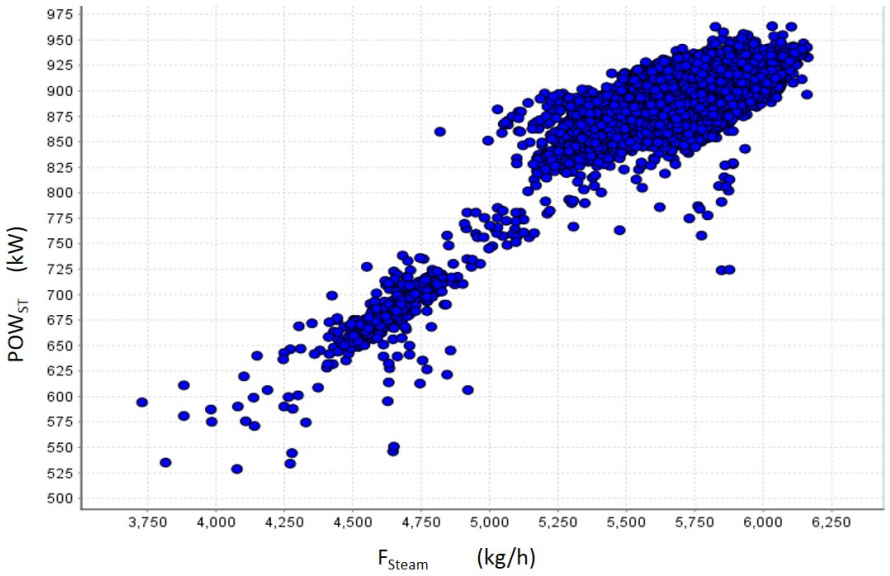



**Figure 4.4:** Scatter plot of two variables related to the steam turbine in the CHP plant.

## Exponential smoothing

Given any number of samples $\{x_i\}_{i=1}^{K}$, the exponential smoothing value $s_i$ in a particular measurement of the dataset is given by the formula:

$$s_i = \alpha\, x_i + (1 - \alpha)\, s_{i-1} \qquad 0 < \alpha \leq 1, \tag{4.1}$$

where $\alpha$ is the smoothing factor. In other words, the smoothed value $s_i$ is a simple weighted average of the current observation $x_i$ and the previous smoothed value $s_{i-1}$. The smoothing factor $\alpha$ controls the balance between new and old information: as $\alpha$ approaches 1, the smoothing retains only the current data point (i.e., the series is not smoothed at all); as $\alpha$ approaches 0, smoothing retains only the smoothed past (i.e., the curve is totally flat).

Figure 4.5 shows the exponential smoothing for the superheated water temperature variable ($T_{H2O\_SH}$) with the smoothing factor $\alpha$ being 0.1. As can be seen, the smoothed signal (blue line) follows the trend of the original variable but has fewer peaks.

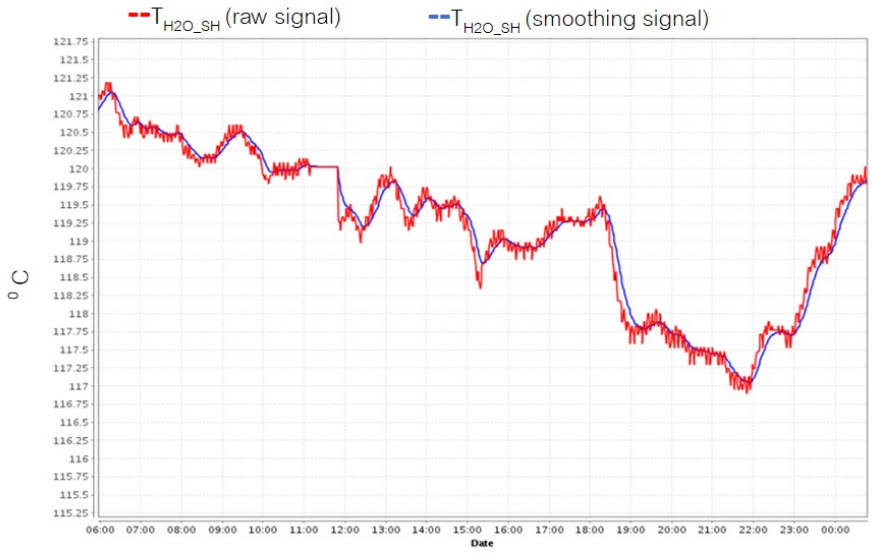



**Figure 4.5:** Exponential smoothing for a temperature variable.

## Moving average

Given any number of samples $\{x_i\}_{i=1}^{K}$, the moving average is the mean of the previous $n$ data (sometimes called window size):

$$s_i = \frac{1}{n} \sum_{j=i}^{i+n-1} x_j. \tag{4.2}$$

The moving average with a window of 200 previous samples for the flow fed into the slurry drying evaporator ($F_{Ev}$) is shown in Figure 4.6. This specific variable has a large number fluctuations and peaks, however, the moving average variable still follows the main trend of the raw data quite well.

Once the constant variables had been removed in the dataset, different exponential smoothing and moving averages were obtained for all the variables. The most appropriate smoothing for each variable was selected. The selected parameters for the exponential smoothing and moving average for the variables involved in the modelling of the CHP plant and the slurry drying process are shown in the Appendix I.

### 4.2.3 Variable Selection

Due to the complexity of the whole plant and systems involved, the modelling of the CHP plant was performed using the different separated systems listed in Table 4.1.

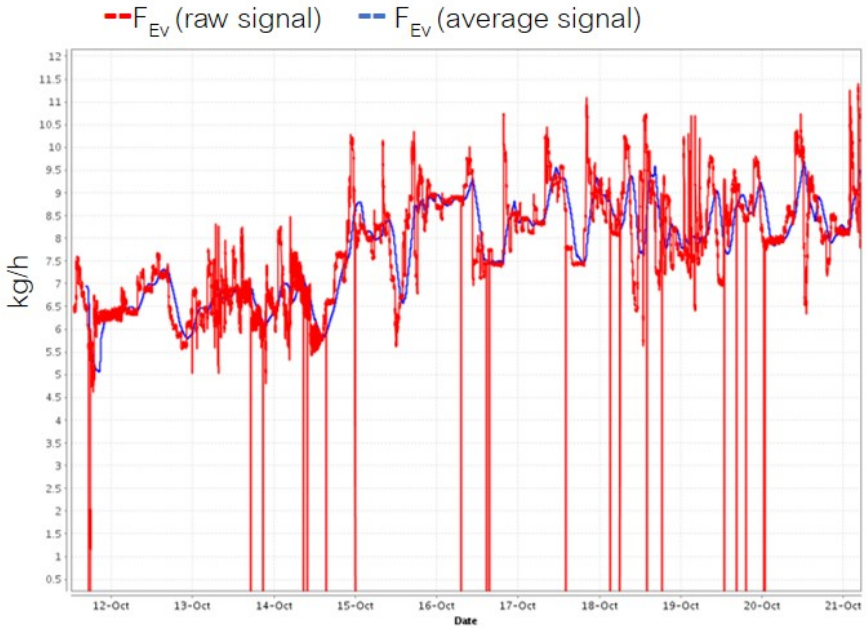



**Figure 4.6:** Moving average for the evaporator feed flow.

For each system, the target variable (output) was selected according to knowledge of the process and the available variables in the database. The input variables for each system were then obtained by selecting those with the greatest influence on each system output. To this end, a combination of mathematical techniques and in-depth knowledge of the process and systems were employed. Some of the mathematical techniques used were:

- Covariance matrix: this is a matrix that gives the covariance between all pairs of variables. Covariance is a measure of the extent to which corresponding elements from two sets of ordered data move in the same direction [201]. The covariance matrix is symmetrical because the covariance between $x$ and $y$ is the same as the covariance between $y$ and $x$. The covariance for two variables $x$ and $y$ with $K$ samples ($1 \leq i \leq K$), can be defined as:

$$\sigma_{xy} = \frac{\sum_{i=1}^{K}(x_i - \bar{x})(y_i - \bar{y})}{K}, \tag{4.3}$$

  where $\bar{x} = 1/K \sum_{i=1}^{K} x_i$ and $\bar{y} = 1/K \sum_{i=1}^{K} y_i$ are the mean for the variables $x$ and $y$, respectively.

- Correlation matrix: this is a matrix that gives the correlation coefficients between all pairs of variables. Like the covariance matrix, the correlation matrix is symmetrical. One of the most widely used correlation coefficients is the linear correlation coefficient, also known as the Pearson correlation coefficient.

The Pearson coefficient, $r_{xy}$, for two variables $x$ and $y$ is:

$$r_{xy} = \frac{1}{K} \frac{\sum_{i=1}^{K}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\frac{1}{K}\sum_{i=1}^{K}(x_i - \bar{x})^2}\sqrt{\frac{1}{K}\sum_{i=1}^{K}(y_i - \bar{y})^2}} = \frac{\sigma_{xy}}{s_x s_y}. \tag{4.4}$$

$s_x = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(x_i - \bar{x})^2}$ is the standard deviation for the variable $x$ (and similarly for the variable $y$). Correlation is a scaled version of covariance. The denominator in the expression of the correlation coefficient amounts to a rescaling of the values of both variables to a standard interval (unlike covariance). For example, correlation between $\mathrm{POW_{ST}}$ and $\mathrm{F_{Steam}}$ is $r_{xy} = 0.95$ (clear linear correlation is shown in Figure 4.4). Due to this clear correlation, $\mathrm{F_{Steam}}$ is one of the inputs selected for the steam turbine.

- Mutual information: this tries to quantify the amount of information shared between two random variables, $x$ and $y$. Mutual Information, $I$, measures how much the knowledge of one of these variables reduces uncertainty about the other. If $x$ and $y$ are independent, then the mutual information is 0. The mutual information for two continuous variables is:

$$I(X;Y) = \sum_{y \epsilon Y} \sum_{x \epsilon X} p(x,y)\, log\left(\frac{p(x,y)}{p(x)p(y)}\right), \tag{4.5}$$

where $p(x,y)$ is the joint probability distribution function of $x$ and $y$, and $p(x)$ and $p(y)$ are the marginal probability density function of $x$ and $y$ respectively. For example, the mutual information value between the air intake temperature engine-A bank1 (variable $\mathrm{T_{B1\_A}}$) and the air output temperature engine-A bank1 ($\mathrm{T_{B1Out\_A}}$) is very high. This means that the information shared between the two variables is very high. For this reason only one of these, $\mathrm{T_{B1\_A}}$, was selected as an input for the engine system. Similarly, for bank 2 of the engine-A, only the intake temperature $\mathrm{T_{B2\_A}}$ was selected as an input. This was also taken into account for the remaining three engines, i.e., only the intake air temperature for the banks was selected as an input variable for the engine systems.

- Principal component Analysis (PCA): this is a mathematical procedure that uses an orthogonal transformation to convert a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. The first principal component obtained using the CHP plant dataset assigns the greatest importance to $\mathrm{POW_{ST}}$ , $\mathrm{F_{Steam}}$ , and the water flow to feed the steam generator ($\mathrm{F_{H2O}}$).

Taking into account all the above mentioned techniques, the knowledge of the CHP process, and certain complementary experiments, the set of input-output variables for each model was selected. Table 4.1 shows these variables.

**Table 4.1:** CHP plant systems and their corresponding variables.

| CHP SYSTEM | MODEL INPUTS | OUTPUT |
|---|---|---|
| Cooling Engine-A | $T_{H2O\_Ex}$ $T_{H2O\_TOW}$ $POW_A$ | $T_{Mixt\_EngA}$ |
| Cooling Engine-B | $T_{H2O\_Ex}$ $T_{H2O\_TOW}$ $POW_B$ | $T_{Mixt\_EngB}$ |
| Cooling Engine-C | $T_{H2O\_Ex}$ $T_{H2O\_TOW}$ $POW_C$ | $T_{Mixt\_EngC}$ |
| Cooling Engine-D | $T_{H2O\_Ex}$ $T_{H2O\_TOW}$ $POW_D$ | $T_{Mixt\_EngD}$ |
| Engine-A | $T_{B1\_A}$ $T_{B2\_A}$ $T_{Amb}$ $H_{Amb}$ LHV $T_{Bank1\_A}$ $T_{Bank2\_A}$ $T_{Mixt\_EngA}$ $POW_A$ $DIV_A$ | $F_{Gas\_A}$ |
| Engine-B | $T_{B1\_B}$ $T_{B2\_B}$ $T_{Amb}$ $H_{Amb}$ LHV $T_{Bank1\_B}$ $T_{Bank2\_B}$ $T_{Mixt\_EngB}$ $POW_B$ $DIV_B$ | $F_{Gas\_B}$ |
| Engine-C | $T_{B1\_C}$ $T_{B2\_C}$ $T_{Amb}$ $H_{Amb}$ LHV $T_{Bank1\_C}$ $T_{Bank2\_C}$ $T_{Mixt\_EngC}$ $POW_C$ $DIV_C$ | $F_{Gas\_C}$ |
| Engine-D | $T_{B1\_D}$ $T_{B2\_D}$ $T_{Amb}$ $H_{Amb}$ LHV $T_{Bank1\_D}$ $T_{Bank2\_D}$ $T_{Mixt\_EngD}$ $POW_D$ $DIV_D$ | $F_{Gas\_D}$ |
| Exhaust Steam Boiler | $P_{StGen}$ $F_{FlueGas}$ | $F_{Steam}$ |
| Steam Turbine Condenser | $T_{H2O\_Tow}$ $T_{ST\_Cond}$ | $P_{Cond}$ |
| Steam Turbine | $P_{StGen}$ $F_{Steam}$ $P_{Cond}$ | $POW_{ST}$ |
| Slurry drying process | $P_{Ev}$ $T_{H20\_SH}$ $T_{H2O\_TH}$ $T_{H2O\_Ex}$ $F_{Cond}$ | $F_{Ev}$ |

## 4.3 Modelling the CHP systems using ANN-BP/ANFIS

In this section, the modelling of the CHP plant and the slurry drying process using ANN-BP and ANFIS is explained.

The multilayer perceptron (MLP) has been selected as a particular neural network model. It has become the most popular architecture for real-world applications, mainly due to the development of the back-propagation learning rule, which is an effective and practical learning algorithm [202]. Moreover, MLP is a universal approximator and hence, it is capable of approximating any measurable function to any desired degree of accuracy [203]. While neural networks are good at approximating any measurable function, they are not good at explaining the behavior of the modelled systems. On the other hand, fuzzy systems which can reason with

imprecise information, are good at explaining the models but they cannot automatically acquire the rules they use to make those decisions. These limitations have been a central driving force behind the creation of intelligent hybrid systems where two or more techniques are combined in a manner that overcomes the limitations of individual techniques [204], such as the ANFIS used in this work.

To model the CHP plant, first the dataset was divided into two subdatasets: a training set with data from February, April, June, August, October and December and a testing set for evaluating the predictability of the models what included the remaining months. Furthermore, due to the large number of samples available, the database was resampled every 30 minutes.

The modelling for all the experiments in this approach was carried out using the OPTIBAT® Trainer tool [205].

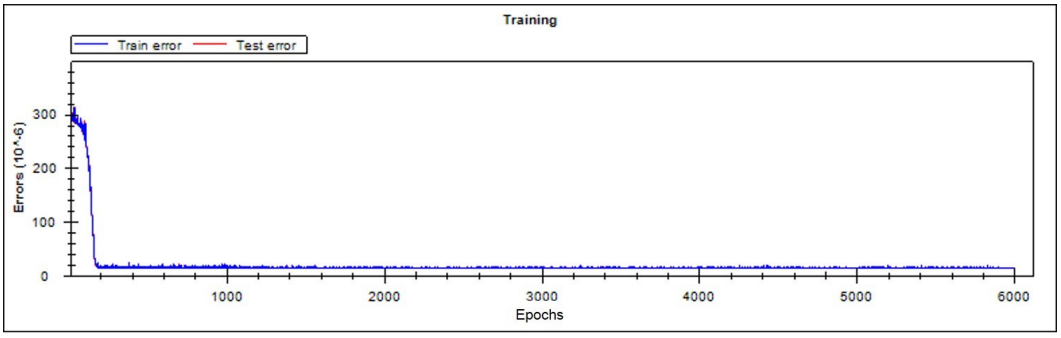



**Figure 4.7:** Training error versus epochs.

### 4.3.1 ANN-BP and ANFIS parameters and structure

An ANN-BP model and an ANFIS model were prepared for each system using the input-output variables shown in Table 4.1. In order to select the structure of the ANN-BP models, some initial tests were carried out using different numbers of hidden layers. It was concluded that the accuracy did not significantly improve when increasing the number of hidden layers. Therefore, the simplest option was selected: single hidden layer. Similarly, some initial tests were conducted to test different numbers of hidden nodes. It was concluded that when the number of hidden nodes is large enough, the accuracy of the model does not change significantly with slight variations in node number. Hence, as a criterion for setting the number of hidden nodes, the common rule of "twice the number of input variables" was finally adopted. Regarding the number of epochs for training the models, it is well known that after a number of epochs, model accuracy does not vary (the error is saturated). This is illustrated in Figure 4.7 for the Cooling A system, where the training error is represented after the weights have been updated using Eqs. 3.4 and 3.5 for each epoch. It can be seen that around epoch number 200 the error stabilises (or

saturates) because a minimum (local or global) is reached. Therefore, the number of epochs was set as 2000 *times the number of inputs* for each system. The error used in this approach was the mean absolute error $MAE = \frac{1}{N}\sum_{i=1}^{k}|y_i - y_i^{'}|$.

**Table 4.3:** Model parameters and modelling results.

| SYSTEM | ANN MODEL | | | | ANFIS MODEL | | |
|---|---|---|---|---|---|---|---|
| | Structure * | Epoch | Train Error | *Test Error* | $N_0$ of rules | Train Error | *Test Error* |
| **Cooling Engine-A** | 3/6/1 | 6000 | 0.21 % | *0.23%* | 16 | 0.21% | ***0.22%*** |
| **Cooling Engine-B** | 3/6/1 | 6000 | 0.28 % | *0.26%* | 16 | 0.23% | ***0.23%*** |
| **Cooling Engine-C** | 3/6/1 | 6000 | 0.10% | *0.13%* | 16 | 0.11% | ***0.12%*** |
| **Cooling Engine-D** | 3/6/1 | 6000 | 0.49% | *0.33%* | 16 | 0.48% | ***0.30%*** |
| **Engine-A** | 10/20/1 | 20000 | 0.39% | ***0.42%*** | 22 | 0.51% | *0.54%* |
| **Engine-B** | 10/20/1 | 20000 | 0.41% | ***0.41%*** | 22 | 0.5 % | *0.51%* |
| **Engine-C** | 10/20/1 | 20000 | 0.38% | ***0.42%*** | 22 | 0.49% | *0.49%* |
| **Engine-D** | 10/20/1 | 20000 | 0.38% | ***0.37%*** | 22 | 0.51% | *0.52%* |
| **Recovery Boiler** | 2/4/1 | 4000 | 0.61% | *0.63%* | 12 | 0.61% | ***0.62%*** |
| **ST Condenser** | 2/4/1 | 4000 | 1.01% | ***0.96%*** | 5 | 2.27% | *1.97%* |
| **Steam Turbine** | 3/6/1 | 6000 | 0.67% | ***0.70%*** | 16 | 0.75% | *0.90%* |
| **Slurry Drying Process** | 5/10/1 | 10000 | 2.35% | ***2.52%*** | 21 | 2.88% | *2.84%* |

*ANN structure: input layer neurons/hidden layer neurons/output layer neurons

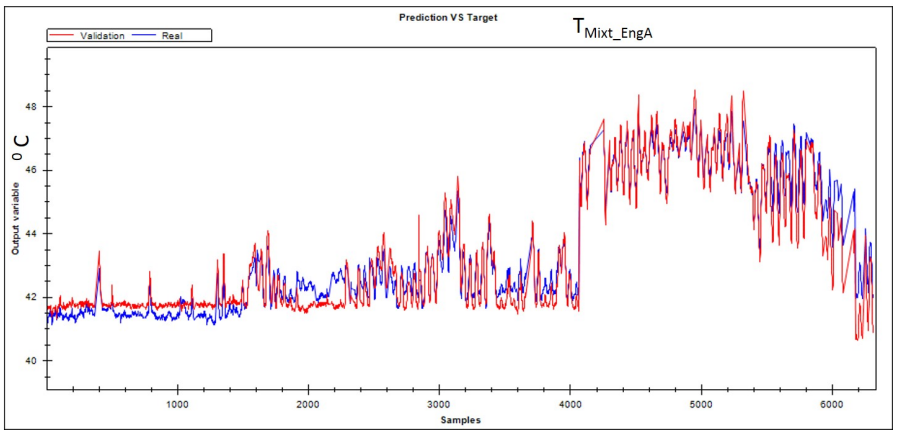
Taking into account everything discussed above, the structure chosen for the ANN-BP models for all the systems was always similar: the number of neurons in the input layer was the same as the number of inputs in the model, the number of neurons in the hidden layer was twice the number of inputs (only one hidden layer), and a single output neuron was used. The stop criteria is the number of epochs.

The ANFIS models, were structured in such a way that the number of rules was increased until the error in the model was stabilised or the maximum training time of 72 hours was reached.

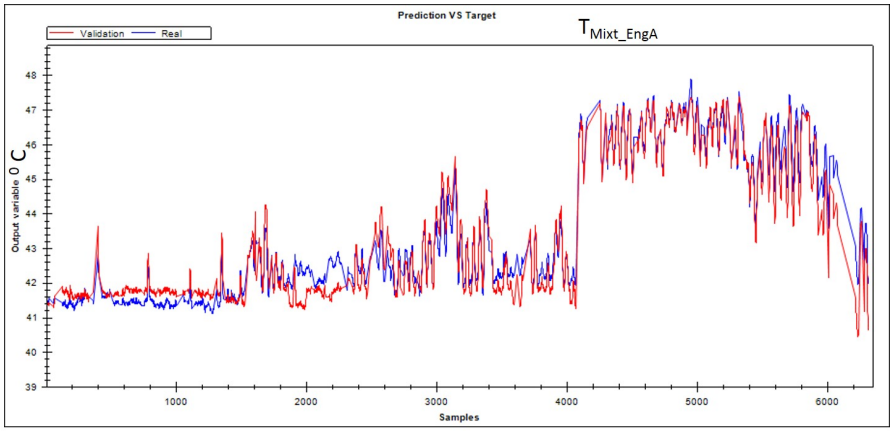
The structure for each model and the number of rules generated for the ANFIS models are shown in Table 4.3. As can be seen, the structures are always more complex in the ANFIS models than the ANN-BP models. One of the main advantages of ANFIS systems, which has already been mentioned in Section 3.5, is the linguistic interpretation. However, in this study, linguistic interpretation is quite difficult because of the large number of rules.

### 4.3.2 Experimental results

The training and testing errors obtained, respectively, using ANN-BP and ANFIS, are shown in Table 4.3 for all systems. The MAE was calculated for each model




**(a)** ANN-BP model, real output vs. predictions.

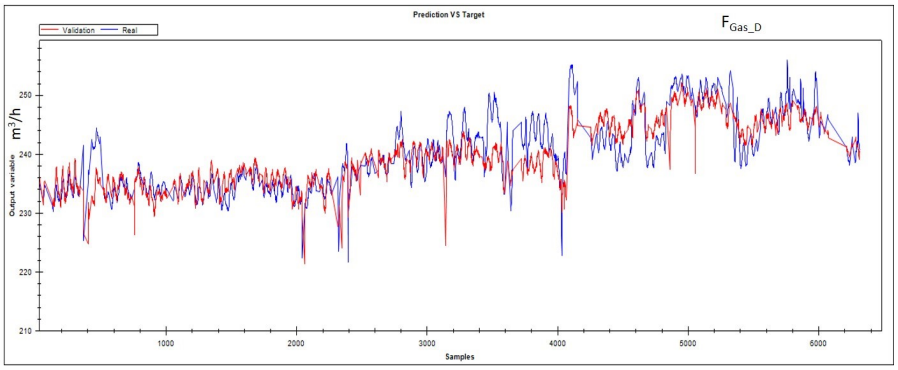



**(b)** ANFIS model, real output vs. predictions.

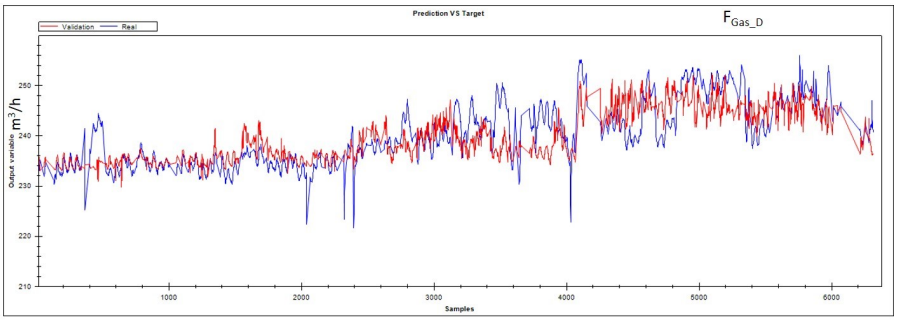
**Figure 4.8:** Graphic results for the ANN and ANFIS models of the cooling engine-A system.

for both the training and the testing dataset. The testing error represents the model's behaviour better than the training error as it contains data that was not seen during the training of the models. For the reason, the testing error is used to evaluate the model's predictive behaviour. We can see that for all the models, the difference between the training error and the testing error was always less than 0.3%. This means that the models were capable of learning the dynamic of the systems and making accurate predictions when dealing with unseen data. The accuracy is similar for both the ANN-BP and ANFIS algorithms.

To graphically illustrate the results of the modelling, Figure 4.8 shows the output of both the ANN-BP model and the ANFIS model for the cooling engine-A system. In this system, the output variable is the cooling water temperature for the mixture in engine-A ($T_{Mixt\_EngA}$). Both figures show the real output in blue and the model's prediction for the test dataset in red. In both graphics, the predicted results are




**(a)** ANN-BP model, real output vs. predictions.




**(b)** ANFIS model, real output vs. predictions.

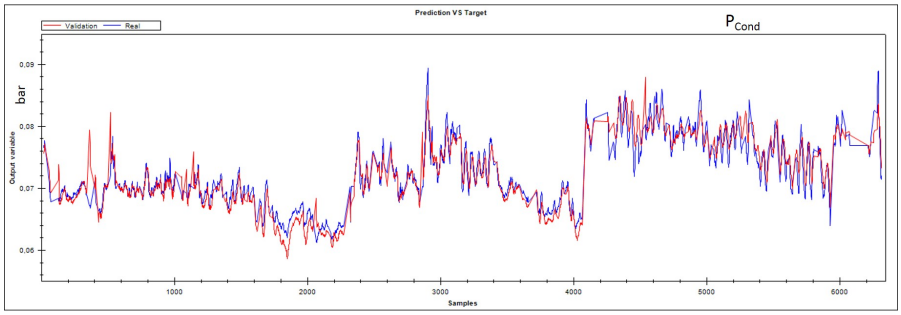
**Figure 4.9:** Graphic results for the ANN-BP and ANFIS models of the engine D system.

very close to the real output therefore revealing the accuracy of the predicted results. The behaviour of the four cooling circuits is analogous. For this reason only one is plotted.

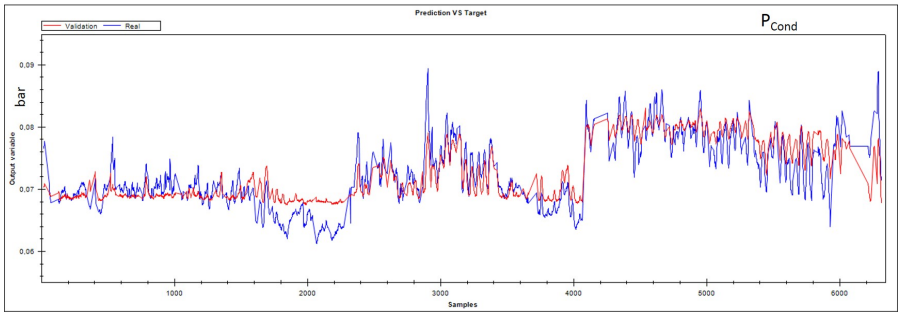
For the engine D system, Figure 4.9 shows both the ANN-BP and ANFIS model predictions for the system output, the natural gas flow feeding engine D ($F_{Gas\_A}$). As can be seen, the predictions for both models follow the main trend, although some peaks are not predicted very accurately. Furthermore, the ANN-BP model is able to predict some areas much better than the ANFIS model, for the example, for samples between 1000 and 2000. This result is consistent with the testing errors, as the ANN-BP testing error (0.37%) is lower than the ANFIS error (0.52%), as can be seen in Table 4.3.

The four engines behave in a similar way so only one is presented.

Figure 4.10 shows the predicted values (red line) and real values (blue line) obtained with both the ANN-BP and ANFIS systems, for the output of the steam turbine condenser system: condenser pressure ($P_{Cond}$). In this case, the best test error 0.96% is obtained with ANN-BP, which is clearly better than the 1.97% provided by ANFIS. Moreover, this difference is the largest for any of the systems, as can be




**(a)** ANN-BP model, real output vs. predictions.




**(b)** ANFIS model, real output vs. predictions.

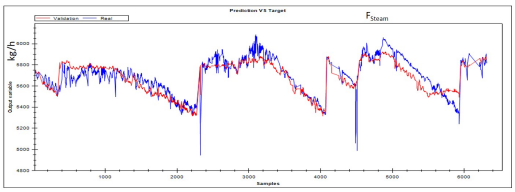
**Figure 4.10:** Graphic results for the ANN and ANFIS models of the ST condenser system.

appreciated in the graphics, where it can be seen that the ANFIS model (Fig. 4.10b) is not able to follow the trend of the system, around sample 2000, as well as the ANN-BP model (Fig. 4.10a).

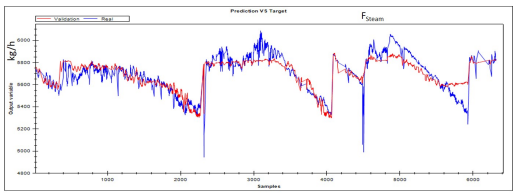
The ANN-BP and ANFIS model simulations for the exhaust steam boiler, the steam turbine and the slurry drying process systems are shown in Figure 4.11. As can be seen, the conclusions for these three systems are similar to those for the above-described systems: all the models follow the main trends of the output variables, although some peaks and ID samples are not very well predicted; for example in Fig. 4.11f between samples 200 and 400, or in the early samples of the steam turbine simulations for both the ANN-BP model (Fig. 4.11c) and ANFIS model (Fig. 4.11d).

The graphic representations and numerical results in Table 4.3, demonstrate that both algorithms were, in general, very accurate. However, when the number of inputs is high, as in the case of the engines, ANN-BP performs better than ANFIS. Even if the number of rules is increased, engine systems are better modelled using ANN-BP. Taking into account the results obtained and the fact that ANFIS has a more complex structure and poor linguistic interpretation ability, it can be concluded that for this modelling problem ANN-BP is a better option than ANFIS.

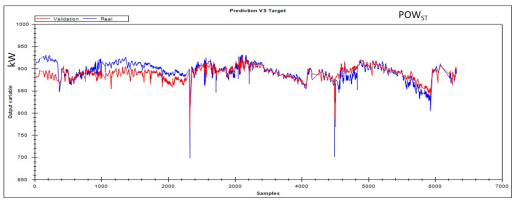
A compressive study of the ANN-BP and ANFIS modelling of the CHP plant can be found in [206].



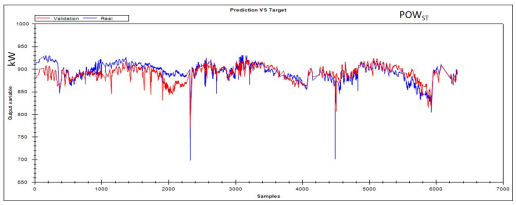
**(a)** ANN-BP model, real output vs. predictions for the exhaust recovery boiler.



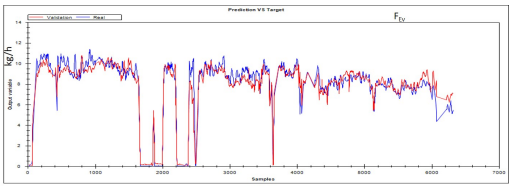
**(b)** ANFIS model, real output vs. predictions for the exhaust recovery boiler.



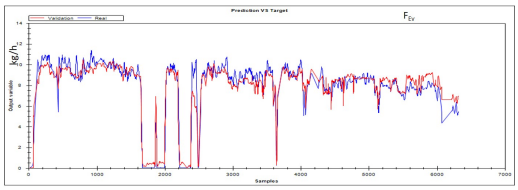
**(c)** ANN-BP model, real output vs. predictions for the steam turbine.



**(d)** ANFIS model, real output vs. predictions for the steam turbine.



**(e)** ANN-BP model, real output vs. predictions for the slurry drying process.



**(f)** ANFIS model, real output vs. predictions for the slurry drying process.

**Figure 4.11:** Graphic results for the ANN-BP and ANFIS models for the exhaust steam boiler, the steam turbine and the slurry drying process systems.

## 4.4 Modelling of the CHP systems using ELM

The conclusion of the previous section is that ANN-BP is a better choice than ANFIS for modelling the CHP plant. Nevertheless, it is important to take into account the fact that ANN-BP has some important disadvantages, such as local minima and the overfitting of training data (Table 3.1). For this reason, this section proposes modelling the CHP plant and slurry drying process using ELM, explained previously in Subsection 3.3.5. ELM provides a robust learning algorithm, free of local minima, with no overfitting problems and is less dependent on human intervention than ANN-BP.

The year-long dataset was resampled every 10 minutes with the aim of analysing the training time for a large number of samples. Thus, a training set and a testing set with almost 19000 data each were obtained.

To model the cogeneration plant, each system was trained with the ELM algorithm using equations from Subsection 3.3.5 and the input-output variables from Table 4.1.

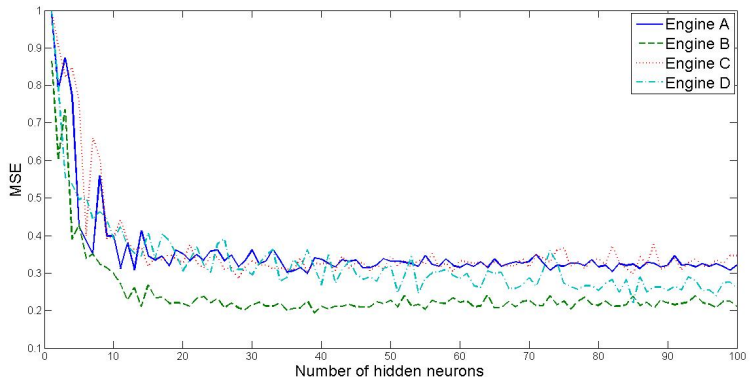
The number of hidden nodes were varied from 1 up to 100 for each system. Finally, the best performance of 10 trials of simulations for each system was selected. The experiments were carried out using the Matlab®tool [207].

For comparison purposes, we have taken into account both the ANN-BP with a single hidden-layer, and a Support Vector Machine (SVM) using a radial basis function kernel. The same topology as in the case of ELM was used for the ANN-BP modelling, but the number of hidden nodes was gradually increased by an interval of 5 up to 100 for comparison purposes. Finally 3000 epochs were selected in each system training. For the SVM the cost parameter $C$ was chosen equal to the range of output values of the training data [208]. The kernel parameter $\gamma$ and the intensive zone $\varepsilon$ values were selected from the most accurate combination of: $\gamma = [2^{-7}, 2^{-6}, ..., 2^{7}]$, and $\varepsilon = [0.1, 0.2, ..., 0.5]$. SVM modelling was carried out using LIBSVM [209]. Table 4.5 compares the overall results among all the models. The table presents the training and testing accuracy (normalised mean square error (MSE) Eq. 3.4), training and testing time, and number of nodes.

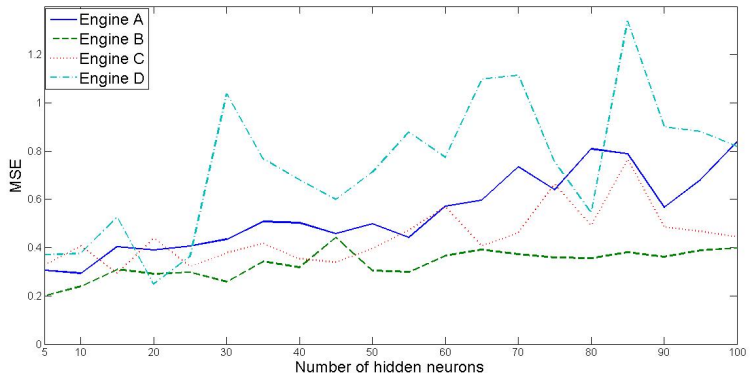
**Table 4.5:** Results and characteristics of the models.

| system | Parameter | ELM | ANN-BP | SVM |
|---|---|---|---|---|
| **Cooling Engine A** | Training MSE | 0.0314 | 0.0465 | 0.0243 |
| | Testing MSE | 0.0504 | 0.0507 | 0.0539 |
| | Training Time (s) | 0.0624 | 14.7109 | 10.4833 |
| | Testing Time (s) | $<10^{-4}$ | 0.0312 | 4.2432 |
| | No. of nodes | 18 | 20 | 7669 |
| **Cooling Engine B** | Training MSE | 0.1530 | 0.2286 | 0.1683 |
| | Testing MSE | 0.1649 | 0.2107 | 0.2139 |
| | Training Time (s) | 0.1649 | 8.2837 | 11.3257 |
| | Testing Time (s) | $<10^{-4}$ | 0.0312 | 5.6472 |
| | No.of nodes | 35 | 10 | 12236 |
| **Cooling Engine C** | Training MSE | 0.2887 | 0.0992 | 0.0673 |
| | Testing MSE | 0.2845 | 0.1313 | 0.1970 |
| | Training Time (s) | 0.1716 | 9.5317 | 12.7141 |
| | Testing Time (s) | $<10^{-4}$ | 0.0312 | 5.1168 |
| | No.of nodes | 10 | 28 | 10434 |
| **Cooling Engine D** | Training MSE | 0.4104 | 0.4324 | 0.4251 |
| | Testing MSE | 0.2911 | 0.2973 | 0.4896 |
| | Training Time (s) | $<10^{-4}$ | 5.6472 | 9.9061 |
| | Testing Time (s) | $<10^{-4}$ | 0.0156 | 1.794 |
| | No. of nodes | 13 | 5 | 10.413 |
| **Engine A** | Training (MSE) | 0.2927 | 0.1897 | 0.2047 |
| | Testing (MSE) | 0.2928 | 0.2933 | 0.2879 |
| | Training Time (s) | 0.1872 | 31.5434 | 18.9073 |

| | | | | |
|---|---|---|---|---|
| | Testing Time (s) | 0.0624 | 0.0312 | 8.2837 |
| | No.of nodes | 38 | 10 | 12860 |
| **Engine B** | Training (MSE) | 0.2133 | 0.2116 | 0.1699 |
| | Testing (MSE) | 0.1934 | 0.2181 | 0.2098 |
| | Training Time (s) | 0.2340 | 9.9373 | 11.1477 |
| | Testing Time (s) | 0.0624 | 0.0312 | 7.9405 |
| | No.of nodes | 39 | 5 | 12456 |
| **Engine C** | Training (MSE) | 0.2887 | 0.2324 | 0.2253 |
| | Testing (MSE) | 0.2845 | 0.2929 | 0.3609 |
| | Training Time (s) | 0.1716 | 12.1681 | 14.1493 |
| | Testing Time (s) | 0.0624 | 0.0468 | 8.9545 |
| | No.of nodes | 28 | 15 | 13357 |
| **Engine D** | Training (MSE) | 0.1595 | 0.2045 | 0.1478 |
| | Testing (MSE) | 0.2210 | 0.2480 | 0.3380 |
| | Training Time (s) | 0.546 | 15.0541 | 12.6921 |
| | Testing Time (s) | 0.1404 | 0.0312 | 7.8781 |
| | No.of nodes | 85 | 20 | 12089 |
| **Exhaust Steam Boiler** | Training MSE | 0.3407 | 0.3687 | 0.2787 |
| | Testing MSE | 0.5157 | 0.6036 | 0.5449 |
| | Training Time (s) | $<10^{-4}$ | 5.0856 | 16.0057 |
| | Testing Time (s) | $<10^{-4}$ | 0.0468 | 5.7876 |
| | No.of nodes | 4 | 5 | 13607 |
| **Steam Turbine Condenser** | Training (MSE) | 0.3728 | 0.2478 | 0.1634 |
| | Testing (MSE) | 0.2979 | 0.2751 | 0.3159 |
| | Training Time (s) | $<10^{-4}$ | 16.1617 | 4.6332 |
| | Testing Time (s) | $<10^{-4}$ | 0.0312 | 1.794 |
| | No.of nodes | 4 | 5 | 2982 |
| **Steam Turbine** | Training (MSE) | 0.1299 | 0.2448 | 0.2484 |
| | Testing (MSE) | 0.2659 | 0.3314 | 0.3024 |
| | Training Time (s) | 0.0624 | 9.6097 | 15.2881 |
| | Testing Time (s) | $<10^{-4}$ | 0.0156 | 6.7704 |
| | No.of nodes | 8 | 10 | 14433 |
| **Slurry drying process** | Training (MSE) | 0.1123 | 0.1434 | 0.0904 |
| | Testing (MSE) | 0.0759 | 0.3645 | 0.1306 |
| | Training Time (s) | 0.3352 | 5.0856 | 13.7749 |
| | Testing Time (s) | $<10^{-4}$ | 0.0468 | 6.1308 |
| | No. of nodes | 11 | 5 | 12105 |




**(a)** ELM performance according to the number of hidden neurons.



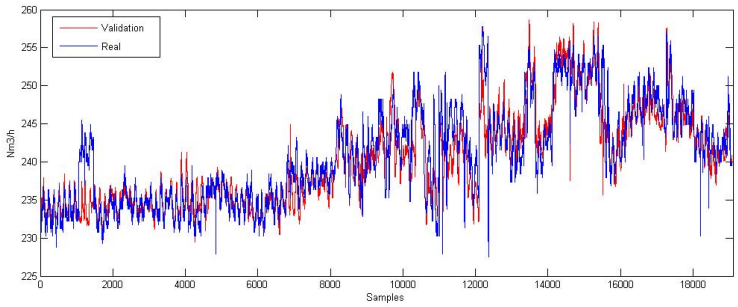

**(b)** ANN-BP performance according to the number of hidden neurons.

**Figure 4.12:** Stability of ELM and BP according to the number of hidden neurons.

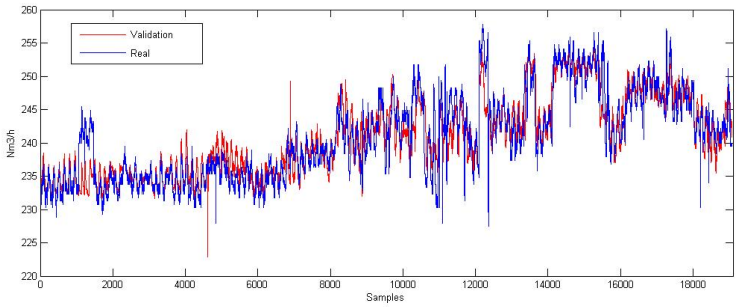
As we can see in Table 4.5, all previous algorithms provide approximately the same accuracy and perform rather well. However, the training and testing accuracy have a significantly larger difference, in most of cases, for ANN-BP and SVM than for the ELM models. This can be explained by the fact that ANN-BP and SVM tend to overfit the training data. The generalisation performance of ELM is very stable over a wide range of hidden node numbers, as Figure 4.12 shows for the four engine models, where the average error for each engine is plotted against the number of hidden nodes.

As a representative example, Figure 4.13 shows the results for the engine-B system. The real values (blue line) and predicted values (red line) are shown for the test dataset. As can be seen, the three models are very similar and follow the main trend of the output variable ($F_{Gas\_B}$). However, the models do not predict some of the peaks very accurately.

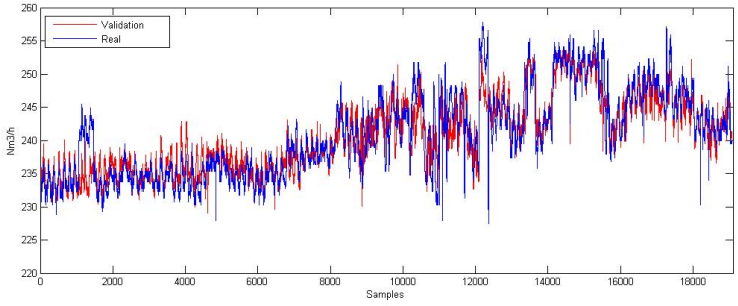
On the other hand, ELM needs more hidden nodes than ANN-BP to achieve a similar performance, while SVM requires many more nodes than either ELM or ANN-BP.




**(a)** ELM model, real output vs. predictions.




**(b)** ANN-BP model, real output vs. predictions.




**(c)** SVM model, real output vs. predictions.

**Figure 4.13:** Graphic results for the ELM, ANN-BP, and SVM models of the engine-D systems.

For training time, ELM is clearly the fastest learning algorithm in all cases, with a training period hundreds of times less than ANN-BP and SVM. Also ELM is the fastest algorithm for testing.

A conclusion that can be drawn from the experimental results, is that the number of nodes is slightly lower for ANN models. However ELM is very stable over a wide range hidden node numbers. The experimental results also show that ELM is by far the fastest, being hundreds of times quicker than SVM and ANN. ELM provides a robust learning algorithm, free of local minima, and has no overfitting problems. All these characteristics make ELM the most suitable algorithms for modelling the CHP plant.

A compressive study of the ANN-BP and ANFIS modelling of the CHP plant can be found in [40].

## 4.5 Modelling of the efficiency using ELM

This section presents the development of a model of the effective electrical efficiency ($\varepsilon_{EE}$) for the real cogeneration plant using ELM. Efficiency is a prominent metric used to evaluate CHP performance and compare it to separated heat and power (SHP) plants. The most commonly used metric for determining the efficiency of a CHP system is $\varepsilon_{EE}$. This metric is used in this work to measure the performance of the plant, and is the multi-objective function used in the optimisation chapter.

The effective electrical efficiency of a CHP plant, $\varepsilon_{EE}$, is defined as:

$$\varepsilon_{EE} = \frac{W_E}{Q_{FUEL} - \sum(Q_{TH}/\alpha)} \cdot 100, \tag{4.6}$$

where $\alpha$ is the efficiency of conventional technology that would otherwise would be used to produce the useful thermal energy output if the CHP system did not exist. The value of $\alpha$, as established in Article 1 of Spanish Royal Decree 661/2007, depends on the type of fuel and the manner in which the heat is used (direct/indirect use). For a gaseous fuel and indirect use of exhaust gases, $\alpha$ is 0.9.

- The net useful power output generated by the four engines and the steam turbine is:

$$W_E = (POW_A + POW_B + POW_C + POW_D) \cdot c_1 + POW_{ST}, \tag{4.7}$$

where $c_1 = \frac{3700kW}{100\%}$ is the conversion factor to change % to $kW$.

- The total fuel (natural gas) input used by the four engines, is defined as:

$$Q_{FUEL} = (F_{GAS\_A} + F_{GAS\_B} + F_{GAS\_C} + F_{GAS\_D}) \cdot c_2, \tag{4.8}$$

  where $c_2 = LHV \frac{kWh}{m^3}$ is the conversion factor to change $\frac{m^3}{h}$ to $kW$.

- The useful thermal energy used in the slurry process is defined as:

$$Q_{TH} = F_{Ev} \cdot c_3, \tag{4.9}$$

  where $c_3$ is the conversion factor to change $\frac{kg}{h}$ to $kW$, being defined as follows:

$$c_3 = \frac{\frac{5730t}{month}\frac{1000kg}{t}\frac{3600s}{h}\frac{825kcal}{kg}}{\frac{7.6297m^3}{h}\frac{60min}{h}\frac{24h}{d}\frac{30d}{month}\frac{860kcal}{kWh}}. \tag{4.10}$$

  The $c_3$ term is obtained taking as reference the performance of the evaporator for a month because 1-minute readings do not exist for the treated amount of slurry, so this has to be approximated. The idea is to relate the amount of slurry treated in the evaporator with the total amount of slurry treated in the slurry drying process (see Subsection 2.4.4). The total amount of slurry treated in the process during the reference month was $5730t$, and $\frac{7.6297m^3}{h}$ is the average flow of slurry treated in the evaporator per hour for this reference month.

  The useful heat assimilated from the slurry drying process is $\frac{825kcal}{kg}$, for pig slurry dried with a humidity of 95%. The term $\frac{860kcal}{kWh}$ relates the thermal energy to the electrical energy. These two values are established in Annex II of the European Commission Directive dated 21st December, 2006 (2007/74/EC) and covered in Article 1 of Spanish Royal Decree 661/2007.

The effective electrical efficiency of the plant was calculated using Eq. 4.6. To do this, firstly the real effective electrical efficiency of each sample in the dataset was calculated using real values. That is: $W_E$, $Q_{FUEL}$, and $Q_{TH}$ are obtained by extracting the variables involved directly from the database.

Next, the effective electrical efficiency was obtained in the same way, but using the predicted values obtained with the ELM models from the previous section, i.e., the power of the steam turbine used was the predicted value from the steam turbine model, the natural gas flow was the predicted value from each engine ELM model, and the thermal energy in the slurry drying process was obtained from the prediction of its respective ELM model.

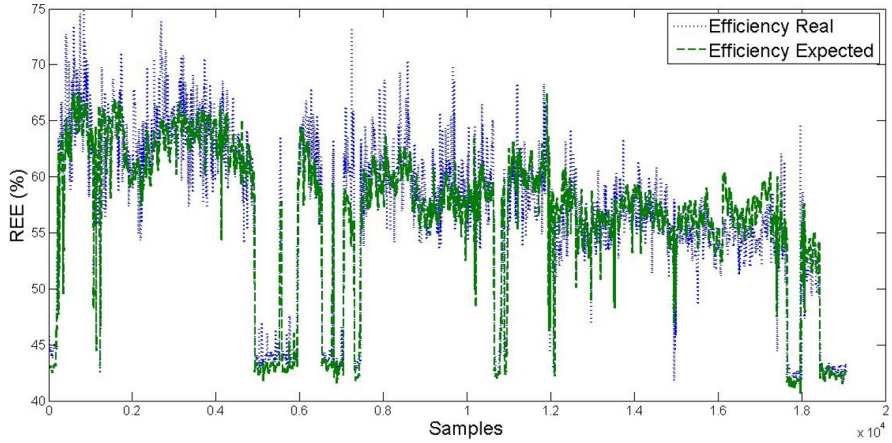



**Figure 4.14:** Results of the ELM models for effective electrical efficiency.

To illustrate the ELM modelling results, Figure 4.14 shows the effective electrical efficiency using Eq.4.6 with real data and employing the predictions of the ELM models. As can be seen, the expected efficiency follows the trend of the real efficiency. The monthly effective electrical efficiency should be greater than 55% on average to be sold to power utilities at a fixed premium price. This is a premium for clean energy that is generated. Some zones are less of 55% efficiency, as between samples 5000 and 6000, but on average, the effective electrical efficiency is greater than this threshold being 56.4% on average obtained with real values and 56,3% on average for predicted values.

## 4.6 A new hybrid feature selection method

An alternative to the variable selection previously developed based on mathematical methods and knowledge of the plant are automated feature selection methods. In this section, a new automated exhaustive hybrid feature selection method is proposed. The method was applied to the steam turbine, in order to check the suitability of the variable selection performed in Subsection 4.2.3.

There are two main categories of feature selection techniques: filter methods and wrapper methods [210]. The latter select a reduced subset of variables by evaluating general data characteristics (i.e., the selected learning algorithm is not involved in the selection process), while the former use the performance of the selected machine-learning algorithm to evaluate each subset of variables. Filters measure the relevance of different subsets of features. Usually they order features individually, or as nested subsets of features, while the filter is assessed by means of statistical tests. They are robust against overfitting, but may fail to select the most useful features for a given

classifier. On the other hand, wrappers measure the usefulness of feature subsets. They perform an exhaustive search of the space of all feature subsets and use cross validation to evaluate the performance of the classifier. Wrappers are able to find the most useful features, but they favour overfitting and are very time-consuming. As will be seen, a combination of both, filter and wrapper, provides an adequate trade-off between usefulness and robustness.

The proposed hybrid feature selection method combines a clustering filter based on the nearest shrunken centroids (NSC) procedure for feature selection in high-dimensional problems [211], with a wrapper around the ELM algorithm, because it has been proven to be the most suitable algorithm from all those previously used.

The NSC method is briefly introduced below.

### 4.6.1 Nearest shrunken centroids method

The NSC method is a modification of the nearest centroid classifier that considers denoised versions of the centroids as class prototypes. The class centroids are increasingly shrunken towards the overall centroid and, as they are shrunk, some features from the initial set no longer contribute to the classification.

$x_{ij}$ are the values for the variables $i = 1, ..., p$, and $j = 1, ..., n$ for the samples. $C_k$ is the set of indices of the $n_k$ samples in class $k = 1, ..., K$. A t-statistic $d_{ik}$ is used to compare each class $k$ to the overall centroid for each variable $i$:

$$d_{ik} = \frac{\bar{x}_{ik} - \bar{x}_i}{m_k(s_i + s)}, \tag{4.11}$$

where $\bar{x}_{ik}$ is $i$-th component of the centroid for class $k$ :

$$\bar{x}_{ik} = \frac{\sum_{j \epsilon C_k} x_{ij}}{n_k}, \tag{4.12}$$

where $\bar{x}_i$ is the $i$-th component of the overall centroid:

$$\bar{x}_i = \frac{\sum_{j=1}^{n} x_{ij}}{n}, \tag{4.13}$$

and where $s_i$ is the pooled within-class standard deviation for each variable:

$$s_i^2 = \frac{1}{n - K} \sum_k \sum_{j \epsilon C_k} (x_{ij} - \bar{x}_{ij})^2, \tag{4.14}$$

where $s$ is the median value of the $s_i$ over the set of variables, and $m_k = \sqrt{1/n_k + 1/n}$. In the shrinkage, $d'_{ik}$ is reduced by soft-thresholding:

$$d'_{ik} = sign(d_{ik})(|d_{ik} - \Delta|)_+, \tag{4.15}$$

where $t_+ = t$ if $t > 0$ and zero otherwise. $\triangle$ is the shrinkage parameter. The absolute value of each $d'_{ik}$ is reduced by an amount $\triangle$, and set to zero if the result is less than zero. According to Eq. 4.11 the shrunken centroids are calculated as follows:

$$\bar{x}'_{ik} = \bar{x}_i + m_k(s_i + s)\, d'_{ik}. \tag{4.16}$$

As the parameter $\Delta$ increases, $d_{ik}$ for some variables are reduced to zero for all classes, and all centroids $\bar{x}_{ik}$ are shrunk to $\bar{x}_i$ as shown Figure 4.15. Those variables are therefore effectively eliminated from the class prediction.

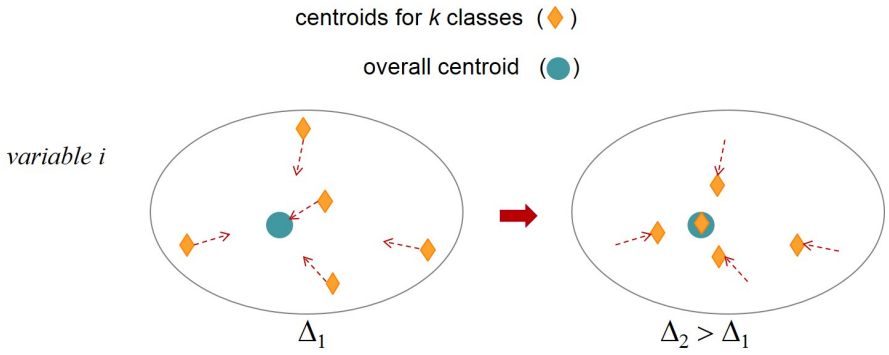



**Figure 4.15:** Progressive shrinkage of centroids towards the overall centroid according to parameter $\triangle$.

### 4.6.2 The hybrid method applied to steam turbine feature selection

The exhaustive hybrid feature selection method proposed combines a clustering filter based on the nearest shrunken centroids (NSC), with a wrapper around the ELM algorithm. The method was applied to one of the systems in the CHP plant, the steam turbine.We will now look at how the method was applied to this steam turbine.

Firstly, a simple correlation study was performed with the aim of detecting strong linear relationships between pairs of signals using the dataset containing 200 signals

after data cleaning tasks. Signals with correlation coefficients higher than $\|r\| > 0.98$ were removed from the original dataset. Taking into account the correlation threshold, a total of 12 signals were eliminated. Henceforth a dataset with 188 signals was available on which to apply hybrid feature selection techniques.

The reduced set of 188 signals was further examined in order to gather informative features.

The output of the steam turbine system is its power, and is a continuous variable. Through a discretisation process, the range of continuous target variable values is transformed into a set of intervals that are used as discrete classes. This enables regression problems to be transformed into classification problems. The discretisation process used was the $k-$means clustering method [212]. This method builds $K$ intervals that minimise the sum of the distances of each element of an interval to the interval's gravity center (the median is used as a centrality statistic). A total of 20 classes were generated.

The shrinkage was applied from $\triangle = 0$ (no shrinkage and no signals eliminated), up to $\triangle = 40$, where only one variable was left with increments of $\triangle = 0.14$ . The training and testing root mean squared error (RMSE) were computed for different $\triangle$ values within this range. The collection of samples was randomly divided into a training set consisting of 75% of the data, and a test set with the remaining 25%. Training and prediction were evaluated using both the nearest shrunken centroids clustering procedure, and the ELM method.

The nearest shrunken centroids classification of a test sample $x^* = (x_1^*, x_2^*, ..., x_p^*)$ was carried out by calculating the standardised distances of sample $x^*$ to each shrunken centroid or prototype of class $k$:

$$\delta_k(x) = \sum_{i=1}^{p} \frac{(x_i^* - \bar{x}'_{ik})^2}{(s_i + s)^2}. \tag{4.17}$$

The prediction for sample $x^*$ was then made using a "winner-takes-all" rule that chooses the class for which the distance $\delta_k$ is the smallest.

**Table 4.7:** Relevant variables obtained by means of the hybrid feature selection method.

| Variables Tag | Description Of The Variables And Units |
|---|---|
| $T_{Mixt\_EngB}$ | Cooling water temperature for the mixture in engine-B (ºC) |
| $T_{Eng\_Room}$ | Gas temperature engines room (ºC) |
| $P_{Crank\_C}$ | Crankshaft pressure of engine C (bar) |
| $F_{Steam}$ | Steam flow to the steam turbine (kg/h) |
| $P_{Cond}$ | Condenser pressure (bar) |
| $P_{St\_Gen}$ | Steam generator pressure (bar) |

ELM training and prediction were performed at each $\triangle$ value over the subset of active variables. In each case, the average accuracy over 100 trials of ELM was computed to provide more stable results and minimise the randomness effect. Figure 4.16 shows the shrinkage results where the ELM performs noticeably well, and some significant variables are pointed out (Table 4.7). The minimum ELM test error was reached for a shrinkage of $\triangle = 32.48$, with a subset of only 3 variables ($P_{Cond}$, $F_{Steam}$, $P_{St\_Gen}$) from the initial 188 variables. The zone with shrinkage between $\triangle = 32.48$ and $\triangle = 38.64$ is very stable. Next, the variable $P_{Cond}$ was eliminated and the error increased slightly. We can conclude that the subset with lowest testing error is the most suitable for generating a model to predict the power generated by the steam turbine. The three variables obtained using the hybrid feature selection method correspond to the variables initially selected as inputs for the steam turbine system, thus verifying the fact, that the initial selection of variables was correct.

A compressive study of the ANN-BP and ANFIS modelling of the CHP plant can be found in [213].

## 4.7 Conclusions

In this chapter different approaches to modelling the whole CHP plant and slurry drying process were presented. To facilitate this modelling, data cleaning tasks were first applied to obtain a suitable dataset. Subsequently, the plant was split into different systems, and the input-output variables for each system were selected.

Then, for each system in the CHP plant and slurry drying process two models were developed: one using ANN-BP and another using ANFIS. The results verified that both the ANFIS and ANN-BP methods are powerful tools when modelling a complex cogeneration plant and are able to make accurate predictions when dealing with unseen data. However, when there is a high number of inputs, ANN-BP performs better than ANFIS. As well as this, ANFIS models always have more complex structures, and their linguistic interpretation is not possible because the number of

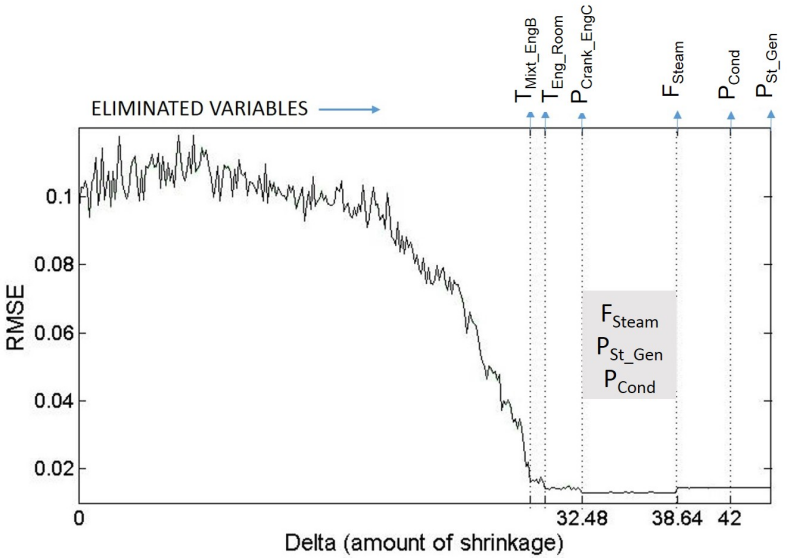



**Figure 4.16:** ELM testing accuracy as a function of the shrinkage parameter $\triangle$, and the significant variables during the hybrid feature selection method.

rules obtained is very high. It can therefore be concluded that for this modelling problem ANN-BP is a better option.

In order to overcome some of the ANN-BP drawbacks, such as overfitting and local minima, the modelling of the CHP plant was developed using a new training algorithm for neural networks: ELM. With the same dataset and variables, models were also constructed using ANN-BP and SVM using a radial basis function kernel. The experimental results show that ELM is by far the fastest algorithm, being hundreds of times quicker than SVM and ANN-BP when to training the models. Also, it is very stable over a wide range of hidden node numbers, with the quantity of nodes being similar to that in ANN-BP and much lower than in SVM models. After this, the effective electrical efficiency of the real CHP process using ELM models was obtained. The results verified that the monthly effective electrical efficiency is greater than 55% on average, then is sold to power utilities at a fixed premium price.

In the last section, a new hybrid feature selection method that combines a clustering filter with ELM as wrapper was applied for the steam turbine system to check the initial variable selection. The results verified that a subset comprising only three variables was the most suitable, which is consistent with the variables selected in the previous approaches.

# Chapter 5: Plant optimisation

## 5.1 Overview

This chapter describes the different approaches used to optimise the cogeneration plant and the slurry drying process modelled in the previous chapter. For this purpose, firstly, the basics of mathematical optimisation, its formulation and principles are introduced. Then, the different optimisation techniques that exist are classified and briefly described. Subsequently, the function to be optimised, the effective electric efficiency, and its three different objective functions are described: 1) the net useful power output of the plant, 2) the quantity of natural gas consumed by the four engines and 3) the useful thermal energy used in the slurry process. Additionally, the selected decision variables used to change the values and optimise the plant are explained.

Two different optimisation approaches are used in this thesis: a single-objective optimisation based on the gradient descent method (GDM), and a multi-objective optimisation using a genetic algorithm (GA), specifically particular the well-known non-dominated sorting genetic algorithm II (NSGA-II). The results of both approaches are explained and analysed.

## 5.2 Optimisation basics

Mathematical optimisation can be defined as the process of finding the "best" combination of available parameters (independent variables), which optimises (minimises or maximises) certain given quantities, possibly subject to some restrictions to the allowed parameter ranges. The general form of a single-objective optimisation problem is as follows: given a vector $\mathbf{x} = (x_1, x_2, ..., x_n) \in R^N$ of decision variables and $f(\mathbf{x}) : R^n \rightarrow R$ being the objective function, $\mathbf{x}$ is found so that [56]:

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad s.t. \ \mathbf{x} \in X \subseteq R^n. \tag{5.1}$$

The set $X$ is the feasible set of decision vectors. The feasible set $X$ is usually defined by $p$ equality constraints:

$$\mathbf{h}_{\mathrm{j}}(\mathbf{x}) = \mathbf{h}_{\mathrm{j}}(x_1, x_2, ..., x_n) = 0 \qquad 1 \leq j \leq p, \tag{5.2}$$

and $m$ inequality constraints:

$$\mathbf{g}_i(\mathbf{x}) = \mathbf{g}_i(x_1, x_2, ..., x_n) \leq 0 \qquad 1 \leq i \leq m. \tag{5.3}$$

When the problem to be optimised has more than one objective function, the problem is called multi-objective-optimisation. The general form of multi-objective optimisation problems is to find $\mathbf{x}$ so that:

$$\underset{\mathbf{x}}{min}\,[f_1(\mathbf{x}),\ f_2(\mathbf{x}), ..., f_k(\mathbf{x})] \qquad s.t.\ \mathbf{x} \in X \subseteq R^n, \tag{5.4}$$

where $k \geq 2$ is the number of objectives.

These objective functions form a mathematical description of performance criteria that usually conflict with one another. Hence, the term optimisation in multi-objective problems means finding a solution that gives the values of all the objective functions acceptable to the decision maker. To do this, optimal decisions need to be made in the presence of trade-offs between two or more conflicting objectives. Most real-word problems involve more than one objective, making multiple conflicting objectives interesting to solve as multi-objectives optimisation problems. An example of practical real-word multi-objective optimisation is a route by car involving two objectives: to minimise both the time needed to make the trip and the fuel consumption. These two objectives are in conflict. If time is decreased, the speed during the trip has to be increased and consequently, fuel consumption increases too.

An element which satisfies all the constraints, $x^* \in X$ , is called a *feasible solution* (or feasible decision) [214]. A vector $[f_1(\mathbf{x}^*),\ f_2(\mathbf{x}^*), ...f_k(\mathbf{x}^*)] \in R^k$ for a feasible solution, $\mathbf{x}^*$, is the *outcome*. The *feasible region* is the set of all the elements that satisfy all the constraints. When objectives are in conflict the notion of Pareto optimal solutions must be introduced [215]. An $\mathbf{x}^* \in X$ vector is said to be a *Pareto optimal* for a multi-objective problem, if all other $\mathbf{x} \in X$ vectors have a higher value for at least one of the objective functions $f_l$, with $l = 1, 2, ..., k$, or have the same value for all the objective functions. A feasible solution $\mathbf{x}^1 \in X$ is said to

be (Pareto) dominant over another solution $\mathbf{x}^2 \in X$, (i.e. $\mathbf{x}^1 \prec \mathbf{x}^2$) if:

$$\begin{array}{ll} f_l(\mathbf{x}^1) \leq f_l(\mathbf{x}^2) & for\, all\, l \in \{1, 2, ..., k\}\, and \\ f_r(\mathbf{x}^1) < f_r(\mathbf{x}^2) & for\, at\, least\, one\, r \in \{1, 2, ..., k\}. \end{array} \tag{5.5}$$

The set of Pareto optimal outcomes is called a *Pareto front*. Figure 5.1 shows the Pareto front for the previous example of the road trip optimisation problem with two objectives (minimisation of both time and fuel) and the trade-off between the two objectives. Once the Pareto front has been found, the goal is to find a point along the Pareto Front that provides suitable trade-offs that satisfy the different objectives, and then to select a single solution that satisfies the preferences of the decision maker.

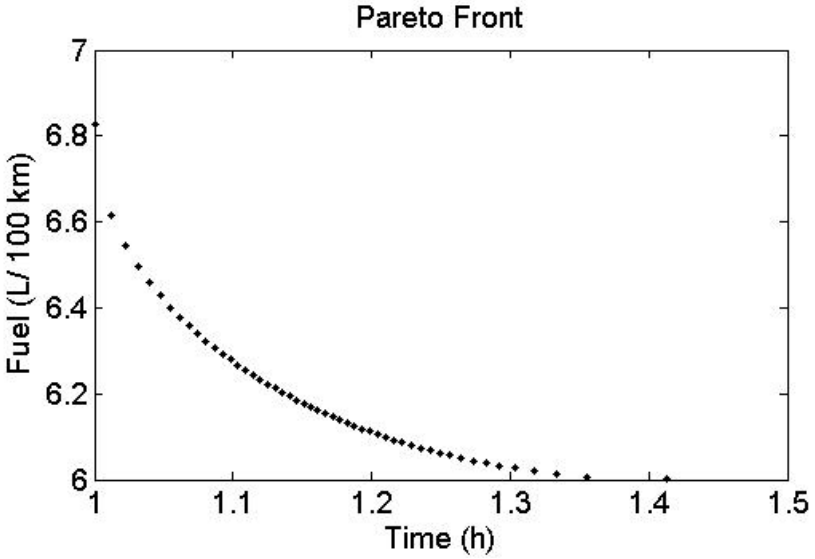



**Figure 5.1:** Pareto front for the optimisation problem of minimising time and fuel consumption for a journey.

Optimisation problems can be classified according to different criteria: type of constraints (constrained vs. unconstrained optimisation problems), nature of design variables (integer programming problem vs real-valued programming optimisation problem), nature of the equations involved and type (linear programming vs. non-linear programming optimisation problem), or number of objective functions (single-objective vs. multi-objective optimisation problem).

There are many optimisation methods available for solving the above problems and these can be classified into two distinct groups: conventional and advanced techniques. We will examine both groups in the following paragraphs [216, 217]:

### 5.2.1 Conventional techniques

Conventional optimisation techniques are useful for single as well as multi dimensional optimisation problems. They are helpful for finding the optimal solution for continuous and differentiable functions. Most of these methods are analytical and make use differential calculus techniques.

- *Direct search methods*: direct methods are a set of methods that do not require the evolution of derivatives at any point [218]. The term "direct search" describes the sequential examination of trial solutions involving comparing each trial solution with the "best" obtained up to that time, together with a strategy for determining what the next trial solution will be (as a function of earlier results). Direct methods can be applied almost immediately to many nonlinear optimisation problems
- *Linear programming*: a linear programming problem may be defined as the problem of maximising or minimising a linear function subject to linear constraints. The constraints may be equalities or inequalities [219]. One of the most widely-used linear programming algorithms is the *simplex* algorithm. It provides a systematic way of examining the vertices of the feasible region to determine the optimal value of the objective function [220].
- *Interior points methods*: this solves linear and non-linear problems and achieves optimisation using an iterative method that moves within the feasible region [221].
- *Derivative-based optimisation methods:* these are a set of algorithms that require the evaluation of first derivatives to determine search directions for maximising or minimising a function. Steepest descent methods, Newton methods and the gradient descent method are examples of derivative-based optimisation methods. The gradient descent method (GDM) is simple and effective and is used in this thesis; it is therefore, explained in the next section.

#### 5.2.1.1 Gradient descent method

GDM is one of the derivative-based optimisation methods mentioned above that use knowledge of derivative information (gradient) to determine search directions according to an objective function.

Given an objective function $f$ and a *n-dimensional* vector $\mathbf{x} = (x_1, x_2, ..., x_n) \in R^n$, the objective is to find a (possibly local) minimum point $\mathbf{x} = \mathbf{x}^*$ that minimizes $f(\mathbf{x})$. GDM is an iterative method that efficiently explores the input space [127]. The next point $\mathbf{x}_{k+1}$ during the search is determined by a step down from the current point $\mathbf{x}_k$ in a *direction vector* $\mathbf{d}$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \eta_k \mathbf{d}_k \qquad (k = 1, 2, 3, ...), \tag{5.6}$$

where $k$ denotes the current iteration number and $\eta$ is the step size. The $\mathbf{x}_k$ is intended to converge to a (local) minimum $\mathbf{x}_k^*$. The iterative descent methods compute at each step $(k)$ the direction $\mathbf{d}$ and the step size $\eta$. The next point $\mathbf{x}_{k+1}$ should satisfy the following inequality:

$$f\left(\mathbf{x}_{k+1}\right) = f(\mathbf{x}_k + \eta\mathbf{d}) < f(\mathbf{x}_\mathrm{k}). \tag{5.7}$$

The direction $\mathbf{d}$ is determined on the basis of the gradient $\mathbf{g}(\mathbf{x})$ of the objective function, and is obtained with the first derivatives of the objectives function. The gradient of an objective function $f(\mathbf{x})$ for a *n-dimensional* vector $\mathbf{x} = (x_1, x_2, ..., x_n) \in R^n$ is defined as:

$$\mathbf{d}(\mathbf{x}) = \mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_1}, \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_2}, \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_3}, ..., \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_n}\right). \tag{5.8}$$

Numerical differentiation is a useful procedure for obtaining the first derivatives when the function is not available or even if there is an underlying function but it is only interesting to know its values for a sampled data set without knowing what the function is itself. The first derivative of a function $f$ in a uni-dimensional vector $\mathbf{x}$ is defined as:

$$f' = \lim_{h \to 0} \frac{f(\mathbf{x} + h) - f(\mathbf{x})}{h}, \tag{5.9}$$

where $h$ represents a small change in $\mathbf{x}$.

An example of the GDM process is illustrated in Figure 5.2 where a 2-dimensional projection of a function $f(\mathbf{x})$ is shown. The blue curves are the contour lines, i.e., the regions where the value of the objective function $f(\mathbf{x})$ is constant. The red arrow originating at point $\mathbf{x}_0$ shows the direction of the negative gradient at that point. The (negative) gradient at a point is orthogonal to the contour line going through that point. As can be seen, the gradient descent leads to the point where the value of the function $f(\mathbf{x})$ is minimal.

The ideal objective for the GDM is to find a value of $\mathbf{x}$ that satisfies:

$$\mathbf{g}(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 0. \tag{5.10}$$

However, most of the time, is not possible to solve this equation analytically. The GDM procedure is repeated until one of the following stopping criteria is achieved:

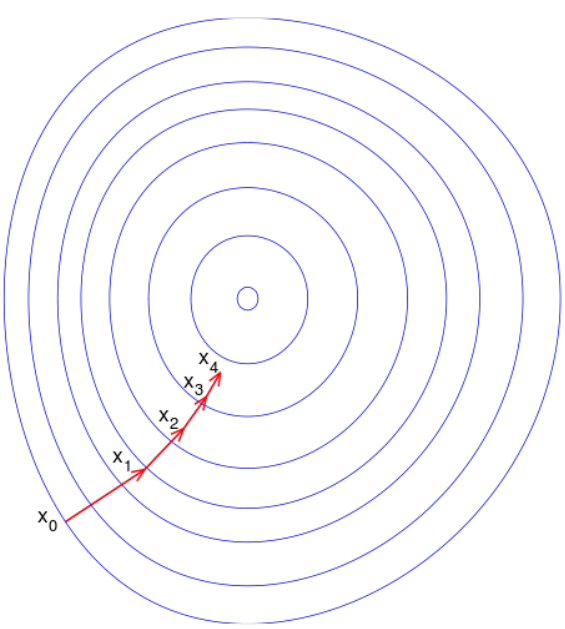



**Figure 5.2:** Directions from the starting point $\mathbf{x}_0$ to $\mathbf{x}_4$ applying the GDM.

- The value of the objective function is sufficiently small.
- The size of the gradient vector is smaller than a specified value.
- A given computing time is exceeded.
- A stop number of iterations is reached.

### 5.2.2 Advanced optimisation techniques

Over the past several years, many optimisation methods based on the observation of certain natural phenomena have been proposed and evaluated. Most of these methods use statistical concepts and random numbers to advance the search toward a solution [222]. Some of these optimisation techniques have been explained in Chapter 3 as simulated annealing, particle swarm optimisation, ant colony optimisation, and one of the most popular (and also used in this thesis): evolutionary algorithms.

Evolutionary algorithms are search methods inspired by the process and mechanisms of biological evolution. The common underlying idea behind all these techniques is the same: given a population of individuals (i.e., possible solutions) environmental pressure leads to natural selection (survival of the fittest). Genetic algorithms are probably the most popular technique in evolutionary algorithms and were explained in a general way in Section 3.6, including the genetic algorithm used in this work, the non-dominated sorting genetic algorithm II (NSGA-II). This algorithm is briefly explained below.

#### 5.2.2.1 Non-dominated Sorting Genetic Algorithm II

NSGA-II is an improved version of the well-known "Non-dominated Sorting Genetic Algorithm" (NSGA) based on the work of Prof. Kalyanmoy Deb for solving single and multi-objective optimisation problems [2]. NSGA was generally criticised for its computational complexity, lack of elitism and for a priori choosing the optimal parameter value for the sharing parameter [223]. NSGA-II provides a solution for all these mentioned shortcomings.

In NSGA-II, initially a random parent population is created ($P_0$). The population is sorted based on the domination concept. An individual is said to dominate another if its objective functions are no worse than the other and at least one of its objective functions is better than in the other, i.e., if it satisfies Eq.5.5. The first front ($F_1$) is a completely non-dominant set in the current population, the second front ($F_2$) is dominated by the individuals in the first front only, and so each front progresses. Each solution is assigned a rank (or fitness), $i_{rank}$, equal to its non-domination level (for the $F_1$ front $i_{rank} = 1$, for the $F_2$ front $i_{rank} = 2$, and for the $F_k$ front $i_{rank} = k$). Thus, minimisation of fitness is assumed. At first, the usual binary tournament selection, recombination, and mutation operators are used to create an offspring population $Q_0$ of size $N$. Since elitism is introduced by comparing the current population with the previously found best non-dominated solutions, the procedure is different after this initial generation.

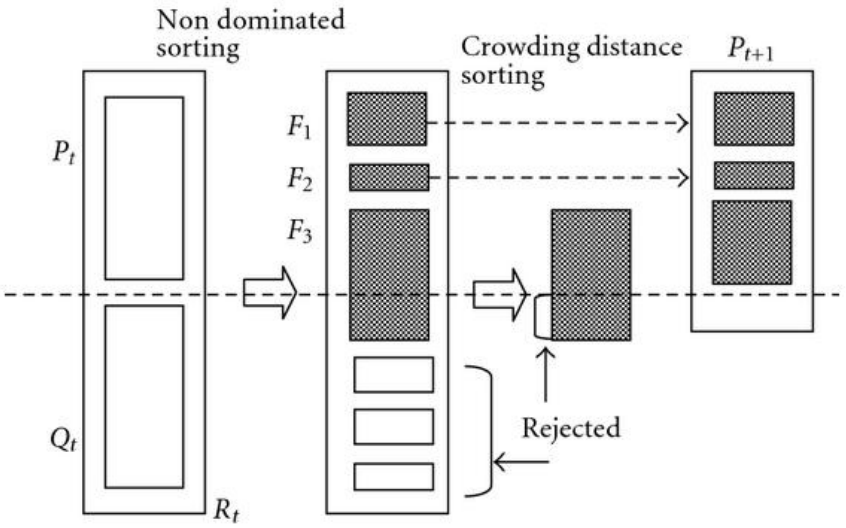



**Figure 5.3:** NSGA-II scheme procedure [2] .

Figure 5.3 shows the process for the $i - th$ generation to illustrate the procedure. Initially a combined population using the current population with previously found best nondominated solutions $R_t = P_t \cup Q_t$ is formed. The population $R_t$ is size $2N$. Then, the population $R_t$ is sorted according to non-domination ($F_1$,$F_2$, $F_3$, ...). Since

all previous and current population members are included in $R_t$, elitism is ensured. Now, solutions belonging to the best non-dominated set $F_1$ are the best solutions in the combined population and must be emphasised more than any other solution in the combined population. If the size of $F_1$ is smaller than $N$, all members of the set $F_1$ for the new population $P_{t+1}$ are selected . The remaining members of the population $P_{t+1}$ are chosen from subsequent non-dominated fronts in the order of their ranking $i_{rank}$. Thus, solutions from set $F_2$ are chosen next, followed by solutions from set $F_3$, and so on, until no further sets can be accommodated.

Defining $F_l$ as the last nondominated set beyond which no other set can be accommodated, the count of solutions in all sets from $F_1$ to $F_l$ would be larger than the population size. However, the number of population members has to be exactly $N$. In this case, the population is sorted using the solutions of the last $F_l$ front according to the crowded-comparison operator $\prec_n$.

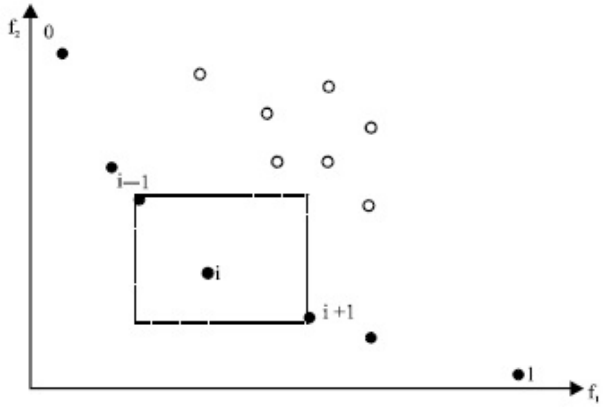



**Figure 5.4:** Crowding distance. Points marked in filled circles are solutions from the same nondominated front [2].

The crowded-comparison operator ensures diversity among population members using two parameters: the non-domination rank ($i_{rank}$) previously mentioned and the crowding distance ($i_{distance}$). The crowding distance serves as an estimation of the perimeter of the cuboid formed by using the nearest neighbours as the vertices. The crowding distance of a point, $i$, in the objective space is equal to the sum of two average side lengths in a rectangle composed of adjacent points $i-1$ and $i+1$, as shown in Figure 5.4. A large average crowding distance will result in better population diversity. The crowded-comparison operator ($\prec_n$) defines a partial order $\prec_n$ as

$$\begin{aligned} i \prec_n j \; if \; (i_{rank} < j_{rank}) \\ or \; ((i_{rank} = j_{rank}) \\ and \; (i_{distance} > j_{distance})). \end{aligned} \tag{5.11}$$

Eq.5.11 means that between two solutions with different non-domination ranks, the

solution with the lower (better) rank is preferred. Otherwise, if both solutions belong to the same front, then the solution located in a lesser crowded region is preferred.

Thus, when the population size is larger than $N$, the last $F_l$ front is sorted according to the crowded-comparison operator $\prec_n$ in descending order and the best solutions needed to fill all the population slots are chosen. After sorting the last front $F_l$, the new population $P_{t+1}$ is obtained. This population $P_{t+1}$ of size $N$ is now used for selection, crossover, and mutation to create a new population $Q_{t+1}$ of size $N$. It is important to note that a binary tournament selection operator is used, but the selection criterion is now based on the crowded-comparison operator. The set of non-dominated solutions after all the generations of the procedure is the Pareto Front.

## 5.3 CHP plant optimisation

Next, the objective functions for optimising the cogeneration process and the slurry drying process are introduced and the different approaches and results obtained are explained. The aim of cogeneration optimisation is to improve plant efficiency. The ANN and ANFIS models selected previously, in Section 4.3, for each system of the plant, were used to simulate the process.

In the present case, the optimisation problem is actually a multi-objective optimisation problem in which three functions need to be optimised: $W_E$, the net useful power output needs to be maximised (Eq.4.7), $Q_{FUEL}$, the total quantity of natural gas consumed by the four engines needs to be minimised (Eq.4.8), and $Q_{TH}$, the useful thermal energy used in the slurry process needs to be maximised (Eq.4.9).

However, since the efficiency of CHP plants is measured in terms of effective electrical efficiency ($\varepsilon_{EE}$), the three above objectives can be formulated as a single optimisation problem:

$$\varepsilon_{EE} = \frac{W_E}{Q_{FUEL} - \frac{Q_{TH}}{\alpha}} 100. \tag{5.12}$$

This expression is explained in depth in Section 4.5.

After carefully studying the process, 12 variables from the entire cogeneration process and slurry process were selected as decision variables in the optimisation process. The decision variables are input parameters whose values can be adjusted to improve the value of the above objectives. These variables are highlighted with a grey circular background in Figure 2.7 to identify where are they located in the cogeneration plant. Figure 5.5 shows the different energy systems of the process with their key parameters and their relationships with the three objective functions.

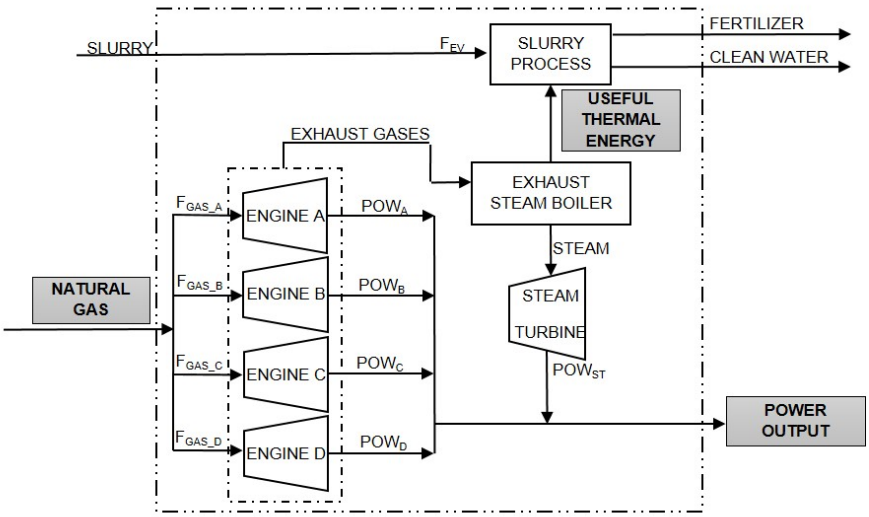



**Figure 5.5:** Scheme for the energy systems and their relationship with the objective functions.

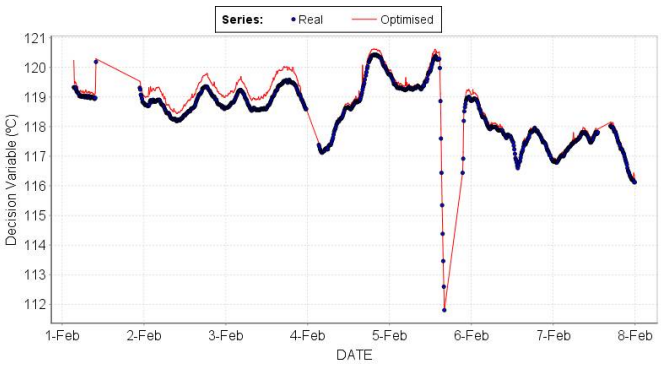
To provide reasonable and realistic optimisation, the decision variables must satisfy the following constraints imposed by the plant owner (as for $T_{A/B/C/D\_1}$ and $T_{A/B/C/D\_2}$ ), or obtained from the real operation variables ranges as for the rest of the decision variables:

- $T_{A/B/C/D\_1}$ and $T_{A/B/C/D\_2}$ (8 intake air temperatures, 2 for each engine): $30 \leq T \leq 38$ºC.
- $T_{H2O\_EX}$(exchange water temperature): $61 \leq T_{H2O-Ex} \leq 65$ºC.
- $P_{St\_Gen}$(steam generator pressure): $20 \leq P_{St-Gen} \leq 22bar$.
- $P_{Ev}$(evaporator pressure): $0.13 \leq P_{Ev} \leq 0.17bar$.
- $T_{H2O\_SH}$ (superheated water temperature ): $110 \leq T \leq 125$ºC.

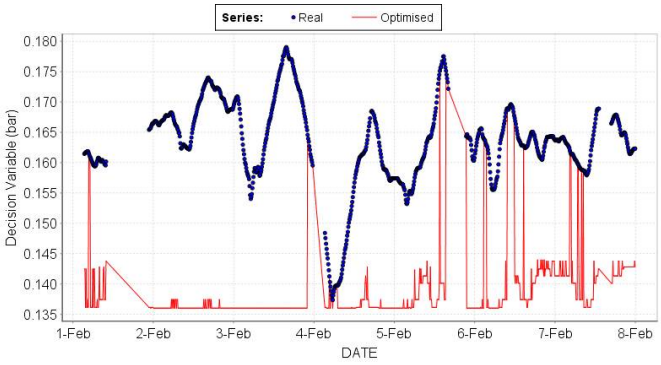
Below, the GDM single-objective and NSGA-II multi-objective optimisation approaches are explained and the results obtained are discussed.

## 5.4 GDM-based optimisation

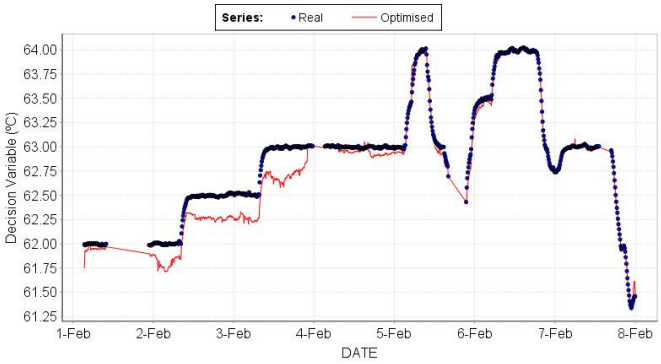
In this section, the GDM technique is used to optimise the cogeneration process. The gradient descent optimisation algorithm calculates the values for the decision variables in order to obtain the maximum $\varepsilon_{EE}$ for each sample of the dataset being optimised. As the three objective optimisation functions ($W_E$, $Q_{FUEL}$, and $Q_{TH}$ )



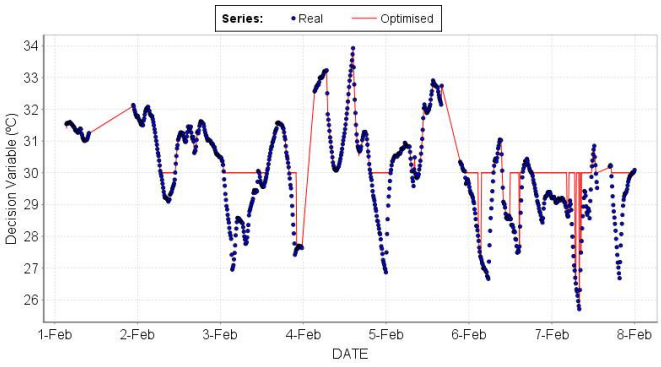

**(a)** $T_{H2O\_SH}$ (superheated water temperature ).




**(b)** $P_{Ev}$(evaporator pressure).




**(c)** $T_{H2O\_EX}$(exchange water temperature).




**(d)** $T_{B1\_D}$ (air intake temperature engine D bank 1).

**Figure 5.6:** Real values (dotted line) versus optimised values (continuous line) for the selected decision variables using GDM.

are evaluated using Eq.5.12, it can be stated as a single objective problem. The experiments in this approach were carried out using the OPTIBAT® Trainer tool [205].

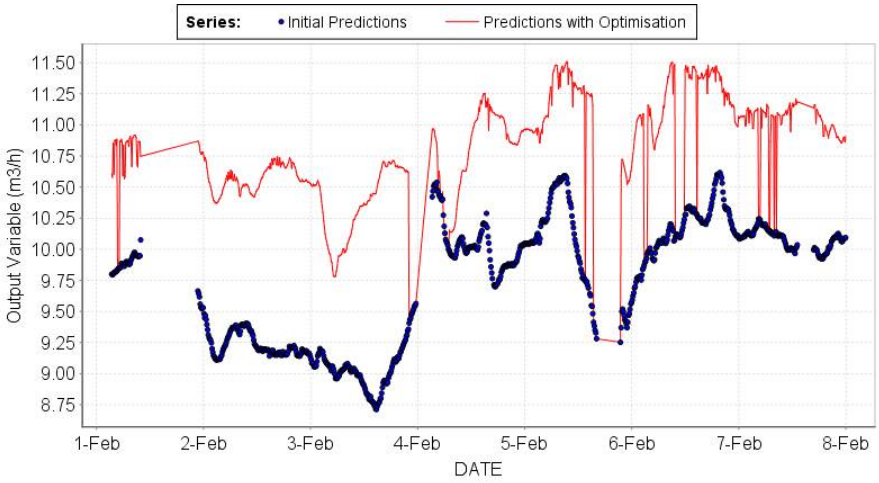


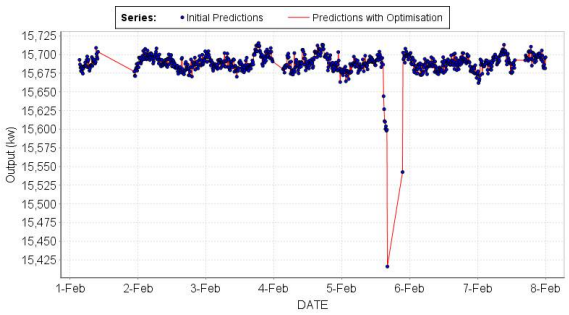

**Figure 5.7:** Output for the slurry process model with and without optimisation ($F_{Ev}$) using GDM.

The GDM optimisation is performed through an exhaustive search, which starts with an initial set of values for the variables and iteratively moves towards a set of values that optimises the cost function using Eq.5.6 to obtain the next point and Eq.5.8 to calculate the direction on the basis of the gradient, as was explained in Subsection 5.2.1.1.

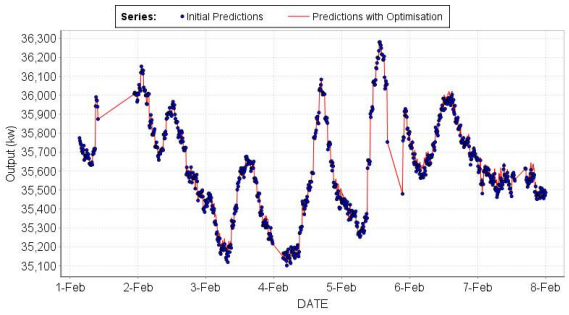
To carry out the optimisation experiments, a subset with data corresponding to one week was selected from the one-year-long dataset and was then resampled at a sample time of ten minutes, (i.e., the values of the variables were read every ten minutes) with the aim of obtaining sufficient data to procure useful results and not spending too long to get them. The one-week dataset was arbitrarily selected in February.

Values for the optimum decision variables were calculated for each sample in the week-long dataset, as shown in Figure 5.6. As all the air intake temperatures were very similar, only one is shown here. Also, the steam pressure in the steam generator is not included because the optimised and real values were practically in agreement for every dataset.

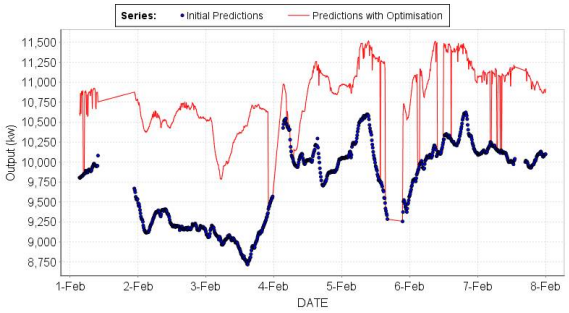
Figure 5.6a shows the temperature of the superheated water used in the slurry drying process and it can be observed that, in most cases, the optimised values were slightly higher than the real values measured in the subset. Figure 5.6b shows the real evaporator pressure and how the optimisation decreased the pressure, thus helping to evaporate more slurry. Similarly, the optimisation decreased the temperature of the exchange water in comparison with the real values (Figure 5.6c). Moreover, with



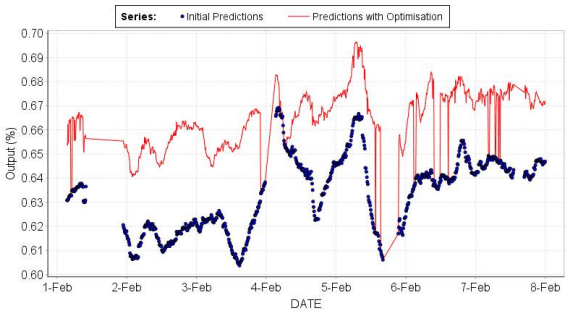

**(a)** Prediction of the power generated by the four engines and the steam turbine with and without recommendations ($W_E$).




**(b)** Prediction of the quantity of fuel used by the four engines with and without recommendations ($Q_{FUEL}$).




**(c)** Prediction of the useful thermal energy used in the slurry process with and without recommendations ($Q_{TH}$).




**(d)** Prediction for the effective electrical efficiency ($\varepsilon_{EE}$) with and without recommendations.

**Figure 5.8:** Terms of the multi-objective function and the multi-objective function with and without recommendations using GDM.

regard to the air intake temperatures for bank 1 in engine D, the optimised values were generally equal to the real values, except when the optimised values were 30ºC. As noted above, a system constraint was that the air intake temperature had to be between 30 and 38 ºC. Therefore, when the real value was lower than 30 ºC, the optimised values were, at least, at the lower limit (Figure 5.6d).

Optimisation causes some of the decision variables to change their values and, therefore some of the model's outputs also change. For example, the system whose output experienced the biggest change was the slurry process. This is shown in Figure 5.7 which predicts that a significantly greater amount of slurry would be treated after the optimisation process. This is consistent with the optimised values for the superheated water temperature, the pressure in the evaporator and the exchange water temperature, because these all favour an increase in the volume of slurry treated.

Finally, Figure 5.8 represents each term in the $\varepsilon_{EE}$ and the effective electrical efficiency, with and without optimisation of the decision variables. The optimisation causes some of the decision variables to change their values and, therefore, the optimisation procedure achieves an average increase in $\varepsilon_{EE}$ of 3.05% with an average of 69.4% of $\varepsilon_{EE}$. This improvement mainly derive from the increased the useful thermal energy used in the slurry drying process, meaning the CHP plant can treat a greater quantity of slurry.

Although the results obtained using the GDM approach improve the effective electrical efficiency of the process, the optimisation problem is also going to be considered here using the NSGA-II algorithm explained previously from a multi-objective perspective, with three objective functions ($W_E$, $Q_{FUEL}$, and $Q_{TH}$ ).

## 5.5 NSGA-II optimisation

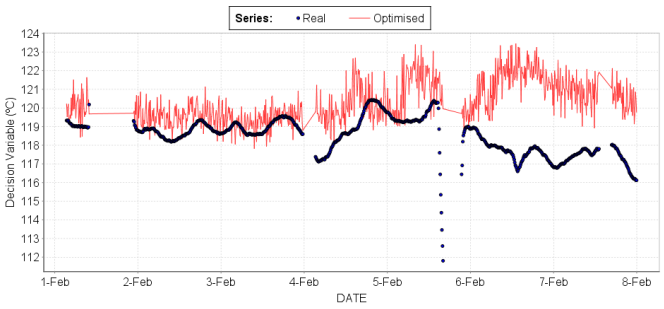
The genetic optimisation algorithm calculates the values for the decision variables in order to maximise the net useful power output $W_E$, to minimise the total natural gas consumed by the four engines $Q_{FUEL}$, and to maximise the useful thermal energy used in the slurry process $Q_{TH}$. The dataset being optimised is the same as that used in the GDM approach, with data from one week and a sample time of 10 minutes. The GA optimisation utilises the NSGA-II algorithm, explained above, using the Matlab® Global Optimisation Toolbox [224]. In particular, the *gamultiobj* function is used, where the number of individuals is the population, i.e. the *population size* is $15 \cdot number\,of\,decision\,variables$ ($15 \cdot 12 = 180$), the maximum number of *generations* before the algorithm stops are $200 \cdot number\,of\,decision\,variables$ ($200 \cdot 12 = 2400$), and the *crossover fraction* is 0.8. The output parameters are the decision variable values for the Pareto front and the corresponding values of the 3 objectives.

As explained previously, NSGA-II begins by creating a random initial population. Each individual represents a point in a search space and a possible solution. The

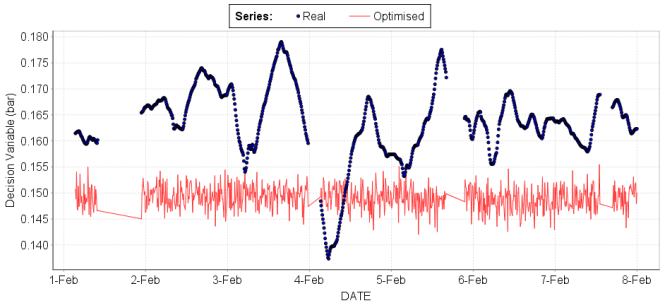
algorithm then creates a sequence of new populations called generations. For each new generation, the algorithm uses the individuals in the current generation to create the next population through a process of evolution via selection, crossover, and mutation. Over successive generations the NSGA-II will converge towards the global (or near global) optimum.

The values of the three objectives for these decision variables constitute the Pareto front. From this set of solutions for the Pareto Front, the user has to select the proper final solution based on knowledge of the process, requirements, and preferences. For representation and analysis purposes, the average value of each decision variable for all the generations at each point of the dataset being optimised was obtained and selected as the optimum value for each decision variable in this sample of the dataset. Similarly, the average values of the decision variables are considered the proper values to be represented in the optimisation procedure.

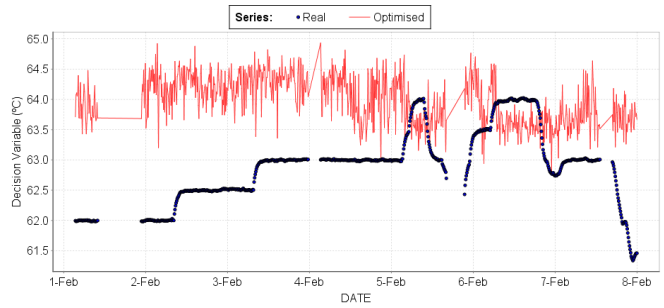
The values for the optimum decision variables were calculated for each sample in the dataset being optimised with the NSGA-II, as shown in Figure 5.9. In Figure 5.9a the temperature of the superheated water used in the slurry drying process is shown, and it can be seen that, in most cases, the optimised values were higher than the real values measured in the subset, as in the GDM approach. Figure 5.9b shows the real evaporator pressure and how optimisation was generally around 0.145-0.155 bar, i.e., lower than the real values, as in the GDM approach. These optimised values are consistent with the knowledge of the process, because by increasing the temperature of the superheated water and decreasing the evaporator pressure, the more slurry can be treated. The temperature values for the exchange water were higher than real values most of the time (Figure 5.9c), in contrast to the GDM approach. From knowledge of the process, if the exchange water temperature decreases, engine refrigeration is better, although a few degrees Celsius difference in the exchange water is not relevant in engine refrigeration and performance. However, the exchange water temperature is used to refrigerate the treated slurry too, and the colder the water, the more slurry is treated. This means that GDM optimisation results are more consistent with the results expected for the exchange water temperature than NSGA-II.



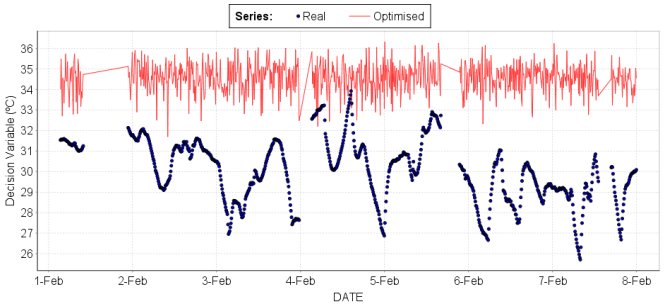

**(a)** $T_{H2O\_SH}$ (superheated water temperature).



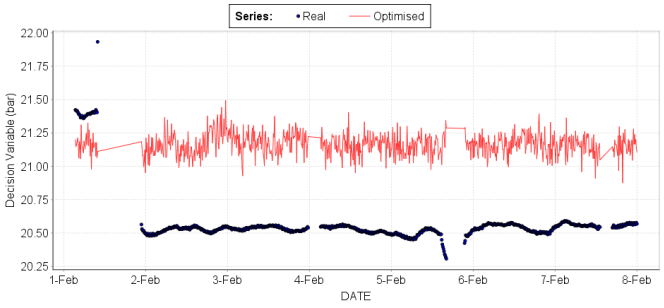

**(b)** $P_{Ev}$(evaporator pressure).




**(c)** $T_{H2O\_EX}$(exchange water temperature).




**(d)** $T_{B1\_D}$ (air intake temperature engine D bank 1).




**(e)** $P_{St\_Gen}$(Steam generator pressure).



**Figure 5.9:** Real values (dotted line) versus optimised values (continuous line) for the selected decision variables using NSGA-II.

Similarly, with regard to the air intake temperatures for bank 1 in engine D, the optimised values were generally higher than the real values, being around 35ºC (Figure 5.9d). As all the air intake temperatures were very similar, only one is shown here. These optimisation values are not the same for both approaches (as mentioned previously, in the GDM approach only the points outside the of constraint values changed in the optimisation), but these differences do not affect engine performance significantly while the values are between the constraint values for the air intake temperatures (30ºC - 38ºC) and this occurs in both optimisation approaches: GDM and NSGA-II. For the steam generator pressure, the optimised values are very stable throughout the week, with values between 21-21.25 bars, unlike in the GDM approach where the optimisation agreed with real values. In the steam turbine system, if the steam generator pressure increases the power generated by the steam turbine increases too.

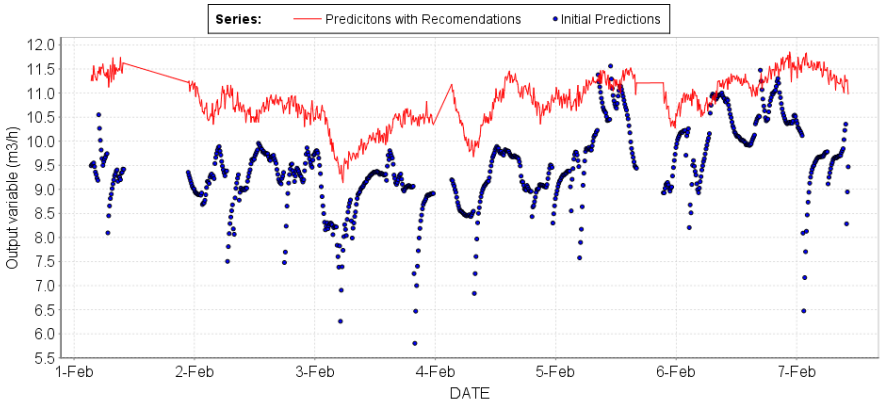
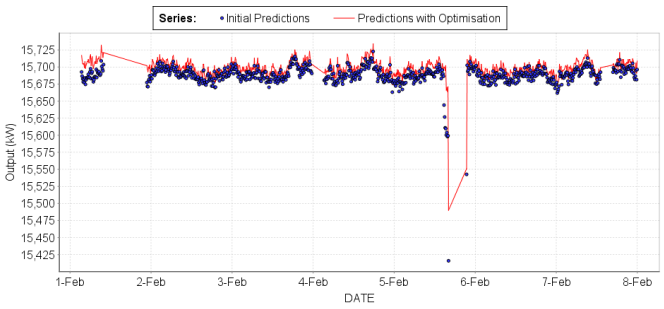
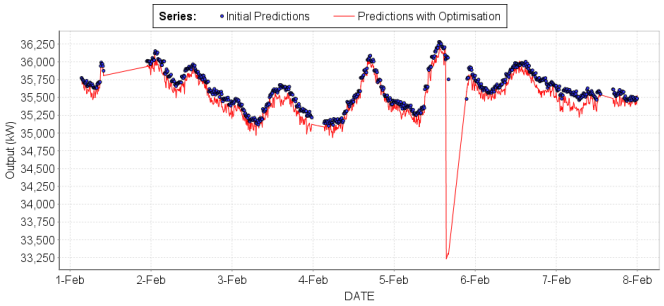



**Figure 5.10:** Output for the slurry process model with and without optimisation ($F_{Ev}$) using NSGA-II.

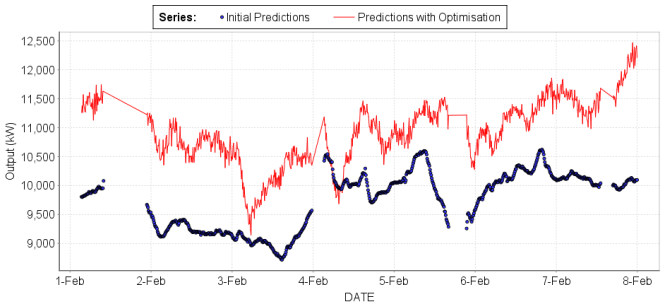
The optimisation causes some of the decision variables to change their values and, therefore some of the model's outputs also change. The system whose output experienced the biggest change was the slurry process, as shown in Figure 5.10. Finally, Figure 5.11 illustrates each term in the $\varepsilon_{EE}$, with and without optimisation of the decision variables. In the NSGA-II approach, the optimisation causes all of the decision variables to change their values and, therefore the procedure achieves an average increase in $\varepsilon_{EE}$ of 4.16% with an average of 70.5% of $\varepsilon_{EE}$, slightly better than with the GDM approach. As in the GDM approach this improvement is mainly derived from the increase in the useful thermal energy used in the slurry drying process, which means that the CHP plant can treat a greater quantity of slurry. With the NSGA-II optimisation, the fuel used by the engines decreases more than in the GDM optimisation, although the power generated by the four engines and the steam turbine increases slightly.



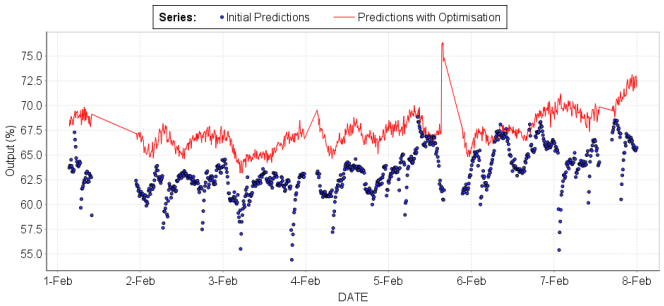

**(a)** Prediction of the power generated ($W_E$) by the four engines and the steam turbine with and without recommendations using NSGA-II.




**(b)** Prediction of the quantity of the fuel used ($Q_{FUEL}$) by the four engines with and without recommendations.




**(c)** Prediction of the useful thermal energy used in the slurry process ($Q_{TH}$) with and without recommendations.




**(d)** Prediction for the effective electrical efficiency ($\varepsilon_{EE}$) with and without recommendations.

**Figure 5.11:** Terms of the multi-objective function and multi-objective function ($\varepsilon_{EE}$) with and without recommendations using NSGA-II.

A closer analysis of the Pareto front reveals possible explanations for the process performance observed. As explained previously, a Pareto front is a collection of all the non-dominated solutions that were retrieved from the final populations during the study. Figure 5.12 shows the Pareto front for the three objective functions of the cogeneration plant using the NSGA-II approach. As can be seen , the shape of the front suggests that most of the Pareto optimal points lie on a plane. From the plane, it can be concluded for the optimal points that the useful thermal energy used in the slurry process increases when the power generated by the engines and the steam turbine increases, and when the fuel used by the engines also increases. This result is consistent with that expected from the process performance. Interestingly, the Pareto front is linear, whereas the problem itself is not.

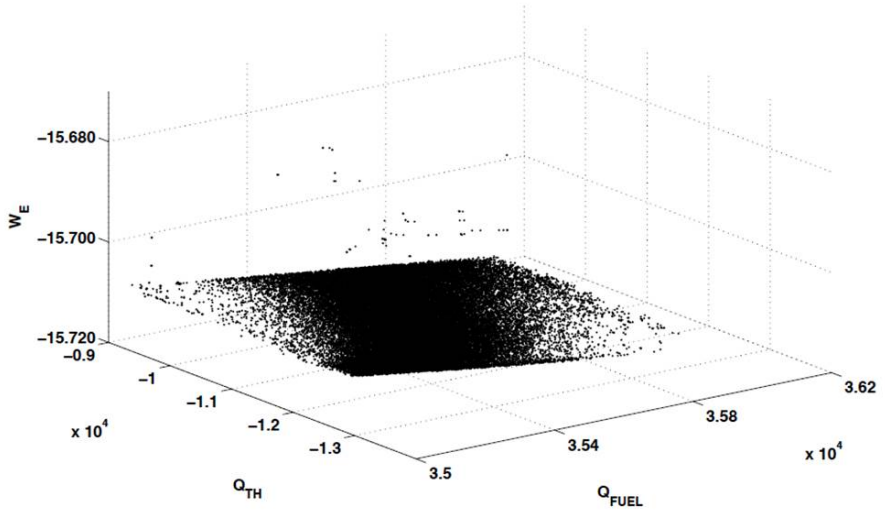



**Figure 5.12:** Pareto front for the three objectives of the cogeneration optimisation problem.

## 5.6 Conclusions

This chapter looks at the optimisation of the cogeneration plant. Two different optimisation approaches were used: a single-objective optimisation using GDM, and a multi-objective optimisation using NSGA-II.

The optimisation approaches provide encouraging results with an average increase of 3.05% optimisation of the energy efficiency with GDM, and 4.16% using NSGA-II. These percentages are a good result when dealing with very large, high-cost industrial processes, such as the CHP plant considered in this work. The dynamic optimisation has the advantage of adaptating to changes in atmospheric conditions and working conditions obtaining the maximum energy efficiency in each time-step.

Although the variation for some decision variables compared with real values was different depending on the optimisation algorithm, as for the exchange water temperature variable, the variations for the decision variables most related to the slurry process agree for both approaches: the superheated water temperature and the evaporator pressure. This is important because in both approaches the improvement in energy efficiency is produced mainly by an increase in the amount of slurry treated.

Comparing the changes in the decision variables for both approaches, GDM and NSGA-II (shown in Figure 5.6 and Figure 5.9), it can be seen that the optimisation values for the decision variables in the NSGA-II approach are very abrupt compared with the GDM approach. This implies that if the optimisation were carried out in the CHP process, these could force the system and cause problems. In summary, the NSGA-II optimisation provides a better average increase in the $\varepsilon_{EE}$ but the optimisation for the decision variables are abrupt. On the other hand, the GDM optimisation results provide a slightly lower average increase in the $\varepsilon_{EE}$ but the optimisation values for the decision variables are more gentle.

From the Pareto front analysis obtained for the three objective functions using the NSGA-II approach, it can be concluded that most of the optimal points of the front lie on a plane. The front reveals for the optimal points of the objective functions a direct relationship between the three objective functions: the net useful power output $W_E$, the total quantity of natural gas consumed by the four engines $Q_{FUEL}$, and the useful thermal energy used in the slurry process $Q_{TH}$.

# Chapter 6: Final conclusions and future work

This thesis presents several computational intelligence approaches for modelling and optimising a complex real-life cogeneration process using real data from the whole process. The main contributions of the thesis are:

- CI algorithms have been applied to both modelling and optimization of a CHP process.
- The modelling process is performed for all the components of a CHP process.
- Data from a real plant have been used.
- The optimisation has been carried out for a multi-objective function.
- It is intended for a continuous on-line operation of the plant.

The main conclusions and future work proposed in order to continue this research are enumerated and discussed below.

## 6.1 Conclusions

After thorough data cleaning to obtain a suitable process dataset and selection of input-output variables for each system using a combination of mathematical techniques and in-depth knowledge of the process and its systems, different modelling approaches were developed. The main conclusions from the modelling are:

- The first approach used to model the combined heat and power (CHP) plant involved an artificial neural networks trained with back-propagation (ANN-BP) as well as with an adaptive neuro-fuzzy inference system (ANFIS). The modelling results demonstrated that both algorithms were, in general, very accurate. However, when the number of inputs was high, as in the case of the plant's engines , ANN-BP performed better than ANFIS. Even if the number of rules was increased, ANN-BP proved better at modelling engine systems. Taking into account the results obtained and the fact that ANFIS has a more complex structure and poor linguistic interpretation ability, it can

be concluded that for this modelling problem ANN-BP is a better option than ANFIS.

- In order to overcome some disadvantages of ANN-BP, such as overfitting and local minima, the modelling of the CHP plant was later developed using a new training algorithm for neural networks: extreme learning machine (ELM). For comparison purposes, models with similar parameters were also constructed using ANN-BP and support vector machine (SVM) with a radial basis function kernel. The experimental results showed that ELM was by far the fastest algorithm, being hundreds of times quicker than SVM and ANN-BP when training the models. Also, it was very stable over a wide range of hidden node numbers, with the quantity of nodes being similar to that in ANN-BP and much lower than in SVM models.
- To check the initial variable selection, a new hybrid feature selection method that combining a clustering filter with ELM as wrapper was applied to the steam turbine system. The results verified that a subset comprising only three variables was the most suitable. This result was consistent with the variables selected in previous approaches.

The optimisation of the cogeneration plant was dealt with using the previously developed cogeneration modelling. The function to be optimised was the effective electrical efficiency $\varepsilon_{EE}$ , which has three different objective functions: 1) the net useful power output of the plant, $W_E$, 2) the quantity of natural gas consumed by the four engines, $Q_{FUEL}$, and 3) the useful thermal energy used in the slurry process, $Q_{TH}$. Two different optimisation approaches were employed: a single-objective optimisation using the gradient descent method (GDM), and a multi-objective optimisation using a non-dominated sorting genetic algorithm II (NSGA-II). The main conclusions from the optimisation are:

- Comparing the changes in the decision variables for both approaches (GDM and NSGA-II) it can be seen that the optimisation values for the decision variables in the NSGA-II approach were very abrupt compared with the GDM approach. This implies that if the optimisation were initiated in the CHP process, it could force the system and cause problems.
- The variations in the decision variables most closely related to the slurry process agreed in both approaches. This is important because in both procedures the improvement in energy efficiency was produced mainly by an increase in the amount of slurry treated.
- The optimisation approaches provided encouraging results with an average increase of 3.05% in energy efficiency with GDM, and 4.16% using NSGA-II. These percentages represent significant results when dealing with very large, high-cost industrial processes, such as the CHP plant considered in this work. Dynamic optimisation has the advantage of adapting to changes in atmospheric and working conditions, obtaining the maximum energy efficiency in each time-step.

- In summary, NSGA-II optimisation provides a better average increase in the $\varepsilon_{EE}$ but the optimisation for the decision variables was abrupt. On the other hand, the GDM optimisation involves a slightly lower average increase in the $\varepsilon_{EE}$ but more gradual values for the decision variables.
- From the Pareto front analysis obtained for the three objective functions using the NSGA-II approach, it can be concluded that most of the optimal points from the objective functions of the front lie on a plane. The front reveals a direct relationship between the optimal points of the three objective functions: the net useful power output, $W_E$; the total quantity of natural gas consumed by the four engines, $Q_{FUEL}$; and the useful thermal energy used in the slurry process $Q_{TH}$.

## 6.2 Future work

Deep learning is a new area of computational intelligence with deep architectures like deep neural networks that have many layers of hidden units and it also allows many more parameters to be used before over-fitting occurs. Some researchers are looking at with deep neural networks applied to industrial purposes [225, 226], and it could be interesting to use these to model the different systems and the energy efficiency of the cogeneration process used in this work.

Another natural continuation of this study would be to focus on the most recent computational intelligence techniques for optimising CHP systems. A proper example is the electrostatic potential energy evolutionary algorithm [227] (ESPEA). The ESPEA is a non-dominated sorting type algorithm that is characterised by providing an excellent distribution of the individuals in the final population. The main difference in the ESPEA compared with NSGA-II is that in the ESPEA algorithm the population diversity is preserved using the physical phenomena of electrostatic potential energy. Some initial tests have been developed in [228] for the CHP process in this study, with encouraging results.

In addition, for the real-life cogeneration plant used in this thesis and taking into account the model prototypes for the different systems and the optimisation simulations, future work should consider real-time plant optimisation for maximum energy efficiency. This would be achieved by installing a prototype in the Optimitive software and connecting via OPC to close the plant's control loop. The main advantage of the proposed approach is that no new investment or changes in the existing plant would be required.

Unfortunately, the CHP plant used in this thesis is no longer working due to a change in the legislation governing premium price in cogeneration plants. Despite this, the method applied for modelling and optimising the CHP process proposed herein could be adapted and extrapolated to other industrial processes with high energy consumption like, for example, paper factories, other energy facilities, or

cement plants. The dynamic optimisation used in this work has the advantage of adapting to changes in atmospheric conditions and working conditions, therefore obtaining the maximum energy efficiency at each time-step.

# Appendix I: cogeneration variables, units, and smoothing

| TAG | Description | Units | Smoothing (Parameter) |
|---|---|---|---|
| $Div_A$ | Diverter engine-A | % | None |
| $Div_B$ | Diverter engine-B | % | None |
| $Div_C$ | Diverter engine-C | % | None |
| $Div_D$ | Diverter engine-D | % | None |
| $F_{Cond}$ | Condensate effluent flow | kg/h | Exponential smooting ($\alpha = 0.3$) |
| $F_{Ev}$ | Evaporator feed flow | kg/h | Moving average (window=200) |
| $F_{FlueGas}$ | Flue gases flow | kg/h | Exponential smooting ($\alpha = 0.5$) |
| $F_{Gas\_A}$ | Natural gas flow engine-A | $m^3/h$ | Exponential smooting ($\alpha = 0.2$) |
| $F_{Gas\_B}$ | Natural gas flow engine-B | $m^3/h$ | Exponential smooting ($\alpha = 0.2$) |
| $F_{Gas\_C}$ | Natural gas flow engine-C | $m^3/h$ | Exponential smooting ($\alpha = 0.2$) |
| $F_{Gas\_D}$ | Natural gas flow engine-D | $m^3/h$ | Exponential smooting ($\alpha = 0.2$) |
| $F_{Steam}$ | Steam flow to steam turbine | kg/h | Exponential smooting |

| | | | |
|---|---|---|---|
| | | | ($\alpha = 0.1$) |
| $F_{H2O}$ | Water flow to feed the steam generator | kg/h | Exponential smooting ($\alpha = 0.2$) |
| $H_{Amb}$ | Ambient humidity | % | None |
| LHV | Low Heating Value | kWh/m$^3$ | None |
| $P_{Cond}$ | Condenser pressure | bar | Exponential smooting ($\alpha = 0.2$) |
| $P_{Crank\_C}$ | Crankshaft pressure of Engine-C | bar | Exponential smooting ($\alpha = 0.1$) |
| $P_{Ev}$ | Evaporator pressure | bar | Exponential smooting ($\alpha = 0.2$) |
| $POW_A$ | Rated power engine-A | % | Exponential smooting ($\alpha = 0.1$) |
| $POW_B$ | Rated power engine-B | % | Exponential smooting ($\alpha = 0.1$) |
| $POW_C$ | Rated power engine-C | % | Exponential smooting ($\alpha = 0.1$) |
| $POW_D$ | Rated power engine-D | % | Exponential smooting ($\alpha = 0.1$) |
| $POW_{ST}$ | Power of the ST | kW | Exponential smooting ($\alpha = 0.1$) |
| $P_{St\_Gen}$ | Steam generator pressure | bar | Exponential smooting ($\alpha = 0.1$) |
| $Q_{Fuel}$ | Natural gas consumed by the engines | kW | None |
| $Q_{TH}$ | Heat used in the slurry process | kW | None |
| $T_{Amb}$ | Ambient temperature | ° C | None |
| $T_{B1\_A}$ | Air intake temperature engine-A bank1 | ° C | Exponential smooting ($\alpha = 0.2$) |

| | | | |
|---|---|---|---|
| $T_{B1Out\_A}$ | Air output temperature engine-A bank1 | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{B2\_A}$ | Air intake temperature engine-A bank2 | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{B1\_B}$ | Air intake temperature engine-B bank1 | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{B2\_B}$ | Air intake temperature engine-B bank2 | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{B1\_C}$ | Air intake temperature engine-C bank1 | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{B2\_C}$ | Air intake temperature engine-C bank2 | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{B1\_D}$ | Air intake temperature engine-D bank1 | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{B2\_D}$ | Air intake temperature engine-D bank2 | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{Bank1\_A}$ | Gases temperature engine-A bank 1 | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{Bank2\_A}$ | Gases temperature engine-A bank 2 | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{Bank1\_B}$ | Gases temperature engine-B bank 1 | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{Bank2\_B}$ | Gases temperature | ° C | Exponential smooting |

| | | | |
|---|---|---|---|
| | engine-B bank 2 | | ($\alpha = 0.1$) |
| $T_{Bank1\_C}$ | Gases temperature engine-C bank 1 | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{Bank2\_C}$ | Gases temperature engine-C bank 2 | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{Bank1\_D}$ | Gases temperature engine-D bank 1 | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{Bank2\_D}$ | Gases temperature engine-D bank 2 | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{Eng\_Room}$ | Gas temperature engines room | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{H2O\_Ex}$ | Exchange water temperature | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{H2O\_SH}$ | Superheated water temperature | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{H2O\_TH}$ | Water temperature tubular heater | ° C | Exponential smooting ($\alpha = 0.1$) |
| $T_{H2O\_Tow}$ | Water temperature cooling tower | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{Mixt\_EngA}$ | Cooling water temperature for the mixture in engine-A | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{Mixt\_EngB}$ | Cooling water temperature for the mixture in engine-B | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{Mixt\_EngC}$ | Cooling water temperature for the mixture in engine-C | ° C | Exponential smooting ($\alpha = 0.2$) |

| | | | |
|---|---|---|---|
| $T_{Mixt\_EngD}$ | Cooling water temperature for the mixture in engine-D | ° C | Exponential smooting ($\alpha = 0.2$) |
| $T_{ST\_Cond}$ | Steam temperature input condenser | ° C | Exponential smooting ($\alpha = 0.1$) |
| $W_E$ | Power generated by the engines and the ST | kW | None |
| $\varepsilon_{EE}$ | Effective electrical efficiency | % | None |

# Appendix II: Publications

In this chapter the disemination works realised during the thesis are listed: the published in International Journals, as well as the International Conferences.

## Journals

- M.A. Braun, S. Seijo, J. Echanobe, P.K. Shukla, I. del Campo, J. Garcia-Sedano, and H. Schmeck. "A neuro-genetic approach for modeling and optimizing a complex cogeneration process". *Applied Soft Computing.* Vol. 48, pp 347-358, November 2016
- S. Seijo, I. del Campo, J. Echanobe, and J. García-Sedano. "Modelling and multi-objective optimization of a complex CHP process." *Applied Energy.* Vol 161, pp 309-319, January 2016.
- S. Seijo, I. del Campo, J. Echanobe, and J. García-Sedano. "Electric Efficiency Modellling of a Complex Cogeneration Process using Extreme Learning Machines." *International Journal of Machine Learning Computing.* Vol. 5, No. 5, pp 399-403, October 2015.
- S. Seijo , I. del Campo, J. Echanobe, J. García-Sedano, E. Suso, and E. Arbizu. "Computational Intelligence techniques for maximum energy efficiency of an internal combustion engine and a steam turbine of a cogeneration process." *International Journal of Energy and Environmental Engineering.* Vol. 5, No. 2, pp 1-10, July 2014

## Conferences

- S. Seijo, V. Martínez, I. del Campo, J. Echanobe, and J. García-Sedano. "Feature Selection and Modelling of a Steam Turbine from a Combined Heat and Power Plant Using ELM." *The International Conference on Extreme Learning Machines (ELM2015)*, Hangzhou 2015.

- S. Seijo, I. del Campo, J. Echanobe, and J. García-Sedano. "Electric Efficiency Modellling of a Complex Cogeneration Process using Extreme Learning Machines." *2nd International Conference on Computational Intelligence and Applications.* (ICCIA2015), Bangkok 2015.
- S. Seijo, I. del Campo, J. Echanobe, J. García, E. Suso, and T.Atienza. "Computational Intelligence techniques for maximum energy efficiency of cogeneration processes." 5th International Congress on Energy and Environment Engineering and Management (CIIEM2013), Lisbon 2013.

# Appendix III: References


[1] "Content available under the creative commons-attribution," Creative Commons Association. [Online]. Available: http://creativecommons.org/

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1109/4235.996017

[3] The European Association for the Promotion of Cogeneration. [Online]. Available: http://www.cogeneurope.eu/

[4] K. Vatopoulos, D. Andrews, J. Carlsson, I. Papaioannou, and G. Zubi, "Study on the state of play of energy efficiency of heat and electricity production technologies," *JRC Scientific and Policy Reports*, 2012.

[5] V. Zare, S. M. S. Mahmoudi, M. Yari, and M. Amidpour, "Thermoeconomic analysis and optimization of an ammonia-water power/cooling cogeneration cycle," *Energy*, vol. 47, no. 1, pp. 271–283, Nov 2012.

[6] M. Feidt and M. Costea, "Energy and Exergy Analysis and Optimization of Combined Heat and Power Systems. Comparison of Various Systems," *Energies*, vol. 5, no. 9, pp. 3701–3722, Sep 2012.

[7] D. Sen, R. Panua, P. Sen, and D. Das, "Thermodynamic analysis and cogeneration of a cement plant in india-a case study," in *2013 International Conference on Energy Efficient Technologies for Sustainability (ICEETS 2013)*, 2013.

[8] I. Noshadi, A. Salahi, M. Hemmati, F. Rekabdar, and T. Mohammadi, "Experimental and anfis modeling for fouling analysis of oily wastewater treatment using ultrafiltration," *Asia-Pacific Journal of Chemical Engineering*, vol. 8, no. 4, pp. 527–538, 2013.

[9] G. Isazadeh, R. Hooshmand, and A. Khodabakhshian, "Design of an Adaptative Dynamic Load Shedding Algorithm using Neural Network in the Steelmaking cogeneration facility," *Iranian Journal od science and technology-Transactions of electrical engineering*, vol. 36, no. E1, pp. 67–82, Jun 2012.

[10] Y. Zhang and J. Ai, "Zeta Potential Modeling of Papermaking Wastewater on Neural Network," in *Proceedings of the 2012 second International Conference*

*on Instrumentation & Measurement, Computer, Communication and Control (IMCCC 2012)*, 2012, Proceedings Paper, pp. 63–66.

[11] M. Budnik, W. Stanek, and H. Rusinowski, "Application of neural modelling in hybrid control model of fluidized bed boiler fired with coal and biomass," in *Proceedings of the 13th International Carpathian Control Conference, ICCC 2012*, 2012, pp. 69–74.

[12] X. Huang and H. Wang, "Modelling of combustion process characteristics of a 600 mw boiler by t-s fuzzy neural networks," in *2009 International Conference on Eenrgy and Environment Technology (ICEET 2009)*, vol. 1, 2009, pp. 737–740.

[13] M. Embrechts and S. Benedek, "Hybrid identification of nuclear power plant transients with artificial neural networks," *IEEE Transactions on Industrial Electronics*, vol. 51, no. 3, pp. 686–693, 2004.

[14] E. Rakhshani, I. Sariri, and K. Rouzbehi, "Application of data mining on fault detection and prediction in boiler of power plant using artificial neural network," in *2nd International Conference on Power Engineering, Eenrgy and Electrical Drives Proceedings (POWERENG 2009)*, Lisbon, 2009, pp. 473–478.

[15] T. Lemma and F. Hashim, "Ifdd: Intelligent fault detection and diagnosis-application to a cogeneration and cooling plant," *Asian Journal of Scientific Research*, vol. 6, no. 3, pp. 478–487, 2013.

[16] Y. Shatnawi and M. Al-khassaweneh, "Fault Diagnosis in Internal Combustion Engines Using Extension Neural Network," *IEEE Transaction on Industrial Electronics*, vol. 61, no. 3, pp. 1434–1443, MAR 2014.

[17] V. N. Ghate and S. V. Dudul, "Cascade Neural-Network-Based Fault Classifier for Three-Phase Induction Motor," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 5, pp. 1555–1563, May 2011.

[18] M. Wolkiewicz and C. T. Kowalski, "On-line Neural Network-based Stator Fault Diagnosis System of the Converter-Fed Induction Motor Drive," in *39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013)*, 2013, Proceedings Paper, pp. 5561–5566.

[19] S. S. Refaat, H. Abu-Rub, M. S. Saad, E. M. Aboul-Zahab, and A. Iqbal, "ANN-Based for Detection, Diagnosis the Bearing Fault for Three Phase Induction Motors Using Current Signal," in *2013 IEEE International Conference on Industrial Technology (ICIT)*, 2013, pp. 253–258.

[20] M. Ballal, Z. Khan, H. Suryawanshi, and R. Sonolikar, "Adaptive neural fuzzy inference system for the detection of inter-turn insulation and bearing wear faults in induction motor," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 250–258, 2007.

[21] M. Kumar Kirar and G. Agnihotri, "Design of high speed adaptive load shedding for industrial cogeneration system," *International Review of Electrical Engineering*, vol. 8, no. 2, pp. 867–874, 2013.

[22] C.-T. Hsu, H.-J. Chuang, and C.-S. Chen, "Artificial neural network based adaptive load shedding for an industrial cogeneration facility," in *Conference Records - IAS Annual Meeting (IEEE Industry Aookications Society)*, 2008, pp. 1692–1699.

[23] G. Isazadeh, R.-A. Hooshmand, and A. Khodabakhshian, "Modeling and optimization of an adaptive dynamic load shedding using the anfis-pso algorithm," *Simulation*, vol. 88, no. 2, pp. 181–196, 2012.

[24] J.-J. Wang, C.-F. Zhang, and Y.-Y. Jing, "Self-adaptive RBF neural network PID control in exhaust temperature of micro gas turbine," in *Proceedings of 2008 International Conference on machine learning and cybernetics*, vol. 1-7, 2008, Proceedings Paper, pp. 2131–2136.

[25] M. Fazlur Rahman, R. Devanathan, and Z. Kuanyi, "Neural network approach for linearizing control of nonlinear process plants," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 2, pp. 470–477, 2000.

[26] A. Mariajayaprakash, T. Senthilvelan, and R. Gnanadass, "Optimization of process parameters through fuzzy logic and genetic algorithm.- a case study in a process industry," *Applied Soft Computing*, vol. 30, pp. 94 – 103, 2015.

[27] F. Rossi, D. Velázquez, I. Monedero, and F. Biscarri, "Artificial neural networks and physical modeling for determination of baseline consumption of chp plants," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4658–4669, 2014.

[28] H. Nikpey, M. Assadi, and P. Breuhaus, "Development of an optimized artificial neural network model for combined heat and power micro gas turbines," *Applied Energy*, vol. 108, pp. 137–148, 2013.

[29] N. Sisworahardjo and M. Y. El-Sharkh, "Validation of artificial neural network based model of microturbine power plant," in *Proceedings of 2013 IEEE Industry Applications Society annual meeting*, 2013, Proceedings Paper.

[30] Y. Ozel, I. Guney, and E. Arca, "Neural network solution to the cogeneration system by using coal," in *International Journal of Energy*, vol. 1, 2007, Proceedings Paper, pp. 105–112.

[31] S. De, M. Kaiadi, M. Fast, and M. Assadi, "Development of an artificial neural network model for the steam process of a coal biomass cofired combined heat and power (CHP) plant in Sweden," *Energy*, vol. 32, no. 11, pp. 2099–2109, Nov 2007.

[32] L. Mastacan, I. Olah, C. Dosoftei, and D. Ivana, "Neuro-fuzzy models of thermoelectric power station installations," in *Proceedings-International Confer-*

*ence on Computational Intelligence for Modelling, Control and Automation, CIMCA 2005*, vol. 1, 2005, pp. 899–904.

[33] A. Tamiru, C. Rangkuti, and F. Hashim, "Neuro-fuzzy and pso based model for the steam and cooling sections of a cogeneration and cooling plant (ccp)," in *Proceeding 2009 3rd International Conference on Energy and Environment: Advancement Towards Global Sustainability*, 2009, pp. 27–33.

[34] Y. Li, B. Yang, Z. Wang, and X. Wang, "Fault pattern classification of turbine-generator set based on artificial neural network," in *presented at the International Conference on Computer Application and System Modeling (ICCASM 2010)*, 2010.

[35] H. Nikpey, M. Assadi, and P. Breuhaus, "Development of an artificial neural network model for combined heat and power micro gas turbines," in *International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, 2012.

[36] G.-B. Huang, D. Wang, and Y. Lan, "Extreme learning machines: A survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[37] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[38] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[39] R. Finker, I. del Campo, J. Echanobe, and M. V. Martínez, "An intelligent embedded system for real-time adaptive extreme learning machine," in *IEEE Symposium on Intelligent Embedded Systems, IES 2014, Orlando, FL, USA, December 9-12, 2014*, 2014, pp. 61–69.

[40] S. Seijo, I. del Campo, J. Echanobe, and J. García-Sedano, "Electric efficiency modelling of a complex cogeneration process using extreme learning machines," *International Journal of Machine Learning and Computing*, vol. 5, no. 5, pp. 399–403, 2015.

[41] R. Zomorodian, M. Rezasoltani, and M. Ghofrani, "Static and dynamic neural networks for simulation and optimization of cogeneration systems," *International Journal of Energy and Environmental Engineering*, vol. 2, no. 1, pp. 51–61, 2011.

[42] A. Jamali, P. Ahmadi, and M. Mohd Jaafar, "Optimization of a novel carbon dioxide cogeneration system using artificial neural network and multi-objective genetic algorithm," *Applied Thermal Engineering*, vol. 64, no. 1-2, pp. 293–306, 2014.

[43] R. Soltani, P. Mohammadzadeh Keleshtery, M. Vahdati, M. Khoshgoftar-manesh, M. Rosen, and M. Amidpour, "Multi-objective optimization of a

solar-hybrid cogeneration cycle: Application to cgam problem," *Energy Conversion and Management*, vol. 81, pp. 60–71, 2014.

[44] H. Gopalakrishnan and D. Kosanovic, "Operational planning of combined heat and power plants through genetic algorithms for mixed 0-1 nonlinear programming," *Computers & Operations Research*, vol. 56, pp. 51 – 67, 2015.

[45] O. Shaneb, P. Taylor, and G. Coates, "Optimal online operation of residential micro-chp systems using linear programicro," *Energy and Buildings*, vol. 44, pp. 17 – 25, 2012.

[46] M. Wang, J. Wang, P. Zhao, and Y. Dai, "Multi-objective optimization of a combined cooling, heating and power system driven by solar energy," *Energy Conversion and Management*, vol. 89, pp. 289 – 297, 2015.

[47] A. Abdalisousan, M. Fani, B. Farhanieh, and M. Abbaspour, "Multi-objective thermoeconomic optimisation for combined-cycle power plant using particle swarm optimisation and compared with two approaches: an application," *International Journal of Exergy*, vol. 16, no. 4, pp. 430–463, 2015.

[48] P. Zhao, Y. Dai, and J. Wang, "Performance assessment and optimization of a combined heat and power system based on compressed air energy storage system and humid air turbine cycle," *Energy Conversion and Management*, vol. 103, pp. 562–572, 2015.

[49] M. Tuba, "Relation between static and dynamic optimization in computer network routing," in *Proceedings of the 8th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, ser. AIKED'09. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 484–489.

[50] H. Cho, R. Luck, S. D. Eksioglu, and L. M. Chamra, "Cost-optimized real-time operation of chp systems," *Energy and Buildings*, vol. 41, no. 4, pp. 445 – 451, 2009.

[51] S. Ikeda and R. Ooka, "Metaheuristic optimization methods for a comprehensive operating schedule of battery, thermal energy storage, and heat source in a building energy system," *Applied Energy*, vol. 151, pp. 192 – 205, 2015.

[52] P. Delgoshaei, S. Treado, and A. Windham, "Real time optimization of building combined heat and power systems." *ASHRAE Transactions*, vol. 118, no. 1, pp. 27 – 33, 2012.

[53] H. Sayyaadi, M. Babaie, and M. R. Farmani, "Implementing of the multi-objective particle swarm optimizer and fuzzy decision-maker in exergetic, exergoeconomic and environmental optimization of a benchmark cogeneration system," *Energy*, vol. 36, no. 8, pp. 4777 – 4789, 2011.

[54] A. Facci, L. Andreassi, and S. Ubertini, "Optimization of CHCP (combined heat power and cooling) systems operation strategy using dynamic programming," *Energy*, vol. 66, pp. 387–400, 2014.

[55] D. G. Brown and B. Morgenstern, Eds., *Algorithms in Bioinformatics - 14th International Workshop, WABI 2014, Wroclaw, Poland, September 8-10, 2014. Proceedings*, ser. Lecture Notes in Computer Science, vol. 8701. Springer, 2014.

[56] D. T. Pham and D. Karaboga, *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, 1st ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1998.

[57] Shipley A., Hampson A., Hedman B, Garland P, and Bautista P, *Combined Heat and Power: Effective Energy Solutions for a Sustainable Future.* U.S. Department of Energy, 2008.

[58] United States Environmental Protection Agency (EPA). [Online]. Available: http://www.epa.gov/

[59] Lerner K. L. and W. L. Brenda, Ed., *The Gale Encyclopedia of science*, 3rd ed. Thomson Gale, 2004, vol. 2, p. 932.

[60] Center for climate and energy solutions (C2ES). [Online]. Available: http://www.c2es.org/technology/factsheet/CogenerationCHP

[61] "Public Utility Regulatory Policies Act of 1978 (PURPA)," *Pub. L. No. 95-617, 92 Stat. 3117*, 1978.

[62] "Energy Policy Act of 2005," *Pub. L. No. 109-58, 119 Stat. 594*, 2005.

[63] Hedman B. (2009, Octuber) CHP: The State of the Market CHP. U.S. EPA Combined Heat and Power Partners Meeting. [Online]. Available: http://www.epa.gov/chp/documents/meeting_100209_hedman.pdf

[64] "Council Regulation (EEC) No. 2008/90 of 29 June 1990 concerning the promotion of energy technology in Europe (thermie programme)," *Official Journal of the European Communities*, vol. L 185, 17 July 1990.

[65] (2000) Green Paper - Towards a European strategy for the security of energy supply. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52000DC0769

[66] "Directive 2004/8/EC of the European Parliament and of the Council of 11 February 2004 on the promotion of cogeneration based on a useful heat demand in the internal energy market and amending Directive 92/42/EEC," *Official Journal of the European Communities*, vol. L 52, 11 February 2004.

[67] "Directive 2012/27/EU of the European Parliament and of the Council of 25 October 2012 on energy efficiency, amending Directives 2009/125/EC and 2010/30/EU and repealing Directives 2004/8/EC and 2006/32/EC," *Official Journal of the European Communities*, vol. L 315, 25 October 2012.

[68] European Comision. (2006) Action Plan for Energy Efficiency: Realising the Potential. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2006:0545:FIN:EN:PDF

[69] ——. (2008) Second Strategic Energy Review. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2008:0781:FIN:EN:PDF

[70] "Directive 2009/28/EC of the European Parliament and of the Council of 23 April 2009 on the promotion of the use of energy from renewable sources and amending and subsequently repealing Directives 2001/77/EC and 2003/30/EC," *Official Journal of the European Communities*, vol. L 140, 23 April 2009.

[71] European Commision. (2011) Energy efficiency plan 2011. [Online]. Available: http://europa.eu/legislation_summaries/energy/energy_efficiency/en0029_en.htm

[72] "Electricity Industry Law 24/2013," *Official State Gazette*, vol. L 310, 27 December 2013.

[73] "Royal Decree 413/2014, of June 6, 2014 regulating the generation of electricity using renewable energy sources, cogeneration and waste ," *Official State Gazette*, vol. L 140, 6 June 2014.

[74] " Ministerial Order IET/1045/2014 of 16 June, which completes the regulatory implementation of the new legal and financial regime applicable to electricity generation facilities based on renewable energy, cogeneration and waste ," *Official State Gazette*, vol. L 150, 16 June 2014.

[75] S. J. M., *Cogeneration, Aspectos termodinámicos, tecnológicos y económicos*, 3rd ed. University of the Basque Country Editorial, 1999.

[76] Pulkrabek, W.W., *Engineering Fundamentals of the Internal Combustion Engine*, Pearson Prentice Hall, Ed., 2004.

[77] EnergyWorks. [Online]. Available: http://www.energyworks.com/

[78] R. F. de la Iglesia, "Efficient electronic implementations of adaptive systems for ambient intelligence environments," Ph.D. dissertation, University of the Basque Country, Spain, May 2015.

[79] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[80] B. Kosko, "Fuzzy systems as universal approximators," *IEEE Transactions on Computers*, vol. 43, no. 11, pp. 1329–1333, 1994.

[81] V. Kreinovich, H. Nguyen, and Y. Yam, "Fuzzy systems are universal approximators for a smooth function and its derivatives," *International Journal of Intelligent Systems*, vol. 15, no. 6, pp. 565–574, 6 2000.

[82] R. Rovatti, "Fuzzy piecewise multilinear and piecewise linear systems as universal approximators in sobolev norms," *Trans. Fuz Sys.*, vol. 6, no. 2, pp. 235–249, May 1998.

[83] D. Dubois and H. Prade, "Unfair coins and necessity measures: Towards a possibilistic interpretation of histograms," *Fuzzy Sets Syst.*, vol. 10, no. 1-3, pp. 15–20, Jan. 1983.

[84] T. A. Runkler, "Selection of appropriate defuzzification methods using application specific properties," *Trans. Fuz Sys.*, vol. 5, no. 1, pp. 72–79, Feb. 1997.

[85] D. P. Filev and R. R. Yager, "A generalized defuzzification method via bad distributions," *International Journal of Intelligent Systems*, vol. 6, no. 7, pp. 687–697, 1991.

[86] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[87] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory.* New York: John Wiley, 1949.

[88] K. Z. Mao, K. C. Tan, and W. Ser, "Probabilistic neural-network structure determination for pattern classification," *Trans. Neur. Netw.*, vol. 11, no. 4, pp. 1009–1016, Jul. 2000.

[89] D. Singh, M. Dutta, and S. H. Singh, "Neural network based handwritten hindi character recognition system," in *Proceedings of the 2Nd Bangalore Annual Compute Conference*, ser. COMPUTE '09. New York, NY, USA: ACM, 2009, pp. 15:1–15:4.

[90] J. Parri and S. Ratti, "Trigonometric function approximation neural network based coprocessor," in *2nd Microsystems and Nanoelectronics Research Conference*, 2009, pp. 148–151.

[91] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *Trans. Neur. Netw.*, vol. 11, no. 3, pp. 586–600, May 2000.

[92] N. Nasrabadi and Y. Feng, "Vector quantization of images based upon the kohonen self-organizing feature maps," in *IEEE International Conference on Neural Networks*, 1988, pp. 101–108.

[93] J. H. Holland, "Outline for a logical theory of adaptive systems," *J. ACM*, vol. 9, no. 3, pp. 297–314, 1962.

[94] ——, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press, 1975, second edition, 1992.

[95] L. Zhao and F. Qian, "Tuning the structure and parameters of a neural network using cooperative binary-real particle swarm optimization," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 4972–4977, May 2011.

[96] C. W. Ahn, S. Member, R. S. Ramakrishna, and S. Member, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, p. 2002.

[97] J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst. Man Cybern.*, vol. 16, no. 1, pp. 122–128, Jan. 1986.

[98] I. del Campo, J. Echanobe, G. Bosque, and J. M. Tarela, "Efficient hardware/software implementation of an adaptive neuro-fuzzy system," *Trans. Fuz Sys.*, vol. 16, no. 3, pp. 761–778, Jun. 2008.

[99] I. del Campo, K. Basterretxea, J. Echanobe, G. Bosque, and F. Doctor, "A system-on-chip development of a neuro-fuzzy embedded agent for ambient-intelligence environments," *Trans. Fuz Sys.*, vol. 42, no. 42, pp. 501–512, Jun. 2011.

[100] J. McCarthy, M. L. Minsky, N. Rochester, and C. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence," 1955.

[101] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[102] B. Widrow and M. E. Hoff, Jr., "Adaptive switching circuits," *IRE WESCON Convention Record*, vol. 4, pp. 96–104, 1960.

[103] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[104] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard University, 1974.

[105] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proceedings of the National Academy of Sciences 79*, 1982, pp. 2554–2558.

[106] ——, "Neurons with graded response have collective computational properties like those of two-state neurons," in *Proceedings of the National Academy of Sciences 81*, 1984, pp. 3088–3092.

[107] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, no. 6, pp. 455–469, 1982.

[108] D. Reilly, L. Cooper, and C. Elbaum, "A neural model for category learning," *Biological Cybernetics*, vol. 45, no. 1, pp. 35–41, 1982.

[109] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, pp. 59–69, 1982.

[110] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.

[111] M. Arbib, S. Amari, N. S. F. (U.S.), and N. G. Shinkōkai, *Competition and Cooperation in Neural Nets: Proceedings of the U.S.-Japan Joint Seminar Held at Kyoto, Japan, February 15-19, 1982*, ser. Lecture Notes in Biomathematics. Springer-Verlag, 1982.

[112] D. B. Parker, "A comparison of algorithms for neuron-like cells," in *AIP Conference Proceedings 151 on Neural Networks for Computing*. Woodbury, NY, USA: American Institute of Physics Inc., 1987, pp. 327–332.

[113] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[114] Y. LeCun, "A theoretical framework for back-propagation," in *Proceedings of the 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988, pp. 21–28.

[115] S. Grossberg, "Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors," *Biological Cybernetics*, vol. 23, no. 3, pp. 121–134, 1976.

[116] ——, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science*, vol. 11, no. 1, pp. 23–63, 1987.

[117] G. Carpenter, S. Grossberg, and D. Rosen, "Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition," *Neural Networks*, vol. 4, no. 4, pp. 493–504, 1991.

[118] ——, "Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, no. 6, pp. 759–771, 1991.

[119] D. S. Broomhead and D. Lowe, "Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks," *Complex Systems*, vol. 2, pp. 321–355, Mar. 1988.

[120] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2. IEEE, 2004, pp. 985–990.

[121] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[122] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[123] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015.

[124] D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986.

[125] ——, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2: Psychological and Biological Models*. Cambridge, MA, USA: MIT Press, 1986.

[126] L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering*. Wiley Interscience, 1997.

[127] J.-S. R. Jang and C.-T. Sun, *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.

[128] S. Rajasekaran and G. A. V. Pai, *Neural networks, fuzzy logic, and genetic algorithms : synthesis and applications*. New Delhi : Prentice-Hall of India, 2003.

[129] T. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 2, no. 6, pp. 459–473, 1989.

[130] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[131] A. Nowé, P. Vrancx, and Y.-M. De Hauwere, *Reinforcement Learning: State-of-the-Art*. Springer, 2012, ch. Game Theory and Multi-agent Reinforcement Learning, pp. 441–470.

[132] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Int. Res.*, vol. 4, no. 1, pp. 237–285, May 1996.

[133] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, pp. 3056–3062, 2007.

[134] D. Serre, *Matrices : theory and applications*, ser. Graduate texts in mathematics. New York, Berlin, Paris: Springer, 2002.

[135] R. Fullér, *Introduction to neuro-fuzzy systems.*, ser. Advances in soft computing. Physica-Verlag, 2000.

[136] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[137] ——, "Fuzzy algorithms," *Information and Control*, vol. 12, no. 2, pp. 94–102, 1968.

[138] R. Bellman and L. Zadeh, "Decision-Making in a Fuzzy Environment," *Management Science*, vol. 17, 1970.

[139] L. Zadeh, "Similarity relations and fuzzy orderings," *Information Sciences*, vol. 3, no. 2, pp. 177–200, 1971.

[140] ——, "Quantitative fuzzy semantics," *Information Sciences*, vol. 3, no. 2, pp. 159–176, 1971.

[141] ——, "Fuzzy-set-theoretic interpretation of linguistic hedges." *J Cybern*, vol. 2, no. 3, pp. 4–34, 1972.

[142] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes." *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-3, no. 1, pp. 28–44, 1973.

[143] E. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant." *Proceedings of the Institution of Electrical Engineers*, vol. 121, no. 12, pp. 1585–1588, 1974.

[144] E. Mamdani and S. Assilian, "Experiment in linguistic synthesis with a fuzzy logic controller." *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[145] L. P. Holmblad and J.-J. Ostergaard, "Control of a cement kiln by fuzzy logic," in *Readings in Fuzzy Sets for Intelligent Systems*, D. Dubois, H. Prade, and R. R. Yager, Eds. San Mateo, CA: Kaufmann, 1993, pp. 337–347.

[146] N. R. Prasad and M. Sugeno, *Fuzzy Modeling and Control: Selected Works of M. Sugeno*, H. T. Nguyen, Ed. CRC Press, Inc., 1998.

[147] M. Sugeno and M. Nishida, "Fuzzy control of model car," *Fuzzy Sets Syst.*, vol. 16, no. 2, pp. 103–113, Jul. 1985.

[148] L.-X. Wang, *A Course in Fuzzy Systems and Control*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.

[149] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-i," *Information Sciences*, vol. 8, no. 3, pp. 199–249, 1975.

[150] B. Roffel and B. Betlem, *Process Dynamics and Control: Modeling for Control and Prediction*. Wiley, 2007.

[151] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*. New York, NY, USA: John Wiley & Sons, Inc., 1997.

[152] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 6, pp. 693–699, 1985.

[153] J.-S. Jang, "Self-learning fuzzy controllers based on temporal back propagation," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 714–723, 1992.

[154] ——, "Anfis: Adaptive-network-based fuzzy inference system," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[155] J.-S. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378–406, 1995.

[156] C.-T. Lin and G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1320–1336, 1991.

[157] H. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 724–740, 1992.

[158] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," *Fuzzy Sets and Systems*, vol. 89, no. 3, pp. 277–288, 1997.

[159] ——, "Neuro-fuzzy systems for function approximation," *Fuzzy Sets and Systems*, vol. 101, no. 2, pp. 261–271, 1999.

[160] C. Darwin, "On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life," *John Murray, London*, no. 1, pp. 1–556, Oct. 1859.

[161] N. A. Barricelli, "Esempi numerici di processi di evoluzione," *Methodos*, pp. 45–68, 1954.

[162] N. Barricelli, "Symbiogenetic evolution processes realized by artificial methods," *Methodos*, pp. 143–182, 1957.

[163] "Simulation of genetic systems by automatic digital computers. i. introduction," *Aust. J. Biol. Sci*, vol. 10, pp. 484–491, 1957.

[164] H. J. Bremermann, "The evolution of intelligence. The nervous system as a model of its environment." Department of Mathematics, University of Washington, Seattle, Technical Report 1, Contract No. 477(17), 1958.

[165] ——, "Optimization through evolution and recombination," in *Proceedings of the Conference on Self-Organizing Systems – 1962*, M. C. Yovits, G. T. Jacobi, and G. D. Golstine, Eds. Washington, DC: Spartan Books, 1962, pp. 93–106.

[166] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM J. Comput.*, vol. 2, no. 2, pp. 88–105, 1973.

[167] ——, "Erratum: Genetic algorithms and the optimal allocation of trials," *SIAM J. Comput.*, vol. 3, no. 4, p. 326, 1974.

[168] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[169] D. Goldberg, "Computer-aided pipeline operation using genetic algorithms and rule learning. part i: Genetic algorithms in pipeline optimization," *Engineering with Computers*, vol. 3, no. 1, pp. 35–45, 1987.

[170] L. Davis and S. Coombs, "Genetic algorithms and communication link speed design: Theoretical considerations," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987, pp. 252–256.

[171] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[172] E. Sánchez, G. Squillero, and A. P. Tonda, *Industrial Applications of Evolutionary Algorithms*, ser. Intelligent Systems Reference Library. Springer, 2012, vol. 34.

[173] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[174] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[175] V. Cerny, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

[176] A. M. Turing, "Computing machinery and intelligence," vol. 59, no. 236, pp. 433–460, Oct. 1950.

[177] I. Rechenberg, *Evolutionsstrategie, Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog, 1973.

[178] L. Fogel, "System for improving intelligibility," Jan. 5 1960, uS Patent 2,920,138.

[179] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[180] J. Kennedy and R. Eberhart, "Particle swarm optimization," vol. 4, 1995, pp. 1942–1948.

[181] M. Dorigo, "Optimization, learning and natural algorithms (in Italian)," Ph.D. dissertation, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[182] W. K. Macura. (2012) Ant colony algorithm. MathWorld. [Online]. Available: http://mathworld.wolfram.com/AntColonyAlgorithm.html

[183] C. Cortes and V. Vapnik, *Support-Vector Networks*, M. Learning, Ed. Kluwer Academic Publishers, 1995.

[184] V. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[185] O. Castillo, P. Melin, W. Pedrycz, and J. Kacprzyk, *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*. Springer Publishing Company, Incorporated, 2014.

[186] P. Angelov and R. Buswell, "Identification of evolving fuzzy rule-based models," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 5, pp. 667–677, 2002.

[187] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, "Ten years of genetic fuzzy systems: Current framework and new trends," *Fuzzy Sets and Systems*, vol. 141, no. 1, pp. 5–31, 2004.

[188] F. Leung, H. Lam, S. Ling, and P. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 79–88, 2003.

[189] D. Floreano, P. Durr, and C. Mattiussi, "Neuroevolution: From architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[190] J.-S. Chiou, S.-H. Tsai, and M.-T. Liu, "A pso-based adaptive fuzzy pid-controllers," *Simulation Modelling Practice and Theory*, vol. 26, pp. 49 – 59, 2012.

[191] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez, "Optimal design of fuzzy classification systems using pso with dynamic parameter adaptation through fuzzy logic," *Expert Systems with Applications*, vol. 40, no. 8, pp. 3196–3206, 2013.

[192] A. Abdelbar, S. Abdelshahid, and D. Wunsch II, "Fuzzy pso: A generalization of particle swarm optimization," vol. 2, 2005, pp. 1086–1091.

[193] C.-F. Juang, "A tsk-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 155–170, 2002.

[194] B. Samanta, K. Al-Balushi, and S. Al-Araimi, "Artificial neural networks and support vector machines with genetic algorithm for bearing fault detection," *Engineering Applications of Artificial Intelligence*, vol. 16, no. 7-8, pp. 657–665, 2003.

[195] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to time series analysis and forecasting*. Wiley-Interscience, Mar. 2008.

[196] Enagas. http://www.enagas.es/portal/site/enagas. [Online]. Available: http://www.enagas.es/portal/site/enagas

[197] Aemet: Agencia estatal de meteorología http://www.aemet.es/en/portada. [Online]. Available: http://www.aemet.es/en/portada

[198] M. North, *Data Mining for the Masses*, ser. A Global text project book. Global Text Project, 2012.

[199] D. Moore, G. McCabe, and B. Craig, *Introduction to the Practice of Statistics*. W. H. Freeman, 2010.

[200] P. K. Janert, *Data Analysis with Open Source Tools*, 1st ed. O'Reilly Media, Inc., 2010.

[201] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[202] A. Khotanzad and C. Chung, "Application of multi-layer perceptron neural networks to vision problems," *Neural Computing & Applications*, vol. 7, no. 3, pp. 249–259, 1998. [Online]. Available: http://dx.doi.org/10.1007/BF01414886

[203] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jul. 1989. [Online]. Available: http://dx.doi.org/10.1016/0893-6080(89)90020-8

[204] R. Fullér, *Introduction to neuro-fuzzy systems.*, ser. Advances in soft computing. Physica-Verlag, 2000.

[205] http://www.optimitive.com/.

[206] S. Seijo, I. del Campo, J. Echanobe, and J. García-Sedano, "Modeling and multi-objective optimization of a complex chp process," *Applied Energy*, vol. 161, no. Complete, pp. 309–319, 2016.

[207] MATLAB, *version 8.1.0 (R2013a)*. Natick, Massachusetts: The MathWorks Inc., 2013.

[208] D. Mattera and S. Haykin, "Advances in kernel methods," B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, ch. Support Vector Machines for Dynamic Reconstruction of a Chaotic System, pp. 211–241.

[209] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[210] D. Mladenic, "Feature selection for dimensionality reduction." in *SLSFS*, ser. Lecture Notes in Computer Science, C. Saunders, M. Grobelnik, S. R. Gunn, and J. Shawe-Taylor, Eds., vol. 3940. Springer, 2005, pp. 84–102.

[211] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Class prediction by nearest shrunken centroids, with applications to dna microarrays," *Statistical Science*, vol. 18, no. 1, pp. 104–117, 2003.

[212] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[213] S. Seijo, V. Martínez, I. Campo, J. Echanobe, and J. García-Sedano, *Proceedings of ELM-2015 Volume 1: Theory, Algorithms and Applications (I)*. Cham: Springer International Publishing, 2016, ch. Feature Selection and Modelling of a Steam Turbine from a Combined Heat and Power Plant Using ELM, pp. 435–445. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-28397-5_34

[214] M. Caramia, P. DellŽOlmo, and SpringerLink, *Multi-objective Management in Freight Logistics*. Springer London„ 2008.

[215] A. Pryke, S. Mostaghim, and A. Nazemi, "Heatmap visualization of population based multi objective algorithms," in *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, ser. EMO'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 361–375.

[216] S. A. Soliman and A.-A. H. Mantawy, *Modern optimization techniques with applications in electric power systems*, ser. Energy systems. Springer, 2012.

[217] Mishra S., "Optimization studies for physics problems in indian PHWRs," Ph.D. dissertation, Homi Bhabha National Institute, April 2012.

[218] R. M. Lewis, V. Torczon, and M. W. Trosset, "Direct search methods: then and now," *Journal of Computational and Applied Mathematics*, vol. 124, pp. 191 – 207, 2000.

[219] G. B. Dantzig and M. N. Thapa, *Linear Programming 1: Introduction*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1997.

[220] S. C. Chapra and R. Canale, *Numerical Methods for Engineers*, 5th ed. New York, NY, USA: McGraw-Hill, Inc., 2006.

[221] J. Gondzio, "Interior point methods 25 years later," *European Journal of Operational Research*, vol. 218, no. 3, pp. 587–601, 2012.

[222] J. Arora, *Introduction to Optimum Design*, ser. MATLAB examples. Academic Press, 2011.

[223] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, Sep. 1994. [Online]. Available: http://dx.doi.org/10.1162/evco.1994.2.3.221

[224] T. Mathworks. Global Optimization Toolbox User's Guide. [Online]. Available: http://in.mathworks.com/help/pdf_doc/gads/gads_tb.pdf

[225] K. Cheon, J. Kim, M. Hamadache, and D. Lee, "On replacing pid controller with deep learning controller for dc motor system," *Journal of Automation and Control Engineering Vol*, vol. 3, no. 6, 2015.

[226] M. Miskuf and I. Zolotova, "Comparison between multi-class classifiers and deep learning with focus on industry 4.0," in *2016 Cybernetics Informatics (K I)*, Feb 2016, pp. 1–5.

[227] M. A. Braun, P. K. Shukla, and H. Schmeck, "Obtaining optimal pareto front approximations using scalarized preference information," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 631–638. [Online]. Available: http://doi.acm.org/10.1145/2739480.2754674

[228] M. Braun, S. Seijo, J. Echanobe, P. Shukla, I. del Campo, J. Garcia-Sedano, and H. Schmeck, "A neuro-genetic approach for modeling and optimizing a complex cogeneration process," *Applied Soft Computing*, vol. 48, pp. 347 – 358, 2016.

# Appendix IV: Nomenclature

$\varepsilon_{EE}$ Effective Electric Efficiency

$CO_2$ Carbon Dioxide

**ACOGEN** Spanish Association for the Promotion of Cogeneration

**ACO** Ant Colony Optimization

**ANN-BP** Artificial Neural Network trained with Back Propagation

**ANNs** Artificial Neural Networks

**ART** Adaptive Resonance Theory

**BP** Back Propagation

**CHP** Combined Heat and Power

**CI** Computational Intelligence

**COGEN** European Association for the Promotion of Cogeneration

**DL** Deep Learning

**DNN** Deep Neural Network

**EAs** Evolutionary algorithms

**EC** European Community's

**EED** Energy Efficiency Directive

**EFs** Evolutionary Fuzzy Systems

**ELMs** Extreme Learning Machines

**EPA** United States Environmental Protection Agency

**ESPEA** Electrostatic Potential Energy Evolutionary Algorithm

**EU** European Union

**FERC** Federal Energy Regulatory Commission

**FISs** Fuzzy Inference Systems

**FPSO** Fuzzy Particle Swarm Optimization

**GAs** Genetic Algorithms

**GDM** Gradient Descent Method

**GFSs** Genetic Fuzzy Fystems

**LSE** Least Squares Estimation

**MAE** Mean Absolute Error

**MF** Membership Function

**MLP** Multiple Layer Perceptron

**NEs** Neuro Evolutionary Systems

**NFs** Neuro-Fuzzy Systems

**NSGA-II** Nondominated Sorting Genetic Algorithm-II

**NSGA** Non-dominated Sorting Genetic Algorithm

**PCA** Principal Component Analysis

**PDP** Parallel Distributed Processing

**PSO** Particle Swarm Simulation

**PURPA** Public Utilities Regulatory Policies Act

**RBF** Radial Basis Function

**SA** Simulated Annealing

**SHP** Separated Heat and Power

**SLFN** Single Hidden-layer Feed-forward Neural Network

**SOM** Self-Organizing Map

**SVM** Support Vector Machines