Human mental states classification using EEG by means of Genetic Programming

Emigdio Z-Flores

Human mental states classification using EEG by means of Genetic Programming

Emigdio Z.Flores López

SEP Tecnológico Nacional de México

Instituto Tecnológico de Tijuana

Departamento de Ingeniería Eléctrica y Electrónica

Clasificación de estados mentales usando EEG por medio de Programación Genética

Trabajo de tesis presentado por:
Emigdio Z.Flores López

para obtener el grado de:
Doctor en Ciencias de la Ingeniería

Director de tesis:
Dr. Leonardo Trujillo

Co-Director de tesis:
Dr. Pierrick Legrand

Tijuana, Baja California, México.
5 de Julio de 2017

SEP Tecnológico Nacional de México

Instituto Tecnológico de Tijuana
Departamento de Ingeniería Eléctrica y Electrónica

Human mental states classification using EEG by means of Genetic Programming

Thesis submitted by:
Emigdio Z.Flores López

to obtain the degree of:
Doctor in Engineering Sciences

Advisor:
Dr. Leonardo Trujillo

Co-Advisor:
Dr. Pierrick Legrand

Tijuana, Baja California, México.
July 5, 2017

SEP
SECRETARÍA DE
EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Tijuana
División de Estudios de Posgrado e Investigación

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

Tijuana, B.C., 10/Junio/2017
ASUNTO: Autorización de impresión de Trabajo de Tesis

LIC. DOROTEO LUNA CASTAÑEDA
JEFE DEL DPTO. DE SERVICIOS ESCOLARES
PRESENTE

En lo referente al trabajo de tesis, **"Clasificación de estados mentales usando EEG por medio de Programación Genética"**, presentado por el **M.C. Emigdio Z.Flores López**, alumno del programa de Doctorado en Ciencias de la Ingeniería, con número de control **D97212354**; informamos a usted que después de una minuciosa revisión e intercambio de opiniones, los miembros del comité manifiestan APROBAR LA TESIS, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias, por lo que se autoriza al interesado para que proceda de inmediato a la impresión del mismo.

ATENTAMENTE
"Por una juventud integrada al desarrollo de México"

Legrand

DR. PIERRICK LEGRAND
PRESIDENTE

DR. LEONARDO TRUJILLO REYES
SECRETARIO

DR. LUIS NÉSTOR CORIA DE LOS RIOS
VOCAL

Ricardo Cárdenas

DR. JOSÉ RICARDO CÁRDENAS
VOCAL

DR. NOHÉ RAMÓN CÁZAREZ CASTRO
VOCAL

DR. VICTOR HUGO DÍAZ RAMÍREZ
SUPLENTE

C.c.p. Oficina de Titulación
División de Estudios de Posgrado e Investigación
Expediente
Interesado
PAVT

RESUMEN

Los avances en el desarrollo de Interfaces Cerebro-Computadora (BCI, por sus siglas en inglés *Brain-Computer Interface*) se han incrementado en años recientes, principalmente porque ha evolucionado el nivel de convergencia de técnicas multidisciplinarias. La electroencefalografía (EEG), una teécnica de grabación de señales cerebrales estudiado en esta tesis, permite la construcción de BCIs, sin embargo las señales son complejas para procesar, lo cual requiere metodologías que extraigan patrones de forma eficiente. Esta tesis explora dos tópicos principales: primero, se propone un sistema para el reconocimiento de convulsiones de epilepsia usando una combinacion de metodos de procesamiento de señales para la extracción eficiente de rasgos; segundo, explora el uso de un algoritmo meta-heurístico, Programación Genética (GP, por sus siglas en inglés *Genetic Programming*), como una alternativa en el diseño de BCIs. Sin embargo, existen temas sin resolver in GP que esta tesis explora: ¿existe una metodologia de búsqueda en GP más eficiente?; ¿cual es una representación apropiada dependiendo del problema a estudiar?; ¿cual son los operadores de búsqueda más adecuados?. De esta forma, se presenta un estudio a fondo con la introducción de un GP memetico aplicado a problemas de regresión. Despues, se extiende adaptandolo a problemas de clasificación. Los resultados son positivos; GP se beneficia fuertemente de la combinación de una metodologia general de busqueda y una local (LS, por sus siglas en inglés *Local Search*). Los ultimos dos cuestionamientos se estudian simultáneamente en el desarrollo de un sistema de reconocimiento para estados mentales usando EEG. Se propone una versión de GP ($^{+}$FEGP) que evoluciona modelos de extracción de rasgos usando operadores especializados de busqueda, representación de individuos y función de aptitud. Los resultados muestran que esta combinación permite una exactitud de clasificación que aporta en el estado-del-arte para la tarea particular del reconocimiento de estados mentales.

Palabras clave: EEG, Extracción de Rasgos, Regularidad Hölderiana, Matching Pursuit, Programación Genética, Regresión, Clasificación.

ABSTRACT

The advances in the development of Brain-Computer Interfaces (BCI) have been increasing in recent years, mostly because the level of convergence from multi-disciplinary techniques has evolved. The electroencephalography (EEG), a brain recording method studied in this thesis, allows the construction of BCIs, however the signals are rather complex to process, which requires methodologies that efficiently extract patterns from them. This thesis explores two directions: first, a system is proposed for the epilepsy seizures recognition using a combination of signal processing methods for an efficient feature extraction; second, it explores the usage of a meta-heuristic algorithm, namely Genetic Programming (GP), as an alternative in the design of BCIs. Nonetheless, there is currently open-issues in GP that this thesis also explores: is there a more efficient search methodology in the exploration by GP?; what is a proper representation depending on the studied problem?; which are the most adequate search operators?. For the first topic, a thoroughly study is presented by introducing a memetic GP applied to regression problems. Then, it is extended by adapting it to classification problems. The results are positive; GP is greatly benefited from the combination of a general and a Local Search (LS) methodology. The last two topics are studied simultaneously in the development of a recognition system for mental states using EEG. A GP version ($^{+}$FEGP) is proposed that evolves feature extraction models by using specialized search operators, individuals representation and fitness function. The results show that the combination of these reaches a state-of-the-art accuracy for the particular task of mental states recognition.

Keywords: EEG, Feature Extraction, Hölderian Regularity, Matching Pursuit, Genetic Programming, Regression, Classification.

To God, my beloved parents, my wife and my daughter

ACKNOWLEDGEMENTS

Firstly, I would like to express my gratitude to my advisor Dr. Leonardo Trujillo Reyes for the continuous support during my doctoral timeline, for his patience, determination, motivation and great vision. During the doctoral road there were several moments where I had hesitations, nevertheless his guidance showed me better paths, revealing me what a great mentor he is.

Besides my advisor, I would like to specially thank to my co-advisor Dr. Pierrick Legrand, for the uncountable times where he gave me fully selfless support with time, guidance, patience, resources and even a good friendship. I thank to his family that gives him the energy to allow him to offer the best of him.

My sincere thanks also goes to the rest of my thesis committee: Dr. Luis Nestor Coria de los Ríos, Dr. Nohé Ramón Cazarez Castro, Dr. José Ricardo Cárdenas Valdez and Dr. Victor Hugo Díaz Ramírez, for their insightful comments, but also for the tough questions which encouraged me to improve my research. My gratitude also goes to other professors from the Postgraduate program that directly or indirectly influenced this dissertation: Dr. Yazmín Maldonado Robles, Dr. Youness El Hamzaoui, Dr. Paul Valle Trujillo and Ms. Arturo Sotelo.

I thank my fellow labmates for the times that we shared joy and frustration along these 4 years: Enrique, *the Elder*; Yuliana, *the Enthusiast*; Victor, *the Joker*; Uriel, *the Kramer*; Angel, *the Instigator*; Perla, *the Dreamer*; Ramón, *the Xolo*; Carlos, *the Conspirator*; Paul, *the Señor*; Luis, *the Erudite*; Pablo, *the Cuban*; Adrian, *the Jungle boy*.

I would like to thank to CONACYT, that allows young prospects to pursue their dreams providing a scholarship (No. 294213) through postgraduate programs across the country. I would like to express my gratitude to the involved people in the FP7-Marie Curie-IRSES 2013 European Commission program, through the ACoBSEC project, whom gave me full support in multiple occasions, including internships in Bordeaux, France. Specially, I thank to Dr. Frédérique Faïta-Aïnseba and her team for making an excellent job during the EEG recording sessions.

In a personal note, I would like to give special thanks to my wife, Anabel, for her unconditional support and patience along this journey; without her I could not succeeded. Last but not the least, special thanks goes to my parents, Emigdio and Olga, that taught me the essential tools to persevere in my goals and their unconditional love; and to my sisters, Celia, Lucía and María, for their encouragement along this journey.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

A Brain-Computer Interface (BCI) is a communication system that allows a person to send and receive control signals toward a external device (usually a computer) by means of a brain generated signal. Such systems have been valuable in different scenarios, like entertainment, security, education, neuromarketing, neuroergonomics and medical (Abdulkader et al., 2015), with the latter serving as the initial inspiration for the development of BCIs. Indeed, the healthcare field is one of the top priorities for human development, where a variety of applications have employed BCI including prevention, detection, diagnostic, rehabilitation and restoration.

However, the interest in other fields has been increasing over time. For example, in the area of neuroergonomics and smart environments, the exploitation of BCIs has been seen to improve the quality of the human's daily life, like the control of smart houses, workplaces or transportation (Peng et al., 2011).

Brain signals can also potentially be used to assist the improvement of workplace conditions by the estimation of an operator' cognitive state (Venthur et al., 2010). The field of intelligent transportation has also been benefited from the cognitive state by monitoring BCI functions. In this domain, the behavior of drivers has been reviewed in numerous studies. It has been found that distraction and fatigue are considered a strong cause for most traffic accidents (Dong et al., 2011). Being recognized that the mental state of a person is critical in several domains, it is one of the main reasons behind the motivation of this thesis.

Designing BCIs systems is a complex task that requires multidisciplinary works related to computer science, neuroscience and psychology. The design of such a system usually involves several elements:

1. **Brain activity capture:** Brain signals are recorded by means of a sensor, such as Electroencephalogram (EEG), the method used in this work.

2. **Pre-processing:** Recorded data is cleaned and denoised.

3. **Feature extraction:** Input data is transformed into a compact feature space.

4. **Classification:** An algorithm is used to build a decision model upon the feature set and the tasks to be recognized.

5. **Translation into a command:** A command is associated to the mental task or state to control a given application.

6. **Feedback:** It provides a feedback to the user, which improves the user's performance.

This thesis is focused on the first four components, with the main original contributions oriented toward the feature extraction, while the last two components are beyond the scope of this work.

1.1 THESIS OBJECTIVES

The work presented in this manuscript belongs to a combination of BCI research, signal processing, machine learning and Genetic Programming (GP) areas. The main goal of this research is focused in the development of EEG recognition systems, first from a classical perspective, that is using tools from signal processing and statistical learning; and second, using an alternative meta-heuristic approach. In the former view, we will explore the current advances in the development of EEG recognition systems and thus propose our own methodology. The design of efficient BCI systems is still considered an open topic, mainly because its complexity is considered high. A wide range of tools and algorithms have been used to design different functional layers of BCIs, either separately or combined. The majority of the research done in this domain is rooted from ideas and concepts derived from traditional optimization and signal processing. An alternative to this point of view are meta-heuristic based algorithms, a superset of tools inspired by a different reasoning: the exact optimal solution is not fundamental to solve the problem at hand. In the broad group of meta-heuristic algorithms, GP stands as a powerful technique that does not require explicit prior knowledge of the data. Despite that GP has been used in different parts of BCI design, the fact remains that GP fundamentally suffers from limitations that we will try to address in the second part of the manuscript. This direction is important because a successful adaptation of GP toward a specific application depends in a large scale on how these drawbacks are handled. Finally, we will combine all prior knowledge to propose a new methodology to solve the recognition of mental states using EEG, based on a GP algorithm.

1.2 THESIS ORGANIZATION AND CONTRIBUTIONS

The diagram in Figure 1.1 shows a complete synopsis of the thesis roadmap in terms of the research directions taken during this journey. This map directly correlates with the presented topics in this manuscript. Each of the roads represents different research areas that contributes to the main goals of the thesis. Understanding this diagram helps, first, to locate each event and the domain where it resides, and second, to comprehend the convergence of the secondary roads into the main road, and their interdependencies along the doctoral timeline.

Parting from this map, a type of guideline for the explored research areas, we now change the direction toward a more pragmatic viewpoint: the manuscript organization.

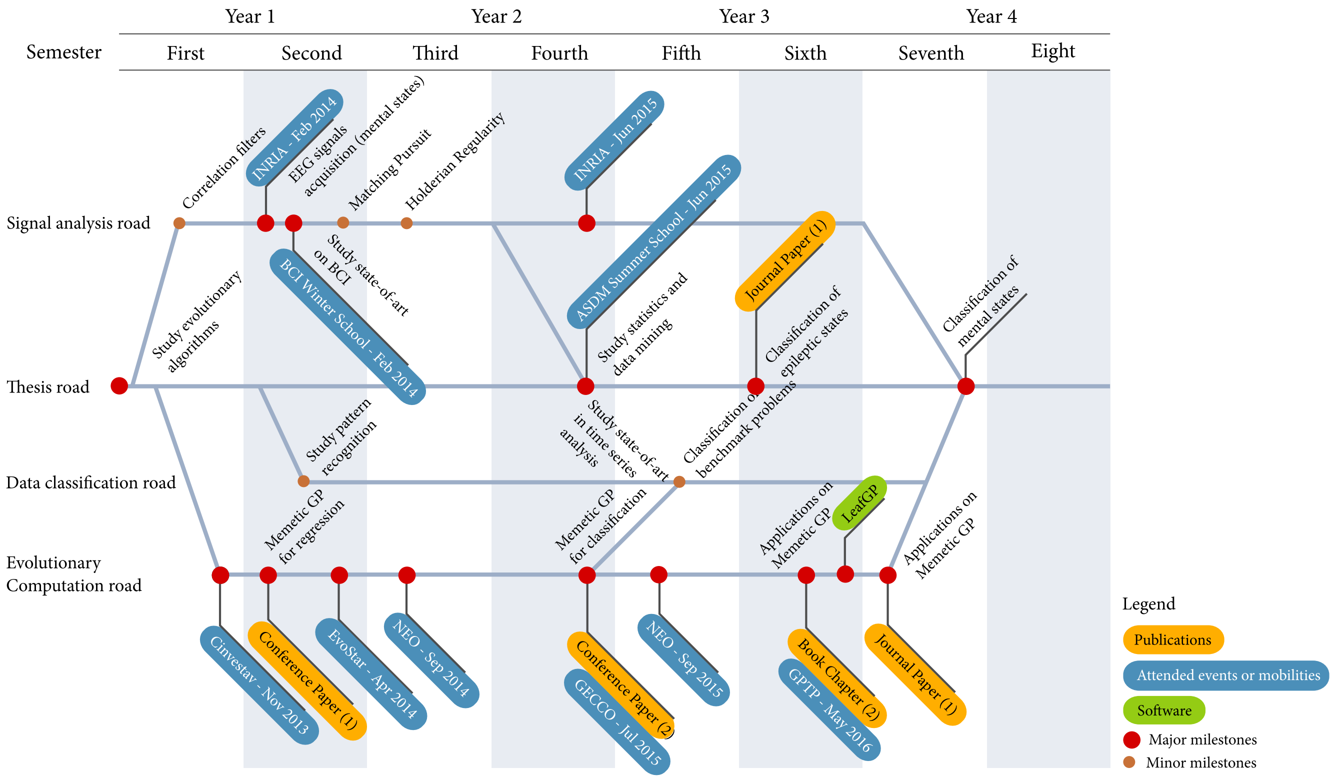


Figure 1.1: PhD roadmap and timeline.

Part I: Epileptic seizures classification

In the first part of the document, we are going to examine the BCI design from a classical perspective, that is the current advances in EEG recognition are going to be discussed, as well as our contribution for a specific application in this domain. In this part, two chapters cover the aforementioned standpoint.

Chapter 2 of this manuscript will be focused on an overview of the current BCI design methodologies, including acquisition, preprocessing, feature extraction and classification.

As briefly mentioned before, one of the main topics that this manuscript is trying to target is the design of efficient feature extraction methods. In Chapter 3, we will present current advances in methods and algorithms used to solve the problem of feature extraction and classification of EEG signals. Moreover, we will focus to the recognition of epileptic seizures, a relative common condition present in a large percentage of the worldwide population. In particular, we propose a novel methodology to design a robust feature extraction block, by employing a set of different measures. In particular, two techniques are combined to build this feature set. First, the Matching Pursuit is used to describe the EEG signals as a global regularity measure; and second, the Hölderian regularity, that works as a local descriptor measure. In the latter measure, an approximation of the measure is calculated with a method never used before in this domain. Additional simpler statistical features are used to compose the full feature set. Our comprehensive study shows that this new methodology outperforms the majority of other recent works (Z-Flores et al., 2016).

Part II: Genetic Programming

The previous part of the investigation worked as a preamble to illustrate the most noteworthy difficulties normally found when handling EEG signals and building a BCI system; which were solved for a specific problem instance. Nevertheless, the complexity of the general problem requires to explore new alternatives. In that sense, the introduction of meta-heuristic techniques are viable options, mainly because they do not necessarily requires that a global optimum solution is found to solve the problem. They are particularly useful when the underlying data size is large and complex, that is each category is not easily distinguishable between them using simpler algorithms. In this part, we introduce GP as a particular meta-heuristic algorithm of interest. GP can adapt to a wide range of applications making it a valuable candidate to address the problem at hand. However, we will state that GP itself requires a series of changes to be successful in the application. Since its conception, it was designed to work as a highly abstract algorithm that eventually could handle very complex situations, but this carries as well some disadvantages. One of these limitations is the way the search is performed (O'Neill et al., 2010); it is not optimal in terms of convergence time or solution complexity. Thus, in this part we try to address this limitation by incorporating the term of Local Search (LS) into GP, a well known technique used in other meta-heuristic algorithms. By combining global and local search operators, the overall performance is increased.

A study of the incorporation of LS strategies into GP is presented in Chapter 4. We will explore different variants in the methodology and what are the consequences for the search by using several performance measurements. The experimental work will target symbolic regression problems as the first of a two-parts study (Z-Flores et al., 2014), with the second part discussed in the following chapter.

Appendix A further extends the experimentation of symbolic regression using LS with GP, applied to real-world problems (Trujillo et al., 2017).

Appendixes B, C and D focus in the modeling of three contrasting real-world problems using the proposed memetic GP. In appendix B, experimental data gathered from an electrical gas turbine is modeled by GP with LS. Furthermore, a comparison between another GP based technique and other machine learning algorithms is presented with outstanding performance for the models produced by our memetic approach (Enríquez-Zárate et al., 2017). Moreover, the effects of amplitude and phase in Radio Frequency (RF) amplifiers are modeled in appendix C producing competitive results compared with more commonly used approaches (Cárdenas Valdez et al., 2017). In the consequent appendix D, models are derived from the removal process of two chemical contaminants commonly present in wastewater. In summary, for all three case studies, GP with LS produced better and compacter models than other state-of-the-art techniques.

In the last appendix, E, a software program is presented with the aim to provide a modeling tool that is modern and user-friendly. The core of this tool is powered by GP with the capability of running in parallel hardware. Currently, it is still in development.

In Chapter 5, the previous study will be extended by addressing classification problems. In there, various modifications are introduced, like the chosen heuristic for LS selection or the proposal for a fitness function that handles binary classification problems. The methodology is tested using real-world benchmark problems (Z-Flores et al., 2015).

Part III: Mental states classification

In the Part II we discussed one of the deficiencies of GP as an efficient global searcher. However, is not the only open issue that still exists in GP (O'Neill et al., 2010). Another important topic is a proper representation of the individuals. The relationship between the representation and the problem at hand is strong if the solution needs to be found in a very efficient way. Additional parameters affect the overall performance of GP, very similarly as other population-based algorithms, like the type of population initialization or the characteristics of the search operators. All these aspects will be discussed in Part III of this manuscript through the mental states classification.

Even within the field of BCI, there are several levels of difficulties and not every problem can be handled using the same approach. In the case of mental states recognition, the problem is considered complex because apart of the common differences in the EEG signals in terms of the cognitive states, the signals carry a combination of other conditions either induced or already present during the experiment. Focusing again in the design of an efficient feature extraction, we will present a GP variant that we expect to handle this problematic in a effective manner. For that, we will introduce several modifications into GP.

In Chapter 6, we will conclude this journey by addressing one of the original objectives of the thesis: to design a BCI system which will be able to recognize human mental states using EEG by means of GP. In particular, we will use GP to design the feature extraction model. However, we will present a full system including the necessary BCI parts like the brain activity capture, pre-processing, feature extraction and classification. This system is build upon a reference system using classical techniques (Vézard et al., 2015). Since GP plays, in this chapter, the main role, we will focus heavily in the discussion of our original contribution: a new representation, new search or genetic operators, a new fitness function and a new methodology for terminal selection; we will name this contribution as Augmented Feature Extraction with Genetic Programming ($^{+}$FEGP). The algorithm will be supported by the necessary experimental evidence.

At Chapter 7, the general thesis conclusions are going to be presented.

1.3 SUMMARY OF PUBLICATIONS

In the following section there is a list of publications produced by the doctoral research.

Journal articles

Published

1. **Z-Flores, E.**, Abatal, M., Bassam, A., Trujillo, L., Juárez-Smith, P., and El Hamzaoui, Y. (2017a). Modeling the Adsorption of Phenols and Nitrophenols by Activated Carbon using Genetic Programming. *Journal of Cleaner Production*, available online

2. **Z-Flores, E.**, Trujillo, L., Sotelo, A., Legrand, P., and Coria, L. N. (2016). Regularity and Matching Pursuit feature extraction for the detection of epileptic seizures. *Journal of Neuroscience Methods*, 266:107-125

3. Enríquez-Zárate, J., Trujillo, L., de Lara, S., Castelli, M., **Z-Flores, E.**, Muñoz, L., and Popovic, A. (2017). Automatic modeling of a gas turbine using genetic programming: An experimental study. *Applied Soft Computing*, 50:212-222

Submitted

4. **Z-Flores, E.**, Trujillo, L., Legrand, P., and Faïta-Aïnseba, F. (2017b). EEG Feature Extraction using Genetic Programming for the Classification of Mental States. *Neurocomputing*

5. Galaviz, J.-A., Roblin, P., Cárdenas, J. R., **Z-Flores, E.**, Trujillo, L., Nuñez, J.-C., and Schütze, O. (2017). Comparison of a Novel Genetic Programming Approach with ANFIS for Power Amplifier Modeling and FPGA Implementation. *Soft Computing*

Conference papers

1. **Z-Flores, E.**, Trujillo, L., Schütze, O., and Legrand, P. (2014). Evaluating the Effects of Local Search in Genetic Programming. In *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, volume 288, pages 213-228

2. **Z-Flores, E.**, Trujillo, L., Schütze, O., and Legrand, P. (2015). A Local Search Approach to Genetic Programming for Binary Classification. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO'15, pages 1151-1158.ACM

3. Castelli, M., Trujillo, L., Vanneschi, L., Silva, S., **Z-Flores, E.**, and Legrand, P. (2015e). Geometric Semantic Genetic Programming with Local Search. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 999-1006. ACM

Book chapters

1. Trujillo, L., **Z-Flores, E.**, Juárez-Smith, P., Legrand, P., Castelli, M., Vanneschi, L., Schütze, O., and Muñoz, L. (2017). Local Search is Underused in Genetic Programming. In *Genetic Programming Theory and Practice XIV*, number January

2. Cárdenas Valdez, J. R., **Z-Flores, E.**, Núñez Pérez, J. C., and Trujillo, L. (2017). Local Search Approach to Genetic Programming for RF-PAs Modeling Implemented in FPGA. In *NEO 2015*, Studies in Computational Intelligence, pages 67-88. Springer

Part I

Epileptic seizures classification

2

AN OVERVIEW OF BCI

2.1 INTRODUCTION

A BCI is an artificial intelligence system that can recognize a certain set of patterns in brain signals following five consecutive stages: signal acquisition, preprocessing or signal enhancement, feature extraction, classification, and the control interface (Nicolas-Alonso and Gomez-Gil, 2012). The signal acquisition stage captures the brain signals and may also perform noise reduction and artifact processing. The preprocessing stage prepares the signals in a suitable form for further processing. The feature extraction stage identifies discriminative information in the brain signals that have been recorded. Once measured, the signal is mapped onto a vector containing effective and discriminant features from the observed signals. The extraction of this interesting information is a very challenging task. Brain signals are mixed with other signals coming from a finite set of brain activities that overlap in both time and space. Moreover, the signal is not usually stationary and may also be distorted by artifacts such as electromyography (EMG) or electrooculography (EOG) (Nicolas-Alonso and Gomez-Gil, 2012). The feature vector must also be of a low dimension, in order to reduce feature extraction stage complexity, but without relevant information loss. The classification stage classifies the signals taking the feature vectors into account. The choice of good discriminative features is therefore essential to achieve effective pattern recognition, in order to decipher the user's intentions. Finally the control interface stage translates the classified signals into meaningful commands for any connected device, such as a wheelchair or a computer. In the following sections, a brief explanation of each stage in BCI is presented, with the exception of the control interface, which is outside of the scope of this manuscript.

2.2 SIGNAL ACQUISITION

The human being is often referred as the subject and is considered the source of the brain signals (Abdulkader et al., 2015). The measurement of the neurophysilogic state of the brain is called signal acquisition. There are two types of brain activities that may be monitored: electrophysiological and hemodynamic.

Electrophysiological activity is generated by electro-chemical transmitters exchanging information between the neurons. The neurons generate ionic currents which flow within and across neuronal assemblies. The large variety of current pathways can be simplified as a dipole conducting current from a source to a sink through the dendritic trunk. These intracellular currents are known as primary currents. Conservation of electric charges means that the primary currents are enclosed by extracellular current flows, which are known as secondary currents (Baillet et al., 2001). Electrophysiological activity is measured by electroencephalography, electrocorticography, magnetoencephalography, and electrical signal acquisition in single neurons.

The hemodynamic response is a process in which the blood releases glucose to active neurons at a greater rate than in the area of inactive neurons. The glucose and oxygen delivered through the blood stream results in a surplus of oxyhemoglobin in the veins of the active area, and in a distinguishable change of the local ratio of oxyhemoglobin to deoxyhemoglobin (Laureys et al., 2009). These changes can be quantified by neuroimaging methods such as functional magnetic resonance and near infrared spectroscopy. These kinds of methods are categorized as indirect, because they measure the hemodynamic response, which, in contrast to electrophysiological activity, is not directly related to neuronal activity.

Most current BCIs obtain the relevant information from the brain activity through electroencephalography. Electroencephalography is by far the most widely used neuroimaging modality, owing to its high temporal resolution, relative low cost, high portability, and few risks to the users. BCIs based on electroencephalography consist of a set of sensors that acquire electroencephalography signals from different brain areas. However, the quality of electroencephalography signals is affected by scalp, skull, and many other layers as well as background noise. Noise is key to electroencephalography and to other neuroimaging methods, insofar as it reduces the SNR and therefore the ability to extract meaningful information from the recorded signals.

In general, there are two classes of brain acquisition methods: invasive and non-invasive methods. Each type will be further discussed below.

2.2.1 *Invasive methods*

Invasive modalities need to implant microelectrode arrays inside the skull that involves significant health risks, which restricts their use to experimental settings. Two invasive modalities can be found in BCI research: electrocorticography, which places electrodes on the surface of the cortex, either outside the dura mater (epidural electrocorticography) or under the dura mater (subdural electrocorticography) (Roland et al., 2013), and intracortical neuron recording which implants electrodes inside the cortex (Hochberg et al., 2006).

2.2.1.1 *Electrocorticography (ECoG)*

ECoG is a recording method that brings a less invasive option while at the same time preserves the advantages of invasive approach. It involves implanting electrode grids or strips over the cortex surface through a surgical operation (Roland et al., 2013). ECoG is an invasive recording modality which requires a craniotomy to implant an electrode grid, entailing significant health hazards. For that reason, the first studies on ECoG were with animals (Yuen et al., 1987). In humans, ECoG has been used for the analysis of alpha and beta waves or gamma waves Crone et al. (1998) produced during voluntary motor action.

2.2.1.2 *Intracortical*

Intracortical neuron recording is a neuroimaging technique that measures electrical activity inside the gray matter of the brain. It is an invasive recording modality that needs to implant microelectrode arrays inside the cortex to capture spike signals and local field potentials from neurons; represents the most invasive method. Three signals can be obtained by intracortical neuron recording: single-unit activity (SUA), multi-unit activity (MUA), and local field potentials (LFPs) (Waldert et al., 2009).

2.2.2 *Non-invasive methods*

These recording methods follow the approach that does not require implanting of external objects into subject's brain. Thus it avoids the surgical procedures or permanent device attachment needed by invasive acquisition. These include functional magnetic resonance imaging, magnetoencephalography, functional near-infrared spectroscopy and electroencephalography.

2.2.2.1 *Functional Magnetic Resonance Imaging (fMRI)*

fMRI is a non-invasive neuroimaging technique which detects changes in local cerebral blood volume, cerebral blood flow and oxygenation levels during neural activation by means of electromagnetic fields. It depends on the fact that any usage of brain part requires the increase of incoming blood flow. It uses blood-oxygen-level-dependent (BOLD) contrast, which is sensitive to the hemodynamic response (Ayaz et al., 2011). The main advantage of the use of fMRI is high space resolution.

2.2.2.2 *Magnetoencephalography (MEG)*

MEG is a non-invasive imaging technique that registers the brain's magnetic activity by means of magnetic induction. MEG measures the intracellular currents flowing through dendrites which produce magnetic fields that are measurable outside of the head (Waldert et al., 2009). The neurophysiological processes that produce MEG signals are identical to those that produce electroencephalography signals. The magnetic signal outside of the head is currently acquired only using the Superconducting Quantum Interference Device (SQUID) (Babiloni et al., 2009).

2.2.2.3 *Functional Near-Infrared Spectroscopy (fNIRS)*

fNIRS is a noninvasive technique that measures blood dynamic in the brain in order to detect the neuronal activity. It uses light in the near-infrared range to determine the blood flood. Infrared light penetrates the skull to a depth of approximately 1–3 cm below its surface, where the intensity of the attenuated light allows alterations in oxyhemoglobin and deoxyhemoglobin concentrations to be measured (Coyle et al., 2007).

2.2.2.4 *Electroencephalography (EEG)*

EEG measures electric brain activity caused by the flow of electric currents during synaptic excitations of the dendrites in the neurons and is extremely sensitive to the effects of secondary currents (Baillet et al., 2001). It was first used in humans by H. Berger in 1924. EEG signals are easily recorded in a non-invasive manner through electrodes placed on the scalp, for which that reason it is by far the most widespread recording modality. However, it provides very poor quality signals as they have to cross the scalp, skull, and many other layers. This means that EEG signals in the electrodes are weak, hard to acquire and of poor quality. This technique is moreover severely affected by background noise generated either inside the brain or externally over the scalp (Nicolas-Alonso and Gomez-Gil, 2012).

The EEG recording system consists of electrodes, amplifiers, A/D converter, and a recording device. The electrodes acquire the signal from the scalp, the amplifiers process the analog signal to enlarge the amplitude of the EEG signals so that the A/D converter can digitalize the signal in a more accurate way. Finally, the recording device, which may be a personal computer or similar, stores, and displays the data (Binnie and Prior, 1994).

The EEG signal is measured as the potential difference over time between signal or active electrode and reference electrode. An extra third electrode, known as the ground electrode, is used to measure the differential voltage between the active and the reference points. The minimal configuration for EEG measurement therefore consists of one active, one reference, and one ground electrode. Multi-channel configurations can comprise up to 128 or 256 active electrodes (Teplan, 2002). These electrodes are usually made of silver chloride (AgCl) (Sinclair et al., 2007). Electrode-scalp contact impedance should be between 1 kΩ and 10 kΩ to record an accurate signal (Usakli, 2010). The electrode-tissue interface is not only resistive but also capacitive and it therefore behaves as a low pass filter. The impedance depends on several factors such as the interface layer, electrode surface area, and temperature (Usakli, 2010). EEG gel creates a conductive path between the skin and each electrode that reduces the impedance. In Figure 2.1, a cap with electrodes is being placed over a studied person. The use of the gel is cumbersome, however, as continued maintenance is required to assure a relatively good quality signal. Electrodes that do not need to use of gels, called 'dry' electrodes, have been made with other materials such as titanium and stainless-steel (Fonseca et al., 2007).

The amplitude of electrical bio-signals is in the order of microvolts. A typical example of such signals is depicted in Figure 2.2. Consequently, the signal is very sensitive to electronic noise. External sources such power-lines or halogen lamps may generate background noise and thermal, shot, flicker, and burst noises are generated by internal sources (Leach, 1994). Design considerations should be addressed to reduce the effects of the noise, such as electromagnetic interference shielding or reduction for common mode signal, among others (Usakli, 2010).

EEG comprises a set of signals which may be classified according to their frequency. Well-known frequency ranges have been defined ac-

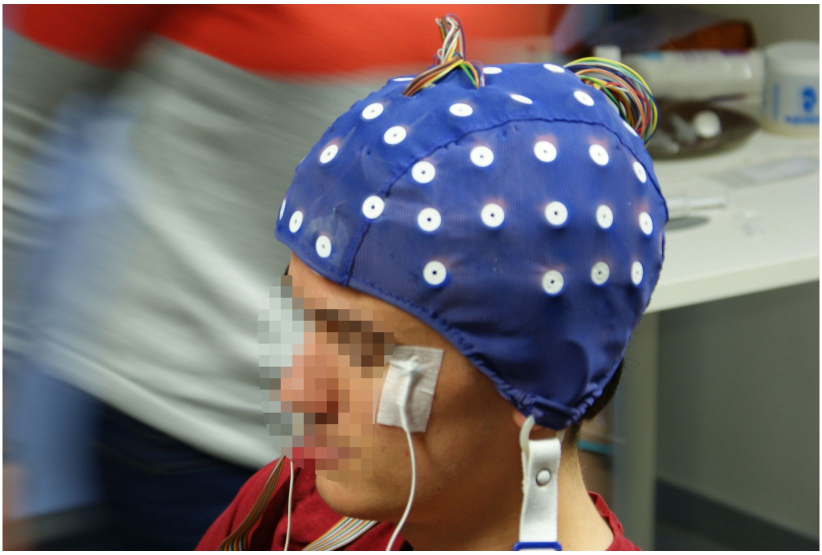


Figure 2.1: EEG cap placement using wet based electrodes.

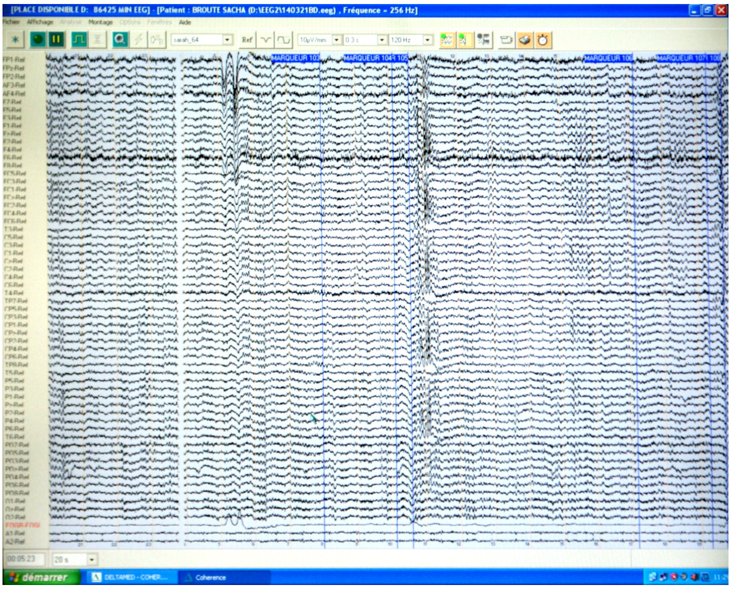


Figure 2.2: EEG digital recordings at the acquisition station.

cording to distribution over the scalp or biological significance. These frequency bands are referred to as delta (δ), theta (θ), alpha (α), mu (μ), beta (β) and gamma (γ) from low to high, respectively. Relevant characteristics of these bands are detailed below.

The delta band lies below 4 Hz, and the amplitude of delta signals detected in babies decreases as they age. Delta rhythms are usually only observed in adults in deep sleep state and are unusual in adults in an awake state. A large amount of delta activity in awake adults is abnormal and is related to neurological diseases (Kübler et al., 2001). Due to low frequency, it is easy to confuse delta waves with artifact signals, which are caused by the large muscles of the neck or jaw.

Theta waves lie within the 4 to 7 Hz range. In a normal awake adult, only a small amount of theta frequencies can be recorded. A larger amount of theta frequencies can be seen in young children, older children, and adults in drowsy, meditative or sleep states (Kübler et al., 2001). Like delta waves, a large amount of theta activity in awake adults is related to neurological disease (Kübler et al., 2001). Theta band has been associated with meditative concentration (Aftanas and Golocheikine, 2001) and a wide range of cognitive processes such as mental calculation (Fernández et al., 1995), maze task demands (Caplan et al., 2001), or conscious awareness (Klimesch et al., 2001).

Alpha rhythms are found over the occipital region in the brain (Pineda, 2005). These waves lie within the 8 to 12 Hz range. Their amplitude increases when the eyes close and the body relaxes and they attenuate when the eyes open and mental effort is made (Black, 1972). Increasing mental effort causes a suppression of alpha activity, particularly from the frontal areas (Venables and Fairclough, 2009). These rhythms primarily reflect visual processing in the occipital brain region and may also be related to the memory brain function (Klimesch, 1997). Mu rhythms may be found in the same range as alpha rhythms, although there are important physiological differences between both. In contrast to alpha rhythms, mu rhythms are strongly connected to motor activities and, in some cases, appear to correlate with beta rhythms (Pineda, 2005; Pfurtscheller et al., 2006).

Beta rhythms, within the 12 to 30 Hz range, are recorded in the frontal and central regions of the brain and are associated with motor activities. Beta rhythms are desynchronized during real movement or motor imagery (Pfurtscheller and Neuper, 2001). Beta waves are characterized by their symmetrical distribution when there is no motor activity. However, in case of active movement, the beta waves attenuate, and their symmetrical distribution changes (Pfurtscheller and Neuper, 2001).

Gamma rhythms belong to the frequency range from 30 to 100 Hz. The presence of gamma waves in the brain activity of a healthy adult is related to certain motor functions or perceptions, among others (Lee et al., 2003). Some experiments have revealed a relationship in normal humans between motor activities and gamma waves during maximal muscle contraction (Brown et al., 1998). This gamma band coherence is replaced by a beta band coherence during weak contractions, suggesting a correlation between gamma or beta cortical oscillatory activity and force Mima et al. (1999). Also, several studies have provided evidence for the role of gamma activity in the perception of both visual and auditory stimuli (Lee et al., 2003; Lutzenberger et al., 1995). Gamma rhythms are less commonly used in EEG-based BCI systems, because artifacts such as electromyography (EMG) or electrooculography (EOG) are likely to affect them (Zhang et al., 2010).

As explained above, EEG is recorded by electrodes. The electrodes placed over the scalp are commonly based on the International 10-20 system (Klem et al., 1958), as seen in Fig. 2.3 which has been standardized by the American Electroencephalographic Society. The 10-20 system uses two reference points in the head to define the electrode location. One of these reference points is the nasion, located at the top of the nose at the same level as the eyes. The other reference point is the inion, which is found in the bony lump at the base of the skull. The transverse and median planes divide the skull from these two points. The electrode locations are determined by marking these planes at intervals of 10% and 20%. The letters in each location corresponds to specific brain regions in such a way that A represents the ear lobe, C the central region, P_g the

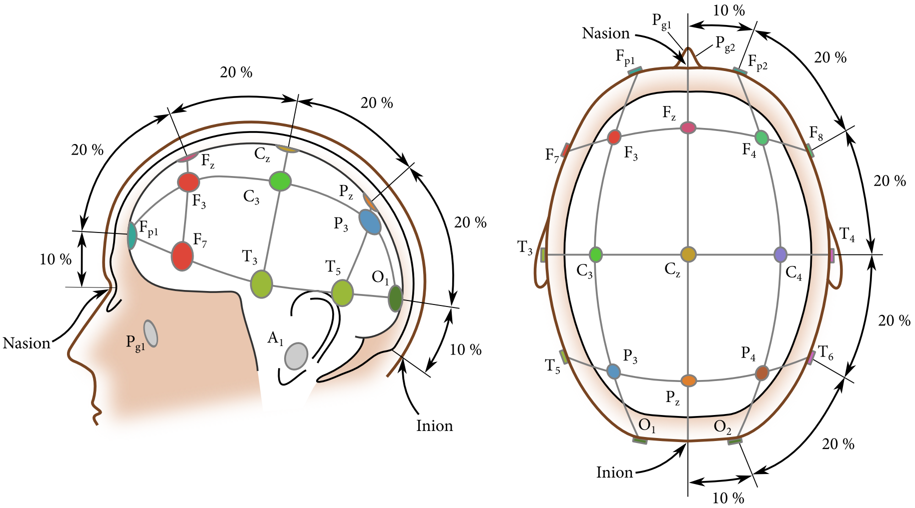


Figure 2.3: Electrodes placement over scalp according with the 10-20 standard.

nasopharyngeal, P the parietal, F the frontal, $\mathbf{F_p}$ the frontal polar, and O the occipital area.

Table 2.1 summarizes the brain acquisition methods with their advantages and disadvantages.

2.3 SIGNAL PRE-PROCESSING

The most common types of pre-processing are artifact detection, spectral filtering and spatial filtering. A summary of these methodologies and their approaches are listed as follows:

- *Artifact detection*. This technique is based on finding confounding signals from sources outside the brain (e.g. eye movement or muscle artifacts), and then removing them from the recording.

- *Spectral filtering*. In this method, noise is removed, such as slow drifts or line noise.

Table 2.1: Characteristics of acquisition methods.

	Invasiveness and medical issues	Spatial resolution	Temporal resolution	Portability	Recorded signal
ECoG	Invasive	~1 mm (High)	~0.003 s (High)	Portable	Electrical
Intracortical	Invasive	~0.5 mm (LFP) ~0.1 mm (MUA) ~0.05 mm (SUA)	~0.003s (High)	Portable	Electrical
fMRI	Non-invasive	~1 mm (High)	~1 s (Low)	Non-portable	Metabolic
MEG	Non-invasive	~5 mm (Medium)	~0.05 s (Medium)	Non-portable	Magnetic
fNIRS	Non-invasive	~5 mm (Medium)	~1 s (Low)	Portable	Metabolic
EEG	Non-invasive	~10 mm (Low)	~0.05 s (Medium)	Portable	Electrical

- *Spatial filtering*. Here, the combination of signals from multiple electrodes for focusing on activities at a particular location of the brain. It is used either to target or reject source-nodes based upon their position.

2.4 FEATURE EXTRACTION

Different activities result in different patterns of brain signals. BCI is seen as a pattern recognition system that classifies each pattern into a class according to its features. BCI extracts some features from brain signals that reflect similarities to a certain class as well as differences from the rest of the classes. The features are measured or derived from the properties of the signals which contain the discriminative information needed to distinguish their different types (Nicolas-Alonso and Gomez-Gil, 2012).

The design of a suitable set of features is a challenging issue. The information of interest in brain signals is hidden in a highly noisy environment, and brain signals comprise a large number of simultaneous sources. A signal that may be of interest could be overlapped in time and space by multiple signals from different brain tasks. For that reason, in many cases, it is not enough to use simple methods such as a band pass filter to extract the desired band power.

Brain signals can be measured through multiples channels. Not all information provided by the measured channels is generally relevant for understanding the underlying phenomena of interest. Dimension reduction techniques such as Principal Component Analysis (PCA) or Independent Component Analysis (ICA) can be applied to reduce the dimension of the original data, removing the irrelevant and redundant information. Computational costs are thereby reduced.

In the first part of this manuscript, we recognize that the most demanding part of a successful BCI design is the feature extraction, hence we particularly focus in this area for the contributions in the consequent chapters.

2.5 CLASSIFICATION

The aim of the classification step in a BCI system is recognition of a user's intentions on the basis of a feature vector that characterizes the brain activity provided by the feature extraction step. Either regression or classification algorithms can be used to achieve this goal, but using classification algorithms is currently the most popular approach (Lotte et al., 2007).

Classification algorithms have traditionally been calibrated by users through supervised learning using a labeled data set. It is assumed that the classifier is able to detect the patterns of the brain signal recorded in online sessions with feedback. However, this assumption results in a reduction in the performance of BCI systems, because the brain signals are inherently non-stationary.

Classifiers also have to face two main problems related to the pattern recognition task: the curse of dimensionality and the bias-variance tradeoff. The curse of dimensionality means that the number of training data needed to offer good results increases exponentially with the dimensionality of the feature vector (Jain et al., 2000). Unfortunately, the available training sets are usually small in BCI research, because training process takes a long time and is a tiring process for users. The bias-variance tradeoff represents the natural trend of the classifiers towards a high bias with low variance and vice versa. Stable classifiers are characterized by high bias with low variance, while unstable classifiers show high variance with low bias. To achieve the lowest classification error, bias and variance should be low simultaneously. A set of stabilization techniques such as the combination of classifiers or regularization can be used to reduce the variance.

2.6 CHAPTER CONCLUSIONS

In this chapter we briefly reviewed some of the techniques employed in state-of-the-art BCI systems. Different neuroimaging approaches have been successfully applied in BCI: (i) EEG, which provides acceptable quality signals with high portability and is by far the most usual modality in BCI; (ii) fMRI and MEG, which are proven and effective methods for localizing active regions inside the brain; (iii) NIRS, which is a very promising neuroimaging method in BCI; and (iv) invasive modalities, which have been presented as valuable methods to provide the high quality signals required in some multidimensional control applications e.g., neuroprostheses control.

A wide variety of signal features and classification algorithms have been tested in the BCI design. Although BCI research is relatively young, many advances have been achieved in a little over two decades, because many of these methods are based on previous signal processing and pattern recognition research. Many studies have demonstrated the valuable accuracy of BCIs and provided acceptable information bit rate, despite the inherent major difficulties in brain signal processing.

In the next chapter we will discuss a particular application of BCI, the recognition of epileptic seizures, a rather common condition. Although this being a well recognized situation from neuroscience and physiology perspectives, is still an open issue in terms of the design of robust automatic recognition systems. Similarly as other BCI applications that use EEG as the brain capture method, the signals also in this domain are complex and their patterns recognition is not trivial by any means. For that, we will introduce our first contribution by employing a novel combination of two different signal processing methods to develop a feature extraction model.

3

CLASSIFICATION OF EPILEPTIC SEIZURES BY MEANS OF SIGNAL PROCESSING

Based on Z-Flores, E., Trujillo, L., Sotelo, A., Legrand, P., and Coria, L. N. (2016). Regularity and Matching Pursuit feature extraction for the detection of epileptic seizures. *Journal of Neuroscience Methods*, 266:107–125

3.1 INTRODUCTION

Epilepsy is a type of neurological disorder that it is characterized by an enduring predisposition to generate unprovoked seizures, each occurring more than 24 hours apart (Fisher et al., 2014). Normal brain activity is considered to be non-synchronous, but during epileptic seizures a group of neurons begins firing in an abnormal, excessive and synchronized manner. This is opposed to what normally happens, when an excitatory neuron fires it becomes resilient to firing again for a short period of time (Else and Hammer, 2013). There are approximately 65 million people worldwide living with epilepsy (Thurman et al., 2011). It varies from region to region, for instance in the United States the annual incidence of epilepsy is 48 per 100,000 inhabitants, whereas the prevalence approximates 710 per 100,000 (Hirtz et al., 2007). Diagnosing epilepsy after a single unprovoked seizure, when there is a high risk for recurrence, may or may not lead to a decision to initiate treatment by the epileptologist. Nonetheless, the diagnosis helps the physician assess the balance between the possible avoidance of a second seizure and the associated risks of actually occurring. According to (Eadie, 2012) about 70% of the cases can be controlled by medication, but even then the patient might suffer negative side effects.

Automated systems can help develop an appropriate therapy plan that could eventually improve the balance of the associated risks and potentially enhance the patient's quality of life (Fisher et al., 2014). Such systems would be able to detect a bodily response that matches the epileptic symptoms, and distinguish them from signals present during a patient's regular activity. There are non-invasive methods that can sense epileptic states before they manifest, as well as to detect the seizure physiologic state, which is also known as the Ictal state. One such method is the Electroencephalogram (EEG), which records electrical brain activity along the scalp. Other methods are also available, such as the Electrocorticogram (ECoG) that is recorded by electrodes that are inserted trough the skull (Sotelo et al., 2013, 2015). ECoG provides more localized readings but can be undesirable for the patient since it is invasive (Ball et al., 2009).

EEGs are very popular because they have several advantages compared to other methods, as stated before in chapter 2 (Tzallas et al., 2009; Übeyli and Güler, 2007; Güler and Ubeyli, 2005; Rajendra Acharya et al., 2012; Acharya et al., 2012a; Guler et al., 2005; Ahammad et al., 2014; Orhan et al., 2011; Guo et al., 2010; Kamath, 2015; Lima et al., 2010), these are: (1) hardware costs are relatively low; (2) electrodes can be positioned based on application needs; (3) it does not expose the patient to high-intensity magnetic fields like magnetoencephalography; and (4) it is non-invasive, relative with other methods. There are some drawbacks as well, like the inherent presence of noise in the signals. Nevertheless, the present work focuses on the use of EEG recordings for automatic epilepsy analysis.

Epileptic seizures can be identified in EEG signals by experienced physicians, but automatic recognition is still not a trivial task. Research for the development of computational systems that perform these tasks usually focuses on three fronts. Either by exploiting the morphology of the signals recorded during the epileptic crisis (Yadav et al., 2012), by applying analytic or numerical methods (Ramgopal et al., 2014), or a combination of the two (Tzallas et al., 2009; Übeyli and Güler, 2007; Güler and Ubeyli, 2005; Rajendra Acharya et al., 2012; Acharya et al., 2012a; Guler et al., 2005; Ahammad et al., 2014; Orhan et al., 2011; Guo et al., 2010; Kamath, 2015; Lima et al., 2010; Murugavel and Ramakrishnan, 2014; Divya, 2015; Kumar et al., 2014).

The design of BCI includes several sections, these being: brain recording method, pre-processing, feature extraction, classification, translation into commands and feedback. As stated earlier in this manuscript, we are going to focus in the first four sections, putting special importance in the feature extraction, which we consider one of the most critical parts for the efficiency of a BCI. Thus, we propose a new methodology for feature extraction, which incorporates two powerful signal analysis tools to construct specialized domain features, namely Hölderian regularity (Mallat and Hwang, 1992) and the Matching Pursuit (MP) algorithm (Mallat, 1993). Each of these tools tackles the posed problem from different perspectives. The former incorporates a local signal regularity measure while the latter employs a time-frequency analysis. These two methods are related, since MP can be used for a global regularity measure. Both of these techniques are considered to be highly nonlinear in their core design, thus the proposed methodology is well suited to analyze a nonlinear process, like the one that produces an epileptic seizure. Moreover, we combine these nonlinear features with much simpler features computed as statistics of the raw signals in time domain, which have been shown to be useful in related tasks (Sotelo et al., 2013).

After the proposed feature extraction process, a classifier is used to solve the automatic detection problem. Experimental results achieve a perfect performance for three of the four tested problem instances. Moreover, on the fourth test case, the most complex, classification accuracy is 97.6%, competitive with the state-of-the-art (Tzallas et al., 2009; Übeyli and Güler, 2007; Güler and Ubeyli, 2005; Rajendra Acharya et al., 2012; Acharya et al., 2012a; Guler et al., 2005; Ahammad et al., 2014; Orhan et al., 2011; Guo et al., 2010; Kamath, 2015; Lima et al., 2010; Tzallas et al., 2007; Guler and Ubeyli, 2007). All of these conclusions are derived from a rigorous experimental validation and comprehensive statistical tests.

In the following subsections a brief description of epileptic states is presented, as well as a short review of previous works related to techniques employed for EEG analysis.

3.1.1 *Epileptic states*

Epiletic seizures are unintentional and disruptive events of mental activity that impair a patient's motor, sensorial and autonomic functions. Seizures develop over several states (Franaszczuk et al., 1998): (1) the Basal state; (2) the Pre-Ictal state; (3) the Ictal state; (4) the Post-Ictal state; and (5) the Inter-Ictal state. These states can be identified by the symptoms exhibited by the patient and the morphology of the EEG signals. The Basal state corresponds to normal brain functions, in this state brain signals are characterized by a low amplitude and a relatively high frequency. The Pre-Ictal state refers to the time period before the seizure symptomatology is evident. Here, the signal amplitude is higher than in the Basal state, with the presence of spikes and transitory activity, also called recruiting rhythms (Kohsaka et al., 2002). The Ictal state is the prominent period where the symptomatology is evident. The EEG signal magnitude is higher than in any other state and displays a dominant low frequency rhythm. The Post-Ictal state refers to the span of time when an altered state of consciousness exists after the active portion of the seizure ended. This period is variable and depends of the seizure duration. The overall amplitude of the signal decreases and the frequency increases. Several symptoms appear during this state, like migraines, depression and a loss of motor functions (Fisher and Schachter, 2000). Finally, the Inter-Ictal state refers to the period of time between seizures.

The problem of automatically detecting the states of epileptic seizures can be posed in different ways (Sotelo et al., 2013, 2015). Here, we recognize that the identification of Ictal states is an important task, as a way to prevent or prepare for a seizure. Moreover, the effects on the patient are most severe in this state. Consequentially, this work focuses on the automatic recognition of Ictal activity among other EEG readings. For this work, we use the Bonn data set (Andrzejak et al., 2001), a public database that contains several types of EEG recordings, to test the proposed approach.

3.1.2 *Previous work*

The automatic classification of epileptic seizures has received much attention over recent years. The problem has been handled from many perspectives, focusing on different aspects of the problem, while using a variety of signal processing and pattern recognition paradigms. Since the number of reported studies is large, here we review the most recent and relevant examples. Moreover, for comparative reasons, we mostly limit to works based on the Bonn data set (Andrzejak et al., 2001). (Acharya et al., 2013) present an extensive survey that summarizes a variety of different methods on how this problem has been addressed. In that survey, the authors also summarize the classification accuracy from each reviewed work, comparing their quality and experimental work. Even though some works achieve strong performance, there is still room for developing new domain-specific patterns recognition methods. Moreover, the insights gathered in epileptic states detection might be extended to other areas of EEG analysis or application domains (Vezard et al., 2014).

Before attempting to solve the classification task, feature extraction must be performed. The simplest features are extracted in the time domain. For instance, (Xie and Krishnan, 2014) proposed an improved Dynamic Principal Component Analysis (DPCA) by means of a non-overlapping moving window. Because they show a highly effective feature extraction, a simple nearest neighbor classifier produces good results. This shows that one of the most difficult tasks is deriving an optimum feature extraction method. Recently, (Kamath, 2015) uses the Hilbert Transform (HT) as a method for a time to time-domain transformation, where the frequency is expressed as a rate of phase change, a dispersion entropy measure, dispersion complexity and forbidden count. These features are then used to distinguish between the Ictal and Inter-Ictal states.

A more common approach is to use time-frequency analysis for epilepsy detection, such as wavelet analysis (Acharya et al., 2012a; Ahammad et al., 2014; Orhan et al., 2011; Tzallas et al., 2009; Guo et al., 2010; Chen et al., 2014; Nunes et al., 2014; Kumar et al., 2014; Gandhi et al., 2012; Zainuddin et al., 2013; Murugavel and Ramakrishnan, 2014; Kumari and Prabin, 2011). A related technique is used by (Kovacs et al., 2014), the Short Time Fourier Transform (STFT), achieving good results and confirming that well established methods are still relevant in this domain. Packet Wavelet Decomposition is also used in this domain, since it is a variant of the same underlying concept (Rajendra Acharya et al., 2012). In (Faust et al., 2015), the authors present a recent comprehensive review of wavelet-based methods used to solve the problem of automatic seizure detection. It is clear that the robust time-frequency analysis provided by the wavelet decomposition is a popular technique among the reviewed literature, because its properties are useful in many areas of signal processing.

Other nonlinear methods not directly related to time-frequency analysis are also used in this domain, (Acharya et al., 2013) suggests that given the nonlinearity of EEG signals these techniques might yield better results, although simpler methods sometimes can outperform them. For example, some works (Bajaj and Pachori, 2012; Alam and Bhuiyan, 2013; Divya, 2015) use the Empirical Mode Decomposition (EMD) to decompose EEG signals into a set of intrinsic mode functions, a method that performs well with highly non-stationary and nonlinear signals, while using a variety of classifiers. (Guler et al., 2005) make use of another nonlinear approach, the Lyapunov exponent approximation as a measure of signal chaosity, with promising results. The use of different type of entropies like Kolmogorov-Sinai Entropy(KSE), Approximate Entropy (ApEn), Sample Entropy (SampEn), Spectral Entropy (SE) among others have been the subject of study in this domain (Acharya et al., 2012a; Nicolaou and Georgiou, 2012; Acharya et al., 2012b; Martis et al., 2013). In (Acharya et al., 2015), the authors survey recent works that apply entropy methods for epilepsy detection. Other related methods include works that use the High Order Spectra (HOS), employing the third order cumulant as a feature (Acharya et al., 2012b), Recurrence Quantification

Analysis (RQA), an analysis based on the topologies derived from recurrence plots in dynamical systems (Niknazar et al., 2013), and the Higuchi Fractal Dimension for measuring signal complexity through fractal concepts (Martis et al., 2013).

The above cited works recognize that a critical step in identifying a brain-related phenomenon using EEG recordings is the feature extraction phase, the main emphasis of the current work. In the following section we present our proposed feature extraction approach and afterwards we evaluate our classification results. Moreover, we will compare our results with those previously reported in the literature, showing that our proposal achieves state-of-the-art performance.

The reminder of this chapter is organized as follows. In Section 3.2 the employed EEG data set is described, the proposed feature extraction methods are presented and the classification task is posed. In Section 3.3 we discuss the experimental details and its results are presented in Section 3.4. Finally, in Section 3.6 we present our chapter conclusions.

3.2 MATERIALS AND METHODS

3.2.1 *Epilepsy EEG Data set*

The data set used in this work was published by the Bonn University (Andrzejak et al., 2001). The data set includes five subsets of signals (denoted as Z, O, N, F and S), each containing 100 single-channel EEG segments with a duration of 23.6s. All segments were recorded using an amplifier system, digitized with a sampling rate of 173.61 Hz and 12-bit A/D resolution, and filtered using a 0.53-40 Hz (12 dB/octave) band pass filter. The normal (Basal) segments (Sets Z and O) were taken from five healthy subjects. The standard surface electrode placement scheme (the international 10-20 system) was used to obtain the EEG from the healthy cases. Volunteers were relaxed in an awake state with eyes opened (Z) and eyes closed (O), respectively. Both states exhibit different characteristics, the EEG readings with the eyes closed show a higher magnitude than when the eyes are opened, as well as the presence of alpha waves

Table 3.1: Summary of the Bonn data set. All classes have 100 epochs per class and 4096 samples per epoch.

Data set label	Class	Description
Z	A	Normal, eyes open
O	B	Normal, eyes closed
N	C	Seizure free, depth electrodes, hippocampal formations
F	D	Seizure free, depth electrodes, epileptogenic formations
S	E	Epiletic activity, depth electrodes, eptic formations

which are common in a relaxation state (Ball et al., 2009). Both the Inter-Ictal and Ictal segments were obtained from five epileptic patients. The Inter-Ictal segments were recorded during seizure-free intervals from the depth electrodes that were implanted into the hippocampal formations (Set N) and from the epileptogenic zone (Set F). The Ictal segments (Set S) were recorded from all sites exhibiting Ictal activity using depth electrodes and also from strip electrodes that were implanted into the lateral and basal regions of the neocortex. For convenience, we refer to subsets Z, O, N, F and S, as class A, B, C, D and E respectively; see Table 3.1.

3.2.2 *Proposed feature extraction*

An appropriate feature extraction is one of the most important tasks in designing classification systems. We propose a feature extraction approach, that to the authors knowledge, has not been used before in the problem domain studied here. In particular, our approach is based on combining the MP algorithm (Mallat, 1993) and signal regularity analysis (Jaffard and Meyer, 1996). The proposed system is depicted in Figure 3.1. Some versions of each method have been used before trying to either analyze or classify EEG recordings, specially using the MP algorithm (Durka and Blinowska, 1995). However, this is the first time that they are used simultaneously to undertake the epilepsy detection problem. The motivation for the simultaneous use of both techniques is that together they can complement the level of detail extracted from the signal, from a broader to a local characterization. This span of detail might produce a robust feature extraction outcome, and its effectiveness is actually validated by experimental results.

Durka has been efficiently using MP decomposition for EEG signal analysis for several years (Durka and Blinowska, 1995; Franaszczuk et al., 1998; Durka et al., 2001, 2005). In particular, he worked on EEG signals derived from subjects exhibiting an epileptic condition with competent results (Durka, 2004). More recently, additional research has been focused on extending Durka's work, like improving the residual inference using source deflation (Wu and Swindlehurst, 2013) or by imposing a restricted dictionary on the MP (Picot et al., 2012), among other proposals (Bénar et al., 2009). It is worth mentioning that the measures derived from the MP decomposition used in our proposal differ from Durka's works, as seen further in Section 3.2.2.2. MP is effective at detecting epileptic spikes due to its ability in finding base functions closely similar to the analyzed signal in time-frequency space (Durka, 2004). In our work, we use a simple approach: a canonical MP (as originally proposed by Stéphane Mallat (Mallat, 1993)) decomposition is performed over an EEG recording (no additional complexity is added) and several statistics are extracted from the obtained decomposition.

Similarly, regularity of EEG signals has also been studied before (Mikaili and Golpayegani, 2002; Popivanov et al., 2006; Natarajan et al., 2004; Mathuvanesan and Jayasankar, 2013), however this topic is broad and there are several ways to measure a signal regularity; including Shannon entropy, spectral entropy, ApEn, Lempel-Ziv complexity and Higuchi fractal, among others. Some works have focused on ApEn concerning EEG analysis (Fan et al., 2011; Abásolo et al., 2005; Chuckravanen, 2014). Closely related to signal regularity, there is also some research for EEG analysis by means of mutifractal theory. For instance, a

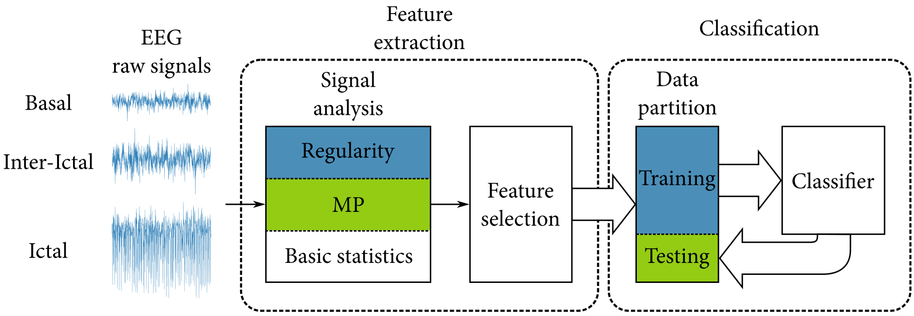


Figure 3.1: Proposed system for automatic epileptic seizures detection.

tool for extracting the regularity spectrum of a time series, the so called Multifractal Detrended Fluctuation Analysis (MFDFA) is employed in different works, by approximating the Hurst exponent (Figliola and Serrano, 2007; Zorick and Mandelkern, 2013; Kantelhardt et al., 2002). Another method for extracting the fractal dimension of a signal, an approach for regularity analysis, is the Wavelet Transform Modulus Maxima (WTMM), used in several works (Ma et al., 2006; Song and Lee, 2005; Dick and Svyatogor, 2012; Popivanov et al., 2006). However, this work represents the first time that the approach by (Jaffard, 2004) is employed to solve the problem of detecting epileptic states.

In summary, the MP algorithm and regularity analysis complement each other by accessing different layers of information embedded in the signals. Hölderian exponent calculation works as a local regularity measure while MP acts as a global regularity measure. In the following subsections we formally describe both feature extraction methods.

3.2.2.1 *Hölder exponent*

In general, the regularity of a signal can be described as the characterization of the singularities it contains. These singularities often carry valuable information, specially for nonlinear and non-stationary signals (Mallat, 2008).

There are several methods to calculate the regularity of a nonstationary time series, either of form local or pointwise (Mallat, 2008).

The precise calculus of them requires of complex numerical methods which are not practically viable and sometimes are not even possible to calculate it. Rather, an approximation is commonly used, where the computational resources are feasible and its characterization is quite accurate. In this work we make use of the regularity measure given by the pointwise Hölder exponent, approximated through a discrete wavelet decomposition. The pointwise Hölder exponent of a function is defined as:

Definition 1 *Let $f : \mathbb{R}^d \to \mathbb{R}, s \in \mathbb{R}^{+*} \backslash \boldsymbol{N}$ and $x_0 \in \mathbb{R}^d$. Then, $f \in C^s(x_0) \Leftrightarrow \exists \eta \in \mathbb{R}^{+*}$, a polynomial P of degree $< s$ and a constant c such that*

$$\forall x \in B(x_0, \eta), |f(x) - P(x - x_0)| \leq c|x - x_0|^s \tag{3.1}$$

The pointwise Hölder exponent of f at x_0 is $\alpha_p = sup_s\{f \in C^s(x_0)\}$.

In some cases, it is necessary to have information of the regularity of a signal, not in a point, but in a neighborhood of that point, namely a local Hölder exponent. Its definition is:

Definition 2 *Let f be a function on the neighborhood of x_0. Let $\{I_n\}_{n \in \boldsymbol{N}}$ be a decreasing sequence of open intervals converging toward x_0. The local Hölder exponent of the function f at x_0 is*

$$\alpha_l(x_0) = \sup_{n \in \boldsymbol{N}}[\alpha_f(I_n)] = \lim_{n \to +\infty} \alpha_f(I_n) \tag{3.2}$$

where

$$\alpha_f(\Omega) = sup\{s : f \in C^s(\Omega)\} \tag{3.3}$$

being $f : \Omega \to \mathbb{R}$ a function, with Ω a open subset of $\mathbb{R}$.

There are, as well, different techniques to estimate the pointwise regularity by considering Hölder spaces. One of them is by using the oscillation method; a mechanism that closely follows the Hölder exponent definition (Jaffard and Meyer, 1996). Another method is the use of discrete wavelet decomposition (DWT). Wavelet analysis can be used to compute an approximation of the Hölder exponent by estimating the decay rate of its wavelet coefficients versus the scales, corresponding to

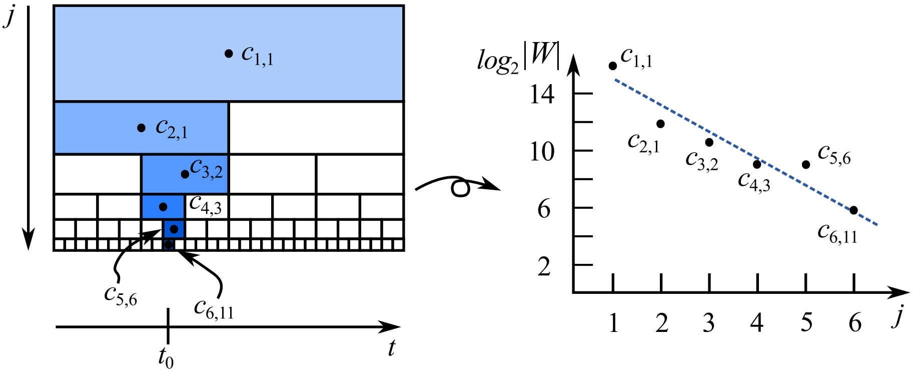


Figure 3.2: Regression calculated over a point of the signal. Left image shows a dyadic wavelet decomposition, and the right image display the actual regression calculated over the point t_0, where each dot corresponds to each log_2 of the wavelet coefficient magnitude located approximately above t_0.

the wavelets localized near the considered point (Jaffard, 2004). Here, we choose the latter method as the preferred way to achieve the regularity estimation of our signals, and specifically using a dyadic decomposition methodology.

Being Ψ a mother wavelet of the classical form $\{\Psi_{j,k}\}_{j,k}$ that makes an orthonormal base of $\mathbf{L}^2$, wavelet coefficients are denoted as $c_{j,k}$ of f, with the latter being a uniformly Hölderian function, where j corresponds to the scales and k corresponds to the temporal location.

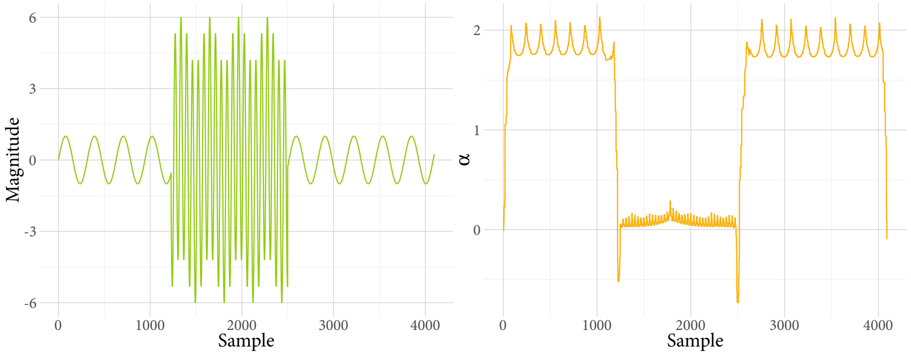
Theorem 1 *Let α be the pointwise Hölder exponent of f at point t_0, then exists a constant $c > 0$ such that the wavelet coefficient satisfy*

$$|c_{j,k}| \leq c2^{-j(\alpha+\frac{1}{2})}(1+|2^j t_0 - k|)^{\alpha} \forall j,k \in \mathbb{Z}^2 \; ; \tag{3.4}$$

reciprocally,

$$\text{if } \forall j,k \in \mathbb{Z}^2 \text{ we have } |c_{j,k}| \leq c2^{-j(\alpha+\frac{1}{2})}(1+|2^j t_0 - k|)^{\alpha'} \,, \tag{3.5}$$

for an $\alpha' < \alpha$ the Hölder exponent of f at t_0 is α. A function f is uniformly Hölderian is there is $\varepsilon > 0$ such that $f \in C^{\varepsilon}(\mathbb{R})$.



(a) Raw signal, superposition of two waveforms with different frequencies.

(b) Regularity measure.

Figure 3.3: (b) Hölderian regularity calculated over a sample signal (a), where α is the estimated Hölder exponent.

Equation 3.4 states that the wavelet coefficients decrease in absolute value by an amount that depends on the Hölder exponent. From this theorem, if we make the hypothesis that the global and local exponents are the same then we are only interested in indices j, k such that $|k - 2^j t_0| < c$ for the point t_0. Then, Equation 3.5 implies the existence of coefficients of the order of $2^{-j(\alpha+\frac{1}{2})}$. This simplifying assumption is satisfied if and only if the local exponent is equal to the pointwise exponent at t_0 (Véhel and Legrand, 2004).

Under this assumption, an estimation of the Hölder exponent is obtained by means of the slope p of the $log_2|c_{j,k}|$ regression with j : $\alpha(n, t_0) = -p - \frac{1}{2}$, $n = \lfloor log_2(N) \rfloor$ being the number of decomposition levels and N the length of the signal.

Definition 3 *In each point t_0 of the signal, decomposed in n scales, the regularity estimation is given by*

$$\alpha(n, t_0) = -\frac{1}{2} - K_n \sum_{j=1}^{n} s_j log_2|c_{j,k}| \,, \tag{3.6}$$

with $K_n = \frac{12}{n(n-1)(n+1)}$ and $s_j = j - \frac{n+1}{2}$. $c_{j,k}$ are the wavelet coefficients located over t_0. The value of k is given by $\lfloor \frac{t_0+1}{2^{n-j+1}} \rfloor$.

Figure 3.2 illustrates this method (Legrand, 2004).

Figure 3.3 presents an example of regularity computation on a time series. The input signal is composed by the superposition of two waveforms with different amplitudes, frequencies and starting times. The regularity measure characterizes the signal singularities. The amplitude given by α corresponds to the regularity of the signal around a given point. Clearly, the low frequency waveform is more regular than the high frequency one. The hard transition between both waveforms is also captured by the presence of highly irregular spikes.

3.2.2.2 *Matching Pursuit*

Matching pursuit refers to a family of greedy algorithms that compute the best nonlinear approximation of a signal (Hussain and Shawe-taylor, 2009). One of these algorithms was introduced by (Mallat, 1993), with the goal of decomposing a signal into a linear expansion of waveforms that are selected from a redundant dictionary of functions.

An unknown signal can be expanded in terms of functions $g_{\gamma n}$, called time-frequency atoms (Chen et al., 1998), given by

$$f(t) = \sum_{n=-\infty}^{+\infty} a_n g_{\gamma n}(t) \, , \tag{3.7}$$

where a_n is an expansion coefficient. In this way $f(t)$ can be explained using functions $g_{\gamma n}$ from a dictionary $\mathcal{D}$. The family $\mathcal{D}= [(g_\gamma(t))_{\gamma \in \Gamma}]$, where the index γ is an element of the set $\Gamma = \mathbb{R}^+ \times \mathbb{R}^2$, it is highly redundant. In general, a family of time-frequency atoms can be generated by scaling, translating and modulating a single window function

$$g_\gamma(t) = \frac{1}{\sqrt{s}} g\left(\frac{t-u}{s}\right) e^{i\xi t} \, , \tag{3.8}$$

where s, u, ξ are the scale, translation and frequency modulation respectively, for the proposed window. The expansion coefficient a_n in Equation 3.7 provide explicit information on certain types of properties of $f(t)$.

Let $f \in \mathbf{H}$, where $\mathbf{H}$ is a Hilbert space, and in our case $\mathbf{H} = \mathbf{L}^2(\mathbb{R})$ (space of complex valued functions), the MP algorithm computes a linear expansion of f over a set of vectors selected from $\mathcal{D}$, by successive approximations of f with orthogonal projections on elements of $\mathcal{D}$. Let $g_{\gamma 0} \in \mathcal{D}$, the vector f can be decomposed into

$$f = \langle f, g_{\gamma 0} \rangle g_{\gamma 0} + Rf \ , \tag{3.9}$$

where Rf is the residual vector after approximating f in the direction of $g_{\gamma 0}$. The MP algorithm sub-decomposes the residue Rf by projecting it on a vector of $\mathcal{D}$ that best matches Rf, as it was done for f.

In general, after m iterations MP decomposes a signal f into

$$f = \sum_{n=0}^{m-1} \langle R^n f, g_{\gamma n} \rangle g_{\gamma n} + R^m f \tag{3.10}$$

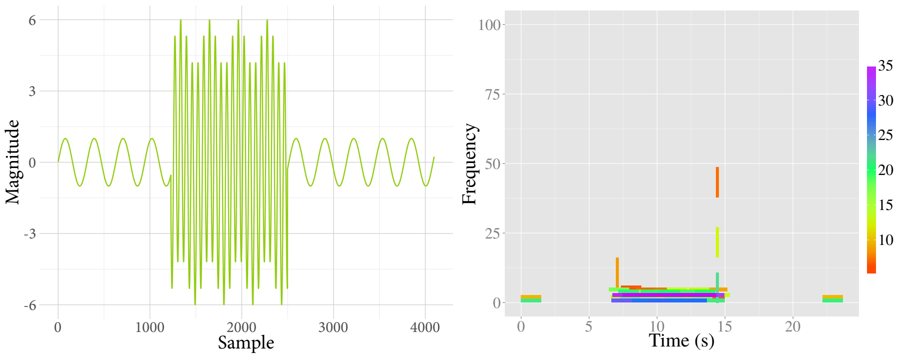
or, in terms of energy,

$$\|f\|^2 = \sum_{n=0}^{m-1} |\langle R^n f, g_{\gamma n} \rangle|^2 + \|R^m f\|^2 \ . \tag{3.11}$$

In Figure 3.4, a visualization of the MP decomposition into time-frequency atoms of one signal is shown, by means of a Heisenberg boxes. Notice that the raw signal is the same used in the example of Figure 3.3. The boxes show specific waveform characteristics: its frequencies, positions, durations and amplitudes, by means of the atoms found during the decomposition. The color scale represents the relative atoms amplitude according with the input signal. The high frequency spikes that exist at the transition of both waveforms are captured by the MP algorithm.

3.2.3 *Proposed feature sets*

The main contribution of this paper is the exploitation of time-frequency content and regularity analysis of EEG signals for the detection of the Ictal state during an epileptic seizure. The pointwise



(a) Raw signal.

(b) MP decomposition.

Figure 3.4: MP Heisenberg boxes over a sample signal. Colors correspond to the relative atom amplitudes.

Hölder exponent and the MP decomposition are not directly dependent on signal amplitude. This is important in real applications, since it is well known that EEG recordings can vary in amplitude as result of changes in the instrumentation, noise induction or skull thickness (Usakli, 2010). Usually, EEG recording systems require a calibration process which can be problematic (Grizou et al., 2014). Therefore, frequency and regularity-based techniques could help alleviate the previously mentioned complications.

Nonetheless, it is evident that the Ictal state presents high amplitude spikes in the time-domain, a characteristic that might help simplify the detection problem. Therefore, we recognize that some useful, and probably necessary, information can only be obtained by explicitly considering the time-domain features.

Thus, we propose two sets of features extracted from each EEG epoch, which are evaluated and compared in our experimental work: (a) Hölder regularity and MP decomposition features (4 features); (b) Hölder, MP and statistical time-domain features (10 features). Additionally, we include an additional set which is automatically chosen by searching for the optimum combination of the proposed 10 feature set, using a meta-heuristic combinatorial optimization algorithm. All of these feature sets are summarized in Table 3.2 and described in the following subsections.

3.2.3.1 *Hölderian regularity and MP decomposition (4 features set)*

These features are presented in the second column of Table 3.2. For the regularity analysis, two statistical measures are used as features: the mean regularity of an epoch $\mu(\mathbf{H})$ and standard deviation calculated by the Median Absolute Deviation (MAD), $\mathrm{MAD}(\mathbf{H})$, being $\mathbf{H}$ the pointwise regularity vector for a given epoch with the same length as number of epoch samples. The MAD is commonly used instead of the standard deviation because it is more resilient to outliers. The MAD can be calculated with

$$\mathrm{MAD}(\mathbf{H}) = (\mu_{1/2})_i(|H_i - (\mu_{1/2})_j(H_j)|)\ , \tag{3.12}$$

where $\mu_{1/2}$ is the median, i and j are indexes of samples in $\mathbf{H}$.

For the MP decomposition, we propose two basic features: the Gabor Atom Density (GAD), first introduced by (Koubeissi et al., 2009), and the frequencies mean $\mu(\mathbf{F})$ of all atoms, where $\mathbf{F}$ is the atom frequencies vector. The frequencies mean is calculated with

$$\mu(\mathbf{F}) = \frac{1}{m}\sum_{n=0}^{m} F_n(g_{\gamma n})\ , \tag{3.13}$$

where $F_n(g_{\gamma n})$ is the frequency of the n atom found during the decomposition.

The GAD is defined as the number of atoms m retrieved during the decomposition, divided by the size of the reconstructed time-frequency space. For each window, the range of time and frequency are, respectively, $R_t = N(1/F_s)$; $R_f = F_s/2$, where N is the number of points in the window and F_s the sampling frequency. The density of atoms over the space is

$$\mathrm{GAD} = \frac{m}{R_t R_f} = \frac{2m}{N}\ . \tag{3.14}$$

3.2.3.2 *Hölder, MP decomposition and time-domain features (10 features set)*

This set is presented in the third column of Table 3.2. These include the regularity and MP features, as well as the mean amplitude of the MP decomposition

$$\mu(\mathbf{A}) = \frac{1}{m}\sum_{n=0}^{m} A_n(g_{\gamma n}), \qquad (3.15)$$

where $A_n(g_{\gamma n})$ is the amplitude of the n atom.

Additionally, five additional features which were successfully used in (Sotelo et al., 2013) for epilepsy state identification are considered. The proposed features are basic statistical measures calculated explicitly in the time domain over the raw epochs **E**, with **E** being the epoch vector in the time domain. They are: the mean $\mu(\mathbf{E})$, median $\mu_{1/2}(\mathbf{E})$, standard deviation $\sigma(\mathbf{E})$, skewness $\gamma_1(\mathbf{E})$ and kurtosis $\gamma_2(\mathbf{E})$.

3.2.3.3 *Automatic feature selection with GA*

An additional feature set is proposed, which is automatically chosen as a subset of the complete 10 feature set. To perform this automatic feature selection procedure we employ a Genetic Algorithm (GA) (Goldberg, 1989), a popular algorithm from the field of Evolutionary Computation (EC) (Eiben and Smith, 2015).

GAs are well established stochastic population-based optimization algorithms, where a large set of candidate solutions compete to survive in each iteration (generation) of the search process. Each solution is ranked by a fitness function that is defined based on the problem domain. For instance, for a classification task fitness can be based on accuracy or total classification error. Each candidate solution (individual) is encoded using a domain specific representation called a genotype, that the search process can manipulate. The genotype is then decoded into a problem domain representation (phenotype), for which fitness can be computed. By means of the search (genetic) operators, solutions evolve by randomly swapping genetic material between pairs of solutions (crossover) or by randomly altering parts of existing solu-

Table 3.2: Proposed feature sets. An additional set is selected automatically, presented in Section 3.3.6

Type	**Number of features**	
	4	**10**
Regularity statistics	MAD($\mathbf{H}$)	MAD($\mathbf{H}$)
	$\mu(\mathbf{H})$	$\mu(\mathbf{H})$
MP statistics	GAD	GAD
	$\mu(\mathbf{F})$	$\mu(\mathbf{F})$
		$\mu(\mathbf{A})$
Raw signal basic statistics		$\mu(\mathbf{E})$
		$\mu_{1/2}(\mathbf{E})$
		$\sigma(\mathbf{E})$
		$\gamma_1(\mathbf{E})$
		$\gamma_2(\mathbf{E})$

tions (mutation). The fitness function is the main determining factor for choosing which solutions will be subjected to the search operators, and to determine which solutions will be kept in the population or will be discarded before the next iteration. This produces a selective pressure towards high-performing solutions. Normally, the search is stopped when a certain number of iterations is reached, and the best solution found at that point is returned.

GAs have been widely used for automatic feature selection (Sun et al., 2004; Lin et al., 2014), in particular given that the canonical binary representation is well suited to represent solutions in this general task. Section 3.3.6 provides further details of our implementation and the resulting feature set.

3.2.4 *Classification*

Once the feature extraction process has been done and consequentially a set of features has been extracted, the classification task must be performed, depicted in the rightmost block of Figure 3.1. This task solves the problem of assigning a class label to unknown data, based on the set of known features and corresponding class labels, basically a supervised learning problem (Duda et al., 2000). Machine learning literature includes a wide variety of supervised classification techniques, from simple Bayes classifiers, to more complex methods like hidden Markov models (Alpaydin, 2010) or genetic programming (Sotelo et al., 2013). In this work, we use the Random Forests™(RF) classifier by (Breiman, 2001), part of the family of decision trees classifiers (Duda et al., 2000). Decision trees creates a model that predicts the class label of an unknown feature vector based on rules derived from the training set and expressed as a tree where each internal node performs a decision based on a particular input feature, and each leaf is labeled with a class or a probability distribution over the classes (Rokach and Maimon, 2008). RF creates a set of decision trees in an ensemble scheme, where a set of weak models work together to build a stronger classifier. The main algorithm parameter is the number of trees used.

This method exhibits several advantages that have made it popular in many domains (Sculley, 2011), including EEG analysis (Chen et al., 2014). It is noteworthy to mention that to the author's knowledge RF has not been applied to the Bonn data set before (Acharya et al., 2013). The advantages of RF are (Breiman, 2001): (1) accuracy on test data is superior to many classifiers when feature selection is optimal; (2) it performs implicit feature selection; (3) it can effectively estimate missing data while maintaining high accuracy; and (4) it has shown resiliency to over-fitting and class imbalance.

3.3 EXPERIMENTAL WORK

This section presents our experimental validation of the proposed feature extraction methods and classification scheme, evaluating each of the proposed feature sets, the effect of signal normalization and considering several problem formulations, using standard performance measures and statistical tests. Moreover, our results are compared with state-

Table 3.3: Classification problems derived from the Bonn data set.

Name	Type	Classes
Problem 1	Binary	A - E
Problem 2	Binary	A,B,C,D - E
Problem 3	Multi-class	A - D - E
Problem 4	Multi-class	A - B - C - D - E

of-the-art works from recent literature. In all of the following work, the reported algorithms and tests were implemented in (MATLAB, 2014).

3.3.1 *Classification problems*

Following the suggestions in (Acharya et al., 2013), the Bonn data set can be used to formulate four distinct problems of varying degrees of difficulty, these are summarized in Table 5.1. Problem 1 and Problem 2 are binary classification problems, the former between class A and E (see Table 3.1) and the latter between all Basal readings (classes A, B, C and D) and Ictal state readings (class E). On the other hand, Problems 3 and 4 are multi-class problems, the former considering classes A, D and E, while Problem 4 considers all five classes, the most difficult case. Note that each group (A-E) contains 100 epochs with 4097 samples each. Therefore, Problem 2 in particular presents an unbalanced classification task.

3.3.2 *Epoch segmentation*

Each epoch in the Bonn database is composed by 4096 samples or measurements, given the duration of each recording and the sampling frequency. However, we pose two different classification problems by treating the data differently. First, we consider each epoch as a single data vector, leaving 100 epochs per class. In the second approach, we split each epoch into 4 non-overlapping segments of roughly 5.9s each with 1024 samples, following (Übeyli and Güler, 2007; Güler and Ubeyli,

Table 3.4: Configuration for MP algorithm.

Description	Value
Number of atoms per dictionary	5000
Sampling rate	173.61 Hz
Stopping criteria	25 dB SNR of reconstruction
Atom types in dictionary	Gabor with Gaussian window
	Gabor with Cosine window
	Chirp with Gaussian window
	Gabor with Rectangle window

2005). Therefore, each class contains 400 total epochs (100×4). Notice that this increases the number of examples we have for each class, but each example contains less information, possibly making it more difficult to characterize each epoch. Hereafter, we refer to the second variant as the segmented method.

3.3.3 *Experimental setup*

For the computation of Hölder regularity we use a Daubechies 10 orthogonal wavelet family and least square linear regression for the coefficients slope calculation in the wavelet decomposition. When we segment the epochs, the maximum decomposition level is lowered because the epoch length is smaller by a factor of 4. The MP algorithm is setup with the parameters in Table 3.4. The atom dictionary can capture different EEG patterns, this helps reduce the number of required iterations to obtain a satisfactory reconstruction.

Usually, there are two stopping criteria for the MP algorithm. One is the number of iterations and the other is the energy level of the signal reconstruction. Here the latter was chosen because we prefer to keep an uniform energy level after reconstruction for all epochs, regardless of its class.

Figures 3.5 and 3.6 show the computation of the pointwise Hölder exponent and MP on the first epoch from each class (A-E). The pointwise Hölder exponent α are similar for class A, B, C and D, with small statistical differences. Class E is distinct, its regularity is noticeably different compared with other groups. For MP decomposition, we can see a clearer difference of atom distribution for all groups. Group B captures more content in higher frequencies, opposed to class E which has a more compact atom location toward lower frequencies despite the fact that it contains more atoms. Other groups exhibit an intermediate composition of atom frequencies, but each shows distinct patterns. To calculate the Hölderian regularity, the FracLab toolbox was used [1].

Figure 3.7 depicts pairwise projections of all possible combinations of the proposed features in Table 3.2, with their corresponding class emphasized by a different color. Effectively, the complete feature space can be visualized, depicting the underlying difficulty of the classification task. From this visualization we can see clearly that no single feature pair is enough to achieve strong discriminant effects. A common pattern found is that variance of class E is large and similar for all combinations of features, although its is usually clearly separated from other classes, thus the uniqueness of the signals generated during epileptic seizures.

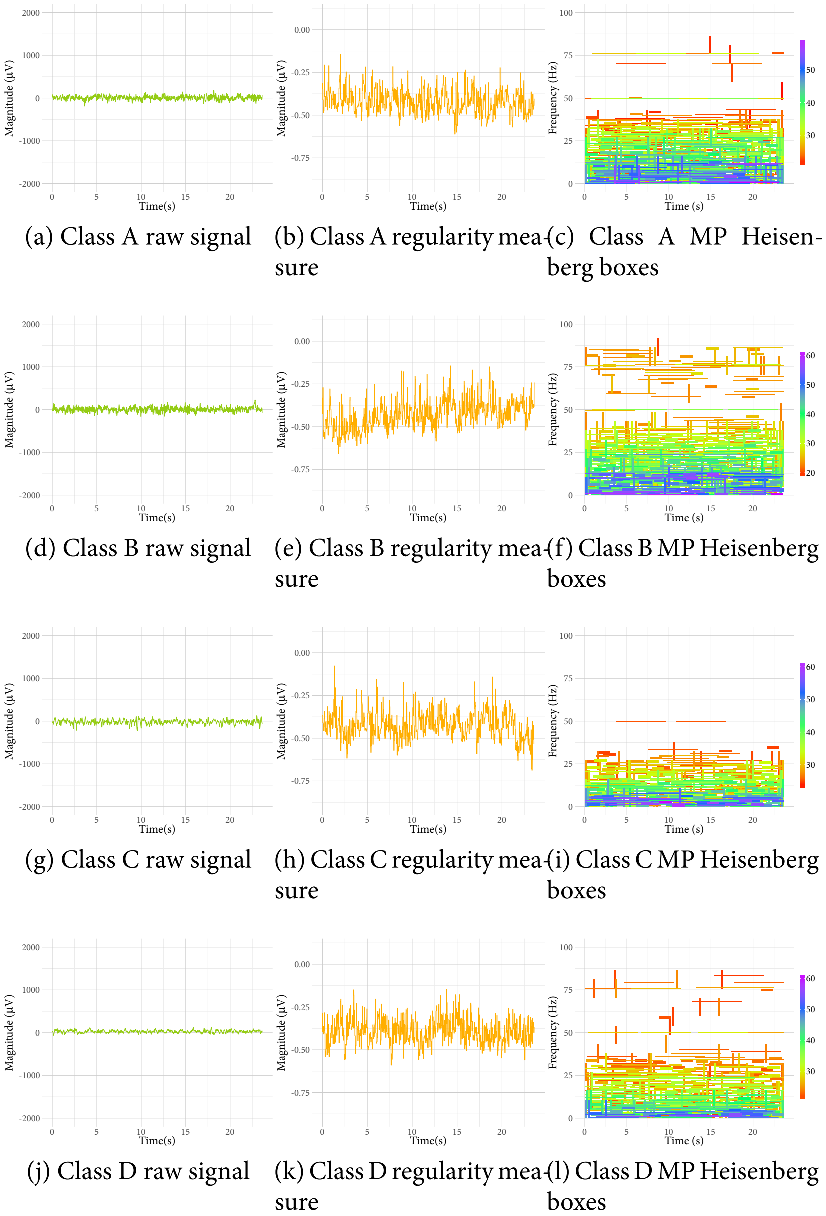
3.3.4 *Pre-processing*

In this work we consider a simple pre-processing stage, where the amplitude is scaled within the range $[0, 1]$ using min-max normalization. To test the importance of this pre-processing step, the proposed features are evaluated both with and without normalization.

3.3.5 *Classifier setup and performance measures*

The experimental work uses the Random Forests™ implementation by (Jaiantilal, 2012) configured to use a maximum of 200 decision trees. The EEG data was partitioned in training and testing sets using a 10-fold

[1] https://project.inria.fr/fraclab/



(a) Class A raw signal

(b) Class A regularity measure

(c) Class A MP Heisenberg boxes

(d) Class B raw signal

(e) Class B regularity measure

(f) Class B MP Heisenberg boxes

(g) Class C raw signal

(h) Class C regularity measure

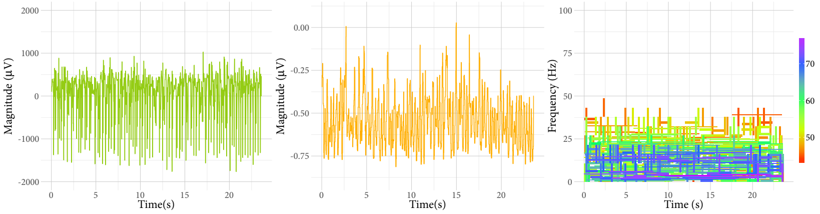
(i) Class C MP Heisenberg boxes

(j) Class D raw signal

(k) Class D regularity measure

(l) Class D MP Heisenberg boxes

Figure 3.5: The pointwise Hölder exponent α (second column) and MP Heisenberg boxes (third column) calculated over the first epoch of each group in the Bonn data set (first column).



(a) Class E raw signal (b) Class E regularity measure (c) Class E MP Heisenberg boxes

Figure 3.6: The pointwise Hölder exponent α (second column) and MP Heisenberg boxes (third column) calculated over the first epoch of each group in the Bonn data set (first column).

cross-validation to obtain a robust estimate of classifier performance. The cross-validation was performed 10 times, this gives us a total of 100 runs per problem instance. The following performance measures are reported based on statistics over all runs.

The main quality measure to determine the classification performance is the accuracy. Given a confusion matrix $A = [A(i,j)]$, where its element $A(i,j)$ is the number of data points whose true class label was i and were classified to class j, the overall accuracy can be calculated with

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{M} A(i,i)\ , \tag{3.16}$$

where N is the number of samples in the data set and $i \in M$ being the i-th label of M classes. Additionally, specificity, F-score and recall were calculated as test classification performance measures. Recall and specificity are measures developed for binary class problems, thus they are calculated for each class, where high values are desired. These are given by

$$\text{Recall}_i = \frac{A(i,i)}{A(i,i)+A(i,j)} \tag{3.17}$$

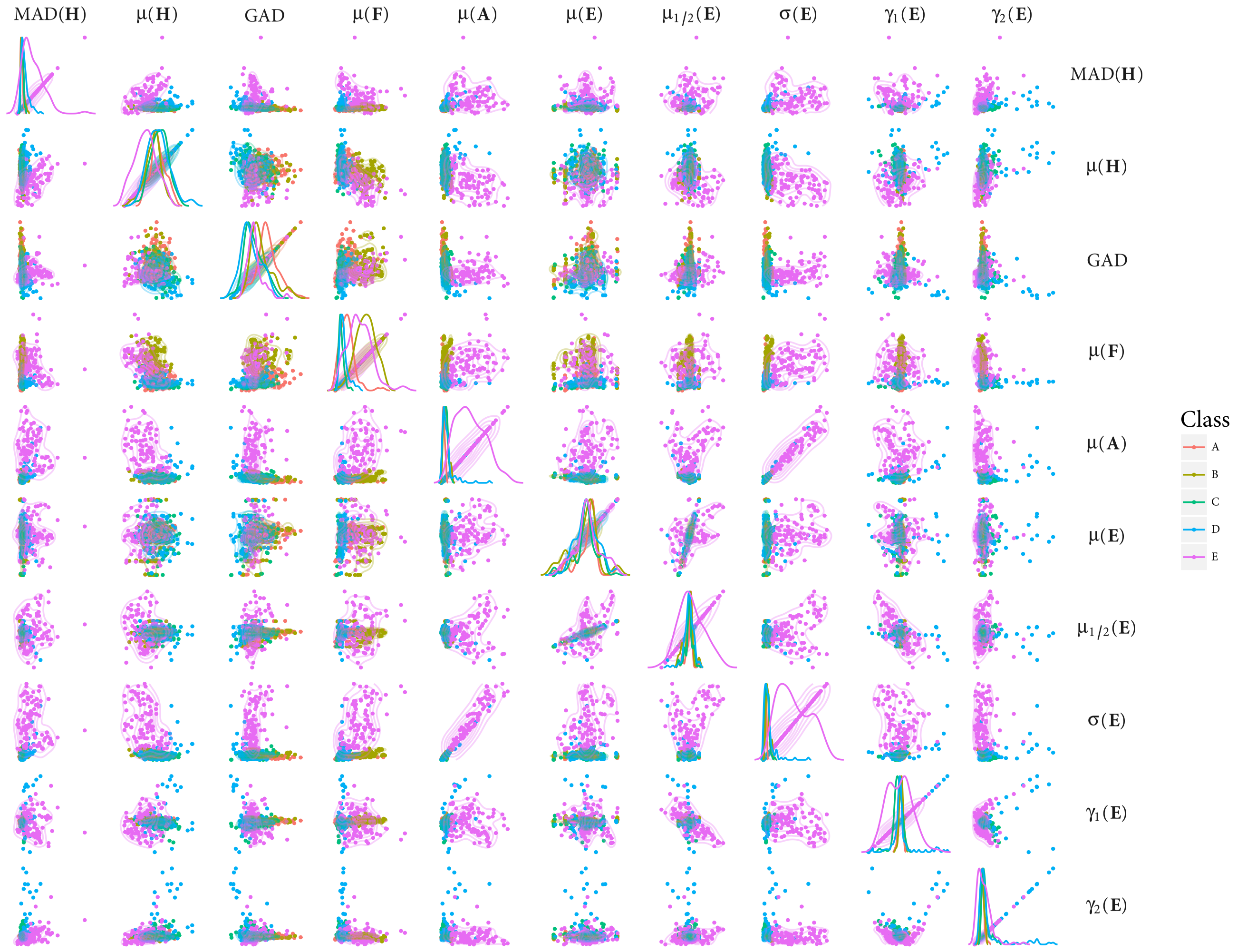


Figure 3.7: Matrix of pairwise projections of all features for all five classes. The diagonal plots correspond to the density distribution per feature.

and

$$\text{Specificity}_i = \frac{A(j,j)}{A(j,j) + A(j,i)} \,. \tag{3.18}$$

The balanced F-score measure can be calculated based on precision and recall by

$$\text{F-score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.19}$$

where precision is given by

$$\text{Precision}_i = \frac{A(i,i)}{A(i,i) + A(j,i)} \,. \tag{3.20}$$

3.3.6 *Automatic feature selection (9 features set)*

To pose the feature selection problem, the genotype of each candidate solution is given by a binary string $\mathbf{b} = [b_1, \ldots, b_{10}]$, where each b_i is associated with one element in the 10 feature set (see the third column of Table 3.2). Each bit in the string determines if the i-th feature is used ($b_i = 1$) or not ($b_i = 0$). The fitness function is based on the classification accuracy (Equation 3.16) achieved on the training set, considering all five classes (Problem 4). Moreover, we apply pre-processing and use the complete epochs. The parameters of the GA are given in Table 3.5. The GA was executed using 10-fold cross-validation, with the goal of detecting which features were chosen with the highest frequency.

Figure 3.8 depicts the frequency of the optimal set of features found by the GA over all folds of the training data. Notice that most features were used at least 80% of the time, with seven of them used with 100% frequency. The only exception was the $\gamma_2(\mathbf{E})$ feature, indicating that it is likely the least useful of all the features. Therefore, in this set we only use the first nine features; hereafter we will refer to this set as the 9 feature set or the automatic feature set.

Table 3.5: GA parameters.

Parameter	Value
Population	100
Generations	50
Chromosome	Binary string
Crossover operator	Intermediate, 0.8 prob.
Mutation operator	Adaptive feasible, 0.15 prob.
Elitism	0.05%
Selection method	Tournament
Fitness function	RF classification accuracy

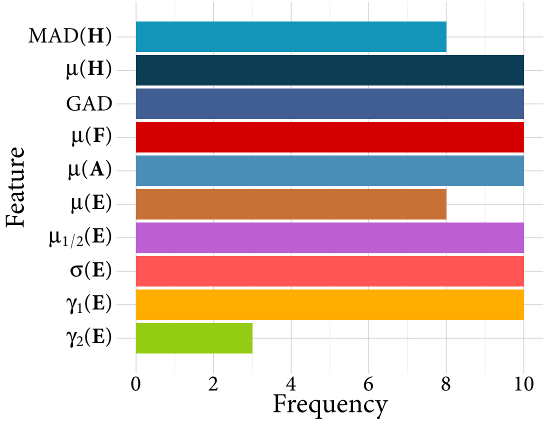


Figure 3.8: Optimal feature frequency for GA algorithm. Each bar value is the accumulative feature appearance after running over all 10 folds.

3.4 RESULTS

The experimental results are organized as follows. Firstly, we perform a statistical analysis on the complete feature set to detect any significant differences among the five classes. For this, the One-way Analysis of Variance (ANOVA) is employed over the feature matrix. These results are presented in Table 3.6.

Secondly, Tables 3.7 and 3.8 summarize the performance of our classification system for each problem (1-4) using each feature set (4, 9 and 10). This table shows results obtained on the raw and normalized signals. In each case, the tables show the average Specificity, Recall and F-score computed over all runs. Moreover, the median, best, worst and inter-quartile spread is given. For the segmented approach, the same performance analysis is presented in Tables 3.9 and 3.10 as well.

Finally, to validate our results, non-parametric statistical tests are used to compare the performance behavior of each system configuration based on the accuracy results. For every problem, the proposed approach was tested considering each feature set (3 variants), whether or not pre-processing is applied (2 variants) and if it is using the full epochs or the segmented approach (2 variants); a total of twelve different groups ($3 \times 2 \times 2$). Afterward, a Friedman test is used to calculate pairwise statistical differences. The p-values for each pairwise comparison are given in Table 3.11 and Table 3.12, applying the Benjamini-Hochberg correction. For these tests, we will reject the null hypothesis that two groups share the same median value with p-values below an $\alpha = 0.05$.

3.5 DISCUSSION

The main contribution of the proposed methodology is the proposed feature set, with Table 3.6 providing some interesting insights. Among all features only the mean of the epoch $\mu(\mathbf{E})$ suffers from poor discrimination properties. Indeed, the variance of the average amplitude of the raw signals is quite similar between epochs, thus it is hard to determine a statistically significant difference between the classes. Although the me-

Table 3.6: One-way ANOVA test for complete feature set. Mean and standard deviation per class are shown in columns 2 through 6. p-values for each feature are shown in last column.

Feature			Class			p-value
	A	B	C	D	E	
MAD(**H**)	0.07±0.01	0.07±0.01	0.07±0.01	0.08±0.02	0.12±0.05	0.0000
$\mu(\mathbf{H})$	-0.46±0.11	-0.53±0.11	-0.44±0.12	-0.45±0.15	-0.63±0.15	0.0000
GAD	0.10±0.02	0.09±0.02	0.06±0.03	0.03±0.01	0.09±0.05	0.0000
$\mu(\mathbf{F})$	0.00±0.00	0.01±0.01	0.00±0.00	0.00±0.00	0.00±0.00	0.0000
$\mu(\mathbf{A})$	6.46±0.74	7.12±0.69	5.30±0.44	4.90±0.24	6.50±0.49	0.0000
$\mu(\mathbf{E})$	-6.26±24.69	-12.51±30.57	-8.88±24.00	-6.20±23.40	-4.75±27.02	0.2437
$\mu_{1/2}(\mathbf{E})$	-6.23±24.63	-13.08±30.63	-8.42±24.04	-8.72±30.09	6.67±81.36	0.0226
$\sigma(\mathbf{E})$	40.73±8.26	61.11±18.12	50.83±19.17	65.62±57.53	306.61±147.23	0.0000
$\gamma_1(\mathbf{E})$	-0.02±0.11	0.06±0.13	-0.15±0.27	0.08±0.76	-0.06±0.76	0.0092
$\gamma_2(\mathbf{E})$	3.23±0.27	3.22±0.27	3.65±0.63	4.33±2.59	3.43±1.19	0.0000

Table 3.7: Summary of classification performance processed over full length epochs, employing a 10-fold cross validation and 10 independent runs. Includes raw and normalized signals. Columns show the average Specificity, Recall, F-score and rank statistics of the accuracy, including median, best, worst and Interquartile Range (IQR).

(a) 4 features

		Raw							Normalized						
		Specificity	Recall	F-score	Accuracy				Specificity	Recall	F-score	Accuracy			
					media	best	worst	IQR				media	best	worst	IQR
Problem 1	Class A	0.9710	0.9700	0.9710	100	100	100	0.00	0.9750	0.9600	0.9668	96.7	100	95	3.80
	Class E	0.9700	0.9710	0.9692					0.9600	0.9750	0.9680				
	Average	0.9705	0.9705	0.9701					0.9675	0.9675	0.9674				
Problem 2	Class A,B,C,D	0.8200	0.9613	0.9582	92.8	92.8	92.8	0.00	0.8000	0.9545	0.9523	92.6	94	91	2.00
	Class E	0.9613	0.8200	0.8305					0.9545	0.8000	0.8040				
	Average	0.8906	0.8906	0.8943					0.8773	0.8773	0.8782				
Problem 3	Class A	0.9287	0.8430	0.8494	87	88.3	86.6	1.00	0.9897	0.9550	0.9665	95.8	96.7	95	0.98
	Class D	0.9203	0.8350	0.8371					0.9650	0.9390	0.9343				
	Class E	0.9655	0.9650	0.9531					0.9610	0.9390	0.9330				
	Average	0.9382	0.8810	0.8798					0.9719	0.9443	0.9446				
Problem 4	Class A	0.8941	0.7160	0.6960	69.6	71	68	2.00	0.8714	0.6420	0.6207	70.4	72	68	2.70
	Class B	0.9291	0.7290	0.7462					0.8593	0.4870	0.4758				
	Class C	0.8652	0.7380	0.6758					0.9171	0.6510	0.6783				
	Class D	0.9222	0.4930	0.5603					0.9370	0.9350	0.8892				
	Class E	0.9413	0.8970	0.8716					0.9480	0.7800	0.8045				
	Average	0.9104	0.7146	0.7100					0.9066	0.6990	0.6937				

(b) 9 features (Automatic)

Problem 1	Class A	1.0000	1.0000	1.0000	100	100	100	0.00	1.0000	1.0000	1.0000	100	100	100	0.00
	Class E	1.0000	1.0000	1.0000					1.0000	1.0000	1.0000				
	Average	1.0000	1.0000	1.0000					1.0000	1.0000	1.0000				
Problem 2	Class A,B,C,D	0.9680	0.9925	0.9923	98	98	98	0.00	0.9690	0.9950	0.9937	100	100	100	0.00
	Class E	0.9925	0.9680	0.9688					0.9950	0.9690	0.9735				
	Average	0.9803	0.9803	0.9805					0.9820	0.9820	0.9836				
Problem 3	Class A	0.9351	0.9410	0.9105	93	93.3	91.7	0.70	1.0000	1.0000	1.0000	100	100	100	0.00
	Class D	0.9625	0.8600	0.8863					0.9930	1.0000	0.9933				
	Class E	0.9926	0.9840	0.9854					1.0000	0.9860	0.9926				
	Average	0.9634	0.9283	0.9274					0.9977	0.9953	0.9953				
Problem 4	Class A	0.9291	0.8590	0.8179	80.7	82	79	1.06	0.9726	0.8970	0.8965	92.9	94	92	0.73
	Class B	0.9626	0.8650	0.8677					0.9807	0.8630	0.8886				
	Class C	0.9119	0.7970	0.7567					0.9729	0.9350	0.9183				
	Class D	0.9408	0.5770	0.6373					0.9819	0.9650	0.9502				
	Class E	0.9866	0.9520	0.9536					0.9946	0.9700	0.9741				
	Average	0.9462	0.8100	0.8067					0.9805	0.9260	0.9255				

Table 3.8: Summary of classification performance processed over full length epochs, employing a 10-fold cross validation and 10 independent runs. Includes raw and normalized signals. Columns show the average Specificity, Recall, F-score and rank statistics of the accuracy, including median, best, worst and Interquartile Range (IQR).

(a) 10 features

		Raw							Normalized						
		Specificity	Recall	F-score	Accuracy				Specificity	Recall	F-score	Accuracy			
Problem 1	Class A	1.0000	1.0000	1.0000	100	100	100	0.00	1.0000	1.0000	1.0000	100	100	100	0.00
	Class E	1.0000	1.0000	1.0000					1.0000	1.0000	1.0000				
	Average	1.0000	1.0000	1.0000					1.0000	1.0000	1.0000				
Problem 2	Class A,B,C,D	0.9680	0.9905	0.9912	98.6	99	98	0.70	0.9700	0.9950	0.9938	100	100	100	0
	Class E	0.9905	0.9680	0.9654					0.9950	0.9700	0.9747				
	Average	0.9793	0.9793	0.9783					0.9825	0.9825	0.9842				
Problem 3	Class A	0.9553	0.9260	0.9200	93.7	95	93.3	0.90	1.0000	1.0000	1.0000	100	100	100	0
	Class D	0.9530	0.8970	0.9006					0.9930	1.0000	0.9934				
	Class E	0.9927	0.9800	0.9828					1.0000	0.9860	0.9926				
	Average	0.9670	0.9343	0.9345					0.9977	0.9953	0.9953				
Problem 4	Class A	0.9504	0.8420	0.8326	83.4	85	82	1.90	0.9729	0.8830	0.8897	92.6	94	92	1.10
	Class B	0.9586	0.8770	0.8719					0.9752	0.8540	0.8702				
	Class C	0.9165	0.8250	0.7743					0.9740	0.9250	0.9143				
	Class D	0.9441	0.6300	0.6788					0.9809	0.9710	0.9522				
	Class E	0.9884	0.9650	0.9634					0.9946	0.9800	0.9800				
	Average	0.9516	0.8278	0.8242					0.9795	0.9226	0.9213				

Table 3.9: Summary of classification performance processed over segmented epochs, employing a 10-fold cross validation and 10 independent runs. Includes raw and normalized signals. Columns show the average Specificity, Recall, F-score and rank statistics of the accuracy, including median, best, worst and Interquartile Range (IQR).

(a) 4 features

		Raw							Normalized						
		Specificity	Recall	F-score	Accuracy				Specificity	Recall	F-score	Accuracy			
					media	best	worst	IQR				media	best	worst	IQR
Problem 1	Class A	0.9625	0.9648	0.9634	97.3	97.5	96.9	0.62	0.9260	0.9263	0.9259	93.5	93.7	93.1	0.60
	Class E	0.9648	0.9625	0.9638					0.9263	0.9260	0.9260				
	Average	0.9636	0.9636	0.9636					0.9261	0.9261	0.9260				
Problem 2	Class A,B,C,D	0.8658	0.9781	0.9724	95.5	95.8	95.2	0.00	0.7580	0.9549	0.9476	91.6	92	91	0.50
	Class E	0.9781	0.8658	0.8863					0.9549	0.7580	0.7817				
	Average	0.9219	0.9219	0.9294					0.8564	0.8564	0.8647				
Problem 3	Class A	0.8866	0.8095	0.7977	83.8	84.6	83.3	0.83	0.9018	0.8320	0.8265	83.7	84.6	82.9	0.83
	Class D	0.9033	0.7663	0.7842					0.9076	0.7915	0.8062				
	Class E	0.9547	0.9420	0.9338					0.9100	0.8630	0.8531				
	Average	0.9149	0.8393	0.8386					0.9064	0.8288	0.8286				
Problem 4	Class A	0.8595	0.6643	0.6312	67.1	67.8	66.5	0.50	0.8070	0.4650	0.4605	56.4	57.2	55.5	0.75
	Class B	0.8933	0.6745	0.6777					0.8025	0.4130	0.4122				
	Class C	0.8743	0.6210	0.6172					0.8383	0.4888	0.5043				
	Class D	0.8765	0.4725	0.5020					0.8460	0.6280	0.6194				
	Class E	0.9522	0.8973	0.8885					0.9001	0.7998	0.7913				
	Average	0.8911	0.6659	0.6633					0.8388	0.5589	0.5575				

(b) 9 features (Automatic)

Problem 1	Class A	1.0000	1.0000	1.0000	100	100	100	0.00	1.0000	1.0000	1.0000	100	100	100	0.00
	Class E	1.0000	1.0000	1.0000					1.0000	1.0000	1.0000				
	Average	1.0000	1.0000	1.0000					1.0000	1.0000	1.0000				
Problem 2	Class A,B,C,D	0.9685	0.9956	0.9939	99	99.2	99	0.00	0.9780	0.9979	0.9962	99.5	99.5	99.2	0.00
	Class E	0.9956	0.9685	0.9752					0.9979	0.9780	0.9845				
	Average	0.9821	0.9821	0.9845					0.9879	0.9879	0.9904				
Problem 3	Class A	0.9476	0.9600	0.9307	94.3	95	94.2	0.42	1.0000	0.9998	0.9999	100	100	100	0.00
	Class D	0.9751	0.8815	0.9123					0.9999	1.0000	0.9999				
	Class E	0.9918	0.9903	0.9876					1.0000	1.0000	1.0000				
	Average	0.9715	0.9439	0.9435					1.0000	0.9999	0.9999				
Problem 4	Class A	0.9367	0.8930	0.8447	83.2	83.8	82.8	0.50	0.9926	0.9530	0.9617	97.6	97.8	97.5	0.25
	Class B	0.9676	0.8593	0.8711					0.9869	0.9683	0.9584				
	Class C	0.9303	0.7868	0.7708					0.9953	0.9723	0.9769				
	Class D	0.9400	0.6570	0.6968					0.9942	0.9840	0.9807				
	Class E	0.9945	0.9743	0.9781					0.9980	0.9928	0.9925				
	Average	0.9538	0.8341	0.8323					0.9934	0.9741	0.9740				

Table 3.10: Summary of classification performance processed over segmented epochs, employing a 10-fold cross validation and 10 independent runs. Includes raw and normalized signals. Columns show the average Specificity, Recall, F-score and rank statistics of the accuracy, including median, best, worst and Interquartile Range (IQR).

(a) 10 features

		Raw							Normalized						
		Specificity	Recall	F-score	Accuracy				Specificity	Recall	F-score	Accuracy			
Problem 1	Class A	1.0000	1.0000	1.0000	100	100	100	0.00	1.0000	1.0000	1.0000	100	100	100	0.00
	Class E	1.0000	1.0000	1.0000					1.0000	1.0000	1.0000				
	Average	1.0000	1.0000	1.0000					1.0000	1.0000	1.0000				
Problem 2	Class A,B,C,D	0.9700	0.9930	0.9928	98.8	99	98.5	0.25	0.9783	0.9983	0.9964	99.5	99.5	99.5	0.00
	Class E	0.9930	0.9700	0.9709					0.9983	0.9783	0.9854				
	Average	0.9815	0.9815	0.9818					0.9883	0.9883	0.9909				
Problem 3	Class A	0.9424	0.9655	0.9295	93.8	94.2	93.3	0.83	0.9970	0.9950	0.9945	99.8	100	99.6	0.42
	Class D	0.9743	0.8658	0.9023					0.9975	0.9940	0.9945				
	Class E	0.9884	0.9830	0.9809					1.0000	1.0000	1.0000				
	Average	0.9683	0.9381	0.9375					0.9982	0.9963	0.9963				
Problem 4	Class A	0.9378	0.8728	0.8360	82.8	83.5	82.5	0.50	0.9908	0.9398	0.9511	96.9	97.2	96.5	0.25
	Class B	0.9639	0.8480	0.8589					0.9827	0.9620	0.9478				
	Class C	0.9291	0.7810	0.7657					0.9927	0.9710	0.9711				
	Class D	0.9362	0.6608	0.6961					0.9929	0.9785	0.9756				
	Class E	0.9903	0.9728	0.9714					0.9984	0.9823	0.9880				
	Average	0.9515	0.8271	0.8256					0.9915	0.9667	0.9667				

dian is related to the mean, the latter is quite sensitive to outliers while the former is more robust, thus the difference in their corresponding p-value. There are also similarities in the skewness feature among the classes, although it is not statistically significant. Furthermore, there seems to be strong agreement between these results and the GA-based feature selection described in Section 3.3.6. Both the ANOVA test and the meta-heuristic feature selection confirm that 8 of the 10 features are relevant to the classification task, partially disagreeing on only two cases ($\mu(\mathbf{E})$ and $\gamma_2(\mathbf{E})$).

Turning to the classification results summarized in Tables 3.7, 3.8, 3.9 and 3.10 (with statistical comparisons given in Tables 3.11 and 3.12) it is clear that Problem 1 is the easiest and Problem 4 is the hardest, as expected. Indeed, for Problems 1-3, at least one configuration achieves perfect accuracy on the test set, while the best performance on Problem 4 is 97.6% accurate. Moreover, these results seem to depend on several factors. For instance, in most cases the full set of 10 features is required to achieve the best performance, with the automatic set of 9 features being the next best option. In all cases, except when using the 4 feature set, performing a pre-processing (normalizing) step is also necessary to achieve the best results.

An interesting result was the performance of the segmented approach. If we compare each row of Tables 3.7 and 3.8 with each row of Tables 3.9 and 3.10, particularly focusing on the normalized data, it is evident that the best results are achieved when we use the full epochs. This is consistent with the idea that with larger epochs the feature extraction process becomes more robust, and with a better ability to extract general properties from each class. There is one exception however, for Problem 4 the best performance is achieved using the segmented approach and the 9 feature set. It seems that the classifier benefits from having a larger amount of examples in this case.

To further validate our work, we take one of the best configurations (9 feature set, full length epochs and pre-processing) and compare it (informally) in Tables 3.13, 3.14, 3.15 and 3.16 with state-of-the-art results reported for the Bonn data set. In the case of Problem 4, we use the segmented variant that produced the best performance. This table presents the authors, the feature extraction approach, whether or not feature post-processing was performed, the classifier used, the experimental training and testing configurations, the number of features employed, and their performance based on average accuracy, recall and specificity. For a fair comparison, only works that explicitly reported all of the above configurations and performance measures are considered.

This comparison is quite favorable for the methodology described in our current work. First, for problems 1, 2 and 3 we achieve perfect performance based on classification accuracy, matching or surpassing all other previous works. Second, the performance on Problem 4 is quite competitive, though slightly outperformed by some works (Übeyli and Güler, 2007; Güler and Ubeyli, 2005; Guler and Ubeyli, 2007). However, as noted in the comparison table, the experimental setup (regarding training and testing partitions) is sometimes not fully given (Übeyli and Güler, 2007; Güler and Ubeyli, 2005; Guler and Ubeyli, 2007; Kamath, 2015; Guo et al., 2010; Lima et al., 2010; Ahammad et al., 2014; Guler et al., 2005), making the relevance of the performance differences less clear in those cases. Furthermore, on Problem 4 our method uses a significantly smaller feature set (9) compared to other works summarized in Table 3.16, none of them use less than 20 features.

3.6 SUMMARY AND CHAPTER CONCLUSIONS

In this chapter we propose a new feature extraction approach for the classification of EEG signals and the detection of epileptic seizures. The detection depends on the ability of the system to recognize seizures among other EEG recordings. The proposed system performs feature extraction using the MP algorithm and Hölderian regularity analysis. This work is the first to combine both methods to effectively extract meaningful traits from epileptic events that are captured by EEG recordings.

Results show that regularity analysis based on the Hölder exponent was able to capture enough information to make a local characterization of the signals and build useful features. One important characteristic of regularity based analysis is its invariance to signal scaling or amplitude

Table 3.11: Friedman pairwise tests, showing adjusted p-values with the Benjamini-Hochberg correction; bold values indicate that the null hypothesis is rejected at the $\alpha = 0.05$ significance level.

(a) Problem 1

			Full						Segmented					
			Raw			Normalized			Raw			Normalized		
		# features	4	9	10	4	9	10	4	9	10	4	9	10
Full	Raw	4	-	**0.0000**	**0.0000**	**0.0072**	**0.0000**	**0.0000**	**0.0000**	**0.0029**	**0.0189**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	1.0000	**0.0000**	1.0000	1.0000	**0.0000**	**0.0189**	**0.0042**	**0.0000**	1.0000	1.0000
		10	-	-	-	**0.0000**	1.0000	1.0000	**0.0000**	**0.0189**	**0.0042**	**0.0000**	1.0000	1.0000
	Norm.	4	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	-	-	-	1.0000	**0.0000**	**0.0189**	**0.0042**	**0.0000**	1.0000	1.0000
		10	-	-	-	-	-	-	**0.0000**	**0.0189**	**0.0042**	**0.0000**	1.0000	1.0000
Segmented	Raw	4	-	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	-	-	-	-	-	-	0.0761	**0.0000**	**0.0189**	**0.0189**
		10	-	-	-	-	-	-	-	-	-	**0.0000**	**0.0042**	**0.0042**
	Norm.	4	-	-	-	-	-	-	-	-	-	-	**0.0000**	**0.0000**
		9	-	-	-	-	-	-	-	-	-	-	-	1.0000
		10	-	-	-	-	-	-	-	-	-	-	-	-

(b) Problem 2

			Full						Segmented					
			Raw			Normalized			Raw			Normalized		
		# features	4	9	10	4	9	10	4	9	10	4	9	10
Full	Raw	4	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.2613	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	0.0577	**0.0000**	0.3139	0.8400	**0.0000**	0.4061	0.3502	**0.0000**	1.0000	0.8400
		10	-	-	-	**0.0000**	**0.0005**	**0.0023**	**0.0000**	0.5234	0.4600	**0.0000**	0.1089	0.0644
	Norm.	4	-	-	-	-	**0.0000**	**0.0000**	**0.0002**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	-	-	-	0.3091	**0.0000**	0.6431	**0.0022**	**0.0000**	0.3595	0.5492
		10	-	-	-	-	-	-	**0.0000**	0.2299	**0.0304**	**0.0000**	0.5340	0.8400
Segmented	Raw	4	-	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	-	-	-	-	-	-	**0.0034**	**0.0000**	**0.0000**	**0.0000**
		10	-	-	-	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**
	Norm.	4	-	-	-	-	-	-	-	-	-	-	**0.0000**	**0.0000**
		9	-	-	-	-	-	-	-	-	-	-	-	0.2132
		10	-	-	-	-	-	-	-	-	-	-	-	-

Table 3.12: Friedman pairwise tests, showing adjusted p-values with the Benjamini-Hochberg correction; bold values indicate that the null hypothesis is rejected at the $\alpha = 0.05$ significance level.

(a) Problem 3

			Full						Segmented					
			Raw			Normalized			Raw			Normalized		
		# features	4	9	10	4	9	10	4	9	10	4	9	10
Full	Raw	4	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.2141	**0.0000**	**0.0000**	**0.0009**	**0.0000**	**0.0000**
		9	-	-	**0.0036**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		10	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
	Norm.	4	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0001**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	-	-	-	0.5316	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0003**	0.0934
		10	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0309**	**0.0001**
Segmented	Raw	4	-	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0005**	**0.0000**	**0.0000**
		9	-	-	-	-	-	-	-	-	**0.0435**	**0.0000**	**0.0000**	**0.0000**
		10	-	-	-	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**
	Norm.	4	-	-	-	-	-	-	-	-	-	-	**0.0000**	**0.0000**
		9	-	-	-	-	-	-	-	-	-	-	-	**0.0000**
		10	-	-	-	-	-	-	-	-	-	-	-	-

(b) Problem 4

			Full						Segmented					
			Raw			Normalized			Raw			Normalized		
		# features	4	9	10	4	9	10	4	9	10	4	9	10
Full	Raw	4	-	**0.0000**	**0.0000**	0.5966	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	**0.0310**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.1111	0.8383	**0.0000**	**0.0000**	**0.0000**
		10	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0068**	**0.0079**	**0.0000**	**0.0000**	**0.0000**
	Norm.	4	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		10	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Segmented	Raw	4	-	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
		9	-	-	-	-	-	-	-	-	0.1735	**0.0000**	**0.0000**	**0.0000**
		10	-	-	-	-	-	-	-	-	-	**0.0000**	**0.0000**	**0.0000**
	Norm.	4	-	-	-	-	-	-	-	-	-	-	**0.0000**	**0.0000**
		9	-	-	-	-	-	-	-	-	-	-	-	**0.0079**
		10	-	-	-	-	-	-	-	-	-	-	-	-

Table 3.13: Comparison of classification performance among several published approaches using the same database, including our work. Accuracy values are shown as percentile. Best value for each problem is shown in bold. Compared works are grouped by problems and sorted by their accuracy.

	Authors	Feature selection	Feature post-processing	Classifier	Training/Testing partition	Number of features	Accura	Recal	Specificity
Problem 1	(Nicolaou and Georgiou, 2012)	Permutation Entropy	No	SVM	60/40 random, 100 times	69-92	93.55	-	-
	(Tzallas et al., 2009)	Time Frequency analysis	PCA	ANN	50/50 random, 10 times holdout	16	94.3	100	100
	(Divya, 2015)	Empirical Mode Decomposition	No	ANN	[a]	15	96.88	93.3	100
	(Zainuddin et al., 2013)	DWT	No	Wavelet Neural Network	10-fold cross validation	20	98.87	94.96	99.43
	(Murugavel and Ramakrishnan, 2014)	Wavelet	Entropy	Optimized Extreme Learning Machine	50/50 cross fold	30	99	93.2	98.9
	(Kovacs et al., 2014)	Discrete Short Time Fourier Transform	No	Bagged Alternating Decision Trees	75/25	483	99.7	-	-
	(Guo et al., 2010)	Multiwavelet	Entropy	MLPNN	50/50 [a]	10	99.8	100	99.2
	(Tzallas et al., 2007)	Time Frequency analysis	PCA	ANN	50/50 random, 10 times holdout	13-66	**100**	100	100
	(Lima et al., 2010)	Wavelet	No	LS-SVM	50/50 [a]	5	**100**	100	100
	(Kumari and Prabin, 2011)	DWT	No	SVM	80/20	15	**100**	-	-
	(Gandhi et al., 2012)	DWT	Discrete Harmony Search	Probabilistic Neural Network	10-fold cross validation	45	**100**	-	-
	(Alam and Bhuiyan, 2013)	Empirical Mode Decomposition	No	ANN	60/35 + 5 of validation partition	48	**100**	100	100
	(Kumar et al., 2014)	DWT	Fuzzy Approximate Entropy	SVM	50/50 random, 10 times hold-out	6	**100**	100	100
	(Chen, 2014)	DTCWT	No	1-NN	50/50	2	**100**	-	-
	(Kamath, 2015)	Hilbert Transform Scatter Plot	No	Statistical analysis	[a]	>1000	**100**	100	100
	This work	Regularity & MP	No	RF	10-fold cross validation	9	**100**	100	100

[a] There is missing information of how the train/test partition was done or other details in the experimental setup.

Table 3.14: Comparison of classification performance among several published approaches using the same database, including our work. Accuracy values are shown as percentile. Best value for each problem is shown in bold. Compared works are grouped by problems and sorted by their accuracy.

	Authors	Feature selection	Feature post-processing	Classifier	Training/Testing partition	Number of features	Accura	Recal	Specificity
Problem 2	(Nicolaou and Georgiou, 2012)	Permutation Entropy	No	SVM	60/40 random, 100 times	69-92	86.1	-	-
	(Kovacs et al., 2014)	Discrete Short Time Fourier Transform	No	Bagged Alternating Decision Trees	75/25	483	96.7	-	-
	(Kumar et al., 2014)	DWT	Fuzzy Approximate Entropy	SVM	50/50 random, 10 times hold-out	6	97.36	98.3	93.5
	(Guo et al., 2010)	Multiwavelet	Entropy	MLPNN	50/50 [a]	10	98.3	99	95.5
	(Murugavel and Ramakrishnan, 2014)	Wavelet	Entropy	Optimized Extreme Learning Machine	50/50 cross fold	30	99	-	-
	(Divya, 2015)	Empirical Mode Decomposition	No	ANN	[a]	15	99	93.3	100
	(Orhan et al., 2011)	DWT	K-means	ANN	cross validation, 5000 times	18-56	99.6	100	98
	(Alam and Bhuiyan, 2013)	Empirical Mode Decomposition	No	ANN	60/35 + 5 of validation partition	48	**100**	100	100
	(Chen, 2014)	DTCWT	No	1-NN	50/50	2	**100**	-	-
	This work	Regularity & MP	No	RF	10-fold cross validation	9	**100**	100	100

[a] There is missing information of how the train/test partition was done or other details in the experimental setup.

Table 3.15: Comparison of classification performance among several published approaches using the same database, including our work. Accuracy values are shown as percentile. Best value for each problem is shown in bold. Compared works are grouped by problems and sorted by their accuracy.

	Authors	Feature selection	Feature post-processing	Classifier	Training/Testing partition	Number of features	Accura	Recal	Specificity
Problem 3	(Ahammad et al., 2014)	Wavelet	No	Linear	62/38 [a]	18	84.2	83.5	85.6
	(Divya, 2015)	Empirical Mode Decomposition	No	ANN	[a]	15	93.9	97.1	89.1
	(Martis et al., 2013)	Higuchi Fractal Dimension, entropy over Intrinsic Time-Scale Decomposition	No	Decision Tree	[a]	6	95.67	99	99.5
	(Murugavel and Ramakrishnan, 2014)	Wavelet	Entropy	Optimized Extreme Learning Machine	50/50 cross fold	30	96	93.5	98.4
	(Orhan et al., 2011)	DWT	K-means	ANN	random subsampling, cross validation, 5000 times & best model	18-56	96.7	93.5	98.3
	(Acharya et al., 2012a)	Entropies	No	Fuzzy	3-fold cross validation	4	98.1	99.4	100
	(Niknazar et al., 2013)	DWT, Recurrence Quantification Analysis	Phase Space Reconstruction	Error-Correction Output Codes	70/30 random, 20 times	30	98.67	98.55	99.33
	(Rajendra Acharya et al., 2012)	Wavelet Packet Decomposition	PCA eigenvalues	Gaussian Mixture Model	10-fold cross validation	9	99	99	99
	(Acharya et al., 2012b)	Entropies, High Order Spectra, Hurst exponent, Higuchi Fractal Dimension	ANOVA filtering	Fuzzy	10-fold cross validation	7	99.7	100	100
	(Alam and Bhuiyan, 2013)	Empirical Mode Decomposition	No	ANN	60/35 + 5 of validation partition	48	**100**	100	100
	This work	Regularity & MP	No	RF	10-fold cross validation	9	**100**	100	100

[a] There is missing information of how the train/test partition was done or other details in the experimental setup.

Table 3.16: Comparison of classification performance among several published approaches using the same database, including our work. Accuracy values are shown as percentile. Best value for each problem is shown in bold. Compared works are grouped by problems and sorted by their accuracy.

	Authors	Feature selection	Feature post-processing	Classifier	Training/Testing partition	Number of features	Accura	Recal	Specificity
Problem 4	(Nunes et al., 2014)	DWT	Relief, InfoGain	Optimum Path Forest	10-fold cross validation	<40	89.2	-	-
	(Murugavel and Ramakrishnan, 2014)	Wavelet	Entropy	Optimized Extreme Learning Machine	50/50 cross fold	30	94	93.6	98.3
	(Übeyli and Güler, 2007)	Eigenvector method	No	Modified of Mixture of expert model	50/50 [a]	64	98.6	98.6	99.6
	(Güler and Ubeyli, 2005)	Wavelet	No	ANFIS	50/50 [a]	20	98.7	98.7	99.7
	(Guler and Ubeyli, 2007)	Wavelet, Lyapunov exponents	No	SVM	50/50 [a]	24	**99.3**	99.3	99.8
	This work	Regularity & MP	No	RF	10-fold cross validation	9	97.6	97.8	97.5

[a] There is missing information of how the train/test partition was done or other details in the experimental setup.

changes, which makes it ideal for EEG processing given the noisy conditions. On the other hand, the MP decomposition provided a more global characterization of the analyzed signals, by building features over the obtained set of waveforms or so called atoms. Certainly, the feature extraction process was shown to be sufficiently informative to be able to simplify the classification problem and achieve state-of-the-art performance.

The experimental work allows us to conclude the following regarding the proposed approach. First, the pre-processing step produces statistically significant performance improvements. Second, using only MP decomposition and Hölderian regularity provided strong performance on all problems, but to achieve optimal accuracy additional information was required. In particular, statistical time-domain features provided sufficient information to solve most problems with state-of-the-art results, that compared favorably with recently published works. Third, the set of proposed features provides the necessary discriminant information to solve the studied problems, with both a statistical analysis and a combinatorial feature selection algorithm suggesting that almost all features are relevant to the epilepsy detection problem in EEG signals. Indeed, on the most difficult classification task (considering five classes), our results achieve close to perfect accuracy (97.6%), while using less than half of the total features used by other approaches that achieve similar performance.

Although the proposed approach was seen as a promising choice among other signal processing/machine learning methods the fact remains that the problem to be solved is complex. In these cases, meta-heuristic methods are viable options compared with classical machine learning methods. Meta-heuristic approaches are highly efficient when the problem exhibits these characteristics: the number of samples and variables in the dataset is high, the search space where the optimal solution exists is massive, an approximation of the optimal solution is sufficient to solve the problem, and generalization is more important than data fitting. Despite that in the literature exists a wide range of meta-heuristic algorithms, GP is one of the most promising options due to one unique asset: the adaptation to a large number of contrasting applications. One of the main GP distinctions compared with other algorithms is the solutions representation; they are built using computing elements that being combined can ultimately execute complex processes according with the problem's requirements. In other words, the same algorithm can be adapted to a learning system solver (regression, classification, etc.), to code synthesis for software engineering, to automatic bug fixer, to game strategies/rules development, to electrical circuits designer, among others.

In Part II of this thesis, GP will be the main topic, as the driver for the following research in this manuscript. Nevertheless, as to be explained later, GP is not free of shortcomings. Its unique traits also carry some associated limitations, at least in the original formulation of the algorithm. The most common representation, tree based, carries a series of drawbacks that, fortunately, have been addressed with promising results. Some of these are further explored in consequent chapters, along unique contributions and experimental evidence. More specifically, the incorporation of LS strategies into GP is analyzed and applied to two different statistical learning problems: regression and classification.

Part II

Genetic Programming

4 REGRESSION

Based on Z-Flores, E., Trujillo, L., Schütze, O., and Legrand, P. (2014). Evaluating the Effects of Local Search in Genetic Programming. In *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, volume 288, pages 213–228

4.1 INTRODUCTION

In the domain of optimization algorithms, it is widely understood that effectively balancing exploration and exploitation is one of the key issues underlying any successful search process. Indeed, a useful taxonomy of search algorithms is based on the idea of differentiating between search methods that rely on exploratory search operators, and search methods that can guarantee (local) optimality by exploiting the local structure of a fitness landscape. Local search (LS) algorithms can guarantee convergence to local optima if their underlying assumptions are satisfied and are computationally efficient. However, the success of a LS strongly depends on the initial point, particularly in highly irregular, multimodal or discontinuous fitness landscapes (Gill et al., 1981). On the other hand, global search algorithms include a variety of deterministic and stochastic strategies. Here, we focus on evolutionary algorithms (EAs), metaheuristic strategies based on an abstract model of Neo-Darwinian evolution (Eiben and Smith, 2015; Luke, 2013). EAs are particularly useful when a good initial solution is difficult to propose, or when LS strategies tend to converge on undesirable local optima. Moreover, EAs are particularly well suited when a single monolithic solution is not sufficient, and instead a set of different solutions is required (Coello et al., 2006; Dunn et al., 2006). In general, EAs cannot guarantee convergence towards local optima in most realistic scenarios. Moreover, they tend to be computationally costly, and in many cases can be highly inefficient. Nevertheless, EAs have produced extremely competitive solutions in difficult domains and problem instances (Koza, 2010).

Considering the strengths and weaknesses of both global and local methods, it is intuitive to conclude that the best strategy should be a hybrid approach, commonly referred to as memetic search. These methods have been extensively studied over recent years, combining EAs with a variety of local searchers (Chen et al., 2011).

While all EAs are based on the same general principles (De Jong, 2006), there is a large variety among current methods, each with their respective strengths and weaknesses. In this chapter, we focuses on an EA paradigm that presents unique properties among other EAs, genetic programming (GP) (Koza, 1992; Poli et al., 2008). The goal of GP is to automatically evolve specialized syntactic expressions that best solve a given tasks, which can be interpreted as mathematical functions or computer programs. In particular, in this chapter we focus on how the standard syntactic search can be improved by including a LS method. While this work is not the first to develop a memetic GP system, it does present the first systematic evaluation of what could be the best strategies to accomplish this, by considering several variants and evaluating them on current benchmark problems for symbolic regression.

The remainder of this chapter proceeds as follows. First, an overview of GP is provided in Section 4.2. Then, Section 4.3 reviews previous work on memetic optimization with EAs. Afterwards, Section 4.4 describes our proposed memetic algorithms to perform real-valued parameter optimization of evolved syntactic expressions, considering several basic variants. The experimental setup is presented in Section 4.5 and results are discussed in Section 4.6. Finally, a summary of the chapter and concluding remarks are outlined in Section 4.7.

4.2 GENETIC PROGRAMMING

GP can be understood as a generalization of the basic genetic algorithm, its main features can be summarized as follows (Poli et al., 2008). First, GP was originally proposed as an EA that evolves simple programs, functions, operators, or in general symbolic expressions that perform some form of computation. GP is basically used to evolve solutions to different types of design problems, with examples as varied as quantum algorithms (Spector, 2006), computer vision operators (Olague and Trujillo, 2011) and satellite antennas (Hornby et al., 2011). Second, solutions are expressed as variable length structures, such as linked lists, parse trees or graphs (Poli et al., 2008). These structures encode the syntax of each individual program. Therefore, in a canonical GP algorithm, search operators, such as crossover and mutation, perform syntactic variations on the evolving population. Third, by considering each individual in a GP run as a program, the evolutionary process is basically attempting to write the best program syntax that solves a given problem. Therefore, a finite set of basic symbols needs to be defined, which is called the primitive set $\mathbb{P}$. Within the primitive set there is a subset of basic operations or functions of different arity, called the function set F, and a subset of input variables, constants or zero arity functions called the terminal set T, such that $\mathbb{P} = F \cup T$.

Given the variety of possible GP configurations and applications, this work focuses on the problem of symbolic regression using a tree representation. In symbolic regression, the goal is to search for the symbolic expression $K^O : \mathbb{R}^p \to \mathbb{R}$ that best fits a particular training set $\mathbb{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ of n input/output pairs with $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. In such problems, for instance, the function set can be defined as $F = \{+, -, /, *\}$ and the terminal set can be composed by each of the features in the input data, such that $T = \{x_i\}$ with $i = 1, \ldots, p$, but other terminal elements can be included such as integer or real-valued constants.

The general symbolic regression problem can then be defined as

$$(K^O, \boldsymbol{\theta}^O) \leftarrow \underset{K \in \mathbb{G}; \boldsymbol{\theta} \in \mathbb{R}^m}{arg\ min}\ f(K(\mathbf{x}_i, \boldsymbol{\theta}), y_i)\ with\ i = 1, \ldots, p\ , \tag{4.1}$$

where $\mathbb{G}$ is the solution or syntactic space defined by $\mathbb{P}$, f is the fitness function based on the difference between a program's output $K(\mathbf{x}_i,\boldsymbol{\theta})$ and the expected output y_i, such as the mean square error, and $\boldsymbol{\theta}$ is a particular parametrization of the symbolic expression K, assuming m real-valued parameters. This dual problem, of simultaneously optimizing syntax (structure) as well as its parametrization is also discussed in (Lohmann, 1991; Emmerich et al., 2001). The authors differentiate between two possible approaches towards solving such a task. The first group is *hierarchical structure evolution* (HSE), when $\boldsymbol{\theta}$ has a strong influence on fitness, and thus a local search is required at each generation of the global (syntactic) search, configured as a nested process. The second group is called *simultaneous structure evolution* (SSE), when $\boldsymbol{\theta}$ has a marginal effect on fitness, in such cases a single evolutionary loop could optimize both syntax and parameters simultaneously. However, such categorizations are highly abstract, and a particular implementation could easily be classified into both groups. Nevertheless, it is reasonable to state that the standard GP approach falls in the SSE group.

In standard GP, however, parameter optimization is usually not performed explicitly, since GP search operators only affect syntax. Therefore, the parameters are only implicitly considered. For instance, a GP individual might have the following syntax $K(x) = x + sin(x)$; in this case, we might propose the following parametrization: $\boldsymbol{\theta} = (\alpha_1, \alpha_2, \alpha_3)$, with $K(x) = \alpha_1 x + \alpha_2 sin(\alpha_3 x)$. In a traditional GP search, these parameters are all set to 1, which does not necessarily lead to the best possible performance for this particular syntax. Indeed, if the optimal solution is $K^O(x) = 3.3x + 1.003sin(0.0001x)$, then individual K might be easily lost during the selection or survival processes [1].

This seems to be a glaring weakness in the standard GP approach. While the syntactic-based search in GP has solved a variety of difficult problems, it is nonetheless very inefficient, leading to important practical limitations, such as search stagnation, bloat and uninterpretable solutions (Silva and Costa, 2009). In other words, GP performs a highly

[1] Some GP systems do include numerical terminals or random ephemeral constants, while these constants can act as coefficients in an evolved expressions, they are not subject to optimization after they are introduced into the syntax of a particular individual.

explorative search, since the search operators can produce large fitness changes with only modest syntactic modifications or vice-versa. Therefore, the inclusion of a LS procedure could enhance the performance of the search. In this chapter, we study what are the effects of including a local optimization strategy within a GP run, configured as an HSE system.

4.3 PREVIOUS WORK

Many works have studied how to combine an evolutionary algorithm (EA) with a local optimizer, usually referred to as memetic algorithms, with impressive results in a variety of domains (Chen et al., 2011). A useful taxonomy of memetic algorithms can be derived based on how inheritance is carried out during the evolutionary process (Chen et al., 2011). Suppose that $(\mathbf{h}, \mathbf{h}^o) \in \mathbb{G}^2$, where $\mathbf{h}$ is an individual solution and $\mathbf{h}^o$ is the solution generated after a LS is applied on $\mathbf{h}$. Obviously, for a minimization problem, $f(\mathbf{h}^o) \leq f(\mathbf{h})$, where f is the objective function. Thus, $\mathbf{h} \neq \mathbf{h}^o$, unless $\mathbf{h}$ was in fact a local optima. Then, a memetic algorithm could proceed in two distinct ways with respect to inheritance. In a *Lamarckian* algorithm, the traits acquired during the local search, captured in $\mathbf{h}^o$, replace those of the original individual $\mathbf{h}$; i.e., the inheritance of acquired characteristics $(\mathbf{h}, f(\mathbf{h})) \rightarrow (\mathbf{h}^o, f(\mathbf{h}^o))$. On the other hand, in a *Baldwinian* algorithm, the local optimization process only modifies the fitness of an individual; $(\mathbf{h}, f(\mathbf{h})) \rightarrow (\mathbf{h}, f(\mathbf{h}^o))$; i,e., ontogenic evolution. In this work, we propose a Lamarckian approach for a GP system.

When applying a LS strategy to tree-based GP, there are basically two broad approaches to follow. (1) To apply a LS on the syntax of a tree or (2) to apply it numerically on the parameters of the tree, as described in Equation (4.1). Regarding the former, two recent works are noteworthy. In (Azad and Ryan, 2014), the authors apply a greedy search on a randomly chosen GP node, attempting to determine the best function to use in that node among all the functions in the primitive set, constrained only by the arity of the initial function. Moreover, to reduce computational overhead the authors apply a heuristic decision rule to decide which trees are subject to local optimization, preferring smaller trees in the population. This heuristic has the positive effect that it biases the search towards smaller trees, so it is adopted in our current proposal. In (Wang et al., 2014), the authors propose a multi-objective GP algorithm to evolve decision tree classifiers, and use two specialized mutation operators that limit their resulting phenotypic variations within a local neighborhood of the parent classifier.

Regarding the optimization of numerical parameters within the tree, the following works are relevant. In (Topchy and Punch, 2001), gradient descent is used to optimize numerical constants within a GP tree, achieving good results on symbolic regression problems. However, the work only optimizes the value of the terminal elements (tree leaves), and it does not consider parameters within internal nodes. Additionally, that work only considers training fitness, a highly deceptive measure of learning. Similarly, in (Zhang and Smart, 2004) and (Graff et al., 2013) a LS algorithm is used to optimize the value of constant terminal elements. In (Zhang and Smart, 2004), gradient descent is used and tested on classification problems, while (Graff et al., 2013) uses Resilient Backpropagation (RPROP) and evaluates the proposal on a real-world problem. In (Zhang and Smart, 2004), the authors apply gradient descent on every individual of the evolving population, an obvious computational bottleneck, while (Graff et al., 2013) only applies RPROP on the best individual from each generation. However, it is not evident which strategy can offer the best results in new scenarios, particularly since both (Zhang and Smart, 2004; Graff et al., 2013) evaluate their approaches on specific problem instances.

Other recent works have completely changed the basic GP framework to include an explicit parametrization of syntactic expressions. The fast function extraction (FFX) algorithm (McConaghy, 2011), for instance, poses the symbolic regression problem as that of finding a linear combination of a subset of candidate basis functions. Thus, FFX builds linear in parameter models, and optimizes using a modified version of the elastic net regression technique, eliminating the evolutionary process altogether. In the prioritized grammar enumeration (PGE) technique (Worm and Chiu, 2013), dynamic programming replaces the the basic search operators of traditional GP, and numerical parameters are optimized using the non-linear Levenberg-Marquardt algorithm.

In this chapter, we propose a very simple parametrization approach for GP trees, by constraining the number of internal parameters of each tree regardless of its size. A similar approach was developed by (Kommenda et al., 2013), with some noteworthy differences; for instance, parameters are only used to replace constants terminals, and each tree is enhanced by adding a linear upper tree, that effectively adds a weight coefficient and a bias to the entire tree. Then, the Levenberg-Marquardt optimizer is used to find the optimal values for these parameters. Similarly, in (Tsai, 2011), the authors propose a weighted GP by inserting coefficients in binary trees. In this work, weight optimization is done by means of GA, which requires parameter tuning itself to reach a global optimum. However, this study has some drawbacks, it lacks the study of the influence of optimized solutions in the evolutionary process, it is tested over an specific problematic and experimental reproducibility is not clear.

Finally, the work presented by (Arnaldo et al., 2014) uses a multivariate linear regression approach to optimize evolved solutions for symbolic regression problems. Particularly, the search is conducted by standard GP, but the linear regression is used as a Baldwinian memetic algorithm, such that the tree is decomposed into a set of subtrees which are then linearly combined discarding the original tree structure, a much more explorative approach then the one presented here. Then, to control code growth a multiobjective approach us included to account for solution complexity.

Here, we propose a Lamarckian memetic algorithm, where the main goal is to combine the explorative capabilities of GP without changing its canonical implementation, and incorporate stronger exploitative features by means of numerical optimization.

4.4 INTEGRATING LOCAL OPTIMIZATION STRATEGIES WITHIN GP

Intuitively, through tree parametrization and local optimization, a GP search should converge faster towards high quality solutions. As stated before, the proposal is to develop an HSE-GP, by parameterizing GP trees and including a Lamarckian memetic strategy. Therefore, the parametrization of GP trees must be defined; then, a particular LS method must be chosen. Finally, a decision strategy must be suggested to determine on which individuals should a LS be applied. The proposals for each of these issues are presented below.

4.4.1 *Parametrization of GP trees*

For this study, we propose a simple and naive approach to add parameters within GP trees. For each function in the function set $g_k \in F$, we add a unique weight coefficient $\theta_k \in \mathbb{R}$, such that each function is now defined by

$$g_k' = \theta_k g_k \tag{4.2}$$

where g_k' is the new parameterized function, $k \in \{1,\ldots,r\}$ and $r = |F|$. Note that each θ_k is linked to a single g_k, such that the parameter vector of tree K_i is given by $\boldsymbol{\theta} \in \mathbb{R}^m$ with m the number of different functions included in K_i such that $m \leq r$. Notice that if a tree contains several instances of a particular function, all instances of this function share the same coefficient. Indeed, this severely constraints the optimization process, particularly for large trees that can include many instances of the same function. To be clear, it is not argued that such a parametrization should be considered as the best possible alternative. Nonetheless, it does have one important consequence, it bounds the size of the search space for each LS process, something that could become overwhelming if the GP trees grew to large, which tends to happen frequently due to bloat (Silva and Costa, 2009).

In all trees, every θ_k is initialized to unity, which would be their implicit value in a standard GP. Since the proposed algorithm is a Lamarckian memetic search, the standard GP search operators (subtree mutation and subtree crossover) still only operate at the syntax level, exchanging g'_k nodes without affecting their respective θ_k.

4.4.2 *Local Search mechanism*

Potentially, any tree can be of linear or non-linear form; however, for convenience we treat every tree as non-linear expression. This is a multi-dimensional non-linear optimization problem, which can be solved using least squares.

The above problem, is formally expressed by the following cost function

$$\min_{\boldsymbol{\theta}} ||K(\boldsymbol{\theta},\mathbf{x}) - \mathbf{y}||_2^2 = \min_{\boldsymbol{\theta}} \sum_i (K(\boldsymbol{\theta},\mathbf{x_i}) - \mathbf{y_i})^2 , \tag{4.3}$$

where $\mathbf{x}$ are the input data points, $\mathbf{y}$ are the output data points, i is the index for the regression instances, and K is the non-linear function. The problem posed in (4.3) can be solved using different methods (Yuan, 1996)(Lawson and Hanson, 1995). In this case, we chose to use a well known technique with good convergence properties and good scalability in complexity for high dimensional problems, called trust region optimization (Sorensen, 1982).

The trust region optimization method tries to minimize a smooth non-linear function subject to bounds on the variables, given by

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^m} K(\boldsymbol{\theta}),\ l_i \leq \theta_i \leq u_i\ \forall i \in \{1..m\} , \tag{4.4}$$

where $l_i \in \{\mathbb{R} \cup \{-\infty\}\}$, $u_i \in \{\mathbb{R} \cup \{\infty\}\}$, and $K : \mathbb{R}^m \to \mathbb{R}$. Conceptually, a trust region approach replaces an m-dimensional unconstrained optimization problem by an m-dimensional constrained problem. This results in an approximate solution, since it does not needed to be solved with high accuracy. One of the appealing points of these methods is their strong convergence properties (Coleman and Li, 1992). The idea behind a trust region method for $min_{\theta \in \mathbb{R}^m} K(\theta)$ is simple. The increment

$s_k = x_{k+1} - x_k$ is an approximate solution to a quadratic subproblem with a bound on the step size

$$\min_{s \in \mathbb{R}^m} \left\{ \psi_k(s) \overset{def}{=} g_k^T s + \frac{1}{2} s^T B_k s : \|\overline{D}_k\| \leq \Delta_k \right\} , \tag{4.5}$$

where $g_k \overset{def}{=} \nabla f(x_k)$, B_k is a symmetric approximation to the Hessian matrix $\nabla^2 f(x_k)$, $\overline{D}_k$ is a scaling matrix, and Δ_k is a positive scalar representing the trusted region size. Solving (5.6) efficiently is not a trivial task, see (Shultz et al., 1982; Sorensen, 1982; Steihaug, 1983; Moré and Sorensen, 1983); Figure 4.1 shows an illustration of the trust region method. Here, the method proposed in (Coleman and Li, 1993) is used, which does not require the solution of a general quadratic programming subproblem at each iteration.

4.4.3 *Integrating LS into GP*

The final issue that must be considered is to determine on which individuals is the LS applied, and at what moment during the evolutionary process. For the latter point, LS is applied after a complete evolutionary loop is completed; i.e., LS is applied after the survival criterion is applied and before the following generation begins. This means that the local optimization of individuals from generation t impacts the search at generation $t + 1$. For the former point, several different approaches are evaluated.

First, three naive approaches are considered, two of which have been used in previous memetic GP systems. Probably the simplest is to apply a LS on all of the individuals, this method is referred to as LSALL-GP. This approach will inherently introduce a large computational overhead. Another approach is to be more selective, and only apply a LS on the best individual from each generation, this method is referred to as LSBEG-GP. Conversely, LS can also be applied on the worst individual from each generation, this variant is called LSWEG-GP. This variant might be useful for extreme cases of individuals that present the correct structure, but a highly suboptimal parametrization.

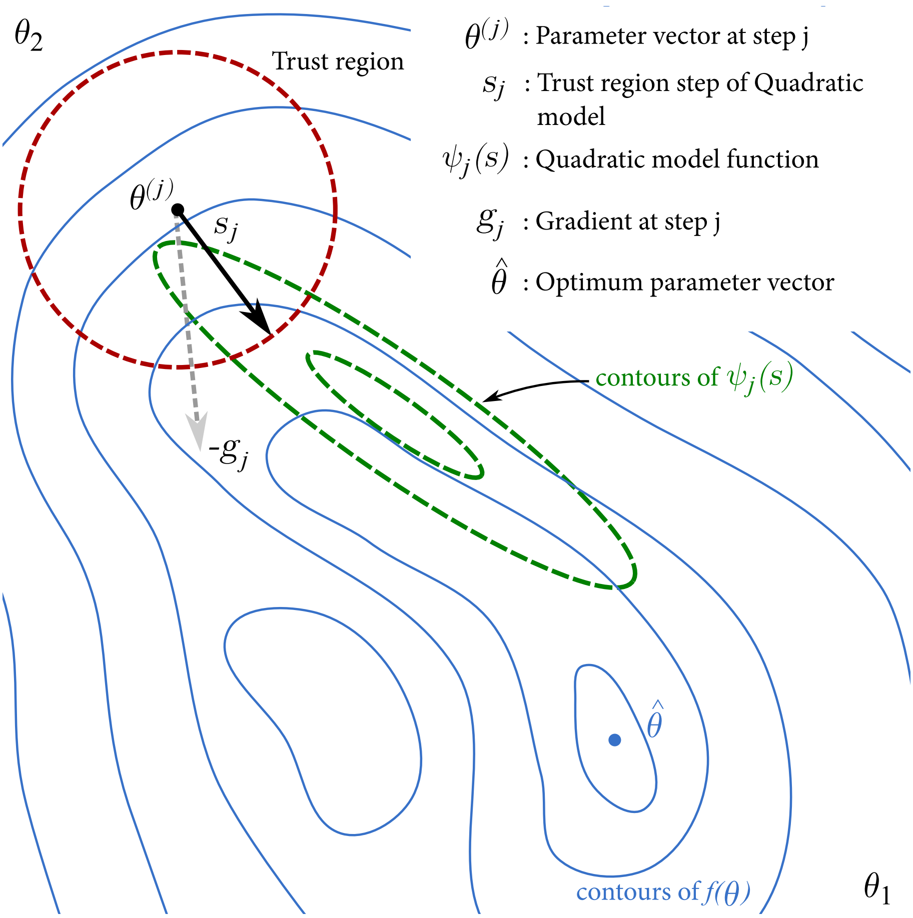

θ_2
Trust region
$\theta^{(j)}$: Parameter vector at step j
s_j : Trust region step of Quadratic model
$\psi_j(s)$: Quadratic model function
g_j : Gradient at step j
$\hat{\theta}$: Optimum parameter vector
$\theta^{(j)}$
s_j
$-g_j$
contours of $\psi_j(s)$
$\hat{\theta}$
contours of $f(\theta)$
θ_1

Figure 4.1: Visualization of the trust region algorithm showing the landscape of $f(\theta)$ as the objective function, equivalent to Equation 4.4, at iteration j.

Table 4.1: GP Parameters for the different methods.

Parameter	LSBEG-GP	LSWEG-GP	LSRP-GP	LSBS-GP	LSWS-GP	LSALL-GP
θ	vector of ones					
F_{BEST}	yes	no	no	no	no	no
F_{WORST}	no	yes	no	no	no	no
p_{LS}	1	1	0.1	0.5	0.5	1
PER_{LS}	0	0	100%	10%	10%	100%

The fourth variant is referred to LS Random Population or LSRP-GP, where every individual of the population is a viable candidate for a LS optimization with a probability p_{LS}, basically implemented as a mutation operator. Here, it is assumed that a low probability is desirable, to minimize the added computational cost. The fifth variant is called LS Best Subset or LSBS-GP, which is the same as LSRP-GP, except that the individuals that are valid candidates for LS are those in the best PER_{LS} percentile of the population, a second algorithm parameter. Finally, the sixth variant is called LS Worst Subset or LSWS-GP, takes the same approach as LSRP-GP, but candidate individuals for a LS are those in the worst PER_{LS} percentile. All of the methods and their parameters are summarized in Table 4.1.

4.5 EXPERIMENTAL SETUP

The algorithms that are evaluated are LSBEG-GP, LSWEG-GP, LSRP-GP, LSBS-GP, LSWS-GP and LSALL-GP; also, a standard GP is included as a control method. All the algorithms were implemented in Matlab using the GPLAB[2] Toolbox (Silva and Almeida, 2005) modifying its core functionality to integrate the LS procedure. The set of experiments cover a series of well known benchmark symbolic regression problems. Five synthetic problems were used: Keijzer-6 (Keijzer, 2003), Korns-12 (Korns, 2011), Vladislavleva-4 (Vladislavleva et al., 2009), Nguyen-7 (Uy et al., 2011), Pagie-1 (Pagie and Hogeweg, 1998); and one real-world

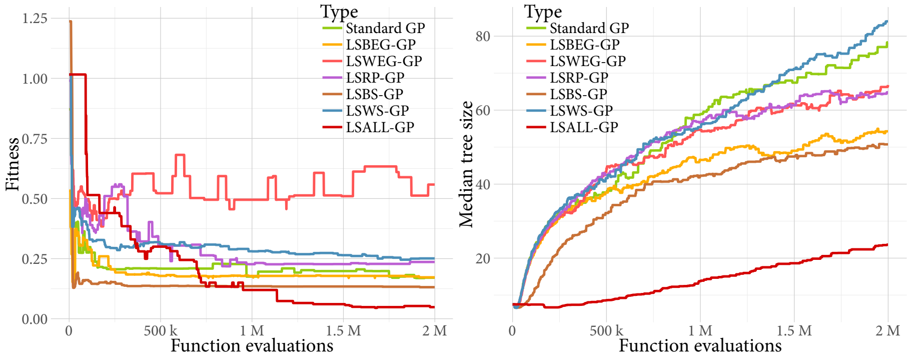
[2] http://gplab.sourceforge.net/

Table 4.2: General GP parameters

Parameter	Value
Runs	30
Population	500
Function evaluations	2'000,000
Training set	as indicated in (McDermott et al., 2012)
Testing set	as indicated in (McDermott et al., 2012)
Crossover operator	Standard subtree crossover, 0.9 probability
Mutation operator	Mutation probability per node 0.05
Tree initialization	Ramped Half-and-Half, maximum depth 6
Function set	as indicated in (McDermott et al., 2012)
Terminal set	Input variables, constants as indicated in (McDermott et al., 2012)
Selection for reproduction	Tournament selection of size 3
Elitism	Best individual survives
Maximum tree depth	17

problem. All of them are suggestions made in (White et al., 2013), a recent survey on GP benchmarking.

Two performance measures are used to evaluate the different algorithms. First, fitness evaluation over the test set for the best solution found thus far. Second, the average population size, a relevant measure regarding the bloat phenomenon and solution complexity (Trujillo et al., 2013). The evolution of these measures is analysed with respect to the total number of fitness function evaluations instead of generations, to account for the LS iterations. The LS performs a maximum of 400 iterations. However, we do not consider any additional computational effort due to the LS, which underestimates the computational cost of performing LS. The stopping criteria is the number of function evaluations. The GP configuration parameters for all variants are shown in Table 4.2. Some of these parameters vary depending on the problem, as suggested in (McDermott et al., 2012). For the case of Tower problem, total samples were divided in a ratio of 30/70% for the training and testing sets. As stated before, Table 4.1 summarizes the parameter values for the memetic GP variants.



(a) (b)

Figure 4.2: Results for problem Keijzer-6 plotted with respect to total function evaluations: (a) Fitness over test data; and (b) Average program size. Both plots show median values over 30 independent runs.

4.6 RESULTS AND SUMMARY

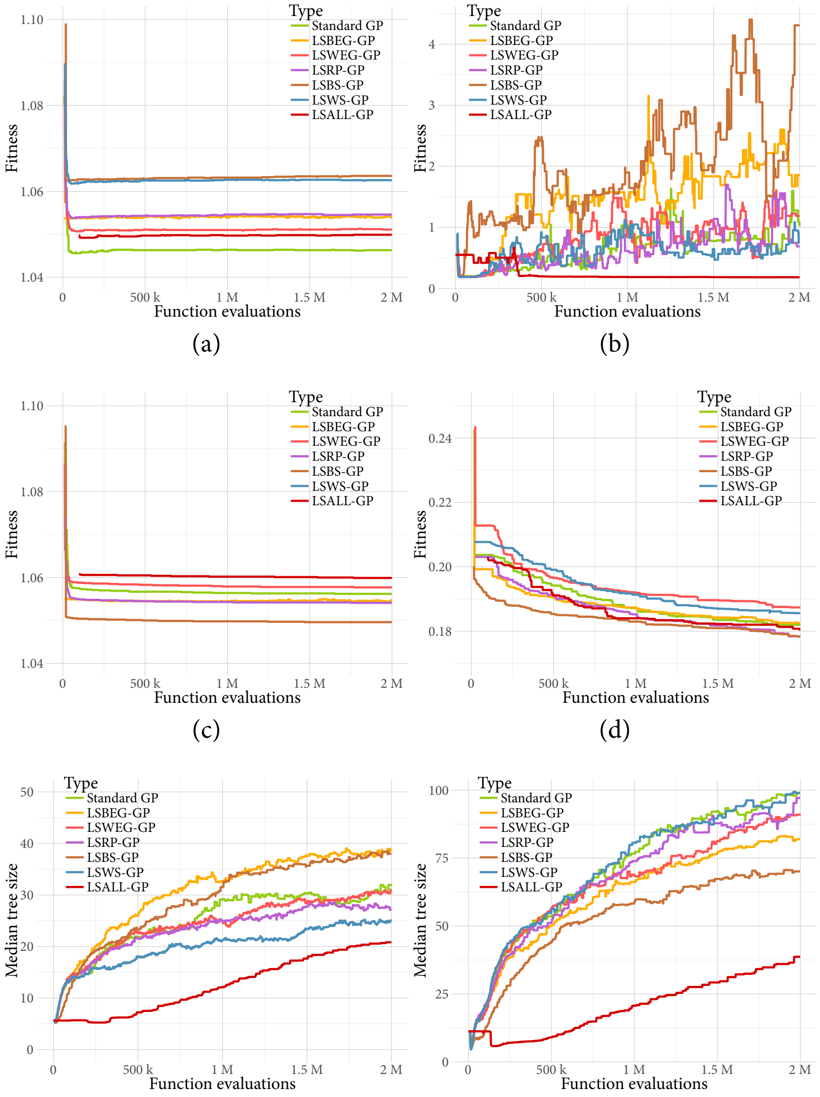
Figures 4.2-4.6 summarize the results of the all tested techniques, showing the median values computed over 30 independent runs. The first problem is Keijzer-6 shown in Figure 4.2, considered a simple or easy problem. The difficulty is raised with the testing data, since the solution must extrapolate outside the training domain. In general, most methods exhibit similar performance, with some notable exceptions. First, LSWEG-GP obviously achieves the worst test performance, while LSBS-GP and LSALL-GP exhibit the best. In particular, LSBS-GP and LSALL-GP converge quickly to very good performance; in fact LSBS-GP could have been halted very early in the run. However, after a million function evaluations LSALL-GP, shows a noticeable improvement, achieving the best results. In terms of size, both of these methods also exhibit the best results, with LSALL-GP producing the smaller trees.

Figure 4.3 presents the results for the Korns-12 problem, considered difficult in GP literature since standard GP usually cannot find the true expression (Korns, 2011). The problem includes redundant input data that does not influence its output. The idea is to test GP algorithms on their ability to avoid over-fitting. With respect to test error, Figure 4.3(a) shows that most algorithms perform quite similarly, all of them converging to their best median values quite early in the runs. What is noticeable is the performance of LSBS-GP, with the worst test performance, but also the best training performance, shown in Figure 4.3(c), indicating a slight over-fitting. Regarding the evolution of size, shown in Figure 4.3(e), all methods exhibit similar sizes, but code growth is noticeably slower for LSALL-GP.

For the Vladislavleva-4 problem shown in Figure 4.3, considered a difficult test case for GP (Vladislavleva et al., 2009), over-fitting is also noticeable. In Figure 4.3(b) we can see that test fitness varies greatly across the runs for most methods, particularly for LSBS-GP. Conversely in Figure 4.3(d) we can observe a smooth monotonic convergence for the training fitness. The case of LSBS-GP is noticeable, particularly since it is clear that the method achieves its best test fitness early in the run, and then starts to over-fit. The only method that did not over-fit was LSALL-GP, exhibiting the best performance over the test data. Finally, with respect to program size (Figure 4.3(f)), once again all methods show similar trends, with LSALL-GP producing substantially smaller trees.

Problem Nguyen-7 and Pagie-1 exhibit similar outcome in terms of fitness and program size. In both, among all methods LSBS-GP exhibits a faster convergence to a better solution quality computed over test data and also achieves the best results along with LSALL-GP. Once again LSALL-GP produces the smallest trees, with LSBS-GP the second best, however in these problems the difference between both of these methods is smaller than in the other cases. All other variants show similar performance based on both measures.

Finally, the Tower problem is an industrial real-world problem on modeling gas chromatography measurements of the composition of a distillation tower. This problem contains 5000 records and 23 potential input variables. The measurements (5000 for each variable) are not treated as time series, but simply used as samples for a regression model. In this case, LSALL-GP again achieves the most interesting results, both in regards to test-fitness and solution size. From the remaining meth-

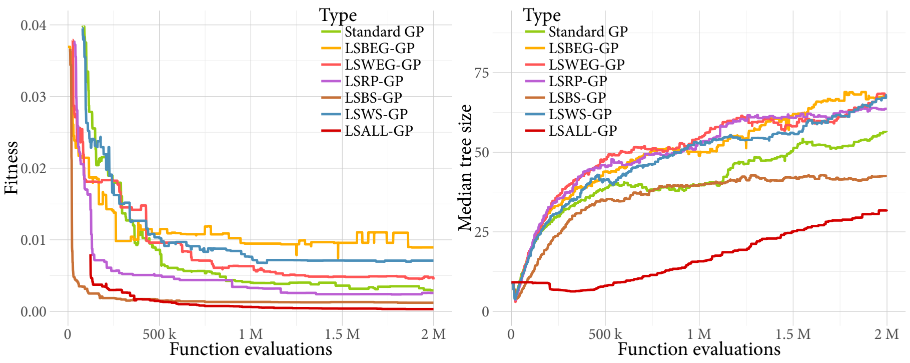


(a) (b)

(c) (d)

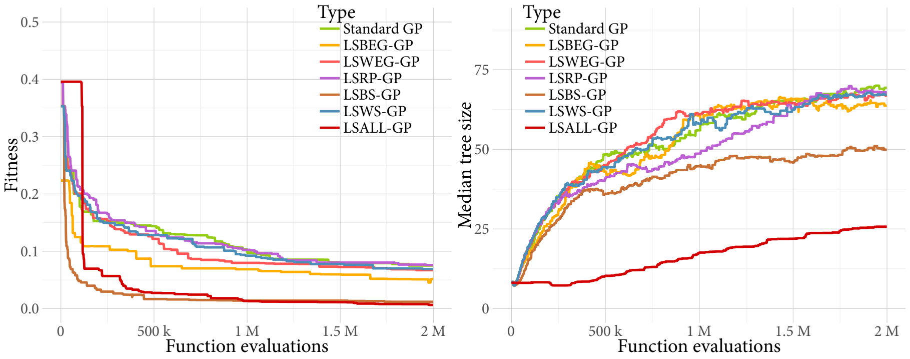
(e) (f)

Figure 4.3: Results for problems Korns-12 (a,c,e) and Vladislavleva-4 (b,d,f): (a,b) Fitness over test data; (c,d) Fitness over training data; and (e,f) Average program size. All plots show median values over 30 independent runs.



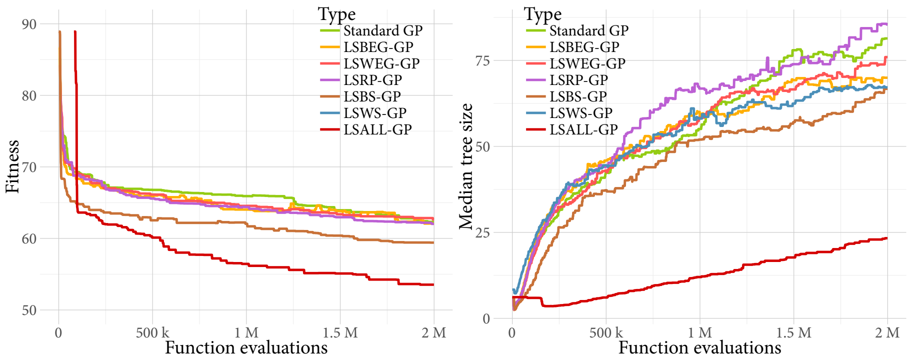
(a) (b)

Figure 4.4: Results for problem Nguyen-7 plotted with respect to total function evaluations: (a) Fitness over test data; and (b) Average program size. Both plots show median values over 30 independent runs.



(a) (b)

Figure 4.5: Results for problem Pagie-1 plotted with respect to total function evaluations: (a) Fitness over test data; and (b) Average program size. Both plots show median values over 30 independent runs.



(a) (b)

Figure 4.6: Results for Tower problem plotted with respect to total function evaluations: (a) Fitness over test data; and (b) Average program size. Both plots show median values over 30 independent runs.

ods, LSBS-GP shows the best performance, but still noticeably worse that LSALL-GP.

A final summary of the performance of each method is presented in Table 4.3, that shows three different snapshots of the median performance of each algorithm, after 250,000, 1 million and 2 million function evaluations. In general, LSALL-GP exhibits the best performance after all of the allowed function evaluations. However, if we only consider 250,000 function evaluations, then LSBS-GP exhibits the best performance on three of the six problems, and also the second best performance on two others.

From the remaining methods, those that show the worst performance are LSWEG-GP and LSWS-GP, that apply LS to the worst solutions in the population, what definitely seems to be a bad strategy. Moreover, while LSBS-GP shows good performance, the deterministic LSBEG-GP that applies LS to the best solution is notably inferior to the best methods, a noteworthy results since several previously published works have employed this memetic strategy.

Table 4.3: Summary of median fitness computed over the test set of each problem. The *Sample* column indicates the number of function evaluations performed; bold indicates the best results.

Problem	Sample	S-GP	LSBEG-GP	LSWEG-GP	LSRP-GP	LSBS-GP	LSWS-GP	LSALL-GP
Keijzer-6	250,000	0.2	0.2	0.5	0.54	**0.15**	0.29	0.46
	1000000	0.17	0.17	0.49	0.22	0.13	0.28	**0.11**
	2000000	0.17	0.17	0.5	0.23	0.13	0.25	**0.04**
Korns-12	250,000	**1.04**	1.05	1.05	1.05	1.06	1.06	**1.04**
	1000000	**1.04**	1.05	1.05	1.05	1.06	1.06	**1.04**
	2000000	**1.04**	1.05	1.05	1.05	1.06	1.06	**1.04**
Vladislavleva-4	250,000	**0.28**	0.47	0.31	0.35	1.03	0.34	0.5
	1000000	1.03	1.33	0.98	0.6	1.89	0.99	**0.18**
	2000000	1.01	1.85	1.18	0.68	4.3	0.74	**0.18**
Nguyen-7	250,000	0.02	0.01	0.02	0.005	**0.002**	0.02	0.003
	1000000	0.004	0.009	0.006	0.003	0.001	0.007	**0.0006**
	2000000	0.002	0.008	0.004	0.002	0.001	0.007	**0.0003**
Pagie-1	250,000	0.15	0.1	0.15	0.16	**0.02**	0.14	0.05
	1000000	0.1	0.06	0.07	0.1	0.01	0.09	**0.01**
	2000000	0.07	0.05	0.06	0.07	0.01	0.06	**0.006**
Tower	250,000	67.5	66.9	67	67.4	63.2	66.8	**61.98**
	1000000	65.92	64	64.54	64.39	62	64.52	**56.43**
	2000000	62.31	62.08	62.64	62.03	59.41	62.42	**53.53**

4.7 CHAPTER CONCLUSIONS

This work studies the problem of integrating a local optimization process into a GP, using a Lamarckian HSE memetic approach. A comparative study is performed, evaluating different ways in which to incorporate a LS during the basic evolutionary process of GP, evaluating performance on symbolic regression problems. A simple tree parametrization is proposed, bounding the size of parameter space for each individual tree, even if bloat occurs during the run. The local optimization is done by a trust region technique that determines the optimal coefficients posing a basic non-linear curve fitting problem. As stated before, it is not clear what might be the best strategy to incorporate LS into GP, so different stochastic and deterministic variants are extensively evaluated over a set of widely used benchmark problems. In particular, each method uses different heuristic decisions to determine which individuals in the GP run should be subject to a local optimization process; experimental results suggest the following.

In general, it seems that a memetic GP almost always outperforms a standard GP, in terms of both solution quality and solution size. Moreover, among the different methods that were tested, several insights can be gathered. First, it does not appear to be beneficial to use a LS as another mutation strategy, such that all individuals might be candidates for a LS, given the average performance of LSRP-GP. Second, it does not seem useful to apply LS to the worst individuals in the population, as seen by the performance of LSWEG-GP and LSWS-GP. Third, many works have used the simple deterministic heuristic of applying LS to the best solution found, either at the end of the run or at each generation. However, this to does not seem to be an adequate strategy, given the performance of LSBEG-GP. Finally, of all the methods, the best performance was achieved when LS is applied to all of the solutions (LSALL-GP) or to random individuals chosen from the top percentile (w.r.t. fitness) of the population (LSBS-GP).

For all problems, the best performance was achieved by LSALL-GP. However, if only a small amount of computational effort is feasible, then LSBS-GP seems to be the best option, given its fast convergence. Moreover, the difference in computational effort between both methods is understated in the results presented here, since only the total iterations in the LS are considered, omitting the total effort devoted towards computing approximate derivatives or matrix inversions. Nevertheless, LSALL-GP also shows a substantial ability to curtail bloating during the search, this was indeed expected. Since the search process in LSALL-GP focuses primarily on parameter optimization, limiting the total of syntactic search performed by the GP crossover and mutation operators; i.e., the total number of generations is quite low for LSALL-GP, basically eliminating the possibility of bloating.

In appendix A, we extend the experimentation to real-world symbolic regression problems, providing additional insights in the evolution process. Consequent appendixes, namely B, C and D focus in three different case studies that uses the integration of LS into GP. The compilation of the results derived from all these works produced a compelling conclusion: the global search from GP is greatly benefited by a LS mechanism. Nevertheless, further investigation is required to extend the domain of representation of LS operators into other applications, which will be discussed in the future directions at Chapter 7.

In the next chapter, we will explain our contribution toward classification problems using a memetic approach to GP, gathering experimental evidence as result. We will explain the differences between the representation used in regression and the needed changes to address binary classification problems.

5 CLASSIFICATION

Based on Z-Flores, E., Trujillo, L., Schütze, O., and Legrand, P. (2015). A Local Search Approach to Genetic Programming for Binary Classification. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO'15, pages 1151–1158. ACM

5.1 INTRODUCTION

The general goal of the genetic programming (GP) paradigm is to evolve specialized syntactic expressions that can solve a user defined problem, mostly used in supervised learning tasks. In particular, GP is widely used to generate mathematical functions, or operators, that solve symbolic regression and classification problems. In the previous chapter we explored different variants of LS integration into GP, providing useful insight on the effectiveness of the hybrid search mechanisms in symbolic regression problems. However, there was still a couple of open questions: is it possible to make a representation to handle classification problems? if so, will the memetic GP approach be efficient as well?. To try to answer these questions, in this chapter we introduce a methodology to parameterize GP trees, employ a LS method to numerically tune them and apply this process for binary classification problems.

Here, a domain specific fitness function is used, based on the Receiver Operating Characteristic (ROC) curve of each GP individual. Experimental results are encouraging in several problems, performance is significantly improved relative to a standard GP search, based both on fitness and solution size. Moreover, by using widely used real-world benchmarks, it is clear that our proposal compares favorably with other proposals from the GP and machine learning literature.

The remainder of this chapter is organized as follows. Section 5.2 describes our approach and Section 5.3 presents the experimental work. Finally, concluding comments are given in Section 5.4.

5.2 INTEGRATING THE LOCAL SEARCH MECHANISM WITHIN GP

For binary classification, GP is used as a supervised learning problem, where a training set $\boldsymbol{x}$ of n-dimensional patterns with a known classification, are used to derive a mapping function $g(x) : \mathbb{R}^n \rightarrow \{0, 1\}$, with a threshold δ as classification decision. Afterwards, for convenience, we transform the classification problem into a regression one by inserting a continuous function in order to get gradient information, and to be able to use a numerical optimizer.

5.2.1 *GP Tree Parameterization*

First, we follow the strategy proposed in (Kommenda et al., 2013), where a small linear subtree is added on top of the root node of the original tree $K(\boldsymbol{x})$, such the new extended tree $K'(\boldsymbol{x})$ is given by

$$K'(\boldsymbol{x}) = \theta_2 + \theta_1(K(\boldsymbol{x})) \, , \tag{5.1}$$

where θ_1 and θ_2 are the first two parameters from $\boldsymbol{\theta}$, as shown in Figure 5.1. This subtree provides shifting and scaling properties, so the optimizer later on can have more freedom in its capabilities to find a better solution. This approach is closely related to the linear scaling technique (Keijzer, 2003), allowing the syntactic search operators to focus on evolving the desired shape of the evolved function without worrying about optimizing the scale or bias of the output.

Afterwards, for each node n_k in a given tree we add a weight coefficient $\theta_k \in \mathbb{R}$, such that each node is now defined by

$$n'_k = \theta_k n_k \, , \tag{5.2}$$

where n'_k is the new modified node, $k \in \{1,...,r\}$, $r = |Q|$ and Q is the tree representation. Notice that each node has an unique parameter that can be modified to help meet the overall optimization criteria of the non-linear expression.

At the beginning of the GP run, each parameter is initialized by $\theta_k = 1 \forall k$, where $|\theta| = r$. During the GP syntax search, subtrees belonging to different individuals are swapped, added or removed (following the standard crossover/mutation rules) together with its corresponding parameters, without affecting their values; a memetic search process with Lamarckian inheritance.

5.2.2 *A Continuous Transfer Function*

In a binary classification problem, the desired output is either 0 or 1, a discrete output. Indeed, this discontinuity will have an homologous discontinuity in parameter space, and thus the LS algorithm can easily fail. To provide a continuous output, and provide gradient information to the LS method, a transfer function is added as a root node.

Here, a sigmoid function is used, producing the following non-linear tree

$$sig(K'(\boldsymbol{x},\boldsymbol{\theta})) = \frac{1}{1 + e^{\rho(K'(\boldsymbol{x},\boldsymbol{\theta})-\delta)}} , \tag{5.3}$$

where K' is the program output evaluated over the input vector $\boldsymbol{x}$, $\boldsymbol{\theta}$ is the parameter vector corresponding to the program K', ρ is the sigmoid slope parameter and δ is the classification threshold, where the function has a value of 0.5. The sigmoid function is normalized to the range [0,1]; therefore, if $sig(K'(\boldsymbol{x},\boldsymbol{\theta})) \leq 0.5$ then the class label is 0, and 1 otherwise. The proposed tree extensions and parameterizations are depicted in Figure 5.1.

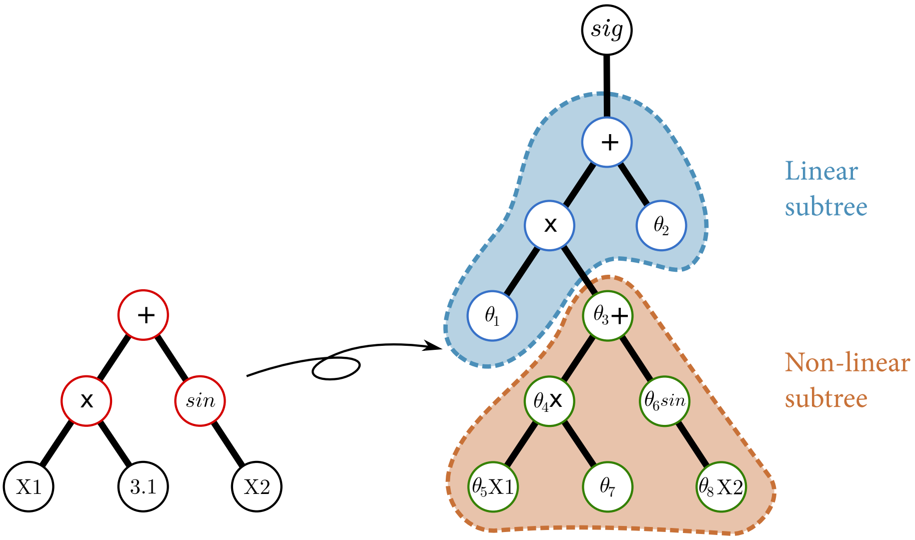


Figure 5.1: Example of tree transformation, product of parameterization and subtree addition.

5.2.3 *The Local Search Mechanism*

In this work we treat each tree as a non-linear expression. We also assume a non-discontinuity for the evaluated tree over the interval constrained by the input training data.

The problem to be solved is an optimization one, where we want to best fit a non-linear function to an output vector containing either zero or one values (each class label). It can be formally defined as

$$\min_{\boldsymbol{\theta}} \| sig(K'(\boldsymbol{\theta},\boldsymbol{x})) - \boldsymbol{y}\|_2^2 = \min_{\boldsymbol{\theta}} \sum_i (sig(K'(\boldsymbol{\theta},x_i)) - y_i)^2 \,, \tag{5.4}$$

where $\boldsymbol{x}$ is the input data vector, $\boldsymbol{y} \in \{0,1\}$ is the known class label, sig is the sigmoid calculated over the program K', i is the index for the problem instances and $\boldsymbol{\theta}$ is the parameter vector.

The problem presented in (5.4) can be solved using different techniques (Yuan, 1996; Lawson and Hanson, 1995). In this work, we follow the same method used in previous chapter, the Trust Region opti-

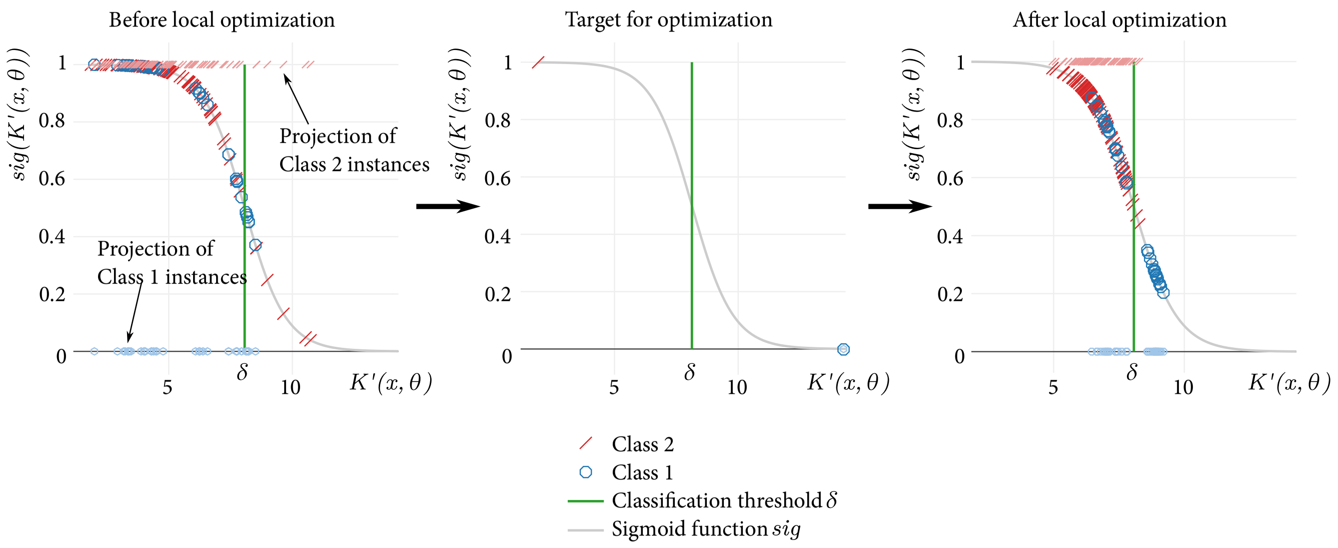


Figure 5.2: Example of the optimization effect over an actual individual for the Parkinsons problem (Little et al., 2007). Even though the solution was clearly a bad classifier, after optimization accuracy improved. Projection of both classes are shown for a clearer visualization on the position of instances relative to the classification threshold.

mizer, belonging to the family of non-linear Gauss-Newton methods (Sorensen, 1982). One example of this family is the recent approach to the Levenberg-Marquardt algorithm, which follows the Trust Region concept.

As a reminder, in the Trust Region algorithm, an iterative process is performed with the following goal

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \|sig(K'(\boldsymbol{\theta},\boldsymbol{x})) - \boldsymbol{y}\|_2^2,\ l_i \leq \theta_i \leq u_i\ \forall i \in \{1,\ldots,m\}\,, \tag{5.5}$$

where $l_i \in \{\mathbb{R} \cup \{-\infty\}\}$, $u_i \in \{\mathbb{R} \cup \{\infty\}\}$, and $K' : \mathbb{R}^m \rightarrow \mathbb{R}$. Conceptually, a Trust Region approach replaces an m-dimensional unconstrained optimization problem by an m-dimensional constrained problem. This results in an approximate solution, since it does not needed to be solved with high accuracy. One of the appealing points of these methods is their strong convergence properties (Coleman and Li, 1992). The idea behind a Trust Region method for $min_{\theta\in\mathbb{R}^m}K(\theta)$ is simple. The increment $s_j = x_{j+1} - x_j$ is an approximate solution to a quadratic subproblem with a bound on the step size

$$\min_{s\in\mathbb{R}^m}\left\{\psi_j(s) \overset{def}{=} g_j^T s + \frac{1}{2}s^T B_j s : \|\overline{D}_j\| \leq \Delta_j\right\}\,, \tag{5.6}$$

where $\psi(s)$ is the quadratic function, $g_j \overset{def}{=} \nabla f(x_j)$, B_j is a symmetric approximation to the Hessian matrix $\nabla^2 f(x_j)$, $\overline{D}_j$ is a scaling matrix, and Δ_j is a positive scalar representing the trusted region size. Solving (5.6) efficiently is not a trivial task, see (Sorensen, 1982; Moré and Sorensen, 1983). Here, the method proposed in (Coleman and Li, 1993) is used, which does not require the solution of a general quadratic programming subproblem at each iteration. Figure 5.2 shows the effect of performing LS over a specific individual.

5.2.4 *The Fitness Function*

In the case of binary classification, a simple method is to consider only accuracy in the training stage (Winkler et al., 2007; Eggermont et al., 1999). In this work we use the following fitness measure

$$fitness = 1 - Acc \ , \tag{5.7}$$

where Acc (accuracy; computed as the ratio of correctly classified instances among total number of cases) is calculated based on the best threshold found using the ROC curve. The ROC curve, which is commonly used by the machine learning community, can provide a robust measure of classifier performance even with an unbalanced dataset. Each operating point on a ROC curve represents the classification accuracy at a single threshold δ_i. The ROC curve is constructed by changing the classification threshold and evaluating its True Positives Rate (TPR) and False Positives Rate (FPR) values. These are calculated with TPR=TP/P and FPR=FP/N, respectively, where TP is the True Positive value, FP is the False Positive value, P is the total of positive instances and N the negative ones.

In the implementation presented here, we used 10 uniformly spaced threshold values within $[min(K'(x,\theta)) - max(K'(x,\theta))]$, which is the range given by minimum and maximum output of the extended tree, over all fitness cases. This methodology is similar as the one presented in (Bhowan et al., 2012).

In the ROC curve, the top-left area represents the most accurate classification results. Ideally, a perfect classifier should have a TPR value of one with zero FPR. Based on this, we propose a simple method to choose the best threshold picked up from the ROC curve. A vector with 45 degree slope is traced with coordinates [0,1]-[1,0], in ROC space, which represent our reference vector v_r. We then build a numerical array $\overline{\omega}$ containing the angle differences between each threshold vector v_i, with coordinates [0,1]-[FPR$_i$,TPR$_i$,] and the reference vector, as seen in Figure 5.3. This is calculated by

$$\Delta v_i = v_i(y) - v_i(x), \Delta v_r = v_r(y) - v_r(x) \tag{5.8}$$

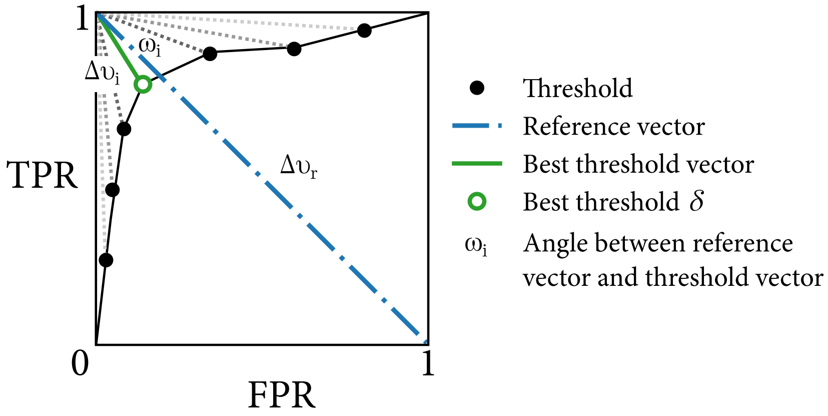


Figure 5.3: Example of a ROC curve and its best threshold. Note that each tree will generate a different ROC curve, thus presenting a different classification threshold.

and

$$\omega_i = acos\left(\frac{(\Delta v_i \cdot \Delta v_r)|\Delta v_r|}{|\Delta v_i|}\right), \tag{5.9}$$

where v_i is the vector corresponding to each threshold δ_i and ω_i is the difference angle expressed in radians. Consequently, the array $\overline{\omega} = [\omega_1, ..., \omega_k]$. The threshold corresponding to the minimum value in this array will represent our chosen threshold for the particular classifier, as seen in Figure 5.3.

5.2.5 *Integrating LS into GP*

Finally, we need to determine which individuals should be improved by the LS method. For instance, some researchers apply the LS to the best solution in the population, but (Trujillo et al., 2014) showed that better results are obtained when it is applied to all the population or a subset of the best solutions in each generation. However, in our case, the number of parameters grows proportionally with the size of the trees, providing incentive to limit the amount of times the LS is applied, especially for larger trees. Therefore, in this work we use the heuristic rule proposed

Table 5.1: Binary classification problems summary used to evaluate proposed algorithm in this work.

Name	# of instances	# of features	Brief description
Parkinsons (Little et al., 2007)	197	23	Biomedical voice measurements related to Parkinson's disease.
Diabetes (Lichman, 2013)	768	8	Diabetes present on Pima Indians patients.
Wine (Lichman, 2013)	178	13	Chemical analysis of wines.
Sonar (Lichman, 2013)	208	60	Sonar signals off metal cylinder at various angles and conditions.
Wholesale (Lichman, 2013)	440	8	Clients information of wholesale distributor.
Banknote (Lichman, 2013)	1372	5	Image information from genuine & forged banknote-like specimens.
LSVT (Tsanas et al., 2014)	126	309	Signal data from speech rehabilitation treatment on Parkinson patients.

by Muhammad and Ryan in (Azad and Ryan, 2014), which they tested with a syntactic LS process. In (Azad and Ryan, 2014), the LS method is applied stochastically for each tree, based on a probability $p(s)$ determined by the tree size s (number of nodes) and the average size of the population $\bar{s}$, based on

$$p(s) = \begin{cases} 1 & \text{if } s < (c-1)\bar{s} \\ \frac{s}{\bar{s}} + c & \text{if } (c-1)\bar{s} \leq s \leq c\bar{s} \\ 0 & \text{otherwise} \end{cases} . \tag{5.10}$$

Based on equation 5.10, shown in Figure 5.4, smaller trees are more likely to be optimized by the LS than larger trees; in particular, by setting $c = 1.5$ trees that are 50% larger than the population average are not subject to the LS process. This heuristic could provide two positive results. First, the LS method is mostly applied on small trees that have a relatively small number of parameters, simplifying the LS problem. Second, by focusing on improving smaller trees, this increases their chances for survival, increasing the chances of finding solutions that are not hampered by the bloat problem (Silva and Costa, 2009). Hereafter, we call this method Local Search Heuristic by Size (LSHS).

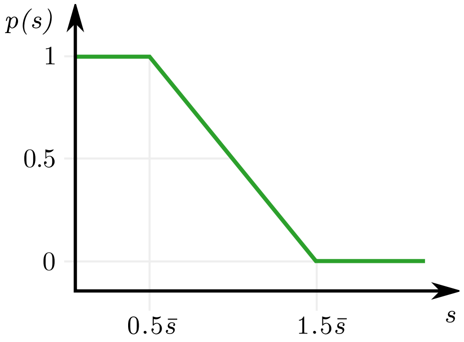


Figure 5.4: Transfer function for $p(s)$ corresponding to the heuristic method based on individuals tree sizes, selected for performing LS.

5.3 EXPERIMENTATION

Table 5.2: GP parameters

Parameter	Value
Runs	20
Population	200
Function evaluations	1’500,000
Training set	70% of complete data
Testing set	30% of complete data
Crossover operator	Standard subtree crossover, 0.8 prob.
Mutation operator	Mutation probability per node 0.15
Tree initialization	Ramped Half-and-Half, max. depth 6
Function set	+,-,×, sin,cos,log,sqrt,tan,tanh
Terminal set	Input features, constants
Selection for reproduction	Tournament selection of size 7
Elitism	Best individual survives
Maximum tree depth	15

5.3.1 *Experimental Setup*

The algorithm LSHS was evaluated and compared with a canonical GP implementation as control method. The GP settings were the same between both variants, including the fitness function.

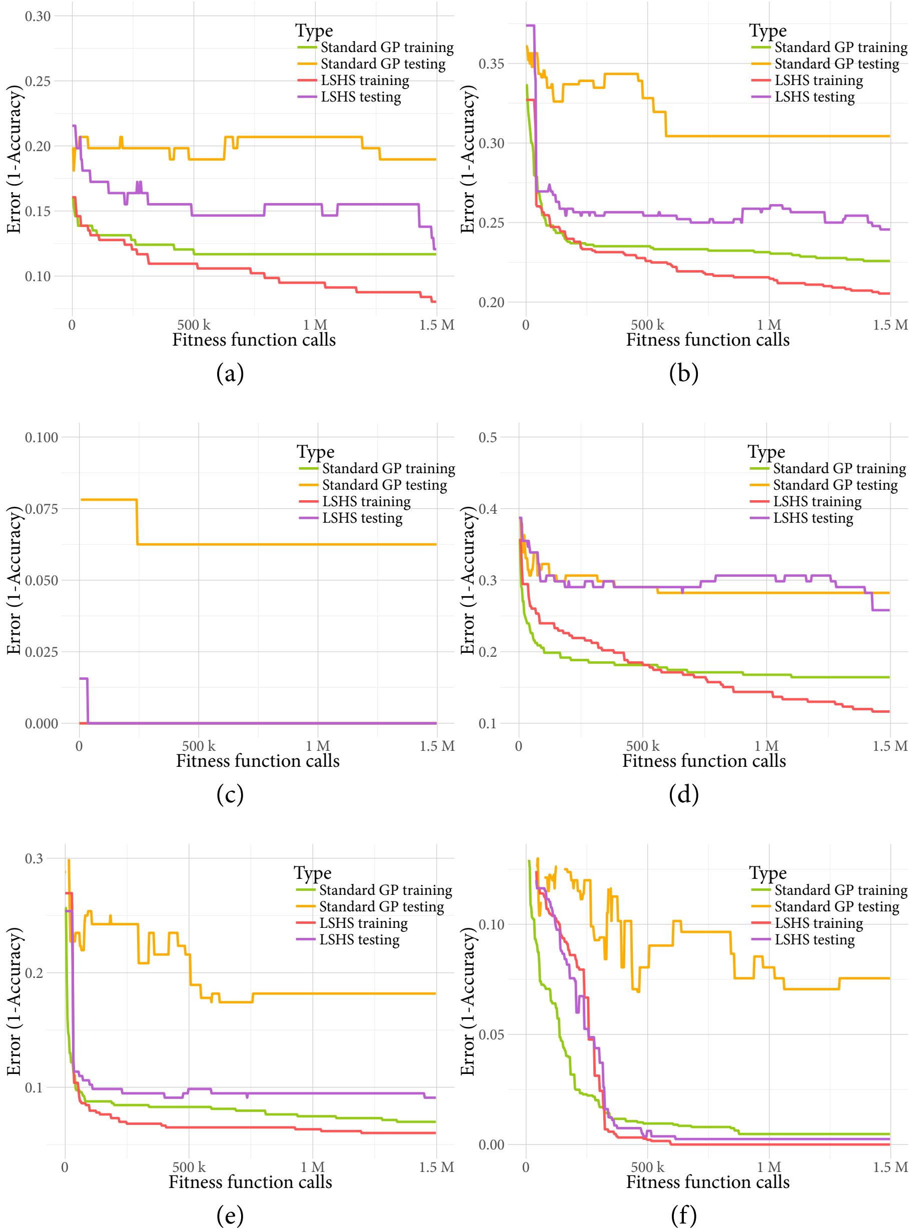
The LSHS algorithm was implemented using the GPLAB[1] Matlab toolbox (Silva and Almeida, 2005). The set of experiments covered a series of benchmark binary classification problems, taken from the UCI repository (Lichman, 2013); real problems with distinct complexity. Table 5.1 shows a summary of each evaluated problem. A couple of quality measurements were used to compare the results of the algorithm, the accuracy calculated over the test data and average population size, needed to assess the performance in terms of bloat and consumed resources. The evolution of these measure is analyzed with respect to the total number of fitness function evaluations instead of generations, to account for the LS iterations. The evaluations performed by the iterative Trust Region algorithm were also taken in account to provide a fair comparison between standard GP and LSHS. The Trust Region algorithm performs a maximum of 500 iterations. The stopping criteria is a predefined number of function evaluations. The GP configuration used in this research is listed in Table 5.2.

5.3.2 *Results*

Fig. 5.5 and Fig. 5.6 summarizes the results of the tested algorithm over several classification problems. The figure shows convergence plots for training and testing fitness with respect to the number of fitness function evaluations, showing the median over all runs. In each of the problems, the accuracy of LSHS testing data is either similar or better compared to standard GP. At least in four of them the improvement in the method is significant.

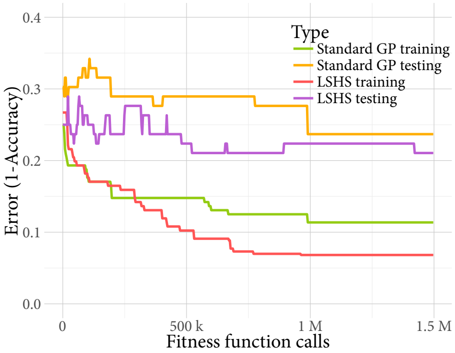
Figures 5.5(a-c) show the results for the problems Parkinsons, Diabetes and Wine. In all of them we can see a superior solution quality evolved by the LSHS method. Notice that for the Wine problem, considered an easy classification problem, standard GP reaches almost zero error for testing data, yet the LSHS algorithm presents a perfect accuracy.

[1] http://gplab.sourceforge.net/



(a) (b) (c) (d) (e) (f)

Figure 5.5: Results of fitness performance of problems Parkinsons (a), Diabetes (b), Wine (c), Sonar (d), Wholesale (e) and Banknote (f). Fitness performance. Plots shows the median over 20 independent runs from the best-so-far solution.



(a)

Figure 5.6: Results of fitness performance of problem LSVT. Fitness performance. Plots shows the median over 20 independent runs from the best-so-far solution.

Figures 5.5(d-f) present the performance results for Sonar, Wholesale and Banknote while Figure 5.6 presents results for LSVT problem. For the Sonar problem the accuracy reached in the testing data is similar to the one obtained by standard GP. In the Banknote problem, the LSHS improvement is significant, where the classification error almost reaches zero.

Figure 5.7(a) discloses descriptive statistics from test fitness, at the end of each run, using notched box plots. The Wilcoxon rank sum test was calculated for the fitness test data, shown in first column of Table 5.3. The average population size is also presented in Figure 5.7(b), and its corresponding rank sum test is given in the second column of Table 5.3.

In both cases, the null hypothesis is that the underlying distributions in each pair of experiments have equal medians.

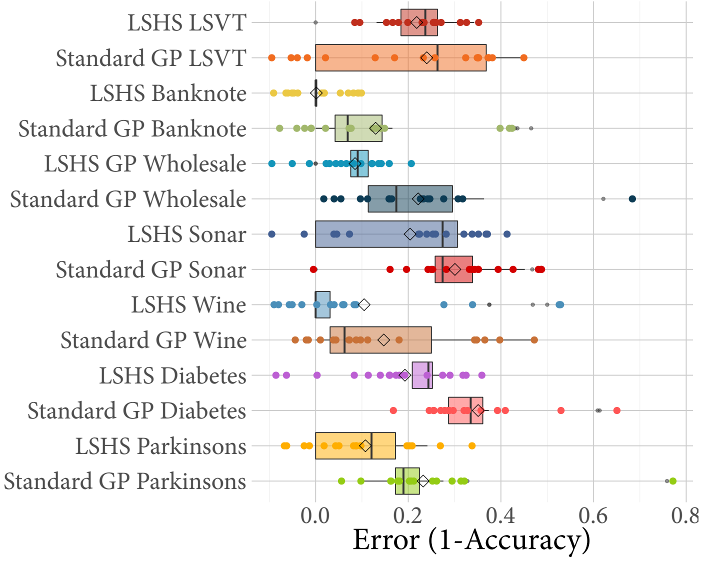
The statistical tests show that that the differences in performance exhibited by the standard GP and LSHS-GP are significant on four of the seven problems, particularly Parkinsons, Diabetes, Wholesale and Banknote, with $\alpha = 0.05$. Regarding size, statistical difference was detected on five problems, namely Parkinsons, Diabetes, Wine, Wholesale and

Banknote. It is interesting to note, that on all four problems where performance improved, size was also reduced. Here we can state that for these problems, the increased average size in standard GP does indeed represent bloat, since better solutions can be found while maintaining smaller populations. The Wine problem seems relatively easy for GP, the median performance of both algorithms is practically a perfect accuracy, but still GP evolves unnecessarily large populations, since the memetic GP with LSHS can produce equivalent performance with smaller populations in terms of average program size. The two outliers seem to be the Sonar and LSVT problems, two difficult problems on which no improvements in terms of size or fitness are obtained. It could be, that this is related to the large number of problem features in both cases, Sonar has 60 features and LSVT has 309, however confirming this hypothesis is left as future research.

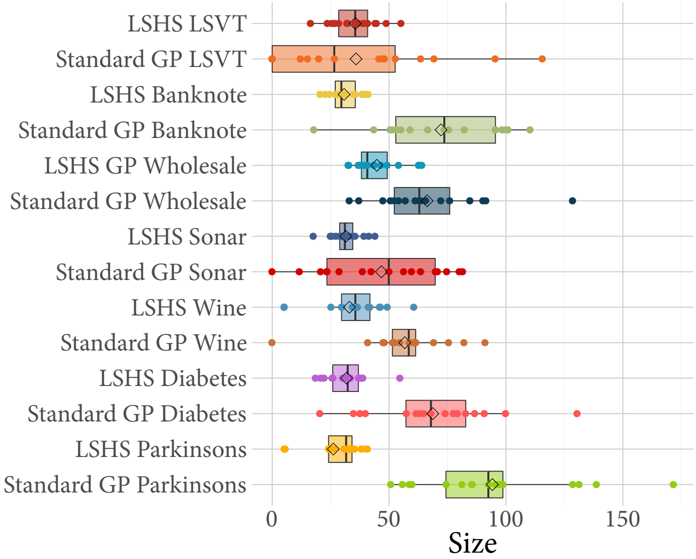
Table 5.4 presents an informal comparison with some recently published results for the seven problems tested here, using average classification accuracy. Thus providing a quick survey to contextualize the results presented above. The comparison is encouraging, the proposed LSHS method shows relatively strong performance on all problems, except on Parkinsons and LSVT. However, the best results on Parkinsons (Ma et al., 2014) uses a domain specific preprocessing stage, while other methods report similar results with our work. Similarly, (Tsanas et al., 2014) uses a state-of-the-art feature selection algorithm to reduce the dimensions of the problem given to the classifier, important for this dataset since it contains 309 features, while GP takes the entire feature vectors as input, a clear disadvantage.

5.4 CHAPTER CONCLUSIONS

In this study, a methodology to incorporate a LS method into GP was proposed, that tackles the challenges imposed by a supervised classification task. A tree parameterization is introduced where a candidate solution in GP is extended by a simple mechanism where a weight is added to each node. Using a well known and robust non-linear opti-



(a)



(b)

Figure 5.7: Box plots from 20 runs at the end of each run; where: (a) presents the test fitness of the best solution found during testing; and (b) the average population size.

Table 5.3: Wilcoxon rank sum test, with $\alpha = 0.05$. Bold values are less than α.

	Fitness	Size
Problem	**p-value**	
Parkinsons	**0.0027**	**0.0000**
Diabetes	**0.0000**	**0.0000**
Wine	0.0630	**0.0001**
Sonar	0.0859	0.1212
Wholesale	**0.0001**	**0.0017**
Banknote	**0.0000**	**0.0000**
LSVT	0.2978	0.7043

mizer, the solution is improved targeting a better classification rate. A heuristic is used to determine which solutions are improved by the LS method, preferring smaller than average trees within the population.

The presented results illustrate the benefits of using a LS method during the GP search. Indeed, in most problems the improvements in terms of classification error was significant, and a survey of recently published results confirm the quality of the evolved solutions. Moreover, the evolution of program growth also provides interesting insights. In particular, GP+LS with LSHS was able to produce high quality solutions, in most cases significantly better than standard GP, while maintaining smaller individuals within the population in terms of average size. This suggests that code growth is not necessary to obtain improved performance if the search space is properly exploited, which the LS method allows us to do. Moreover, it is important to notice that the proposed LSHS-GP system should not increase the complexity, or more precisely reduce the interpretability of the evolved solutions relative to standard GP. In fact, it might help produce simpler trees (results at least show that they are smaller), since the LS process allows simple numerical tuning of the evolved programs to improve fitness; something that otherwise might require complex syntactical changes. This contrast nicely with other recent approaches based on geometric semantic operators (Moraglio et al., 2012), that improve GP performance at the cost of lowering interpretability.

Table 5.4: Classification accuracy comparison among several state-of-art methods, including from the GP community.

Problem	Study	Accuracy (%)	Brief description of the study
Parkinsons	Ozcift and Gulten (Ozcift and Gulten, 2011)	87.7 (5-fold CV)	Dirichlet process mixtures
	Ozcift and Gulten (Ozcift and Gulten, 2011)	87.1 (10-fold CV)	CFS-RF
	Ma et al. (Ma et al., 2014)	**99.49** (10-fold CV)	SCFW-KELM with feature preprocessing
	Dufourq and Pillay (Dufourq and Pillay, 2013)	86.7 (testing)	GP based (arithmetic, logical and decision trees)
	Koshiyama et al. (Koshiyama et al., 2013)	89.33 (testing)	GP based (GPF-CLASS)
	Wang et al. (Wang et al., 2014)	86.96	GP based (MOGP maximizing ROCCH)
	This work	88.9 (testing)	GP + LS (LSHS)
Diabetes	Eggermont et al. (Eggermont et al., 2004)	74.1 (10-fold CV)	GP based (decision trees)
	This work	**75.5** (testing)	GP + LS (LSHS)
Wine	Tan and Dowe (Tan and Dowe, 2004)	93.2 (90/10 training/testing)	Minimum message length Decision Trees
	This work	**100** (testing)	GP + LS (LSHS)
Sonar	Tan and Dowe (Tan and Dowe, 2004)	76 (90/10 training/testing)	Minimum message length Decision Trees
	Koshiyama et al. (Koshiyama et al., 2013)	**77** (testing)	GP based (GPF-CLASS)
	This work	74.2 (testing)	GP + LS (LSHS)
Wholesale	Jayadeva (Jayadeva, 2015)	**92.72** (5-fold CV)	Minimal Complex Machine
	This work	91 (testing)	GP + LS (LSHS)
Banknote	Ghazvini et al. (Ghazvini et al., 2014)	95.99 (10-fold CV)	MLP
	This work	**100** (testing)	GP + LS (LSHS)
LSVT	Tsanas et al. (Tsanas et al., 2014)	**90**	SVM with feature subset
	This work	78 (testing)	GP + LS (LSHS)

In the last two chapters, we tried to provide a bird-eye view in terms of the current memetic GP approaches, together with our own analysis. Before, we stated that GP with a tree representation suffered with some limitations, being the search process one of them. Indeed, the standard search operators are blind to the effect to the individual outputs. Only after evaluation and further selection the search can be steered toward the objective. This produces a slow convergence, that normally is correlated with the underlying data characteristics. Therefore, LS strategies accelerate this convergence without affecting much the solution structure.

Since, conceptually, GP allows us to use it in a wide range of applications, the need for a proper representation is of vital importance. Identifying this representation is one of the several open questions that still exist (O'Neill et al., 2010). Additionally, appropriate search operators, types of population initialization, selection and replacement methods are as well important at the time of adapting GP to a new problem. For example, in our previous proposals, between the regression and the classification approaches, there was a slight variation in the individual representation along the necessary change in the fitness function. In the next chapter, we introduce more drastic modifications. The particular problem of feature extraction for EEG signals requires an overhaul in several GP elements. The canonical individual representation simply will not work. A new set of search operator will be needed as well; first, to fully take advantage of the solutions representation, and second, to make the search more efficient. Furthermore, the fitness function has to be adapted to the new objective. Details on the proposal will be further discussed across the next chapter.

Part III

Mental states classification

6 CLASSIFICATION OF MENTAL STATES BY MEANS OF GENETIC PROGRAMMING

6.1 INTRODUCTION

There are human activities that are considered to have a high risk because they require a response within a given time window with an associated cost. An incorrect reaction could pose different risks in terms of security, health or financial costs, among others. For example, in a surgery, a physician is expected to be fully alert, as is the case for a heavy-machinery operator; a technician in a manufacturing process, a prolonged state of alertness is desired when safety and productivity is the target. Some of the causes of unintentionally low states of vigilance are associated to sleep deprivation (Alhola and Plo-Kantola, 2007), monotonous tasks (Schmidt et al., 2007) or stress (Selye, 1965).

Human consciousness has two main components: wakefulness and awareness of the environment (Baars, 1988). Wakefulness is associated to the content of consciousness and awareness to the level of consciousness. These two, in the majority of situations, are heavily correlated with normal physiological states (with the exception of dream activity during REM-sleep)(Laureys, 2005). Derived from this, a custom model can be built where a high awareness/wakefulness represents an alert or normal state and a lower awareness represents a relaxed but awake state, which are the two conditions studied in this work.

To make use of this model, an approach would be to develop a computing system that would interact to a human and be able to identify each condition. A BCI system fits quite well in this kind of scenario.

In particular, different brain signal acquisition methods have been developed during the last decades, being the Electroencephalogram (EEG) the most used among the non-invasive approaches. It is worth mentioning that research has shown that BCI systems themselves can be affected when the alertness of the user is low (Myrden and Chau, 2015).

However, the task of distinguishing both states by means of EEG recordings is not trivial. These type of signals exhibit several complex characteristics: they are highly irregular, non-periodic, non-stationary, divergent between persons and distinctive among trials (Siuly et al., 2017).

In this work, we propose a new methodology for the automatic classification of two mental states, involving a novel algorithm for feature extraction based on the Genetic Programming (GP) paradigm. The study is dissected in two parts: first, a reference system is built based on our previously published work (Vézard et al., 2015); and second, an enhanced system is proposed with the objective of improving the already competitive results exhibited by the reference system. Both systems include these functional blocks: acquisition of signals, pre-processing, feature extraction and classification. However, this work makes the following contributions: first, the system includes a classifier independent feature extraction algorithm called $^{+}$FEGP; second, the algorithm avoids over-fitting during training using a regularized fitness function, which is unique in GP based search; third, unique components in the algorithm design are introduced to evolve multidimensional transformations of the data, extending a previous approach (Muñoz et al., 2015), with specialized search operators and dynamic parameters; fourth, this is the first time that a GP based algorithm is used in the classification of mental states.

6.1.1 *Related work*

Although the use of an Event-Related Potential (ERP) is a common practice in the development of BCI systems, is not always the case, and basically depends on the mental task that needs to be interpreted. ERPs are brain measurement captured after a cognitive, motor or sensory event has occurred. Some cognitive functions in humans can be detected after some milliseconds in response to a particular stimulus, which is where ERP based methods can be useful (Woodman, 2010). However, ERPs are not suitable for the goal of this work. Certainly, mental states might not be triggered by any particular stimulus, rather they are a prolonged condition in brain activity, activated by psychological reasons like mood change, or by physiological like exhaustion.

Literature states that the most promising EEG based systems have to contain at least the following components: signal pre-processing, feature extraction/selection and classification (Nicolas-Alonso and Gomez-Gil, 2012). Moreover, hybrid systems usually are the ones with the best performance, making it difficult to classify them into well defined groups (Amiri et al., 2013). Because literature related to EEG classification can be overwhelming, we are going to discuss here only the works that try to classify some kind of mental state rather than a particular mental task (e.g. motor-imagery).

The majority of the research in this area has focused on techniques from signal processing, supervised/unsupervised learning, statistical learning or a hybridization of these fields; either for feature selection, feature extraction or classification (Gupta et al., 2015). Particularly for EEG feature extraction in terms of mental states, several approaches have been followed. In (Trejo et al., 2015), a Kernel Partial Least Squares (KPLS) algorithm was used as a feature extraction step for alerted and fatigued states. Wavelet Transform (WT) coefficients and some measures of approximate coefficients, like Shannon entropy, are used in (Zarjam et al., 2015) for feature extraction, for a workload mental states classification problem. Garcés-Correa et al. (Garcés Correa et al., 2014) also use the WT for feature extraction in drowsiness detection. In these methods, coefficients from the WT decomposition are used as a spectral granularity measure of the signals, suitable as feature vectors. Saidatul and Sazali (A.Saidatul, Paulraj, 2013) used a modified Auto Regressive (AR) model by compensating forward and backward prediction errors, then using simpler statistics over the obtained eigenvectors, with very competitive results. Based on a valance/arousal framework, general mental states are identified by using the Power Spectral Density (PSD) in the work by Mallikarjun et al. (Mallikarjun et al., 2016), focusing on feature extraction rather than classification. Other common approaches to construct new features is the Common Spatial Pattern (CSP), Source Power co-Modulation (SPoC) and Spatio-Spectral Decomposition (SSD) which are presented in the work by Schultze-Kraft et al. (Schultze-Kraft et al., 2016). Deep Learning (DL) based methodologies have been recently used successfully in this type of problems. For example, the work by Hajinoroozi et al. (Hajinoroozi et al., 2016) present different variants of deep neural networks (DNN), like Deep Belief Network (DBN), Convolutional Neural Networks (CNN) and some specific channel-wise CNNs (CCNN). By stripping the label layer, the DNNs are effectively used as feature extractor elements.

In terms of feature selection, the number of works is rather small. For example, in (A.Saidatul, Paulraj, 2013) authors use Principal Component Analysis (PCA) to extract a feature vector subset. This is opposed to what happens in mental task research, where there is a high occurrence on feature selection methods, those being filter or wrapper based (Gupta et al., 2015).

Regarding the classifier, the community tends to use a wide range of tools from the machine learning field: Linear Discriminant Analysis (LDA) (Schultze-Kraft et al., 2016)(Vézard et al., 2015), k-Nearest Neighbors (k-NN) (A.Saidatul, Paulraj, 2013)(Guo et al., 2011), Support Vector Machine (SVM) (Gupta et al., 2015)(Shen et al., 2008), Artificial Neural Network (ANN) (Zarjam et al., 2015) and bagging-based methods (Hajinoroozi et al., 2016), among others.

According with the surveyed literature, the majority of works use classical algorithms to solve the problem of mental states recognition, however alternatives like meta-heuristic methods seem to help when the complexity of the problem is high. From an optimization perspective, algorithms from the Evolutionary Computing (EC) field have been employed as channel reduction or feature selection approaches. Genetic Algorithms (GA) have been found to be suitable, mostly in wrapper approaches (Vézard et al., 2013)(Fang et al., 2013)(Rezaee, 2016)(Hongxia et al., 2016). Other algorithms like Ant Colony Optimization (ACO)

(Erguzel et al., 2014) or Particle Swarm Optimization (PSO) (Lin and Hsieh, 2009) have also been used. Adaptive approaches have been employed as well like the auto-reinforced system introduced in (Hassani and Lee, 2014), by incorporating a feed-backed PSO and a rule-based classifier.

GP is a particular variant of EC algorithms, with the unique characteristic that solutions are concatenated computing elements. An inherent strength of the GP paradigm is its symbolic representation that can be adapted to a wide range of problems. For example, GP can be used either for feature selection, feature extraction or classification. In (Bhardwaj et al., 2014), a GP multi-tree classifier is used in combination with Empirical Mode Decomposition (EMD) in an EEG dataset for epilepsy detection. A continuation of this work with modified search operators is presented in (Bhardwaj et al., 2016). A decision-tree based model is presented in (Fernández-Blanco et al., 2013), where arithmetic and logical rules are inferred in a multi-level GP representation. Although Electrocorticogram (ECoG) was used instead of EEG by the authors in (Sotelo et al., 2013), GP was employed as a classifier to solve an epileptic seizure recognition problem.

More closely related to this work, Guo et al. (Guo et al., 2011) built a feature extraction model using GP for an epilepsy EEG dataset. In (Firpi et al., 2006), GP is used similarly, this being a feature extraction task. These cited works are further discussed in Subsection 6.3.1, since there are some differences worth reviewing compared with our proposal. Apart from feature extraction methods, GP can be used as a simple feature selection tool like in (Sabeti et al., 2009). Although some work has been done in EEG classification using GP, to the authors's knowledge this is the first time it is used to specifically tackle the mental state recognition problem.

The chapter is organized as follows. In Section 6.2 the reference system based on CSP and LDA is described and analyzed. In Section 6.3 we discuss in detail the proposed system, in particular the GP based feature extraction method called $^{+}$FEGP. Experimentation and results are presented in Section 6.4. In Section 6.5 results are discussed. Finally, in Section 6.6 we present our conclusions.

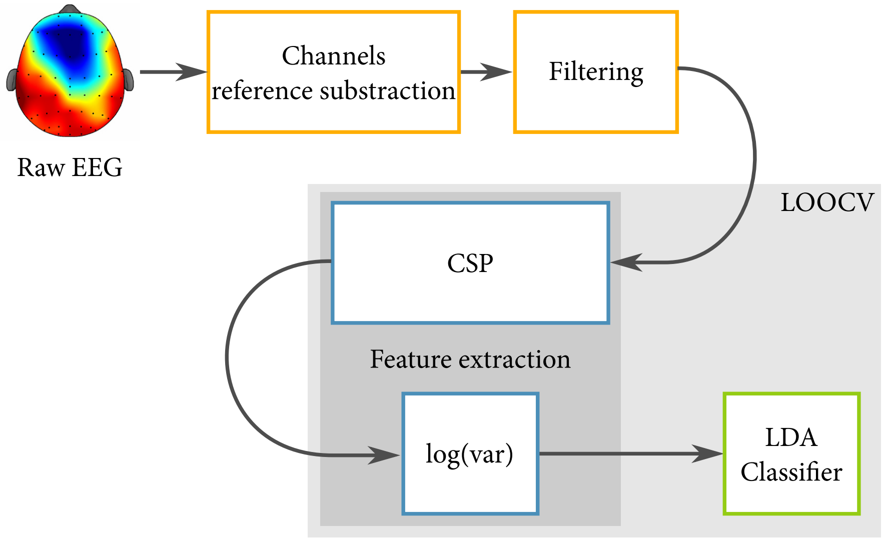


Figure 6.1: Methodology of the reference system

6.2 REFERENCE SYSTEM

The reference system is basically the first part of our previous work (Vézard et al., 2015). Particularly, a complete classification system is built based on several stages. First, a data acquisition protocol was followed; second, a preprocessing step involving spectral filtering; third, a feature extractor based on the CSP is used; and fourth, a classification step using LDA. This reference system is described in Fig. 6.1. In the following subsections, each of these elements are explained in detail.

6.2.1 *Data acquisition*

6.2.1.1 *Acquisition protocol*

A selected group of individuals (critical personal information was protected) were chosen to collaborate in the experiments during a 2011 campaign at the Université de Bordeaux, France. Each of the involved persons were subjected to the following procedure to build the dataset:

First, a cap was placed over the scalp using 58 electrodes in positions following the international standard system 10/10 (Jurcak et al., 2007), as shown in Fig. 6.2. This procedure lasted approximately 30-40 mins, since the electrodes are wet based, illustrated in Fig. 6.3. The used recording system was a Deltamed Coherence 3NT, by Natus[1].

Second, the participants were seated alone in a soundproof room, in front of a screen where different information was fed directly trough visual cues.

Third, the first EEG recording started at this point. The individual had to look at a fixed cross at the center of the screen, avoiding eye movements as much as possible. This recording corresponds to the alert, basal or normal state.

Fourth, a Contingent Negative Variation (CNV) protocol (Walter et al., 1964) was used, depicted in Fig. 6.4. The CNV is a special protocol where specific steps are followed by using ERPs. The person is instructed to press the space bar as quickly as possible each time the cross is replaced by a square on the screen. For each appearance of the square, a warning sound (beep) is given 2.5s before it appears, allowing the person to prepare her/his response. The experimental session included 50 pairs of stimuli (S1: beep, S2: square), with a random amount of time elapsing between each pair. Depending on the experiment's requirements, number of repetitions or trial duration could vary. An average signal of the superimposition of each trial is extracted, synced at the stimuli triggering starting time. For our experiment, although the subject was instructed to react as fast as possible to the visual cue, the reaction time is not of interest, rather what happens after the audio cue, that is the prolonged mental state. The protocol is designed this way to avoid self-training by the subjects to the studied state. In the CNV test, the amplitude of the resulting signal is directly associated to the relaxation level of the subject (Naitoh et al., 1971), as illustrated in Fig. 6.5.

Fifth, a relaxation session was performed in order to bring the subject to a lower vigilance state. The subject was guided by a soundtrack broadcast through speakers placed in the room. The soundtrack sug-

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

gested the subject to perform three successive exercises of self-relaxation, based on muscular exercises and mental visualization. The first exercise is autogenic training (Carruthers, 1979) where the subject has to mentally repeat some sentences favoring self-hypnosis. The second exercise is a progressive relaxation (Elton et al., 1978), which consists on flexing and unflexing some muscles. The final exercise is mental visualization, where the subject imagines that she/he is moving in a familiar and lovely place.

Sixth, the subsequent EEG recording is obtained afterwards, which corresponds to the relaxed state. Approximately 3 mins of recording was captured at this step, fairly matching the normal state recording length.

Finally, a second CNV collection was gathered, using the same protocol as before.

Notice that a pair of CNV tests was performed during the data acquisition procedure. This dual measurement allows us to properly assess the validation of the actual subjects brain state compared with the expected state. Differences in amplitudes from the first and the second CNV test should display a presence of an induced relaxation state or its absence. If this difference exist then both EEG recordings are considered valid and are tagged as "normal" and "relaxed" states, respectively. To do this, a visual and numerical inspection of superimposed CNV tests was performed, exemplified in Fig. 6.6 at the interval between T1 (360 ms after the stimulus S1) and T2 (900 ms).

6.2.1.2 *Subjects*

The experiment involved 44 subjects (mixed gender), non-smoking, aged from 18 to 35 years old. They are right-handed, to avoid variations in the characteristics of the EEG due handedness linked to functional inter-hemispheric asymmetry. After the CNV test, several subjects were rejected since they were not in compliance with the expected state. Therefore, only 13 valid individuals were used to build the dataset.

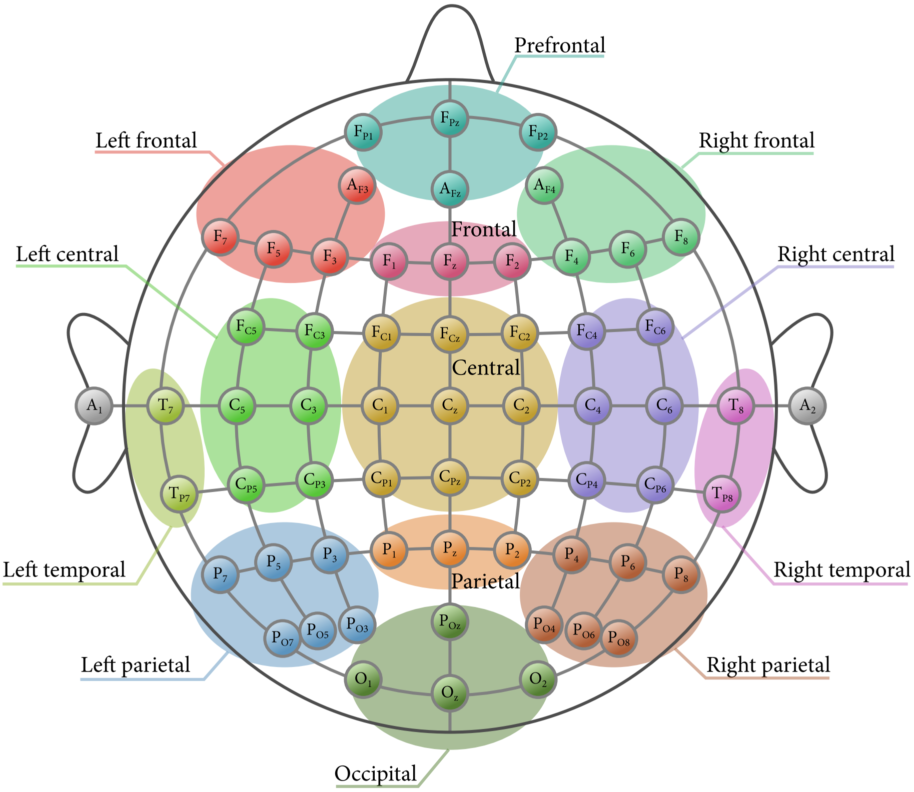


Figure 6.2: Relative position of the electrodes based on the international standard system 10/10.

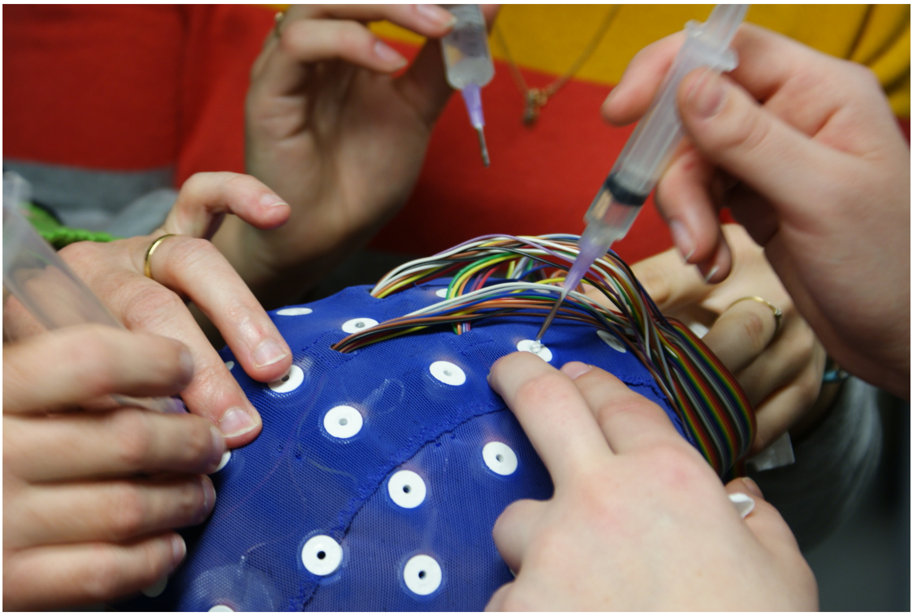


Figure 6.3: Photograph that illustrates the cap positioning procedure. Conductive gel is used to reduce the electrical impedance between the scalp and the electrode.

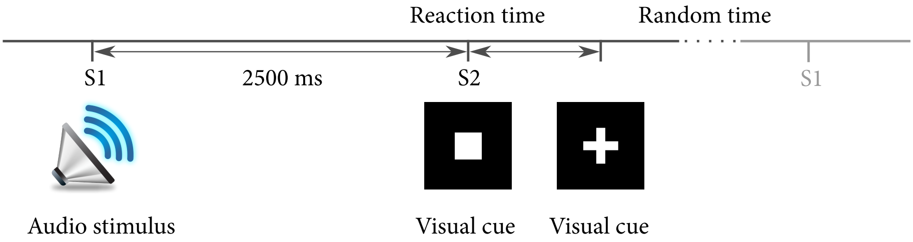


Figure 6.4: Acquisition protocol of the CNV, describing the stimuli presented to each subject. Each task consists of 50 pairs of stimuli.

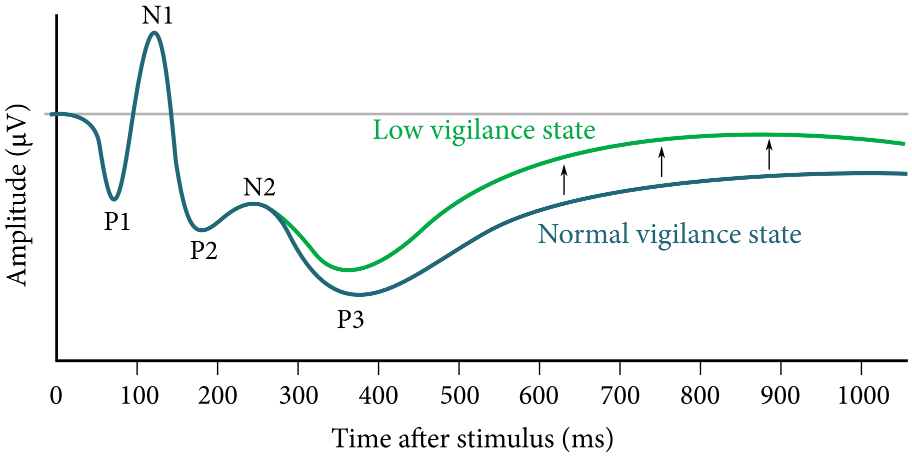


Figure 6.5: Relationship of the vigilance level and the amplitude on the CNV test. Main ERP components are shown here: P1, N1, P2, N2 and P3.

6.2.1.3 *Raw data*

The recorded data, considered as the raw data in this study, contains 26 records of approximately 3 mins each (13 corresponding to the normal state and 13 more for the relaxed state) of 58 channels for each subject. A sampling frequency of 256 Hz was employed during the acquisition by the Deltamed system. Since the length of the recordings slightly varies from subject to subject, approximately 46,000 samples were obtained for each recording.

6.2.2 *Pre-processing*

For this type of systems, the automatic detection should be accomplished by just analyzing a small portion of a signal and classifying it as quickly as possible, particularly in an online scenario. By splitting the signal in small packets of data (commonly referred as trials), training and testing a supervised learning system is possible.

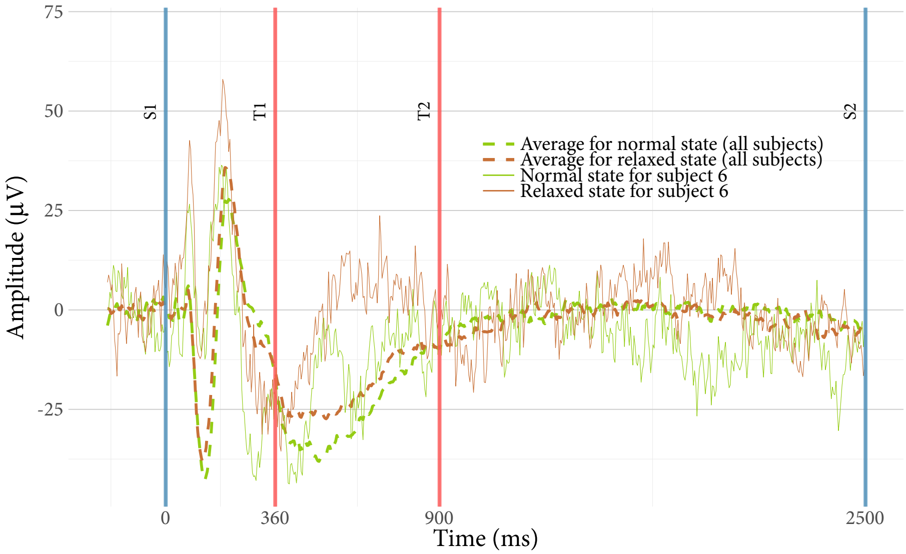


Figure 6.6: CNV obtained for subject 6. Validation interval is achieved between T1 and T2. S1 and S2 belong to the corresponding stimulus triggering time, as shown in Fig. 6.4. In this example, the EEG recordings are considered valid since the expected relaxed state has lower amplitude than the normal state.

However, the length of a trial cannot be predefined a priori, thus a quick analysis to determine an appropriate size for the trials is required. In (Vézard et al., 2015), different trial lengths were checked and the optimal one was found based on the classification accuracy. Lengths of 1024, 2048 and 4096 samples were analyzed and a value of 2048 samples (8 seconds) was found to be the best, resulting in $\approx$ 22 trials per subject/class. Consequently, this value was also used in this work. Therefore, our dataset is stored in a matrix $\mathbf{X}_{58\times(26\times\approx 22)\times 2048}$.

The recorded EEG signals usually contain noise and unrelated frequencies to the task at hand. The origin of these unrelated frequencies are partly due the oscillatory rhythms present during brain activity (Pfurtscheller et al., 1997). The most studied rhythms are alpha (8-12 HZ), beta (19-26 Hz), gamma (1-3.5 Hz) and theta (4-8 Hz), which are associated to different psycho-physiological states. The alpha waves are characteristic of a diffuse awake state for healthy persons and can be used to discern between the normal and relaxed states. Actually, in some recordings alpha waves start to appear when the subject is starting to relax.

In this work, band-pass filtering is applied to discriminate frequencies outside the alpha and beta bands. Again, an analysis is required to find out which cutoff frequencies are useful to build the filter, by sweeping a range of candidate values. In (Vézard et al., 2015) we present such analysis, with the resulting values of 7 Hz and 30 Hz corresponding to the low and high cutoff frequencies using a 5th order Butterworth filter; matching what other researchers have found as well(Ramoser et al., 2000; Blankertz et al., 2008). The filter response is illustrated in Fig. 6.7.

6.2.3 *Common Spatial Patterns*

Like other techniques that derive a projection or transformation matrix based on some specific requirements (e.g Independent Component Analysis (ICA) or PCA), the CSP is a technique where a matrix **W** is built, that simultaneously maximizes the variance of a multivariate subset with an arbitrary label and minimizes the variance for another subset with dif-

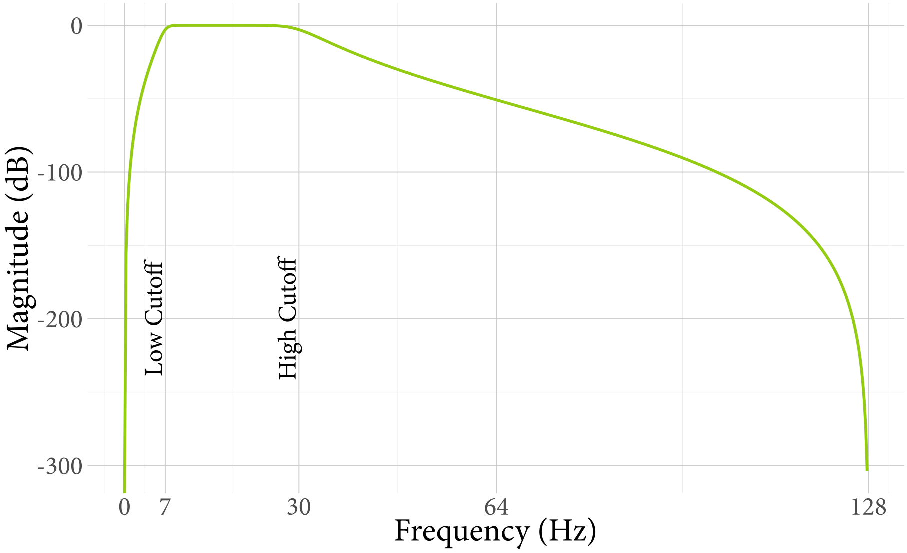


Figure 6.7: Frequency response for the Butterworth filter used in this work.

ferent label, belonging to the same data-set. Projected data is given by $\mathbf{F} = \mathbf{XW}$, where $\mathbf{X}$ is the original data with dimensions $p \times n \times T$, being p the number of channels, n the number of trials and T the trials length. This technique is useful in a binary classification problem because data is projected to a space where both classes are optimally separated in terms of their variance.

The CSP can be briefly defined as the following optimization problem:

$$\mathbf{w} = \underset{\mathbf{w}}{argmax} \frac{||\mathbf{wX}_{C1}||^2}{||\mathbf{wX}_{C2}||^2} \tag{6.1}$$

and

$$\mathbf{v} = \underset{\mathbf{v}}{argmax} \frac{||\mathbf{vX}_{C2}||^2}{||\mathbf{vX}_{C1}||^2}, \tag{6.2}$$

where $\mathbf{X}_{C1}$ is the matrix that contains the trials for class one and $\mathbf{X}_{C2}$ the matrix for class two; respectively the normal and relaxed conditions in our case. Coefficients found in $\mathbf{w}$ and $\mathbf{v}$ define a set of filters that project the covariances of each class orthogonally.

This problem can be solved by a single eigen-decomposition of $\mathbf{M} = \Sigma_{C2}^{-1}\Sigma_{C1}$, where

$$\Sigma_{C1} = \frac{1}{n_{C1}}\mathbf{X}'_{C1}\mathbf{X}_{C1} \tag{6.3}$$

is the estimation of the covariance corresponding to the average of n_{C1} trials of class one. Similarly,

$$\Sigma_{C2} = \frac{1}{n_{C2}}\mathbf{X}'_{C2}\mathbf{X}_{C2} \tag{6.4}$$

is the homologous for class two. In a single eigen-decomposition, the first k eigenvectors (corresponding to the k largest eigenvalues) of $\mathbf{M}$ are also the last k eigenvectors of $\Sigma_{C1}^{-1}\Sigma_{C2}$. Sorting the eigenvalues in descending order we can build our filter matrix with

$$\mathbf{W} = [\mathbf{w}_1,\dots,\mathbf{w}_k,\mathbf{v}_1,\dots,\mathbf{v}_k], \tag{6.5}$$

in which filters are considered pairwise $(\mathbf{w}_j,\mathbf{v}_j)$ for $j = 1,\dots,k$. The number of filter pairs can greatly affect the classification accuracy, thus a tuning step is needed to find the optimal number of filters. This step was performed in (Vézard et al., 2015) resulting in a value of $k = 3$, which we employed as well in this work.

6.2.4 *Classification*

To approximate a normal distribution of the data, a logarithm calculation is performed over the variance obtained from the input data transformation by $\mathbf{XW}$. This is stored in a matrix $\mathbf{Z}$, and is given by

$$\begin{gathered}\mathbf{Z} = log(var(\mathbf{F})) = \\ \begin{pmatrix} log(var(\mathbf{X}_1\mathbf{W}_1)) & \cdots & log(var(\mathbf{X}_1\mathbf{W}_j)) \\ \vdots & \ddots & \vdots \\ log(var(\mathbf{X}_i\mathbf{W}_1)) & \cdots & log(var(\mathbf{X}_i\mathbf{W}_j)) \end{pmatrix}_{n\times 2k} .\end{gathered} \tag{6.6}$$

The evaluation of the classification process is performed using the Leave One Out Cross-Validation (LOOCV) method. In this methodology, data is split into q folds ($\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_q]$) where the training subset has a size of $q - 1$ and the testing has a size of one. In our case $q = 13$, the number of subjects. The process is repeated q times by changing the index m for testing and keeping the remaining folds for training. The reason for using this type of partitioning is because the recognition is performed at the individual level (grouping trials from the same individual) rather than at the trial level. We want to avoid to train and test the classifier using trials from the same subject, which could lead to misleading results.

The classifier used is LDA. This technique is commonly used in BCI systems, due to its ease of implementation and competitive performance compared with more sophisticated algorithms (Duda et al., 2000). Briefly, in LDA a hyperplane is calculated based on the covariances of the data distribution that optimally separates two classes. If we assume that both classes belong to a normal distribution and have the same covariance, then we can calculate

$$w = \Sigma^{-1}(\mu_{C1} - \mu_{C2}) \tag{6.7}$$

and

$$b = w^T \frac{\mu_{C1} + \mu_{C2}}{2}, \tag{6.8}$$

where Σ is the shared covariance matrix for both classes, while μ_{C1} and μ_{C2} are the corresponding means for each class. Classification is performed by assigning a label (either C1 or C2) to a label vector $\hat{y}$ depending on a score vector s, given by

$$s = b + w\mathbf{Z}, \tag{6.9}$$

and

$$\hat{y} = \begin{cases} C1 & \text{if } s \geq 0 \\ C2 & \text{otherwise} \end{cases}. \tag{6.10}$$

Until this point, we employed the training data to obtain the CSP filter set $\mathbf{W}$ and the vector $\hat{y}$. During testing phase, a prediction vector $\hat{y}$ is

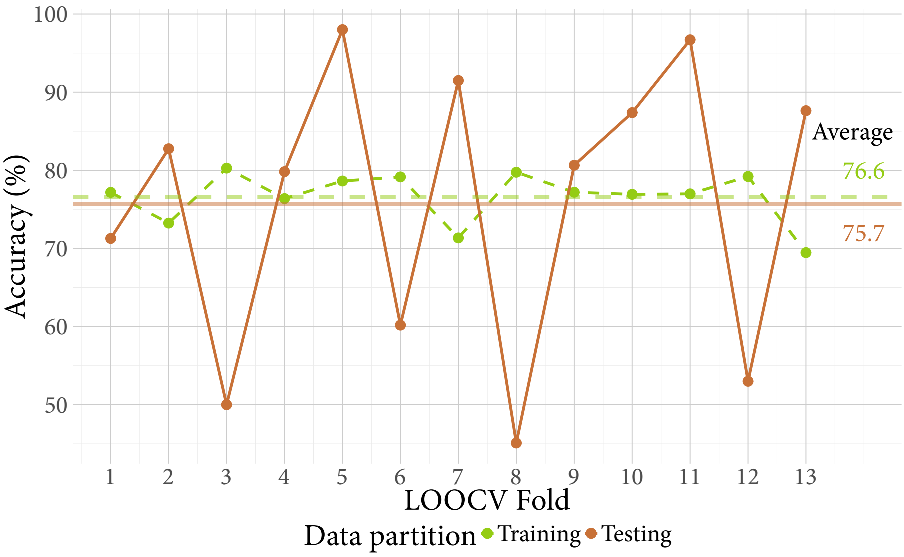


Figure 6.8: Classification performance for the reference system.

computed using the hyperplane found during training but using a projected testing data based on the pre-calculated **W**.

Since this is calculated for a given fold in the LOOCV strategy, after q repetitions, an average of the accuracy results of all folds is computed and is considered as the prediction accuracy. Results from the classification performance can be seen in Fig. 6.8, where the average and fold wise accuracy are presented corresponding to the training and testing partitions. For training, an average accuracy of 76.6% was obtained, and 75.7% for testing. Please note that over-fitting exists (or a failure to generalize) for some folds, namely 3, 6, 8 and 12.

6.3 PROPOSED ENHANCED SYSTEM

In this section we present a novel algorithm based on GP that extends the reference system to improve the classification results, particularly for the prediction over unseen data. From a general point of view, the enhanced system is described in Fig. 6.9, where it can be seen that the

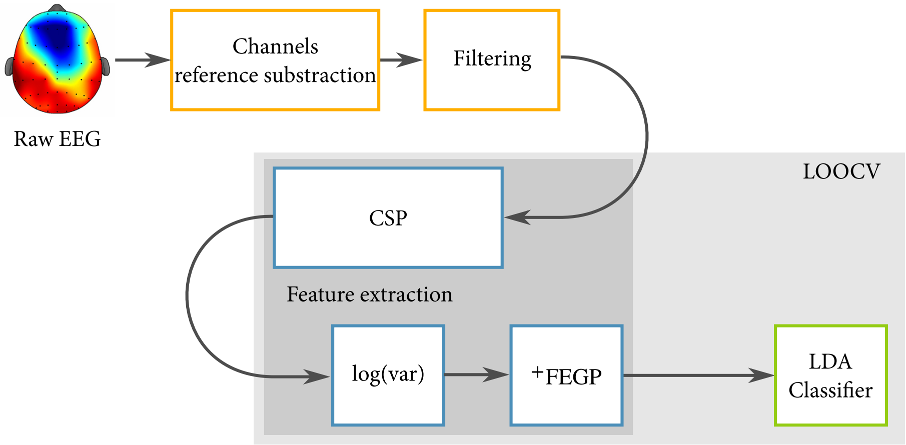


Figure 6.9: Methodology for the proposed enhanced system.

proposed Feature Extraction with GP (${}^{+}$FEGP) follows the CSP block. In other words, we perform a post-transformation of the CSP output, with the goal of improving the classification results.

6.3.1 *Genetic Programming*

Part of the broader Evolutionary Algorithms (EA) research area, GP (Koza, 1992) is sometimes considered to be a generalization of the popular GAs (Goldberg, 1989). In GP, a pool of candidate solutions (population) are syntactically evolved towards a specific objective by applying selection pressure based on a quality assessment or fitness. GP differentiates from other population based algorithms on the following fronts. First, the solutions or individuals are expressed with a symbolic structure. This is mainly where its strength comes from, since it can represent many things, depending on the problem's domain. The only restriction is that the expression must be computable, that is to say for a given input it produces a valid output. Second, the size of the solution is variable, meaning that during the search their length or size changes. This provides GP its second strength, solution structures are not predefined, rather they are evolved simultaneously with their performance. A general diagram for GP is presented in Fig. 6.10.

GP has been used successfully in many applications with different representations (Koza, 2010), whereas the tree representation is the most common among them. In this representation, building blocks are attached following a hierarchical pattern, where the leaves are normally inputs such as variables, features or constants (terminal set), and the internal nodes are computing elements (function set). The tree evaluation is calculated in a reverse order, from leaves to the root node, being the latter where the output is produced.

Although the canonical GP achieves competitive performance compared with other algorithms (Koza, 2010)(Sotelo et al., 2013), including hand-made designs, it is not suitable for all types of problems, specially when the underlying data has complex properties. Practically, GP can suffer from some shortcomings, also common to other machine learning algorithms, like the stagnation of the search in a local optima(Dauphin et al., 2014) or model over-fitting in under-sampled data (Hastie et al., 2009). For these reasons, a new algorithm is proposed based on an extension of our previous works (Muñoz et al., 2015)(Naredo et al., 2016).

6.3.2 *Augmented Feature Extraction with Genetic Programming: $^{+}$FEGP*

Recognized as one of the most important tasks in a classification system, our main proposal is focused on the improvement of the feature extraction stage. One of the discovered complexities found during the implementation of the reference system is that the data of the study is highly varied and uncorrelated within the same class, and overlapped between both classes. This yields a non-trivial problem for any supervised learning system. A suitable strategy would be a non-linear model that constructs new features which further increases the inter-class distance and reduces the intra-class spread.

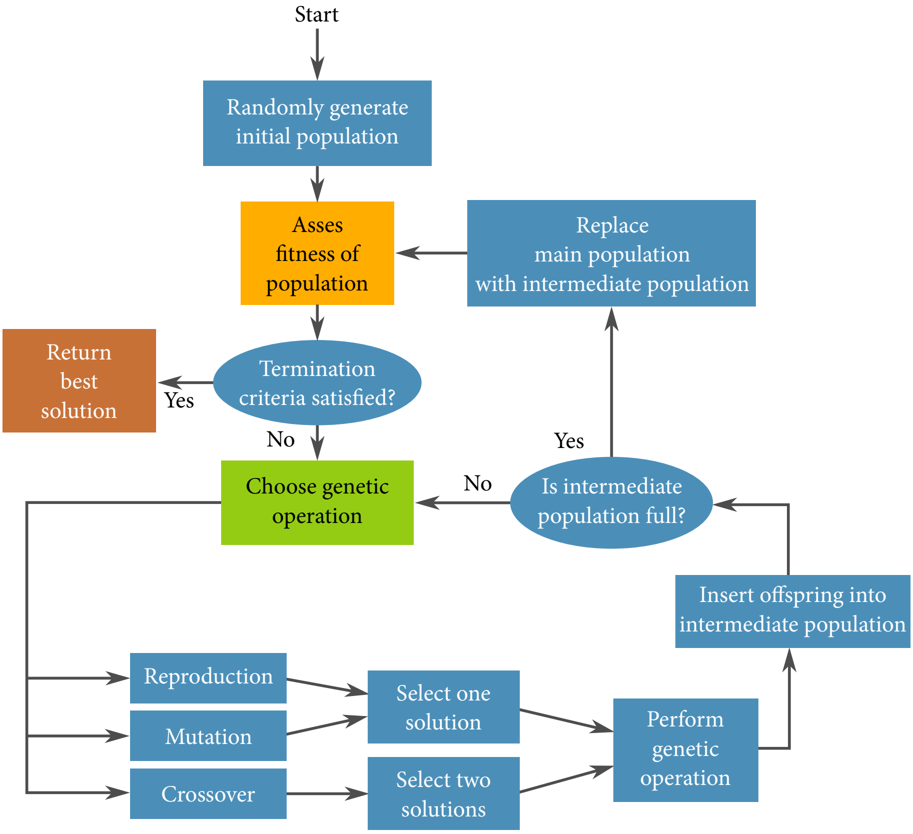

Start
Randomly generate initial population
Asses fitness of population
Replace main population with intermediate population
Return best solution
Yes
Termination criteria satisfied?
No
Yes
Choose genetic operation
No
Is intermediate population full?
Insert offspring into intermediate population
Reproduction
Mutation
Crossover
Select one solution
Select two solutions
Perform genetic operation

Figure 6.10: GP flowchart in a nutshell.

Based on this, a GP-based algorithm is proposed with the objective of improving the classification accuracy for the reference system. Another goal for the proposal is to be highly flexible, not explicitly depending on any specific pre-processing or classifier blocks. A block diagram for this algorithm is depicted in Fig. 6.11, hereafter referred to as $^{+}$FEGP, for augmented feature extraction with GP.

Similar research has been conducted in terms of feature extraction with GP. Guo et al. (Guo et al., 2011) propose first an individual representation with a single root node, where a special node called 'F' can be inserted in any part of the tree, which produces a new feature. Besides this enhancement, no special search operators were used for their system. On the other hand, Firpi, et al. (Firpi et al., 2006) used a canonical GP individual representation, where a single tree produces a new feature, a rather simple proposal that nevertheless produced competitive results on the classification of signals from epileptic seizures. A similar approach was followed by Poli et al. in (Poli et al., 2011), where a single root node representation with a custom fitness function was used to evolve mouse trajectories by employing EEG signals. The authors used a strongly-typed GP, a variant in which there is an explicit constraint on data type allowed in the terminals and functions. In a previous work (Sotelo et al., 2013), we used GP as a classifier for the detection of epileptic stages, taking as input simple statistics of each epoch.

The present proposal builds upon our previous works in (Muñoz et al., 2015)(Naredo et al., 2016), where the M3GP algorithm is presented. M3GP also uses some additional search operators, to add or remove complete features during the run, but mostly relies on a canonical GP implementation with a single objective function that only takes in account the training accuracy and a static parametrization of the search. On the other hand $^{+}$FEGP includes a wider variety of search operators, a specialized local fitness measure to account for the fact that each solution is a collection of subtrees, dynamic parameters that control how the search is performed at different moments during the run, a regularized fitness function to control over-fitting and achieve better generalization, and a specialized initialization procedure to account for the extremely

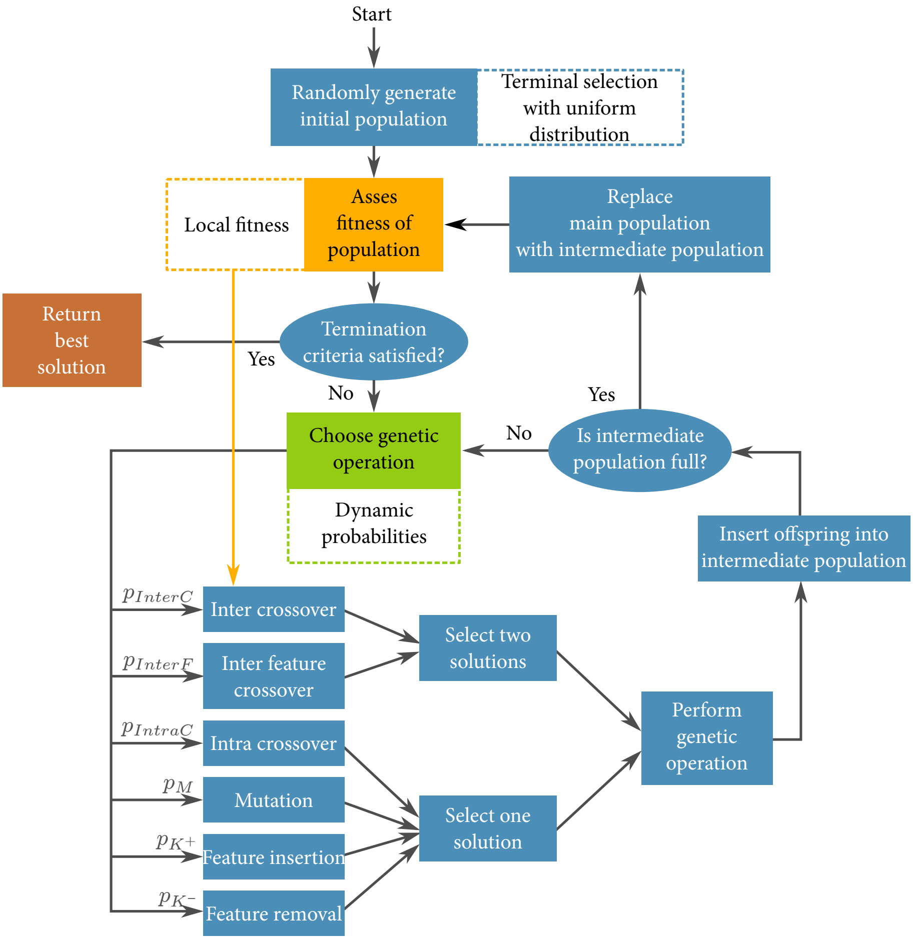

Start
Randomly generate initial population
Terminal selection with uniform distribution
Local fitness
Asses fitness of population
Replace main population with intermediate population
Return best solution
Yes
Termination criteria satisfied?
No
Yes
Choose genetic operation
No
Is intermediate population full?
Dynamic probabilities
Insert offspring into intermediate population
p_{InterC}
Inter crossover
p_{InterF}
Inter feature crossover
Select two solutions
p_{IntraC}
Intra crossover
Perform genetic operation
p_M
Mutation
Select one solution
p_{K^+}
Feature insertion
p_{K^-}
Feature removal

Figure 6.11: Block diagram for the proposed $^{+}$FEGP.

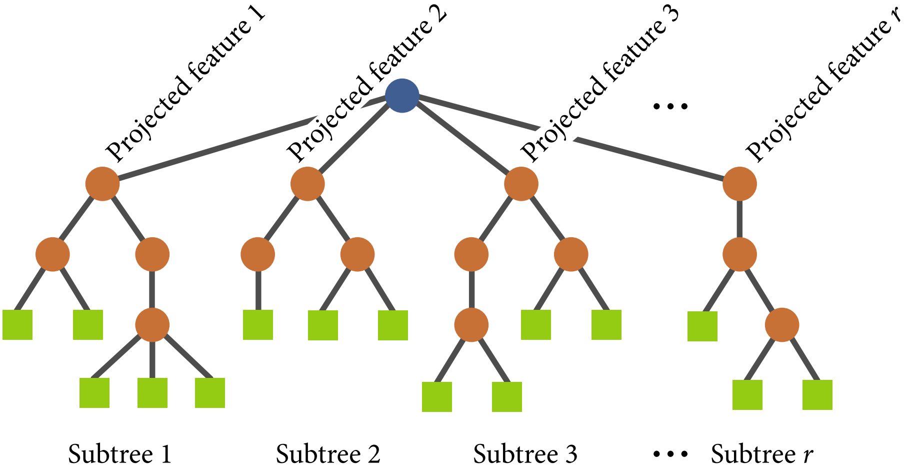


Figure 6.12: $^{+}$FEGP individual representation.

large feature space in this problem domain. Each of these features will now be explained in greater detail.

1. **Individual representation.** The representation is also based on a tree structure, but it involves a multi-tree with a single root node, such that each individual defines a mapping of the form $K : \mathbb{R}^k \rightarrow \mathbb{R}^r$, being k the number of input features and r the number of newly constructed features. Each of these subtrees are constructed in the same way as in a canonical GP (see Fig. 6.12). No calculation is executed at the root node, it is rather used as a container of the evolving r subtrees. Please note that each subtree works as a non-linear model that transforms input features into a new space that is expected to simplify the classification. Moreover, note that r can vary among individuals.

2. **New genetic operators**. Because of the manner in which individuals are represented, it is necessary to introduce new search operators. For any of these operators the root node is avoided during the operation to preserve the representation.

(a) Inter-crossover. This is a crossover performed between subtrees belonging to different individuals and produces two offsprings. The selection for the crossover points are based on a local fitness measure (later discussed in detail), which allows the best subtrees to interchange genetic material. This can be visualized in Fig. 6.13.

(b) Intra-crossover. For a given individual, genetic material can be swapped within the same individual using this operator, depicted in Fig. 6.14. The rationale behind this is that genetic material from one feature might help the evolution of another within the same individual, improving the overall fitness. The subtree selection is performed randomly, choosing the crossover points from different subtrees. This operator generates two offsprings.

(c) Inter-individual feature crossover. It can be desirable to keep complete features with good performance. This operator interchanges complete subtrees between two selected individuals. This is performed randomly rather than using a deterministic approach. Preliminary experimentation suggested that not using a fitness-based selection of the subtree allowed the algorithm to explore the solution space better. This operator can be seen in Fig. 6.15.

(d) Subtree mutation. This is performed the same way as in canonical GP. A portion of a selected subtree is substituted with a randomly generated tree.

(e) Feature insertion mutation. This operator allows the insertion of a new randomly generated subtree expanding the dimensionality of the transformed feature space; Fig. 6.16 depicts this operator.

(f) Feature removal mutation. Similarly, a deletion operator is needed to reduce the projected feature space. The subtree to be removed is selected randomly. This operator can only be used if $r > 1$.

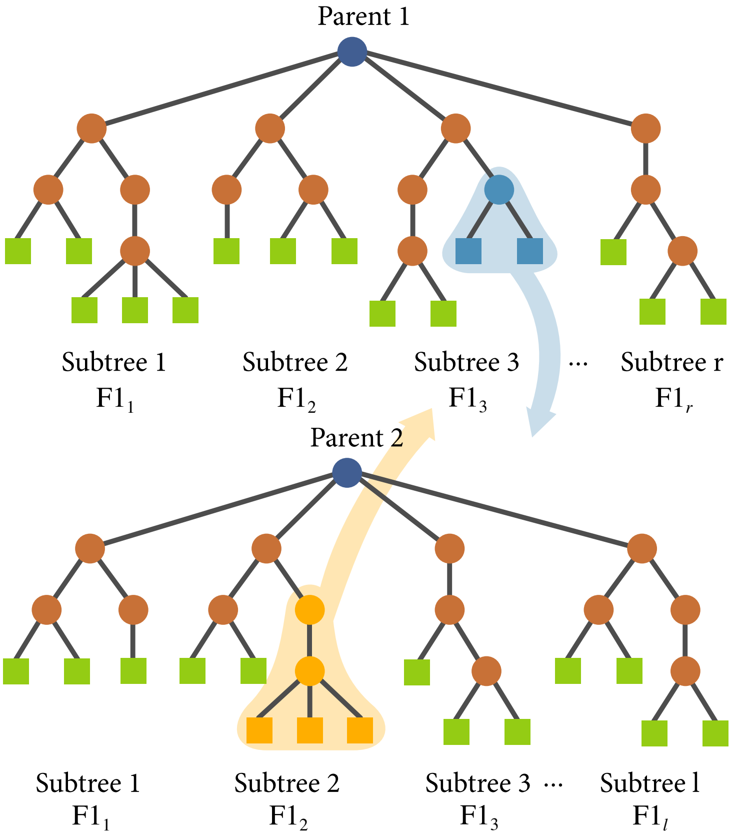


Figure 6.13: $^+$FEGP inter-crossover, swapping subtrees between different individuals. F1$_i$ is the local fitness; the crossover points are chosen from the subtrees with max(F1).

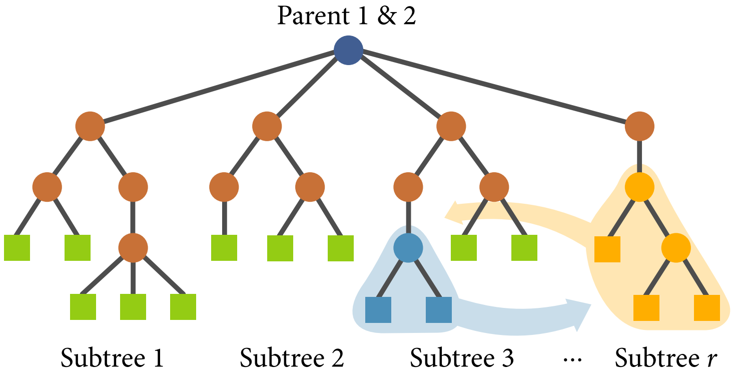


Figure 6.14: $^+$FEGP intra-crossover, swapping subtrees within the same individual.

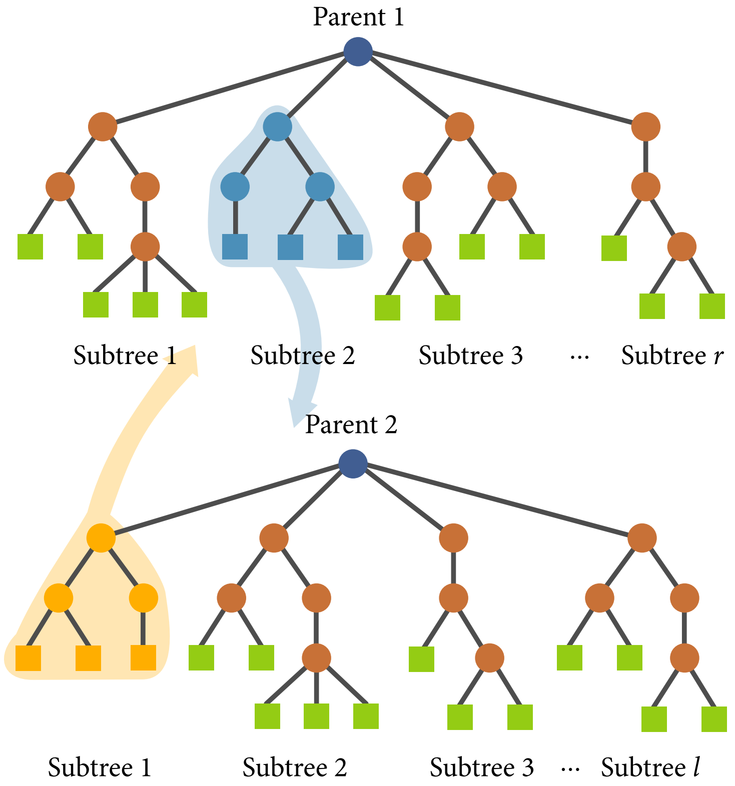


Figure 6.15: $^+$FEGP inter-individual feature crossover.

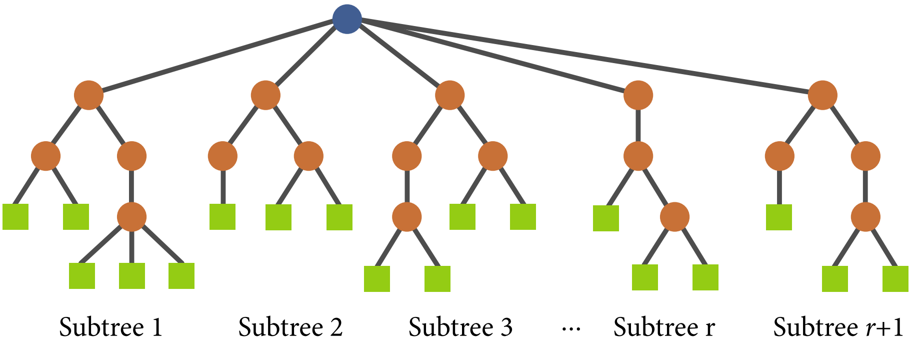


Figure 6.16: $^{+}$FEGP feature insertion mutation.

3. **Terminal selection during population initialization.** To generate the initial population, trees are built by randomly picking functions and terminals when needed. However, when the data set contains many input variables there is a high probability that some variables are less frequently selected or may not be used at all. This greatly reduces the exploration in syntax space, specially during the initial generations. Mutated trees follow the same mechanism, suffering from the same shortcomings. To avoid this, a new terminal selection method is introduced by employing a sampling without replacement procedure. When a terminal is randomly selected then it is excluded on the subsequent terminal selection step. Once all available terminals have been chosen, then the procedure is repeated by making all terminals available. This produces a uniform distribution in the selection of terminals, defined by $\boldsymbol{freq\_ter}$, which can be seen in detail in Alg. 6.3.1. The algorithm's output is a chosen terminal T from the terminal set $\boldsymbol{T}$.

4. **Genetic operator selection.** In canonical GP, operators are chosen at each generation with an user-defined probability. The probability of choosing each operator is usually static throughout the search. In the work by Tuson and Ross (Tuson and Ross, 1998), their operator probabilities are changed at each generation by rewarding or penalizing operators according with their success his-

Algorithm 6.3.1 Terminal selection. $\boldsymbol{freq\_ter}$ is a global array during the tree construction process.

```
Input: Terminal set ($\boldsymbol{T}$),Terminal frequencies ($\boldsymbol{freq\_ter} = \{0\}$)
Output: $T$
$b \leftarrow$ minimum element in $\boldsymbol{freq\_ter}$
$\boldsymbol{index2fill} \leftarrow$ indexes from $\boldsymbol{freq\_ter}$ equal to $b$ (least frequent elements)
$\boldsymbol{t} \leftarrow$ all indexes in $\boldsymbol{T}$
$index \leftarrow$ random index from $\boldsymbol{index2fill}$
The frequency of the chosen terminal is incremented by one:
    $\boldsymbol{freq\_ter}(\boldsymbol{index2fill}(index)) = \boldsymbol{freq\_ter}(\boldsymbol{index2fill}(index)) + 1$
Return the corresponding index in $\boldsymbol{T}$:
    $T = \boldsymbol{t}(\boldsymbol{index2fill}(index))$
```

tory to produce good offsprings. After an initial preliminary experimentation, we introduce probability functions that mimic this aforementioned adaptation.

Based on the implementation in (Tuson and Ross, 1998) three operator probabilities (intra-crossover: p_{IntraC}; inter-individual feature crossover: p_{InterF}; and subtree mutation: p_M) have a common dynamic behavior through the initial experimental runs, with the following characteristics: it was periodic, the variation range was rather small and it was stationary. This means that the probabilities for these operators had to be almost constant to produce good offsprings during the search. Therefore, choosing static values for these operators seemed an appropriate decision for our work. Moreover, it produced almost identical results during initial experimentation compared with the algorithm by (Tuson and Ross, 1998).

The other three operators (inter-crossover: p_{InterC}; feature insertion mutation: p_{K^+}; and feature removal mutation: p_{K^-}) exhibited unique dynamic behaviors, which were implemented as

$$p_{InterC}(gen) = -\Delta_{InterC} \frac{log(gen)}{log(gen_{max}+1)} + \hat{p}_{InterC} \tag{6.11}$$

$$p_{K^+}(gen) = \frac{-sin(\frac{10gen}{gen_{max}} + 2.5)}{\frac{40(gen+\frac{5gen_{max}}{100})}{gen_{max}}} + 0.25 \tag{6.12}$$

$$p_{K^-}(gen) = 1 - p_{InterC} - p_{InterF} - p_{IntraC} - p_M - p_{K^{(+)}}, \tag{6.13}$$

where gen is the current generation, gen_{max} is the maximum generation number, $-\Delta_{InterC}$ is a decay parameter and $\hat{p}_{InterC}$ is an initial probability. At the beginning of the search, the genetic exchange between different individual promotes a quick exploration, hence the high probability for the inter-crossover operator. However, in latter generations, this could be ineffective, where exploitation is desired; it is more beneficial to reuse material within same individual rather than from others. A decaying logarithmic function for the inter-crossover probability replicates this behavior.

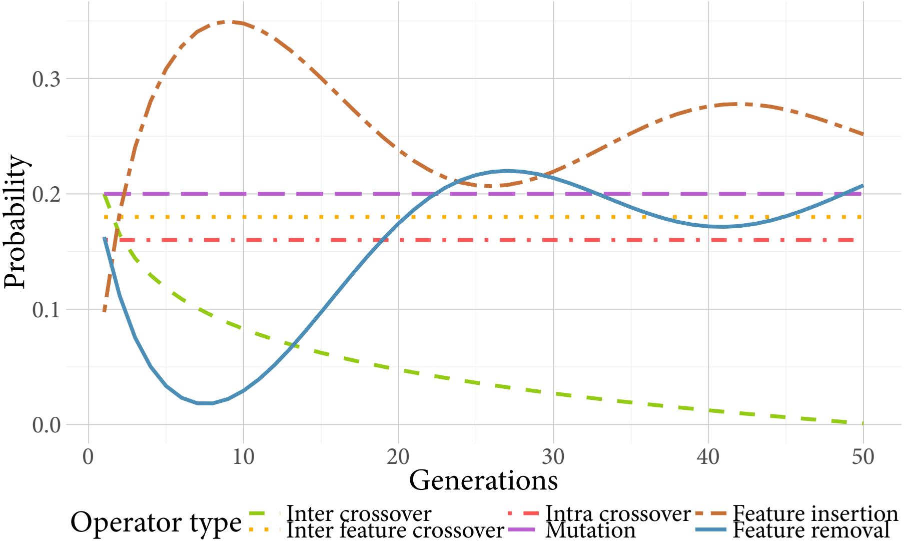


Figure 6.17: Operator probabilities through the search.

Two of the most important operators are the feature insertion and removal mutation. Indeed, these are the ones responsible for the flexibility of the new feature space size. There is a higher change that the hyperplane of the classifier has a good class separability when the number of dimensions is high. A high probability for the generation of new features (p_{K^+}) at the start of the run greatly promotes the search for the most promising size of the new feature space. The ratio between these operators and the static ones decreases after the initial generations, giving more importance to the evolution of the solutions without changing the feature space too much. Because evolved models stagnate at the end of the run, increasing the feature space size is required once more. Moreover, p_{K^-} is seen as the complementary part of p_{K^+} and p_{InterC}. The operator functions are plotted in Fig. 6.17 for the case that $gen_{max} = 50$.

5. **Fitness calculation.** The fitness is computed by a local and a global measure, which are used differently by the search. More specifically:

   (a) Local fitness measure (Fisher's Discriminat Ratio). This is computed at the subtree level, which provides a separability value for each constructed feature. Certainly, the quality of a solution depends on the quality of their individual features. The local measure f' is only used as the criterion for the inter-crossover subtree selection, that is crossover points are selected within the subtrees with the best local fitness. Although the GP selection process chooses two individuals with similar fitness rank it does not guarantee that all individual subtrees have a good performance. We are not interested to interchange material from transformation subtrees with poor performance, rather we prefer a greedy approach where only material from the best solutions are used. This is given by

   $$\mathrm{F1}_i = \left.\frac{(\mu_{C1_i} - \mu_{C2_i})^2}{\sigma^2_{C1_i} + \sigma^2_{C2_i}}\right|_{i=1\ldots r}, \qquad (6.14)$$

   where μ_{C1}, μ_{C2}, σ_{C1} and σ_{C2} are the means and standard deviations corresponding to class one and two for the ith feature. For any given tree K, r local fitness measures are calculated. It was found that the use of the local fitness in other operators did not help the search.

   (b) Global fitness measure. Due to the complexity of the problem which could lead to over-fitting, the global fitness depends on two regularization procedures. Let the global fitness be defined by

   $$f' = (100 - Acc) + \epsilon, \qquad (6.15)$$

   where Acc is the classification accuracy and ϵ is a first regularization term. Moreover, Acc is calculated with

   $$Acc = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}, \qquad (6.16)$$

where TP (True Positive), TN (True Negative), FN (False Negative) and FP (False Positive) are values from the confusion matrix as a result of the labels $\hat{y}_{train}$ obtained by the LDA classifier. The term ϵ is proposed as the first regularization term, given by

$$\epsilon = \left|0.5 - \frac{\text{FN}}{\text{FN} + \text{FP}}\right|, \tag{6.17}$$

which becomes zero if FP and FN are equal. In preliminar experimentation it was found that the algorithm tended to avoid early over-fitting when there was a balance in the FN and FP scores.

In this work, we need first to compute the second regularization, defined as $K \in \mathbb{R}^r \to \boldsymbol{\beta} \in \mathbb{R}^r$ and $\boldsymbol{\beta} = \beta_i \forall i \in \{1 \ldots r\}$, to calculate the accuracy and the confusion matrix. This is achieved by artificially changing the covariance of the tree outputs, and it is defined by

$$\begin{aligned}\beta_i(\boldsymbol{x}) = [&\alpha_{C1_i}(\{K(\boldsymbol{x})|_{C1}\} - \mu_{C1_i}) + \mu_{C1_i},\\ &\alpha_{C2_i}(\{K(\boldsymbol{x})|_{C2}\} - \mu_{C2_i}) + \mu_{C2_i}],\end{aligned} \tag{6.18}$$

and

$$\alpha_{C1_i} = 1 + \frac{2\sigma_{C1_i}}{max(\{K(\boldsymbol{x})|_{C1}\}) - min(\{K(\boldsymbol{x})|_{C1}\})}, \tag{6.19}$$

$$\alpha_{C2_i} = 1 + \frac{2\sigma_{C2_i}}{max(\{K(\boldsymbol{x})|_{C2}\}) - min(\{K(\boldsymbol{x})|_{C2}\})}, \tag{6.20}$$

where $\{K(\boldsymbol{x})|_{C1}\}$ denotes the tree output corresponding to all the data samples from class $C1$ (normal) and $\{K(\boldsymbol{x})|_{C2}\}$ is the same but corresponding to class $C2$ (relaxed). The main reason for this is that data variance is high in this problem and the classifier training has to be relaxed so that prediction accuracy on unseen data increases. Please note that the actual penalty is given by a factor of twice the standard deviation of

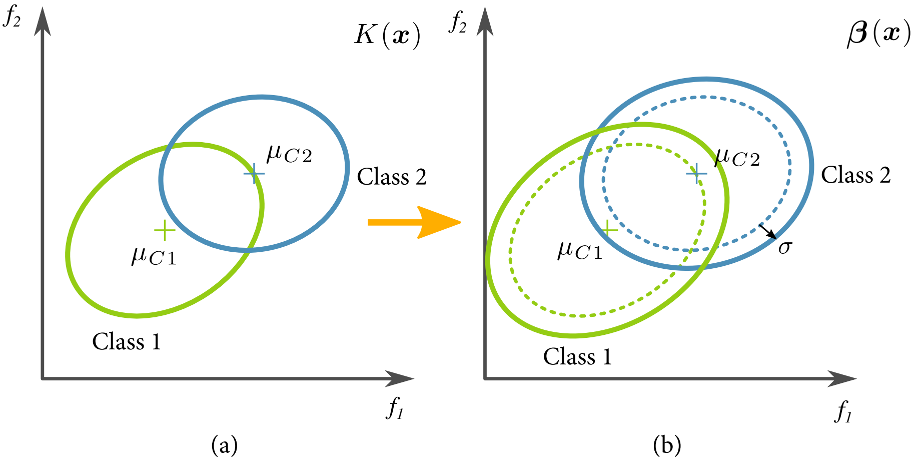


Figure 6.18: Class variance penalization, where (a) corresponds to the tree output, in this example $f_1, f_2 \in i = 1 \ldots r$ are new features; and (b) corresponds to the data modification with a larger distribution variance used to compute the global fitness.

data produced by the tree outputs. This is exemplified in Fig. 6.18.

Afterwards, the LDA hyperplane can be computed using Eqs. 6.7 and 6.8 by using δ as the input data and then obtaining predicted labels. These labels are then used to compute the accuracy.

Finally, it is important to mention that although the LDA classifier was used during training, it was meant only for driving the search so the class separability increases. Once $^{+}$FEGP finds a new transformation model, any classifier can be used during testing. This offers flexibility in terms of the algorithmic choice, specially when dealing with a broader range of classification problems.

6.4 EXPERIMENTATION AND RESULTS

In this section we present experimental details regarding the proposed system. The reported algorithms were implemented in (MATLAB, 2015). Specifically for the $^{+}$FEGP algorithm the GPLAB[2] (Silva and Almeida, 2005) toolbox was used.

Given the stochastic nature of GP, a series of multiple runs were executed in order to statistically determine the algorithm performance. Following the LOOCV scheme for data partition and validation, the experiments involve 30 independent runs per each fold. Results presented in this work contain 390 runs in total (13×30). Details of the values used for filtering, CSP and LDA were mainly discussed in Section 6.2. Furthermore, $^{+}$FEGP parameters are summarized in Table 6.1. Some comments in the third column are derived from experimental tuning of the algorithm.

The training classification performance of the proposed enhanced system is shown in Fig. 6.19, with the overlapped performance of the reference system for comparison. Predictive performance calculated over the testing partitions is presented in Fig. 6.20. Basic statistical results for the $^{+}$FEGP based system per LOOCV fold are given in Fig. 6.21. To validate our results, non-parametric Friedman and Wilcoxon rank sum tests were used to calculate pairwise statistical differences at a LOOCV fold level. The results are presented in Table 6.2 and Table 6.3 corresponding to the training and testing phases, with p-values corrected by applying the Benjamini-Hochberg correction for multi-group comparisons.

$^{+}$FEGP fitness performance and LDA classification accuracy during the training phase are shown in Fig. 6.22. Although the $^{+}$FEGP algorithm only uses the fitness measure (bottom plot), the top plot shows the convergence of the actual classification accuracy achieved by LDA across the generations. Please note that the bottom convergence plot condenses all 390 runs by first computing the median of 30 runs per fold and finally calculating the mean of all fold results. Similarly, in Fig. 6.23,

[2] http://gplab.sourceforge.net/

Table 6.1: $^{+}$FEGP parameters used in the experiments.

Parameter	Value	Comments
Generations	50	If the max depth for the evolved trees is small then more generations are not needed since the solution size is restricted and the overall search stagnates.
Population size	100	A small population was found to be sufficient to evolve good solutions with larger populations reaching the same performance.
Population initialization	Full	Since the search space can be huge, very small individuals are not beneficial to explore quickly the solution space.
Population init max depth	5 levels	A good compromise between structure diversity and complexity.
Initial features	3	
Search operators	Inter crossover, Inter feature crossover, Intra crossover, Mutation, Feature insertion, Feature removal	
Initial probabilities	0.2, 0.2, 0.2, 0.05, 0.15, 0.1	Dynamic probabilities.
Delta values for decay functions	0.2	This only applies to the Inter-crossover operator.
Function set	+,-,×,/,log,cos,sin,tan,$\sqrt{}$,abs	
Terminal set	random [0,1], input features	
Max depth level	20	Since individuals can have several features, big values for the allowed depth is not desired. If a child tree is above this limit after the genetic operation, then it is discarded and a reproduction operation is performed instead.
Selection	Tournament, size 5	A small tournament size provides a richer diversity in the population.
Elitism	Only best individual survives	Assures that the best solution is not lost from generation to generation.

prediction performance in terms of classification accuracy is presented as well.

Two examples of the data distribution of each class are depicted in figures 6.26 and 6.27, corresponding to the folds with the best and worst performance on the testing data. Left scatter plots show the first two principal components from PCA calculated over the raw signals. The middle plots correspond to the data after the CSP (matrix **Z**) calculation. Note a slightly different distribution between both folds, because the CSP is performed with different training data. The plots on the right depict the data distribution after a randomly chosen run of the $^{+}$FEGP algorithm. The classification accuracy for Fig. 6.26 is 82.8% for training and 99.1% for testing. The homologous values for Fig. 6.27 are 81.4% (training) and 44.1% (testing).

6.5 DISCUSSION

One of the main contributions of this work is the proposed feature extraction method, $^{+}$FEGP. Generally speaking, the classification accuracy in the training phase, as seen in Fig. 6.19, surpasses the reference system. Furthermore, the system's classification accuracy on unseen data also improved over the reference system, as seen by the testing performance reported in Fig. 6.20.

If we take a look at the system performance for each fold (Fig. 6.21), we immediately see that some data partitions lead to better classification accuracies than others. Fig. 6.26 and Fig. 6.27 show the best and worst cases in terms of quality in more detail. After dimensionality reduction with PCA we can recognize that the volume of the overlap region for both classes is high in both cases for the raw data. Intuitively, we can deduce that for the majority of state-of-the-art classifiers the performance will be poor in such circumstances. The effectiveness of CSP can be clearly seen afterwards, with a substantial increase in the separability of the classes. At this point we can see that the testing cluster is quite different between almost all folds, being fold 4 and 8 the extreme cases. For fold 4 (Fig. 6.26), the testing data matches the distribution of

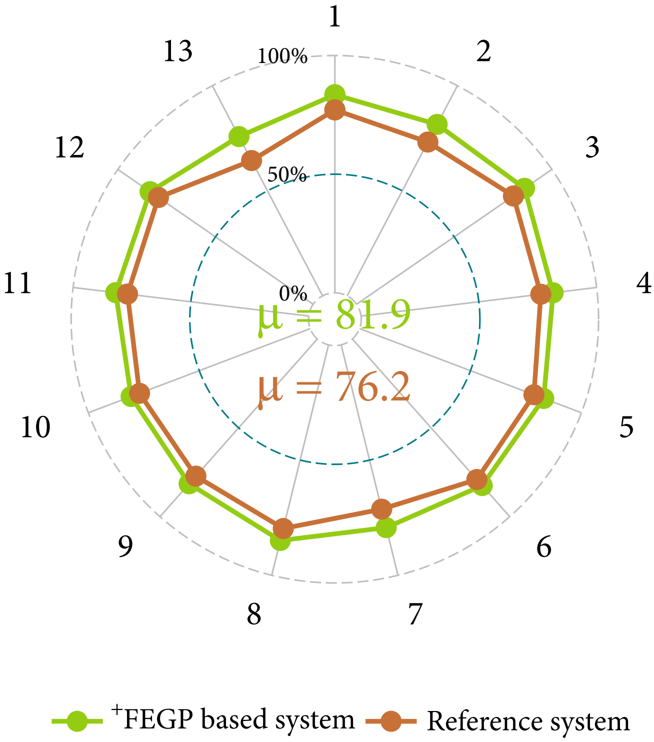


Figure 6.19: Training performance of the reference system and the proposed enhanced system for each LOOCV fold. In the center, mean values for all fold are shown.

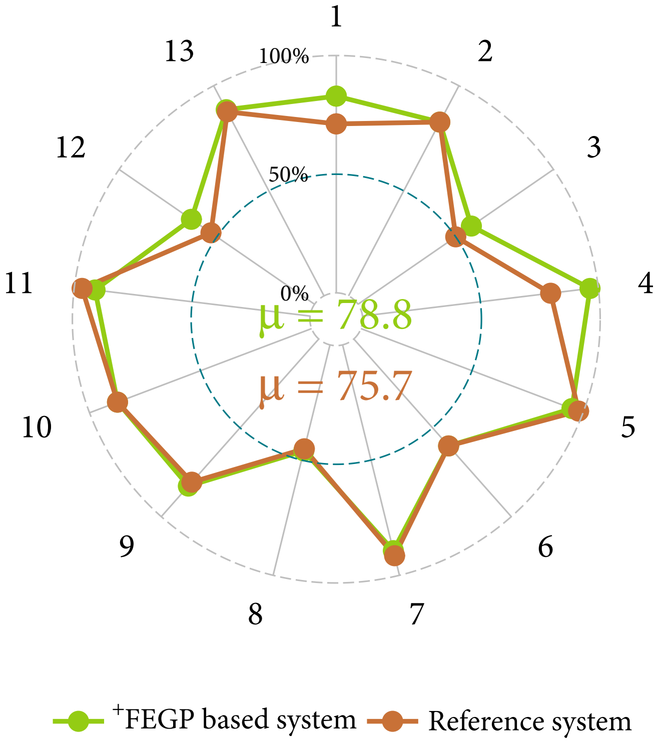


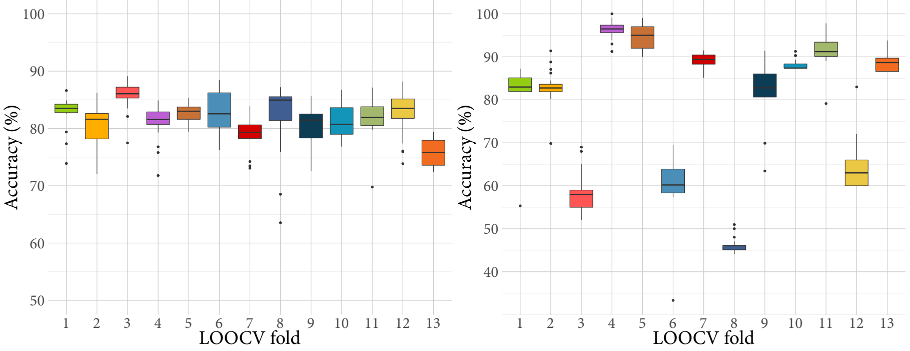
Figure 6.20: Testing performance of the reference system and the proposed enhanced system for each LOOCV fold. In the center, mean values for all fold are shown.

Table 6.2: Statistical pairwise tests corresponding to the training performance. The upper triangular matrix encloses the adjusted p-values with the Benjamini-Hochberg correction of the Friedman test and the lower triangular matrix consist of the p-values for a Wilcoxon rank sum test. Bold values indicate that the null hypothesis is rejected at the $\alpha = 0.05$ significance level for both tests.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-	**0.0412**	**0.0033**	0.2102	0.5796	0.2102	**0.0010**	0.2102	**0.0412**	0.3834	0.5796	0.5796	**0.0003**
2	**0.0168**	-	**0.0033**	0.8273	0.0966	0.3834	0.2102	0.5796	0.3834	0.8273	0.2102	0.3834	**0.0033**
3	**0.0001**	**0.0000**	-	**0.0003**	**0.0003**	0.0966	**0.0001**	**0.0412**	**0.0003**	**0.0010**	**0.0033**	**0.0412**	**0.0001**
4	**0.0206**	0.6327	**0.0000**	-	**0.0412**	0.3834	**0.0412**	0.3834	0.5796	0.3834	0.8273	0.5796	**0.0010**
5	0.5799	**0.0442**	**0.0000**	0.0741	-	0.8273	**0.0033**	0.5796	0.0966	0.2102	0.8273	0.5796	**0.0001**
6	0.6326	0.1589	**0.0103**	0.2683	0.8602	-	0.0966	0.5796	0.2102	0.8273	0.5796	0.5796	**0.0003**
7	**0.0001**	0.1020	**0.0000**	**0.0048**	**0.0000**	**0.0021**	-	**0.0033**	0.0966	0.3834	**0.0033**	**0.0010**	**0.0010**
8	0.2370	**0.0180**	**0.0030**	**0.0206**	0.1445	0.6689	**0.0005**	-	0.0966	0.0966	0.8273	0.3834	**0.0033**
9	**0.0071**	0.9198	**0.0000**	0.4970	**0.0221**	0.1130	0.0741	**0.0111**	-	0.8273	0.0966	0.5796	**0.0033**
10	0.0871	0.6873	**0.0000**	0.8405	0.0782	0.2370	**0.0441**	**0.0469**	0.6326	-	0.5796	0.0966	**0.0010**
1	0.2176	0.1908	**0.0004**	0.5973	0.4657	0.6507	**0.0005**	0.2576	0.1743	0.2177	-	0.8273	**0.0003**
12	0.6689	0.0559	**0.0046**	**0.0497**	0.4063	0.8014	**0.0008**	0.7820	**0.0268**	0.1445	0.3023	-	**0.0001**
1	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0004**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	-

Table 6.3: Statistical pairwise tests corresponding to the testing performance. The upper triangular matrix encloses the adjusted p-values with the Benjamini-Hochberg correction of the Friedman test and the lower triangular matrix consist of the p-values for a Wilcoxon rank sum test. Bold values indicate that the null hypothesis is rejected at the $\alpha = 0.05$ significance level for both tests.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-	0.5262	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.1335	**0.0000**	**0.0000**	0.0003	**0.0000**
2	0.4484	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0003**	**0.0000**	0.8273	**0.0000**	**0.0000**	**0.0000**	**0.0000**
3	**0.0000**	**0.0000**	-	**0.0000**	**0.0000**	**0.0182**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0013**	**0.0000**
4	**0.0000**	**0.0000**	**0.0000**	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0052**	**0.0000**	**0.0000**
5	**0.0000**	**0.0000**	**0.0000**	0.0521	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0182**	**0.0000**	**0.0000**
6	**0.0000**	**0.0000**	**0.0284**	**0.0000**	**0.0000**	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.0537	**0.0000**
7	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	-	**0.0000**	**0.0000**	0.1335	0.0537	**0.0000**	0.8273
8	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	-	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
9	0.3501	0.8996	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	-	**0.0052**	**0.0003**	**0.0003**	**0.0000**
10	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.1258	**0.0000**	**0.0000**	-	**0.0003**	**0.0000**	0.5262
1	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0101**	**0.0000**	**0.0012**	**0.0000**	**0.0000**	**0.0000**	-	**0.0000**	**0.0013**
12	**0.0000**	**0.0000**	**0.0001**	**0.0000**	**0.0000**	**0.0232**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	-	**0.0000**
1	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.5435	**0.0000**	**0.0000**	0.7784	**0.0007**	**0.0000**	-



(a)

(b)

Figure 6.21: Box plots of the enhanced system performance for 30 runs per fold, where (a) corresponds to the classification accuracy for the training partition and (b) is the homologous for the testing partition.

the training data, making it easier for a classifier to obtain good generalization results. On the other hand, in fold 8 (Fig. 6.27), the case is the opposite; the data cluster belongs to a multi-modal distribution, where the obtained model is evolved over a different mode than where the testing data resides, suggesting that the EEG recordings for that particular subject are quite different from the rest. Lets recall that although the experiments were done in a controlled environment we cannot fully constrain the physiological activity of each subject. This is a more realistic scenario, but it makes the problem more difficult to solve. Furthermore, we can see the benefits of the $^{+}$FEGP algorithm in the third scatter plot of each figure. For fold 4, the new features make the problem quite trivial for almost any classifier. The robustness here is that the testing data does not shift nor varies, rather it integrates to the counterpart training samples. The more difficult case is fold 8, which given an already problematic situation from the CSP output, the $^{+}$FEGP improvement is relative small. Here, an important observation is that even for this worst case, the $^{+}$FEGP performance is at least the same or better, but not worse than the performance of the reference system.

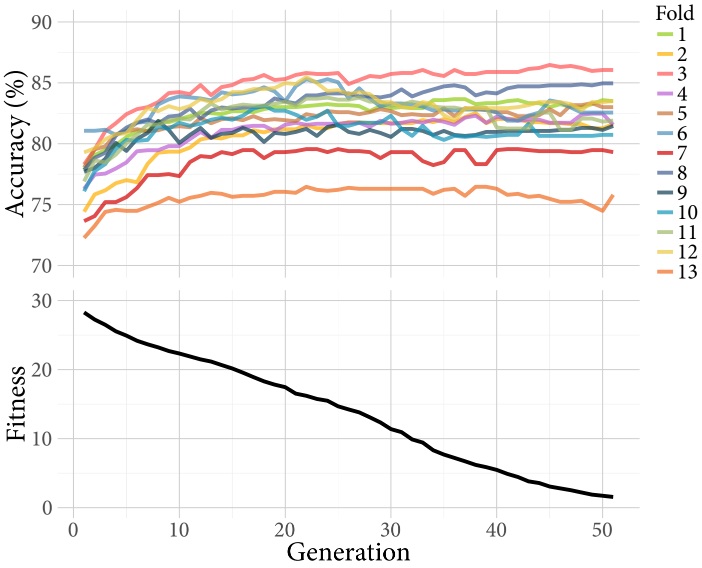


Figure 6.22: LDA classification accuracy and convergence plots for the evolution of fitness on the training data. Top plot shows the history of LDA accuracy calculated over the best individual per fold; it represents the median over 30 independent runs. The bottom plot shows the evolution of fitness from the best individual representing the mean of the 13 (folds) medians over 30 runs.

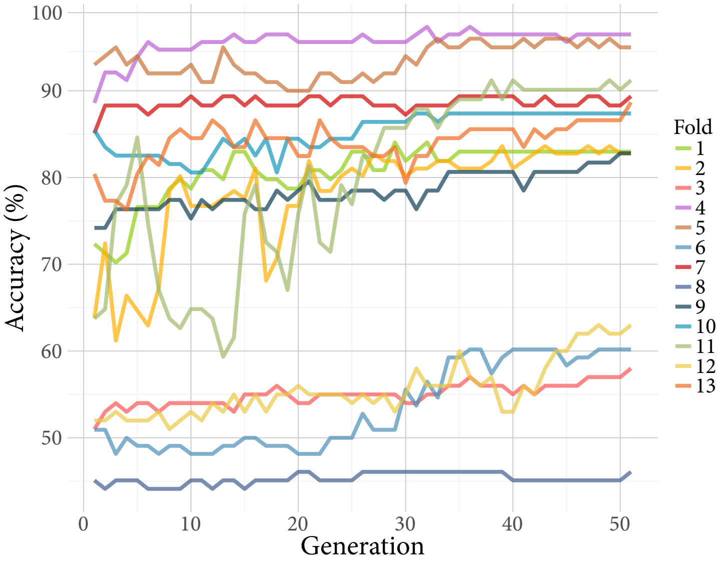


Figure 6.23: Convergence of LDA classification accuracy on the testing data using the best individual found by $^+$FEGP at each generation. Each line in the plot represents the median over 30 independently runs.

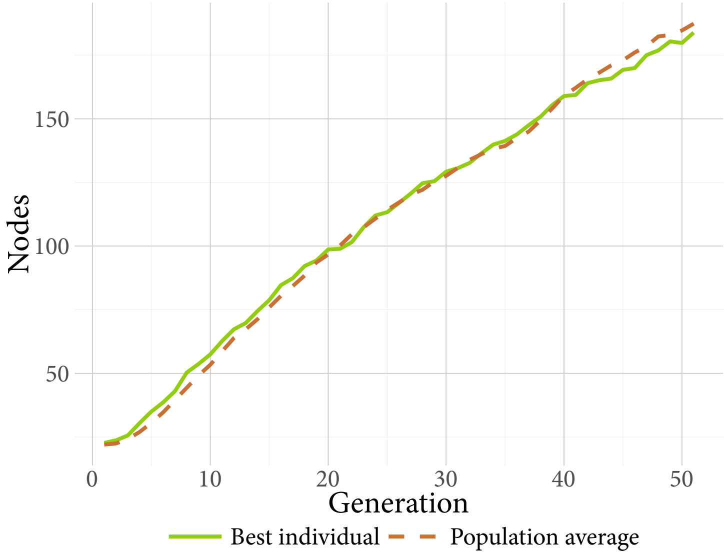


Figure 6.24: Evolution of best individual and population average sizes, showing the mean of 13 medians over 30 runs.

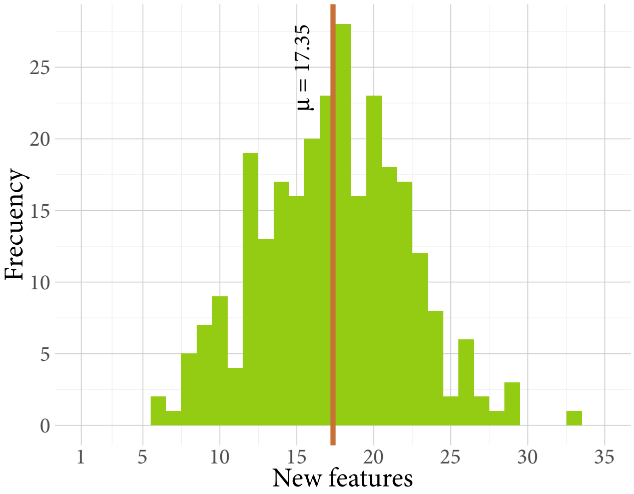


Figure 6.25: Frequency histogram of the number of new features produced by the $^{+}$FEGP algorithm.

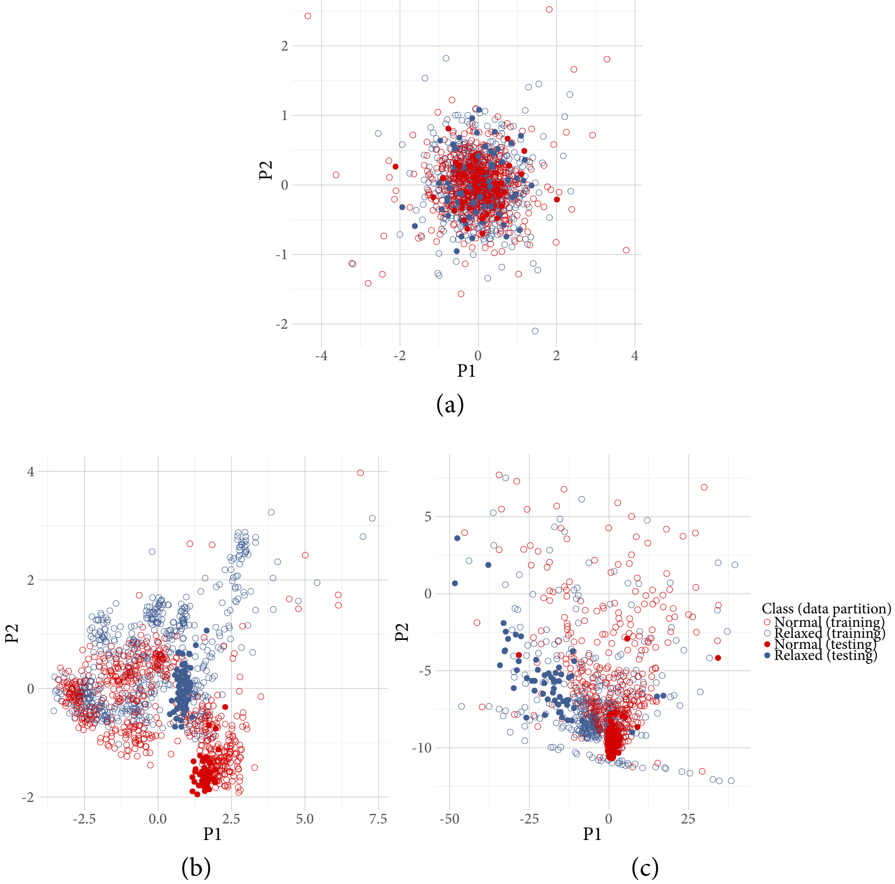
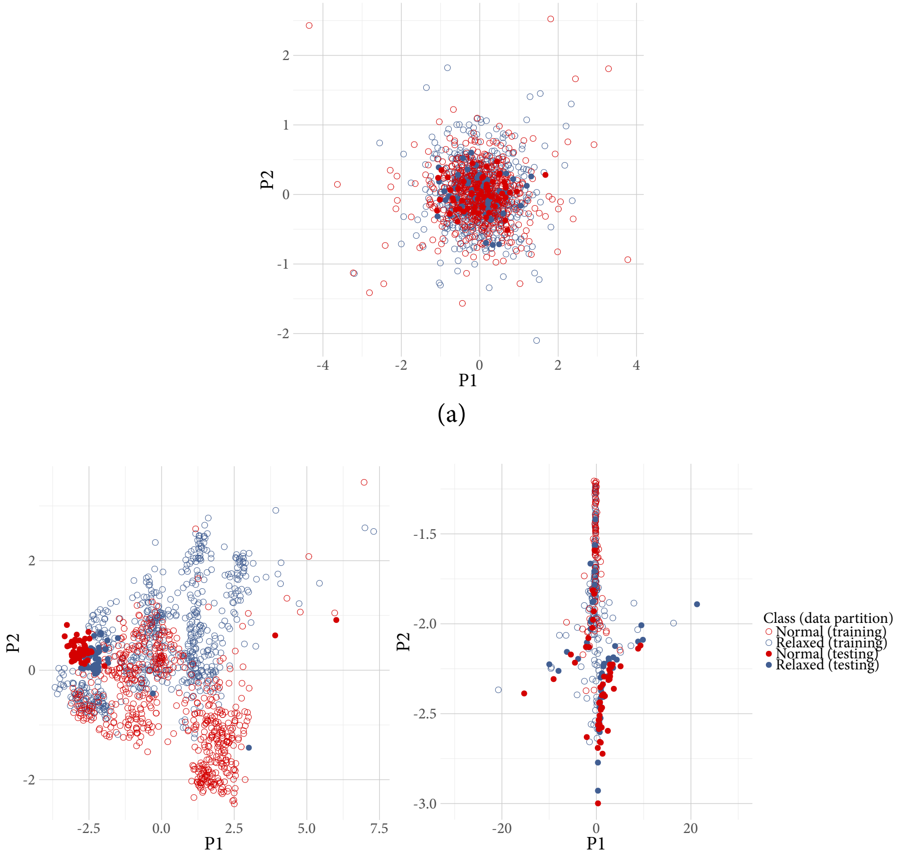


Figure 6.26: The first two components from PCA corresponding to fold 4 in the LOOCV. (a) is the class distribution for the raw data; (b) after the CSP; and, (c) after the best transformation found by $^{+}$FEGP. Classification accuracy for training is 82.8% and 99.1% for testing.



(a)

(b)

(c)

Figure 6.27: The first two components from PCA corresponding to fold 8 in the LOOCV. (a) is the class distribution for the raw data; (b) after the CSP; and, (c) after the best transformation found by $^{+}$FEGP. Classification accuracy for training is 81.4% and 44.1% for testing.

We can analyze further the performance for the remaining folds in the LOOCV. The Wilcoxon test infers that the majority of the folds have a similar statistical performance in terms of their medians in the training phase, with the exception of fold 3 and 13. Although these are the extreme cases for training, their performance is an improvement upon the CSP output. This can be extended to all folds; indeed $^+$FEGP produced almost a linear improvement over all folds compared with the reference system. In the testing phase, almost all null hypothesis combinations were rejected, something expected giving the nature of CSP output. However, the improvement was not linear among all folds; in folds 1, 3, 4 and 13, the improvement by $^+$FEGP was significant, but in folds 5, 7 and 11 there was a small reduction in performance (see figure 6.20), suggesting the obtained models over-fitted. Moreover, according to the Friedman test, the hypothesis that samples from the algorithm's outcome belong to the same distribution was accepted for almost all folds during training, with the exception of folds 3 and 13, if we compare them with the rest. During testing, the hypothesis was rejected for almost all combinations.

The performance of the enhanced system can be seen from different angles. At the bottom plot of Fig. 6.22 we can see a steady minimization of the global fitness. At the same time, we can analyze the performance of the LDA classifier calculated directly over the transformed data. Although training fitness is decreasing during the runs, specially at the end, the classification accuracy does converge, and more importantly, the accuracy on unseen data does not decrease in the final iterations of the search (Fig. 6.23).

Moreover, if we consider solution sizes in $^+$FEGP, shown in Fig. 6.24 (plot lines represents the mean of 13 medians over 30 runs), solutions grow almost linearly during the search. This is a common result in GP with tree representations, where the increase of solution size is a product of fitness improvement (Koza, 1992). A side effect of this phenomena is that sometimes fitness becomes stagnated with unnecessary growth of trees, referred to as bloat (Silva and Costa, 2009). However, in our experiments with a relatively small number of generations for the search, $^+$FEGP does not exhibit any bloat, that is the increase in solution size is always accompanied by a steady improvement in fitness. This again is made possible by the regularized fitness measure employed by $^{+}$FEGP.

The increase of solution sizes also means that there is an increment in the number of newly created features, the depth in the tree structure or both. Specifically, we can seen the frequency histogram for all runs and folds in Fig. 6.25, showing a nearly normal distribution. All individuals were started with 3 features at the beginning of the runs. The evolved models produced a minimum of 6 and a maximum of 33 new features, with an average of 17.35.

As mentioned earlier in Section 6.3, there is an inherent flexibility in terms of the classifier selection, after using the $^{+}$FEGP. Therefore, additional classifiers were evaluated: Tree Bagger (TB), Random Forests[3] (Jaiantilal, 2012) (RF), k-NN and SVM. All of them produces non-linear decision functions, as opposed to the relative simpler LDA. With the exception of RF, all classifiers are based on their MATLAB implementations. No parameter tuning was performed for these, rather their corresponding default settings were used. The classification accuracies are summarized in Table 6.4. A one-way Analysis of Variance (ANOVA) was performed among the classifiers to estimate statistical significance on the null hypothesis that samples belong to the same distribution. Correlating the results from Table 6.4, Fig. 6.26 and Fig. 6.27 we can deduct that for this particular problem, complexer classifiers do not produce better accuracies for unseen data, rather they all over-fit. One reason for this, is that most of these algorithms perform accurately if there is enough data to learn, however for the case of under-sampled data, sometimes simpler models produces better results. The simpler assumptions of LDA (which in this case are actually infringed: class distributions are not normal nor share the same covariance) help to relax the model and behaves better as a generalization method.

[3] https://github.com/jrderuiter/randomforest-matlab

Table 6.4: Classifiers performance for the $^{+}$FEGP algorithm. Values represent means and standard deviation over 13 folds. Last row shows the one-way ANOVA test between classifiers; bold value indicates that the null hypothesis is rejected at an $\alpha = 0.05$ significance level.

Name	Training	Testing
LDA	81.9 ± 2.5	78.8 ± 16.2
TB	100 ± 0	70.7 ± 15.9
RF	100 ± 0	70.8 ± 16
k-NN	100 ± 0	63.2 ± 10.9
SVM	96.1 ± 1.2	63.5 ± 14.7
p-value	**0.0000**	0.0567

6.6 CHAPTER CONCLUSIONS

In this chapter we propose a new system for the classification of two mental state based on EEG recordings, with the introduction of a novel GP algorithm that performs as a feature extraction method. The proposed system involves several functional blocks: acquisition of signals, pre-processing, feature extraction and classification. This work is the first to use a GP based algorithm as feature extraction for the identification of mental states; specifically, normal and relaxed states.

A reference system was implemented based on our previous work (Vézard et al., 2015), which produced competitive results among similar research in this domain. Nevertheless, the proposed system outperformed our previous results by exploiting further the separability of classes using evolved transformation models. Certainly, apart from the appropriate choice of the classifier, one of the most important blocks in these type of systems is the feature extraction process, where the pattern recognition of the underlying data is performed.

Evidence found during experimentation allow us to conclude in the following directions. First, the $^{+}$FEGP algorithm in a wrapper scheme produced improved performance compared to the system with just CSP, reaching a classification accuracy of 78.8% for unseen data. Certainly, any system could efficiently benefit from the hybridization of different tools to achieve competitive results; a sole technique usually cannot tackle such a complex problem. Second, the proposed genetic operators allows us to evolve transformation models with good testing performance. Third, the unique fitness function steered the search toward solutions that avoided over-fitting in most of the cases. Fourth, the sizes of the evolved solutions are relative small in terms of the number of new features and the number of nodes per subtree. Moreover, in our tests $^{+}$FEGP was not affected by bloat (Silva and Costa, 2009). Fifth, although from the system design point of view the classifier can be interchanged, other classifiers did not produce better results compared with LDA. Nevertheless, when examining the whole scenario we can state that most of the shortcomings were derived from the underlying data properties. In this case, we could opt for three options: increase the system's complexity by incorporating an additional processing step and help reduce further issues like over-fitting, use adaptive classifiers or increase the number of EEG recordings. In either of the circumstances, we can foresee applications of the $^{+}$FEGP outside of this domain, that is by merely adjusting the fitness function it can be used in any data transformation scenario.

7 CONCLUSIONS AND FUTURE WORK

7.1 GENERAL DISCUSSION

We can say that there were two original motivations behind this research. First, the implications that a certain human mental state can have in a scenario associated with a high risk, thus the importance to design automatic human-computer interaction systems; and second, the design of new algorithms aimed to improve existing BCI systems. Despite those being our original incentives, this journey turned out to be richer in terms of the obtained insights in several aspects. While the majority of proposals that address the problem of mental states recognition rely on methods from statistical learning and signal processing, other methods are usually required to deal with the highly complex nature of the task. In particular, meta-heuristic algorithms can be a viable alternative to achieve state-of-the-art results. However, there is only a small number of works covering meta-heuristic techniques, if we compared them with classical approaches. This fact may be due to the following reasons: there are still open issues in the meta-heuristic algorithms that prevent them to efficiently solve the problem; the field has not matured to the same level as classical statistical learning and signal processing; or simply, there is not sufficient interest to use these techniques, possible to the lack of theoretical foundations behind their success. We feel that the third reason is too harsh, since important theoretical results have been published (Langdon and Poli, 2001), while other heuristics are commonly used in similar fields. Therefore, in our work we present contributions to help address the first two concerns.

This work began with the study of the classical approaches used in the design of BCI systems. The advances in such techniques have been increasing in recent years, where the combination of different techniques produce the most successful results. We recognized that designing a hybrid methodology might deliver competitive results.

In the first part of the manuscript, we proposed a new feature extraction methodology to deal with the recognition of epileptic seizures. By combining three different types of measures we build a feature set. MP allows us to extract a global characterization of the EEG signals, using density values of the number of Gabor atoms for a given epoch. On the other hand, the Hölderian regularity provides a more detailed characterization, the local regularity (around the neighborhood of a given sample) of the signals. The last set of measures, given by mere statistical calculations over the raw EEG signals, reside in the time domain, as opposed to the first measure which is produced in the frequency domain. Notice that this particular combination of measures collects different levels of information from the raw signals. The experimentation produced competitive results that outperformed, in the majority of the cases, a vast collection of other state-of-the-art methodologies. Nevertheless these encouraging results, the proposal allows us to identify the difficulties normally found when designing BCI systems, a rather common overstatement that new researchers in the field need to consider.

Afterward, we proceeded to study the current state of meta-heuristic algorithms and their application in the domain of BCI design. There is a large number of such algorithms, that are derived from very contrasting rationale. Instead of analyzing this vast group, we focused on a particular bio-inspired algorithm, namely GP. This is a paradigm from the field of EC, in which GP stands out because the solutions representation make it ideal to handle a wide variety of applications. In particular, GP can be used for feature selection, feature extraction or classification; almost every part of a complete BCI system.

However, GP has several open issues that have been the focal point of much research in the field. One of these open issues is the way the search is performed. It has been studied before that the search is not optimal in terms of the solution size or convergence time. The majority of the meta-heuristic algorithms perform a type of exploration search, however the exploitation can become cumbersome mainly because it is not explicitly expressed as a mechanism of search. A meme, a term introduced by Richard Dawkings, was adapted to the computational field and incorporated into meta-heuristic algorithms, namely memetic algorithms, where exploration and exploitation techniques are combined. By incorporating a local neighborhood search, the synergy produced by these two mechanisms results in a faster convergence. The majority of works focus on representations that are rather simple to implement, such as real valued representations. In that sense, GP is still behind, mainly because its representation is rather general, where the adaptation of LS procedures is more difficult.

We recognized that GP is able to evolve the solutions syntax, but the optimal parametrization is never explicitly included as an objective during the search. Thus, we proposed the following hybridization:

1. parameters are added to each node in the tree;
2. an upper linear tree is inserted to the original tree;
3. a heuristic to select which individual has to be optimized is defined.

In the first item, the aforementioned space is explored by a numerical optimizer guided by the same objective function as the global searcher, essentially producing a LS in the solution (program) output landscape. The outcome is that for a particular solution, the optimal parameters are found without changing the structure.

The second item, a more trivial idea, was meant to help the search during the numerical optimization, by adding a bias and a scalar value to the non-linear part.

For the third item, the heuristic for the LS selection is considered to play an important role in the memetic mechanism. For that, we tested a series of different variants, concluding that two different options can be opted depending on the needs: a fitness and a size based heuristics. In the former, a subset of the best solutions in the population turned out to produce the best performance. In the latter, choosing relative small solutions in the population produced a faster convergence. Extensive experimental evidence was collected to validate the initial conjectures. First, in a set of regression problems and later in classification problems. Derived from these experiments, we conclude that GP greatly benefits from the incorporation of LS strategies. In summary, the best solution outperformed canonical GP in terms of quality and size. If this particular issue in GP cannot be considered as closed, we can surely state that this contribution is a small step towards this important goal.

As previously stated, the recognition of mental states is a non-trivial task, that requires the involvement of different approaches. For that, we separated the last part of the thesis in two studies. We first built a BCI system based on classical techniques. Specifically, we used CSP as feature extraction and a LDA classifier. Then we designed an improved system which incorporates a GP based algorithm ($^{+}$FEGP) with the purpose of augmenting the CSP output, in a wrapper scheme. $^{+}$FEGP was built with the rationale behind to evolve transformation models of the form $K : \mathbb{R}^k \rightarrow K' : \mathbb{R}^r$, being K' a new feature set. The search was guided with the objective of improving the classification accuracy for the obtained K' output. By trying to tackle the problem of mental states recognition, another open issue in GP was addressed; a proper representation, which was designed according with the needs of the problem at hand. New search operators were also introduced with independent and specific functions that combined produced a more efficient search for the analyzed problem. A fitness function was defined as well, to complement the other introduced elements into GP. This function included components that helped to reduce over-fitting issues and allowed to evolve models that performed well with testing data. If we oversee this work, we can discover two important contributions: the proposal of a novel approach for an efficient BCI design, and the improvement provided to GP as an stochastic optimization algorithm. We can satisfactorily conclude that we fulfilled the original objective of the thesis.

7.2 FUTURE WORK

The conclusions obtained during the development of this thesis could be considered compelling and produced new opportunities in terms of future research. For that, we will focus first on each part, following the same order as the document, and finally propose a general future research.

For the first part of this thesis, one important aspect to consider is to reduce the computational cost of the feature extraction process, for online or real time implementations. However, there are several ways to cut corners and produce a more efficient method. Hölderian regularity computation can be simplified substantially by relaxing the approximation accuracy and imposing a smaller analyzing window, as done in other works. Alternatively, accurate and efficient approximations can be obtained by using meta-heuristic methods (Trujillo et al., 2012), for which the computational costs are dramatically reduced. MP can also be improved by using a pre-built dictionary employing an optimized family of functions according with a subset of already recorded signals; such a task can be performed offline. Improvements of both techniques would improve the possibility for an online implementation of the proposed approach. Future work will also focus on feature post-processing, which might further improve classification accuracy. The post-processing might be useful to fully exploit the decorrelation hidden in the extracted features. Additional pre-processing techniques might also be effective in boosting performance; e.g., Hölderian regularity has proved to be a powerful tool for denoising signals (Legrand and Vehel, 2003).

For the second part of the manuscript, there are several future directions that can be taken. In terms of efficiency of the memetic GP approach, future work will be centered on exploring and evaluating other parametrization schemes. Moreover, a method must be implemented that determines if a tree requires linear or non-linear parameter optimization, in order to simplify the process whenever possible and to use the parameter values as decision criteria to prune unnecessary subtrees. Although our proposal focused in a single algorithm as the LS optimizer, more efficient algorithms could be more appropriate alternatives. It was found that a global optimum in the LS process is not needed, rather an approximation is sufficient. Thus, a gradient-free algorithm could replace the LS mechanism. Another idea regarding the efficiency aspect, is to develop a surrogate LS mechanism along the GP evolution, where the characteristics of the LS influence can be modeled by another GP population and then replaced as an artificial LS. This would reduce greatly the computational burden of the memetic approach. As previously stated in this manuscript, one of the open issues in GP is a proper representation, which was briefly discussed as a possible difficulty when adapting a LS mechanism to other type of data representations. For that, further research is needed to expand the LS design to cover a wide range of representations.

In the third part, there are several aspects to consider in future work. One is the introduction of feature selection into the system; ultimately, channel reduction is very important for any real-world scenario, mainly for practical reasons. It is also of our interest to increase the number of mental states in future research. Although we emphasize the benefits of the proposed $^{+}$FEGP algorithm in this work, the exploration of different tools in the pre-processing and post-processing is also important. Moreover, we do not express by any means that the $^{+}$FEGP algorithm in conjunction with CSP is the best option, rather a choice, that proved to be very competitive in this domain. Another aspect is the improvement of the $^{+}$FEGP algorithm itself. Further investigation is required to study the evolution behavior; there are still open questions: what is the relationship between the number of features and the efficiency of individual subtrees? Is there a way to fully avoid bloat in longer searches? Are there better genetic operators that allow to search more efficiently? In our previous works, by implementing a type of LS into GP we achieved better results in regression Z-Flores et al. (2014) and classification problems Z-Flores et al. (2015), suggesting that GP in general is benefited from the hybridization of search operators. This encourages us to extend these approaches into this domain. Despite not mentioning computational costs along the presentation of this work, which at the moment is considered as an off-line methodology, it is important to reduce algorithm complexity and at the same time increase its accuracy. In future work, a full study will be done balancing the consumed resources and efficiency, such as by using intrinsic parallel implementations, such as Graphics Processing Units (GPUs) or Field Programmable Gate Arrays (FPGAs).

Finally, generally speaking, the future research direction derived from this thesis can be quite abroad. If we want the influence of BCIs to be extended further into our daily lives, their design must be improved in terms of accuracy and efficiency. For instance, given the complex nature of EEG data for these type of problems, a GP algorithm can be designed in a way that it benefits from specific patterns extracted from the raw signals and dynamically adapts to them, without hand-tuning.

However, to develop an online system, a fast learning algorithm is needed. Here, the advances in approaches for time series forecasting can be really useful, where a delay in the processing might exists but the prediction is performed in real-time.

A more ambitious goal is to develop a complete BCI system by using meta-heuristics, memetic algorithms and a hyper-parametrization mechanisms. An example of this are the memetic algorithms of second and third generation, namely hyper-heuristic and meta-Lamarckian strategies. The actual number of works in GP addressing such systems are rather scarce, a situation that encourages us to work toward this ambitious area.

A

ADDITIONAL REGRESSION EXPERIMENTS USING GP WITH LS

Based on Trujillo, L., Z-Flores, E., Ju, P., Legrand, P., Castelli, M., Vanneschi, L., Sch, O., and Mu, L. (2017). Local Search is Underused in Genetic Programming. In *Genetic Programming Theory and Practice XIV*, number January

A.1 INTRODUCTION

In this appendix we present additional experiments for the integration of Local Search into GP for regression real-world problems. In chapter 4 we showed a study for different strategies on LS selection and we tested in problems where the data was generated by pre-defined functions, the so called benchmark problems. However, in this chapter we proceed to follow the strategies in chapter 5, that is a linear subtree is added to the root node for any selected individual to be optimized and the LS selection in the population is given by a tree size related heuristic. In the following sections we discuss more detail in the implementation, experimentation and results.

A.2 PROPOSAL

First, as suggested in (Kommenda et al., 2013), for each individual K in the population we add a small linear uppertree above the root node, such that

$$K' = \theta_2 + \theta_1(K) \,, \tag{A.1}$$

K' represents the new program output, while $\theta_1, \theta_2 \in \mathbb{R}$ are the first two parameters from $\boldsymbol{\theta}$. Second, for all the other nodes n_k in the tree K we add a weight coefficient $\theta_k \in \mathbb{R}$, such that each node is now defined by

$$n'_k = \theta_k n_k \ , \tag{A.2}$$

where n'_k is the new modified node, $k \in \{3, ..., r + 2\}$ and r is the size of tree K. Notice that each node has an unique parameter that can be modified to help meet the overall optimization criteria of the non-linear expression. At the beginning of the GP run each parameter is initialized by $\theta_k = 1$. During the GP syntax search, subtrees belonging to different individuals are swapped, added or removed together with their corresponding parameters, often called Lamarckian inheritance (Z-Flores et al., 2014, 2015). We consider each tree as a non-linear expression and the local search operator must now find the best fit parameters of the model K'. The problem could be solved using a variety of techniques, but following (Z-Flores et al., 2014, 2015) we use a trust region algorithm.

Finally, it is important to consider that the LS could increase the computational cost of the search, particularly when individual trees are very large. While applying the LS strategy to all trees might produce good results (Z-Flores et al., 2014, 2015), it is preferable to reduce the amount of trees to which it is applied. Therefore, we use the heuristic proposed in (Azad and Ryan, 2014), where the LS is applied stochastically based on a probability $p(s)$ determined by the tree size s and the average size of the population $\bar{s}$ (details in (Azad and Ryan, 2014; Z-Flores et al., 2015)). In this way, smaller trees are more likely to be optimized than larger trees, which reduces the computational cost and improves the convergence of the optimizer by keeping the parameter vectors relatively small. Hereafter, we refer to this version of GP as GP-LS.

A.3 EXPERIMENTS AND RESULTS

We evaluate this proposal on a real-world symbolic regression task, the Yacht problem that has 6 features and 308 input/output samples (Ortigosa et al., 2007). The experiments are carried out using a modified

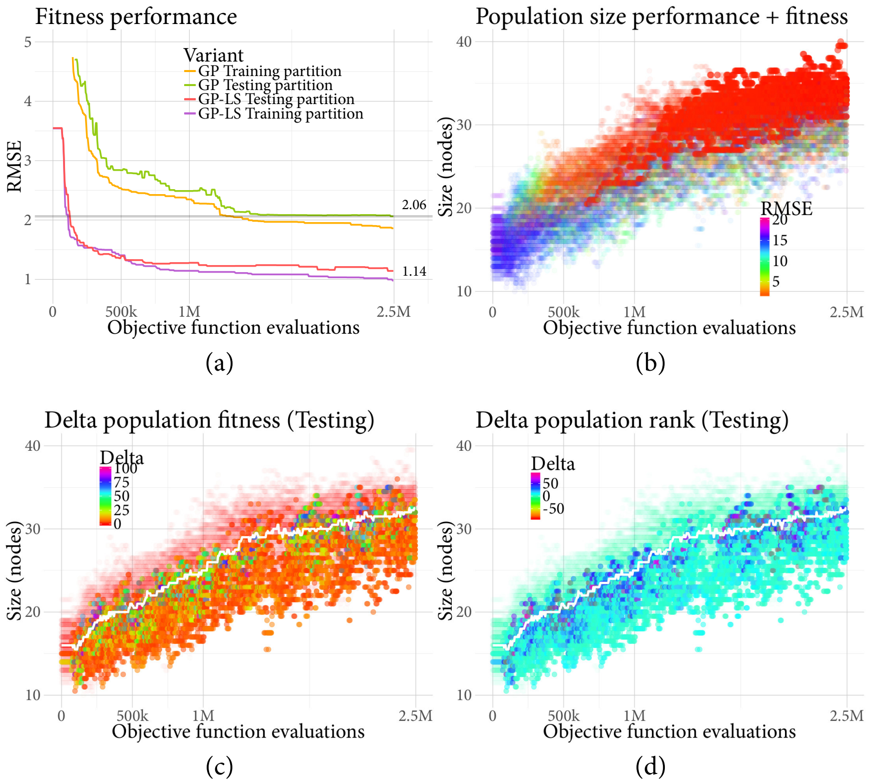
Table A.1: GP parameters.

Parameter	Value
Runs	30
Population	100
Function evaluations	2’500,000
Training set	70% of complete data
Testing set	30% of complete data
Crossover operator	Standard subtree crossover, 0.9 prob.
Mutation operator	Mutation probability per node 0.05
Tree initialization	Full, max. depth 6
Function set	+,-,×,÷,exp,sin,cos,log,sqrt,tan,tanh
Terminal set	Input features, constants
Selection for reproduction	Tournament selection of size 7
Elitism	Best individual survives
Maximum tree depth	17

version of the Matlab GP toolbox GPLab (Silva and Almeida, 2005). The GP parameters are given in table A.1. In what follows, we will present results based on the median performance over all runs. The fitness function used is the RMSE, and the stopping criterion is the total number of fitness function evaluations. Function evaluations are used to account for the computational cost of the trust region optimizer, which in this case is allowed to run for 100 iterations. Results are compared with a standard GP.

Figure A.1 summarizes the main results. The convergence plots of GP and GP-LS are shown in figure A.1(a), showing the median training and testing performance. The figure clearly shows that GP-LS converges faster to a lower error, and at the end of the run it substantially outperforms standard GP, consistent with (Z-Flores et al., 2014, 2015). Figure A.1(b) presents a scatter plot (each point is one individual) of all individuals generated in all runs. The individuals are plotted based on function evaluations and size. Each individual is color coded using a heat map based on test performance, with the best individuals (lowest error) in dark gray. Figure A.1(b) shows that the best performance is achieved by the largest individuals.

However, our strategy is to apply the LS on the smallest individuals of the population. This is clearly validated in figures A.1(c) and A.1(d). Figure A.1(c) shows the raw improvement of test fitness for each individual



(a)

(b)

(c)

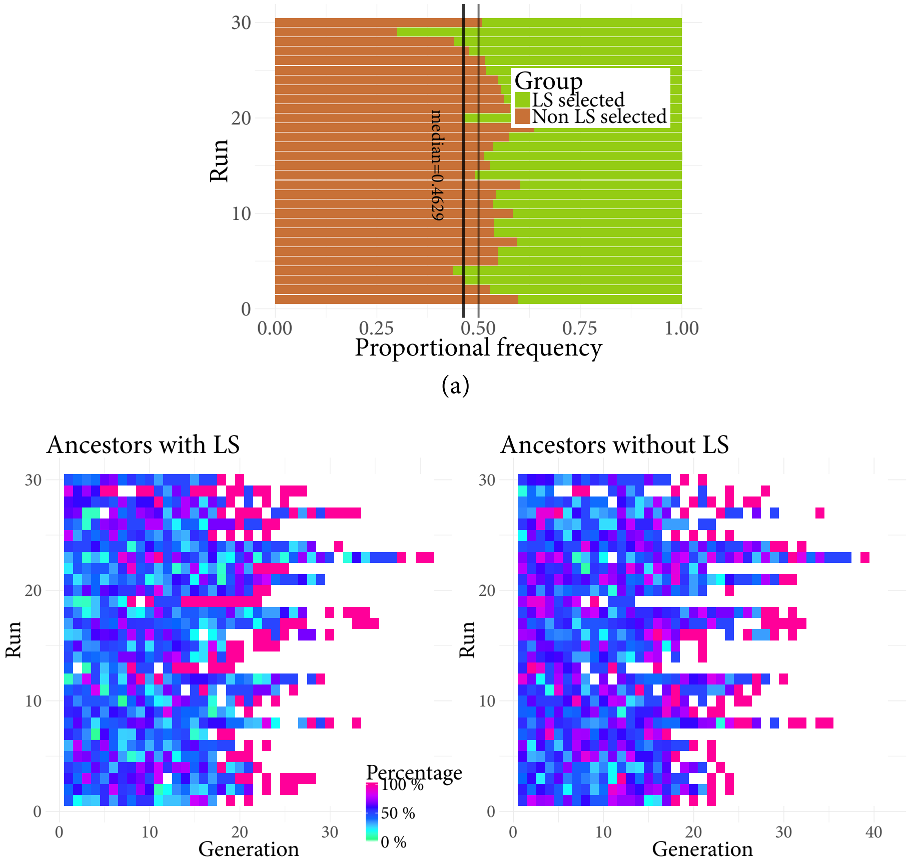
(d)

Figure A.1: Experimental results for GP-LS on the Yacht problem.

before and after the LS. A similar plot is shown in figure A.1(d), however instead of showing the raw improvement, this figure plots the improvement in rank within the population. In both plots the average program size is plotted with a white line, and individuals that were not passed to the local optimizer have a zero value. These plots reveal that: (1) most individuals below the average size are improved by the LS; and (b) the largest improvement is exhibited by individuals that are only slightly smaller than the average program size. The median of the average program size produced by standard GP on this problem is 123.576, which is substantially higher than what is shown by GP-LS.

These results present three interesting and complimentary results. First, GP-LS clearly outperforms standard GP, in terms of convergence, solution quality and average solution size. Second, the LS is clearly improving the quality of the smallest individuals in the population, in some cases substantially. On the other hand, and thirdly, the best solutions are still the largest trees in the population. This means that while the LS operator improves smaller solutions, the best solutions are not necessarily directly subjected to the LS process. This means that the LS process should be seen as an important additional operator, that complements the other search operators. While many previous works have applied a LS process on the best solutions found, our results indicate that this is insufficient, the LS should be applied more broadly to achieve the best results.

Figure A.2 summarizes how the LS operator influences the construction of the best solution. First, figure A.2(a) shows how many times the best solution in the population was chosen by the LS selection heuristic for each run. The plot indicates that the best solution was chosen about 50% of the time. Second, we track all of the ancestors of the best individual from each run, and figure A.2(b) plots the percentage of ancestors that were subjected to the LS. This plot also suggests that, on average, about half of the ancestors were subjected to the LS and half were not.



(a)

(b)

Figure A.2: Influence of the LS operator on the construction of the best solution found for the Yacht problem. Left plot (a) shows the individual ratio of the best individuals that have been chosen for LS versus the ones that do not. The right plot (b) shows the same behavior but in a generation by generation basis, highlighting the ratio in the Z-axis.

B

CASE STUDY: GAS TURBINE MODELING

Based on Enríquez-Zárate, J., Trujillo, L., de Lara, S., Castelli, M., Z-Flores, E., Muñoz, L., and Popovič, A. (2017). Automatic modeling of a gas turbine using genetic programming: An experimental study. *Applied Soft Computing*, 50:212–222

B.1 INTRODUCTION

Gas turbines (GTs) are one of the key elements in the electrical power industry since they transform mechanical power into the desired electrical output. In particular, GTs are widely used in so-called Open Cycle Plants and Combined Cycle Plants. They are considered to be high-risk units since they operate at high speed, temperature and pressure, conditions that impose strict control requirements. The control systems executes the startup and shutdown sequences, control speed, power and temperature, while also performing functions related to safety and protection. Startup and shutdown sequences consist of a series of complex events where each step has to be satisfactorily completed before continuing to the next. Moreover, in central power generation, it is of utmost importance to ensure high levels of availability and reliability of the generating units, particularly due to the high costs associated to system failure or an interruption in the availability of electrical power. Therefore, condition monitoring of GTs within power plants is highly desirable to guarantee the long-term safety of the units and their efficient operation (Sekhon et al., 2006).

For instance, a condition monitoring system can be connected in parallel to the control system for the early detection of potential abnormalities. This can allow supervisors to perform an orderly shutdown of the unit, thus avoiding unexpected stops or failures of a greater magnitude (Sekhon et al., 2006). An important element of such a monitoring system is the model that is used to describe or predict the behavior of the unit. If the models are sufficiently accurate, they can be used to detect unwanted deviations from what is considered to be a normal behavior. Indeed, many different physical models of GTs have been developed in the past (Sarwar et al., 2014). These models capture the dynamics of the turbine, but their precision can significantly vary based on the specific unit that is used. In practice, it is not always possible to have complete mathematical models that describe all of the turbine behaviors accurately. Therefore, another alternative is to develop statistical or computational models that might provide a more accurate characterization of the behavior of such units.

Several authors have developed dynamic mathematical and computational models, as well as specialized test equipment for GTs (Sun et al., 2012). The modeling of GTs for diagnostics and prognostics has been carried out with a variety of techniques, which can be categorized into two major groups: physics-based methods and data-driven methods. The physics-based methods are limited to the cases where failure mechanisms can be quantified (Sun et al., 2012), and require specialized personnel to aid in the modeling process as well as a detailed description of the particular system to be modeled with specific technical information from the manufacturer of the equipment, which are often not available. The data-driven methods attempt to derive the models directly from recorded system data without digging into the physics of the system, using tools derived from artificial intelligence, machine learning, statistical analysis and computational intelligence (Sun et al., 2012). For instance, a popular approach is to use neural networks (DePold and Gass, 1999; Delgadillo and Hernandez, 2002).

This work proposes the use of genetic programming (GP) (Koza, 1992; Langdon and Poli, 2002; Poli et al., 2008) to model a Mitsubishi single shaft Turbo-Generator, Model MS6001, with a 30MW generation capacity. GP has recently gained popularity given its ability to generate syntactic models in a wide variety of problem domains (Koza, 2010), in many cases outperforming more common machine learning paradigms (Trujillo et al., 2012; Castelli et al., 2015c,b,d) while using very little in the way of prior knowledge. While EAs have been used before in GT systems, they are mostly used for optimizing or improving system efficiency using previously derived models (Chaquet et al., 2012). To the extent of our knowledge, this is the first work to address the GT modeling problem directly using a GP approach. Moreover, this work compares recent proposals in GP literature, including the fairly recent geometric semantic GP (GSGP) (Moraglio et al., 2012; Vanneschi et al., 2014a) and improved variants of standard GP (Z-Flores et al., 2014, 2015). The experimental work compares several variants of these methods, particularly emphasizing algorithms that combine the global search process provided by the basic GP paradigm with greedy local search operators. Such combinations have been shown to improve algorithm convergence and predictive accuracy (Castelli et al., 2015e), while also reducing model size and complexity (Z-Flores et al., 2014, 2015). Comparisons are also made with seven non-GP modeling methods, showing that GP search outperforms all other methods in terms of predictive accuracy and model size, particularly on the most difficult modeling task.

The remainder of this work proceeds as follows. Section B.2 describes the power generation process of GTs in general and the particular unit studied in this work, describing how data was collected to apply the data-driven approach. Section B.3 focuses on the specialized variants employed in this work and developed by the authors. The experimental work and results are presented in Section B.4. Finally, conclusions are outlined in Section B.5.

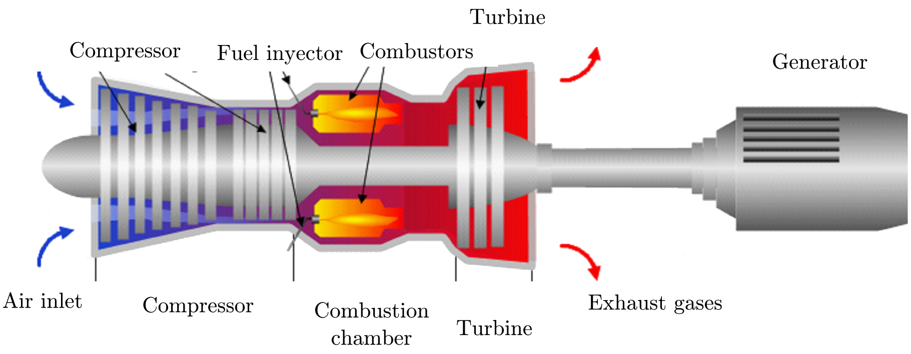


Figure B.1: Graphical depiction of a generic GT system.

B.2 GAS TURBINE OPERATION AND CASE STUDY

This section provides a brief introduction to GTs and how they are used for power generation, as well as a detailed description of how the dataset used in this study was extracted.

B.2.1 *Process Description*

A GT power unit is mainly composed of a starting device, a compressor, a combustion chamber, a turbine, an electric generator and other auxiliary components such as a fuel system, a lube oil system, a starting device and other subsystems. A graphical depiction of a generic GT power unit is shown in Figure B.1.

The starting device is a diesel engine, it assists the GT unit during the startup phase to overcome inertia and to accelerate the GT up to the ignition speed. The centrifugal compressor pressurizes the incoming air and sends it to the combustion chamber. High-pressure combustion exhaust gases are expanded within the turbine stages to rotate the unit. The electric generator that is coupled to the GT can generate the desired

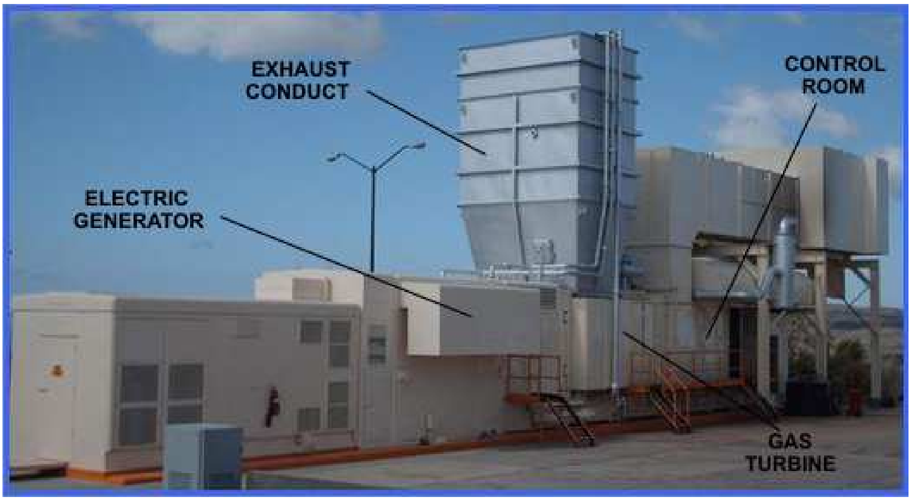


Figure B.2: Turbo-Generator at the Chankanaab Turbo-Gas Central.

electricity when it operated at the rated speed. Finally, the combustion exhaust gases are discharged to the atmosphere (CFE, 1998).

B.2.2 *Case Study*

The case study of the current paper considers a Mitsubishi single shaft Turbo-Generator Model MS6001, which has a 30MW generation capacity. The plant is located in Cozumel, Quintana Roo, in the southeast of Mexico in the complex known as the Chankanaab Turbo-Gas Central (see Figure B.2).

For this model, the combustion zone is formed by 10 combustion chambers and nozzles with their respective baskets. It also has two spark plugs and two flame detectors. In this area the oxygen supplied by the compressor is combined with the diesel fuel delivered by spray nozzles. Afterward, the spark plugs ignite the mixed fuel to produce the required combustion, generating a large volume of gases.

The gases produced by the combustion pass through the turbine which consists of three stages of blades. The power of the exhaust gases is thus converted to mechanical energy. However, the temperature at the turbine inlet is not directly regulated or monitored. The control of the gas turbine is done by monitoring the exhaust temperature, which should not exceed 1067 °F. This temperature is then used to regulate the fuel supply to the turbine.

The liquid fuel system delivers a high pressure flow modulated by a servo controlled bypass valve assembly. The servo valve controls the bypass valve stroke according to the difference requirement and the sensed fuel flow. Fuel oil goes to a flow divider whose purpose is the apportion of fuel oil to each of the fuel nozzles of the turbine. The distribution has ten pump elements (in line) with the inlet port located at the mid-point where the fuel oil enters the unit and is distributed by an internal manifold to the inlet side of each pump element. The flow from each pump element is proportional to the speed at which the unit operates.

Fuel flow is measured by magnetic pickups and represents the feedback signal in the outer control loop of the system. The speed of the flow divider is a direct measure of the fuel flow nozzles in the combustion chambers, as depicted in Figure B.3. The correct measurement of the fuel flow is crucial for a good control performance. Concerning the exhaust gases temperature, it reflects the fuel flow variations; therefore, it is also critical for the control system to maintain a temperature profile that preserves the unit from any thermal excess. In both cases (fuel flow and exhaust temperature), it is also desirable to have accurate predictive models that can be used as monitoring tools. These models can be used to detect deviations over time from the expected behavior of the GT unit, thus anticipating possible failure situations.

The unit is equipped with a control system that offers a user interface to operators. The interface collects data from the system using specialized acquisition modules, providing a historical database for analysis purposes. The monitoring process of this plant can therefore be used to collect a representative dataset related to flow and temperature in the GT. Table B.1 shows the available data related to fuel flow that is provided by the system, while Table B.2 shows the same for the exhaust gas temperature.

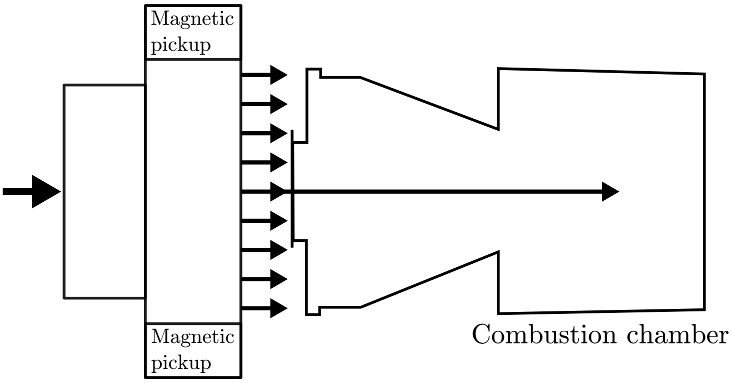


Figure B.3: Fuel flow path in a GT power unit.

Variable	Description	Operating Range	Units
ff	Fuel flow	$[0.0, 61.1]$	GPM
rf	Reference Fuel flow/	$[0.0, 61.1]$	GPM
ts	Turbine speed in %	$[0, 110]$	%
ff_1	Fuel flow at splitter No. 1	$[0.0, 61.1]$	GPM
fp_1	Fuel pressure at splitter No. 1	$[0, 1500]$	PSI
fp_{10}	Fuel pressure at splitter No. 10	$[0, 1500]$	PSI
fdv	Final demand at the valve	$[0, 100]$	%

Table B.1: Monitored variables in the Mitsubishi single shaft Turbo-Generator Model MS6001 that are related to fuel flow.

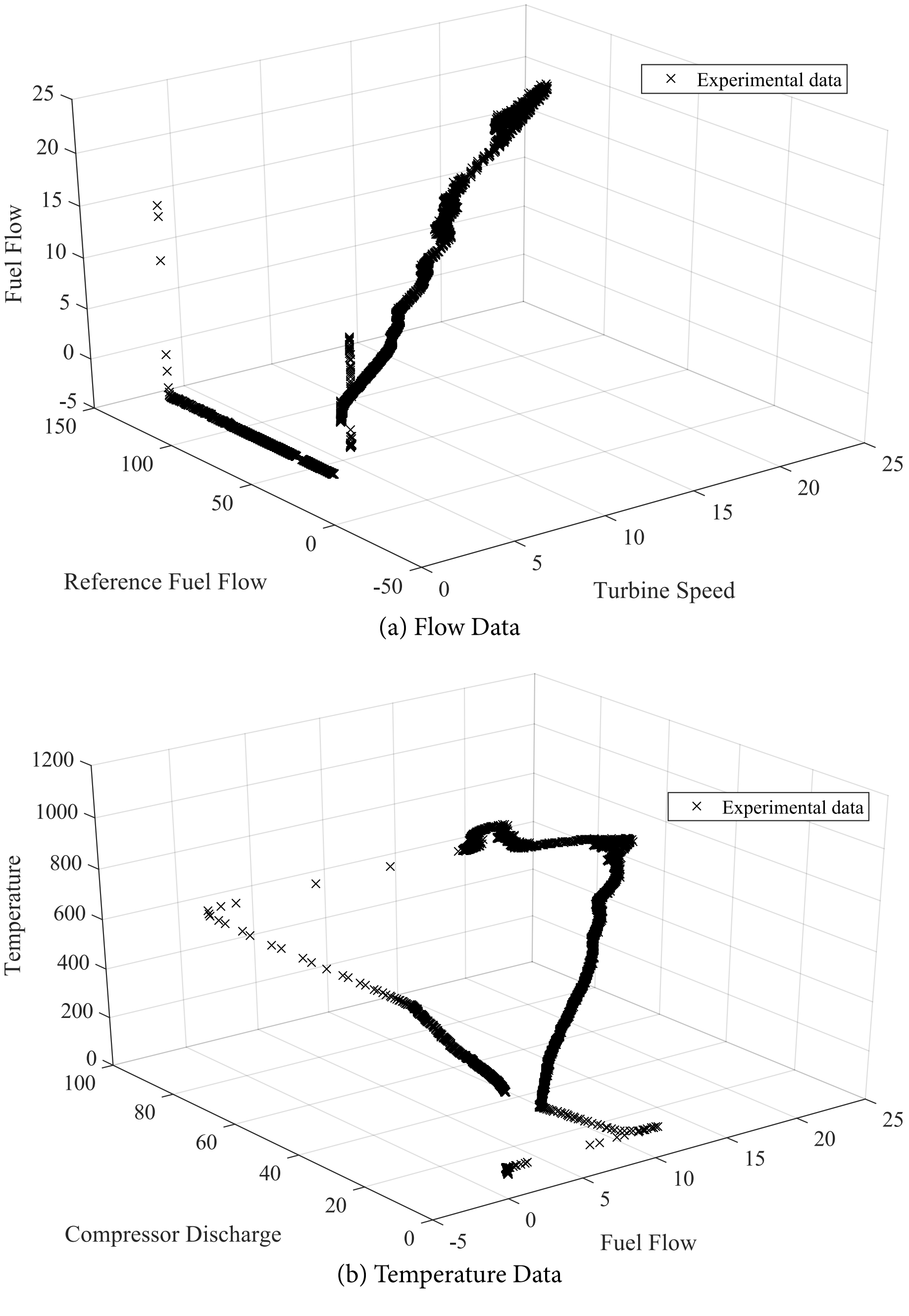
Variable	Description	Operating Range	Units
et	Average of exhaust gas temperature	$[0, 1370]$	°F
cp	Compressor discharge pressure	$[0, 175]$	PSI
ff	Fuel flow	$[0.0, 61.1]$	GPM
al	Alarm limit of exhaust gas temperature	$[0, 1370]$	°F
tl	Trip limit of exhaust gas temperature No. 1	$[0, 1500]$	°F
sp	Speed set point	$[0, 6000]$	RPM
t	Turbine Speed	$[1, 6000]$	RPM

Table B.2: Monitored variables in the Mitsubishi single shaft Turbo-Generator Model MS6001 that are related to exhaust gas temperature.

B.2.3 *Problem Statement and Data*

The goal of this work is to derive predictive models of both the fuel flow and the exhaust gas temperature. In particular, the intention of the fuel flow prediction problem is to find a function K_f that takes the reference fuel flow rf and the turbine speed ts as inputs and produces as output an approximation of the actual fuel flow ff of the GT unit; i.e., $K_F(rf,ts) \approx ff$. Conversely, the goal of the exhaust gas temperature prediction problem is to find a function K_T that takes the fuel flow ff and the compressor discharge pressure cp as inputs and produces as output an approximation of the actual exhaust gas temperature et of the GT unit; i.e., $K_T(ff,cp) \approx et$. In both cases, both K_T and K_F should approximate the fuel flow and the exhaust gas temperature, respectively, within the normal operating ranges of the independent variables rf, ts, ff and cp. In both cases, our intent was to use a small amount of independent variables that could be easily and reliably monitored, while capturing sufficient information to predict the dependent variables. Therefore, in both problems the desired models ($K_F(rf,ts)$,$K_T(ff,cp)$) take two measurements as inputs and produce a single output.

The GT unit was allowed to operate in normal conditions after a control tuning process during the commissioning phase. At this time, the above mentioned independent variables were recorded. In the case of fuel flow prediction, the GT unit operated for approximately 23 minutes, using a sampling period of 100ms, recording a total of 13,788 records (input/output pairs). For the prediction of the exhaust gas temperature, the unit operated for approximately the same amount of time with the same sampling period, producing a total of 13,820 records. These datasets allowed us to pose two supervised learning problems and solve them with GP, as will be described next. Figure B.4 shows a graphical representation of each data set plotted as 3D point clouds, to visualize what is the behavior that needs to be modeled. Notice that in both cases the datasets show highly irregular behavior with some discontinuities.



(a) Flow Data

(b) Temperature Data

Figure B.4: Plots show 3D point clouds of the real data collected from the GT unit.

B.3 GENETIC PROGRAMMING

B.3.1 *Applying GP to Model Real-World Systems*

Providing an exhaustive review of the cases where GP has been used to solve real-world modeling problems is beyond the scope of this work. Our goal is more modest, and provide some recent examples of successful applications of GP to solve complex modeling problems in real-world domains. In (Cao et al., 2000) the authors use a hybrid GP and GA algorithm to derive models based on systems of ordinary differential equations and higher-order differential equations, using several real-world applications to illustrate their approach which is based on tree-based GP. Recently, GP was applied to model the consumption of natural gas consumption in a real-world chemical plant (Kovačič and Dolenc, 2016), outperforming the specialist prediction in minimizing costs to the plant. Another related example to the work presented here is (Arellano et al., 2014), where GP is used to model the exhaust gas temperature of a commercial jet engine, and (Babaelahi and Sayyaadi, 2016) where GP is used to aid in the practical modeling process of Stirling engines. The prediction of fuel consumption for a gasoline engine was addressed with GP in (Togun and Baysec, 2010), achieving very high correlation with the real behavior of the engine and a low predictive error. Modeling the start-up process of an aero-engine is studied in (Li and Wei, 2006), using a specialized GP system to generate linear (in parameter) models, comparing with support vector machines shows that GP models outperform the standard technique and produce smaller and more parsimonious solutions. Similarly, (Cava et al., 2016) presents an approach based on a stack-based GP system for the automatic identification of the dynamics of a wind turbine. The approach posed multi-objective search, deriving models that are as accurate as those produced with competing techniques, whereas are relative more parsimonious, easier to analyze and interpret. Research has also shown that GP models can outperform standard regression techniques in predicting the material characteristics of specialized alloys (Gusel and Brezocnik, 2011), and focus on the fact that since GP is a population-based and stochastic method it can generate a variety of different models that can be considered for further analysis.

B.3.2 *Genetic Programming with Local Search*

In standard GP, only the symbolic expression is evolved by the search process, while the underlying parametrization $\boldsymbol{\theta}$ is not considered. Recent works have shown that concurrently optimizing both the structure and the parametrization of the evolved models can speed up convergence and improve performance. In particular, we adopt our proposal in (Z-Flores et al., 2014, 2015), which has been applied to both symbolic regression and classification problems. The approach can be summarized as follows.

First, as suggested in (Kommenda et al., 2013), for each individual K in the population we add a small linear uppertree above the root node, such that

$$K' = \theta_1 + \theta_2(K)\ , \tag{B.1}$$

where K' represents the new program, while θ_1 and θ_2 are the first two parameters from $\boldsymbol{\theta}$.

Second, for all the other nodes n_k in the tree K we add a weight coefficient $\theta_k \in \mathbb{R}$, such that each node is now defined by

$$n'_k = \theta_k n_k\ , \tag{B.2}$$

where n'_k is the new modified node, $k \in \{3, ..., r\}$, $r = |K|$ which is the total number of nodes (size) of K. Notice that each node has a unique parameter that can be modified to help meet the overall optimization criteria of the non-linear expression. At the beginning of the GP run, each parameter is initialized as $\theta_k = 1$. During the GP syntax search, subtrees belonging to different individuals are swapped, added or removed (following the standard crossover/mutation rules) together with their corresponding parameters, without affecting their values. This follows a memetic search process with Lamarckian inheritance (Z-Flores et al., 2014, 2015), where its main goal is to preserve the information acquired during the local search process and pass it to the following generation. Therefore, we consider each tree as a non-linear expression and the local search operator will search for the best fit parameters of the model K'. The problem can be solved using a variety of techniques, but following (Z-Flores et al., 2014, 2015) we employ a trust region algorithm.

Finally, it is important to consider that the local search optimizer can substantially increase the underlying computational cost of the search, particularly when individual trees are very large. While applying the local search strategy to all trees might produce good results (Z-Flores et al., 2014, 2015), it is preferable to reduce to a minimum the amount of trees to which it is applied. Therefore, we use the heuristic proposed in (Azad and Ryan, 2014), where the local search is applied stochastically based on a probability $p(s)$ determined by the tree size s (number of nodes) and the average size of the population $\bar{s}$, based on

$$p(s) = \begin{cases} 1.5 - \frac{s}{\bar{s}} & \text{if } 0.5\bar{s} \leq s \leq 1.5\bar{s} \\ 1 & \text{if } s < 0.5\bar{s} \\ 0 & \text{otherwise} . \end{cases} \tag{B.3}$$

In this way, smaller trees are more likely to be optimized than larger trees, which also reduces the computational effort and improves the convergence of the trust region optimizer by keeping the parameter vectors relatively small. We will refer to this version of GP as GP-LS.

B.3.3 *Geometric Semantic Genetic Programming*

In the last few years, a rising topic in the field of GP has been the use of what are referred to as semantic methods (Moraglio et al., 2012; Vanneschi et al., 2014a; Castelli et al., 2015f, 2013). Semantics in GP can be loosely defined as a description of what a program actually does, instead of a description of its syntax. Specifically, for supervised learning problems with a training set $\mathbb{T}$ with n input/output pairs $(\mathbf{x}_i, y_i)$, GP literature usually defines the semantics of a program K as the vector of program outputs $\mathbf{s} = \{K(\mathbf{x}_1), ..., K(\mathbf{x}_n)\}$ (Moraglio et al., 2012). From this perspective, the vector of ideal outputs $\mathbf{y} = \{y_1, ..., y_n\}$ can be referred to as the target semantics and the fitness of a program K can be quantified by a distance between vectors $\mathbf{y}$ and $\mathbf{s}$.

In particular, let us consider regression problems where $\mathbf{x} \in \mathbb{R}^p$, and $\mathbf{s}, \mathbf{y} \in \mathbb{R}^n$. Then, the aim of GP is to find a program syntax that produces the target semantics of the problem. However, the search operators of standard GP operate at the syntax level with an indirect effect on the semantics of a program; i.e., standard GP operators cannot control or even constrain what will be the semantics of the offspring based on the semantics of the parents. This makes the GP search process inefficient, in the sense that while fitness is uniquely and directly defined by semantics the mapping between syntax and fitness is much more complex and difficult to predict. Therefore, Moraglio et al. (Moraglio et al., 2012) proposed important new search operators that operate at the level of syntax but have a predictable and bounded behavior on program semantics. This is achieved by imposing geometric constraints within semantic space. These operators are referred to as geometric semantic operators (GSOs) and a GP algorithm that uses them is called geometric semantic GP or GSGP. These operators are defined as follows.

Definition 4 Geometric Semantic Crossover (GSC). *Given two parent functions $K_1, K_2 : \mathbb{R}^n \to \mathbb{R}$, the geometric semantic crossover returns the real function $K_{XO} = (K_1 \cdot K_R) + ((1 - K_R) \cdot K_2)$, where K_R is a random function with normal distribution, domain $\mathbb{R}$ and codomain $[0,1]$.*

Definition 5 Geometric Semantic Mutation (GSM). *Given a parent function $K : \mathbb{R}^n \to \mathbb{R}$, the geometric semantic mutation with mutation step ms returns the real function $K_M = K + ms \cdot (K_{R1} - K_{R2})$, where K_{R1} and K_{R2} are random real functions with normal distributions.*

Using these operators the semantics of the offspring are completely defined by the semantics of the parents. The semantics of an offspring K_{XO} produced by GSC will lie on the segment between the semantics of both parents. On the other hand, GSM defines a mutation such that the semantics of the offspring K_M lies within the n-sphere of radius ms that surrounds the semantics of the parent. Finally, the computational cost of evolving a population of n individuals for g generations is $O(ng)$ (Vanneschi et al., 2014b), while the cost of evaluating a new and unseen instance is $O(g)$ (Vanneschi et al., 2014b). This is the implementation used in this work.

B.3.3.1 *GSGP with Local Search*

While GSGP has been shown to produce strong results (Vanneschi et al., 2014a; Castelli et al., 2015f, 2013), its original formulation has several disadvantages. It is easy to see that GSC will not be useful when the semantics of the parents do not surround the target semantics. Similarly, GSM can sometimes produce offspring that have a worse fitness than the parent, which is unnecessary since the target semantics is known. Moreover, the offspring produced by GSOs will always be larger than the parents, meaning that program growth cannot be eliminated.

This work follows our proposal in (Castelli et al., 2015e), where a local search approach is integrated into GSM, an operator we will refer to as GSM-LS. This operator exploits the fact that the geometric mutation defines a linear combination of the parent K program with random programs, expressed as

$$K_M = \alpha_0 + \alpha_1 \cdot K + \alpha_2 \cdot (K_{R1} - K_{R2}) \tag{B.4}$$

where $\alpha_i \in \mathbb{R}$; notice that α_2 replaces the mutation step parameter ms of GSM. GSM-LS poses a basic multivariate linear regression problem, which could be solved, for example, by Ordinary Least Square regression (OLS). However, in this case we have n linear equations given by the number of fitness cases, and only three unknowns (the α_is). This gives an overdetermined multivariate linear fitting problem, which can be solved through singular value decomposition (SVD) [1]. In this way the GSM-LS will be guaranteed to produce the best linear fit based on the target semantics of the program. Hereafter, we will refer to a GSGP with GSM-LS mutation as GSGP-LS. This version of GSGP has performed very well in a variety of domains (Castelli et al., 2015c,b,d), in many cases outperforming standard GP, GSGP and several other regression techniques.

[1] In this work, the GNU Scientific Library is used `http://www.gnu.org/software/gsl/`

B.4 EXPERIMENTS

This work focuses on applying GP to model a GT unit using a data-driven approach following a supervised learning formulation. Therefore, we will use the datasets described in Section B.2 to pose two symbolic regression problems and solve them with the variants of GP described in Section B.3. Hereafter, we will refer to the problem of modeling the fuel flow of the GT unit as the Fuel problem, and the problem of modeling the exhaust gas temperature as the Temperature problem.

Five different algorithms are compared based on the GP systems described in the previous section; these are:

1. The standard GSGP algorithm as described in (Moraglio et al., 2012) and implemented in (Castelli et al., 2015a);
2. The GSGP-LS algorithm proposed in (Castelli et al., 2015e) and described in the previous section;
3. An algorithm we refer to as HYBRID, that combines GSGP and GSGP-LS, by applying GSGP-LS for the first *l* generations and then switches to GSGP, as suggested in (Castelli et al., 2015e);
4. HYBRID-LIN is the same as HYBRID, but the training data is pre-processed using the linear scaling approach of (Keijzer, 2003).
5. GP-LS is the GP algorithm with LS integrated.

All GSGP variants are implemented using the GSGP-C++ library (Castelli et al., 2015a), while GP-LS is implemented using Matlab and the GPLab toolbox (Silva and Almeida, 2005).

B.4.1 *Experimental Setup*

All algorithms were used to solve both the Flow and Temperature problems, performing 30 independent runs of each. In each case, the problem data was split into a training and testing set, where the former contains 70% of the data samples and the latter the remaining 30%. The

Parameter	GSGP all variants	GP-LS
Population size	200	100
Initialization	Ramped Half & Half	Ramped Half & Half
Max. initial depth	6 levels	6 levels
Crossover rate	0.6	0.9
Mutation rate	0.4	0.05
Reproduction rate	0	0.05
Function set	+,-,×,÷	+,-,×,÷,sin,cos,log,sqrt,tan,tanh,exp
Terminal set	Input variables & random constants	Input variables & random constants
Selection	Tournament of size 6	Tournament of size 7
Elitism	Best individual survives	Best individual survives
Max. tree depth	none	17

Table B.3: Parameters for all algorithms; all GSGP variants (GSGP, GSGP-LS, HYBRID, HYBRID-LIN) share these parameters.

datasets were randomly partitioned before each run and, while training performance will be reported, the critical result will be the prediction accuracy of the evolved models over the test data. Table B.3 summarizes the parameters for all tested algorithms. For GSGP variants mutation and crossover rates have been chosen after a preliminary tuning phase that has been performed following the indication reported in (Castelli et al., 2015g). GP-LS parameters were set based on the values reported in (Z-Flores et al., 2014, 2015), with only minimal manual tuning. Values for other parameters were selected based on author's suggestions and library default values. For the stopping criterion all GSGP variants were executed for a maximum of 300 generations. On the other hand, the stopping criterion for GP-LS is 100 generations. Notice also that the function sets in both cases are different, since we follow the best reported results for each algorithm. For HYBRID and HYBRID-LIN $l = 10$ as in (Castelli et al., 2015e,c). Finally, in GSGP variants, when the standard GSM mutation is used (instead of GSM-LS), the mutation step is randomly set at each mutation event in the range $[0, 1]$.

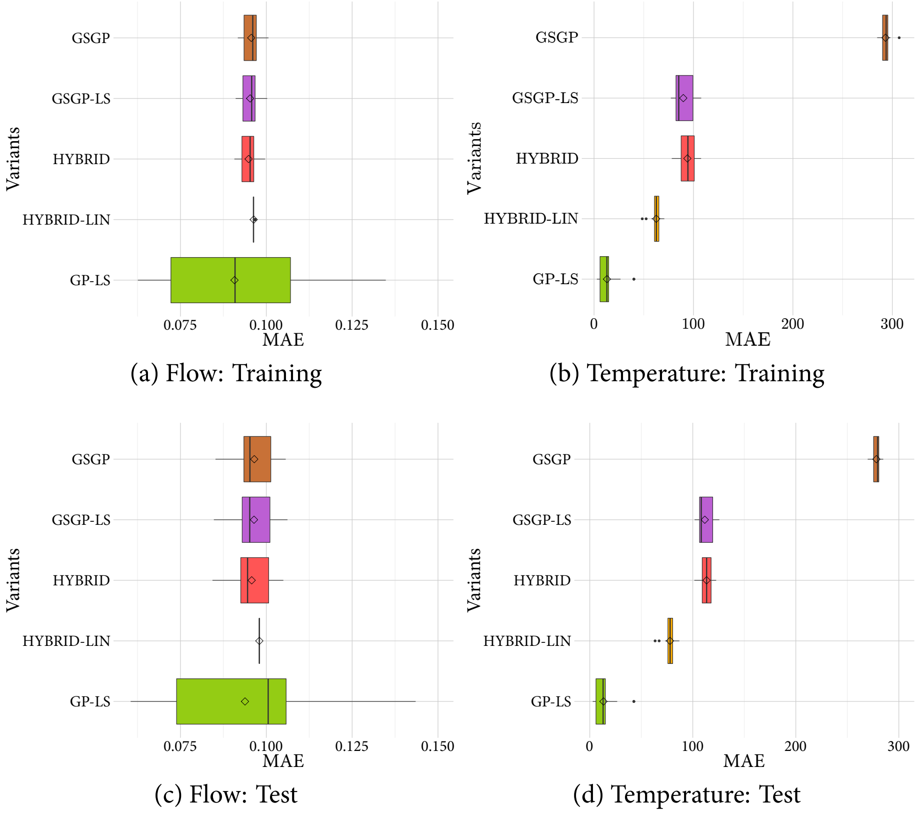
It is important to mention that in all cases fitness was determined using the MAE between the real output and the estimated output of each problem when evaluated over the training set. Moreover, in all subsequent comparisons, performance will be measured using the MAE, computed on the training or testing data.

B.4.2 *Results and Analysis*

Figure B.5 presents a summary of all results as bloxplots that show the median, first and second quartiles, as well as the maximum and minimum performance over all thirty runs for both problems. Figure B.5(a) shows the training error on the Flow problem and Figure B.5(b) shows the same for the Temperature problem. Conversely, Figure B.5(c) shows the test error on the Flow problem while Figure B.5(d) shows the same for the Temperature problem.

Moreover, Table B.4 summarizes these results numerically, reporting the median train and test performance of each algorithm on each problem, showing in bold the best (lowest) results. What immediately stands out from these results is that GP-LS outperforms all other algorithms in all cases except one, testing performance on the Flow problem, however there are runs that still are better than the other algorithms. Also notice that while performance on the Flow problem is very similar for all algorithms, on the Temperature problem performance varies quite substantially, with GSGP performing the worst and GP-LS performing the best. Another relevant observation is related to the variance in performance of each method over all runs. First, on the Flow problem, at one extreme HYBRID-LIN exhibits very stable behavior in all runs, suggesting that the linear scaling approach increases the robustness of GP search. The other extreme is GP-LS, which shows the largest variance on this problem, with several runs achieving substantially smaller error values than the other methods. Second, on the Temperature problem, all methods exhibit comparatively less variance, even GP-LS behaves quite consistently over all runs.

To test the significance of these results, we use the Friedman test to perform all pairwise comparisons of the methods, focusing specifically on test performance for both problems. The null hypothesis in each pairwise comparison is that the groups have equal medians, and the p-values



(a) Flow: Training

(b) Temperature: Training

(c) Flow: Test

(d) Temperature: Test

Figure B.5: Experimental results showing the train (a,b) and test (c,d) performance of all algorithms on each problem, Flow (a,c) and Temperature (b,d).

Algorithm	Flow Train	Flow Test	Temperature Train	Temperature Test
GSGP	0.0961	0.0952	293.75	279.36
GSGP-LS	0.0957	0.0952	85.18	108.23
HYBRID	0.0953	0.0945	94.41	113.41
HYBRID-LIN	0.0963	0.0980	62.72	77.97
GP-LS	0.0909	0.1006	12.94	12.92

Table B.4: Summary of median performance by each algorithm on each problem.

	GSGP	GSGP-LS	HYBRID	HYBRID-LIN	GP-LS
GSGP	-	0.7150	0.0000*	0.1358	0.715
GSGP-LS	-	-	0.0000*	0.1358	0.715
HYBRID	-	-	-	0.1358	0.715
HYBRID-LIN	-	-	-	-	0.715
GP-LS	-	-	-	-	-

Table B.5: Summary p-values of the pairwise Friedman test with Benjamini-Hochberg correction for the test performance on the Flow problem; an asterisk (*) indicates that the null hypothesis can be rejected at the $\alpha = 0.05$ significance level.

	GSGP	GSGP-LS	HYBRID	HYBRID-LIN	GP-LS
GSGP	-	0.0000*	0.0000*	0.0000*	0.0000*
GSGP-LS	-	-	0.4652	0.0000*	0.0000*
HYBRID	-	-	-	0.0000*	0.0000*
HYBRID-LIN	-	-	-	-	0.0000*
GP-LS	-	-	-	-	-

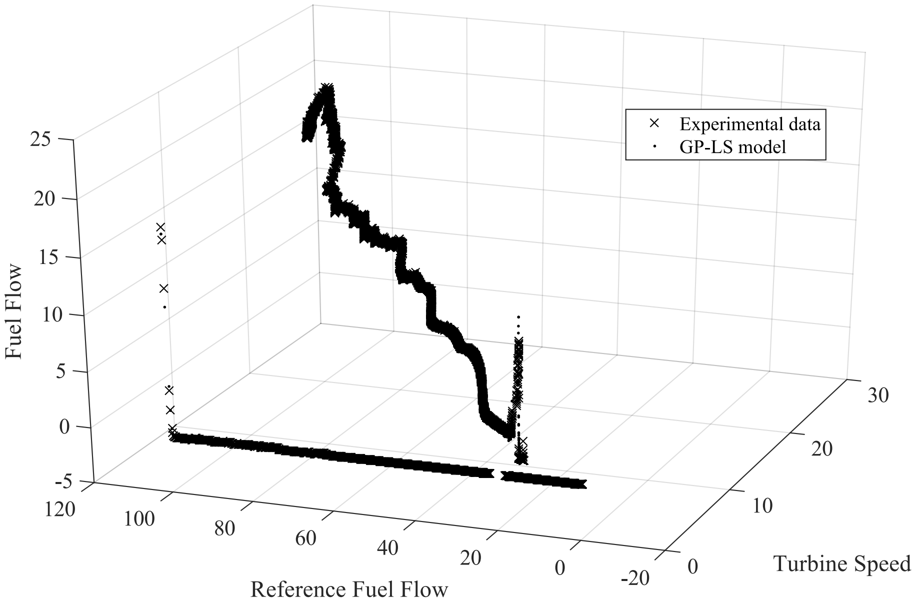
Table B.6: Summary p-values of the pairwise Friedman test with Benjamini-Hochberg correction for the test performance on the Temperature problem; an asterisk (*) indicates that the null hypothesis can be rejected at the $\alpha = 0.05$ significance level.

are corrected using the Benjamini-Hochberg algorithm. The resulting p-values are given in Tables B.5 and B.6, the former for the test performance on the Flow problem and the latter for the test performance on the Temperature problem. In both tables an asterisk (*) indicates that the null hypothesis can be rejected at the $\alpha = 0.05$ significance level.

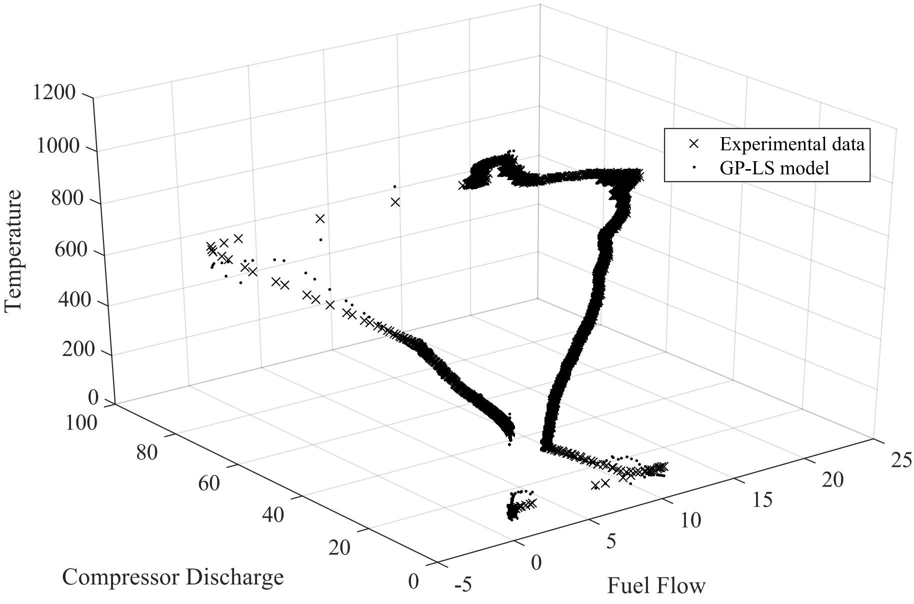
This statistical comparison suggest the following. On the Flow problem, basically all methods exhibit statistically similar results, since the null hypothesis was not rejected in all pairwise comparisons except two. In particular, the test suggests that a statistically significant difference exists between HYBRID and GSGP, and HYBRID and GSGP-LS. While the boxplots of all three methods appear quite similar (see Figure B.5), under further inspection the distribution of the results of GSGP and GSGP-LS are actually bi-modal, which explains the rejection of the null hypothesis. On the other hand, on the Temperature problem it is clear that GP-LS exhibits performance that is significantly different with respect to all other methods, confirming its superiority on this problem. Taken as a whole, we can say that GP-LS is the best method, since its median performance is not statistically different with respect to other methods on the Flow problem, while achieving the best performance on the Temperature problem.

Finally, it is important to comment on the size of the evolved models, which for GP-based systems is measured by the number of nodes of the best solution found. In particular, GSGP is known to produce extremely large models, the size of the models grows exponentially with the number of generation (Moraglio et al., 2012). This means that GSGP solutions are basically black boxes. On the other hand, one of the goals of GP is to produce models that can be interpreted or easily implemented in real-world systems. In this case, GP-LS produces vastly more compact solutions. The median size of the evolved models are 33 and 36 nodes for the Flow and Temperature problems, respectively.

Figure B.6 presents a graphical depiction of the best models evolved by GSGP-LS on each problem, showing 3D point clouds of the real data compared with the predicted values of the GSGP-LS models over the entire dataset. The GP-LS models were chosen because they achieved the minimum error (see the boxplots of Figure B.5) and because the models are the most compact, which is desirable in a real-world deployment of the models. Notice that in both cases the models are able to closely approximate the real behavior of the GT unit, accurately predicting the behavior of both variables of interest: fuel flow and exhaust temperature.



(a) Flow Data



(b) Temperature Data

Figure B.6: Plots show 3D point clouds of the real data collected from the GT unit and the output of the best models produced by the GP-LS method.

The actual best symbolic model obtained by the GP-LS technique for the Flow problem is given by

$$\begin{aligned} f_F =&((\theta_1*(((\theta_2(X_1))+(((\theta_3((log((\theta_4((log((\theta_5((sin((\theta_6((sin(((\theta_7(X_1))/\\ &((\theta_8((log((\theta_9((log((\theta_{10}(X_2)))))))))))/(\theta_{11}(X_1)))))))))))))))))))))*\\ &((\theta_{12}((log((\theta_{13}(X_2)))))))+(((\theta_{14}((log((\theta_{15}((log((\theta_{16}(X_2))))))))))+\\ &((\theta_{17}((log((\theta_{18}((log((\theta_{19}((log((\theta_{20}((log((\theta_{21}(X_2))))))))))))))))))+\\ &(\theta_{22}((log((\theta_{23}((log((\theta_{24}((log((\theta_{25}(X_2)))))))))))))))+(\theta_{26}(X_2)))))+\\ &(\theta_{27}((exp((\theta_{28}((log(((\theta_{29}(X_2))+((\theta_{30}(X_1))/(\theta_{31}(X_2))))))))))))))-\\ &(\theta_{32}(X_1))))+\theta_{33})\,, \end{aligned} \tag{B.5}$$

where f_F is the fuel flow function, $\theta_1 = 0.1377$, $\theta_2 = -2.7002$, $\theta_3 = 0.5654$, $\theta_4 = -17.8777$, $\theta_5 = 1.0151$, $\theta_6 = -2.0230$, $\theta_7 = -0.0188$, $\theta_8 = 2.1919$, $\theta_9 = 4.8047$, $\theta_{10} = 0.0550$, $\theta_{11} = -8.2834$, $\theta_{12} = -0.1644$, $\theta_{13} = -0.0255$, $\theta_{14} = -7.4487$, $\theta_{15} = 4.7472$, $\theta_{16} = 0.0472$, $\theta_{17} = -0.1766$, $\theta_{18} = 4.8017$, $\theta_{19} = 0.5157$, $\theta_{20} = 21.6877$, $\theta_{21} = 0.0480$, $\theta_{22} = 14.5261$, $\theta_{23} = 1.4830$, $\theta_{24} = 3.1747$, $\theta_{25} = 0.0689$, $\theta_{26} = 0.0085$, $\theta_{27} = 2.1211$, $\theta_{28} = 758.9091$, $\theta_{29} = -0.0101$, $\theta_{30} = 1.6733$, $\theta_{31} = 0.3230$, $\theta_{32} = -9.9588$, $\theta_{33} = -0.0007$. X_1 represents the reference fuel flow and X_2 the turbine speed.

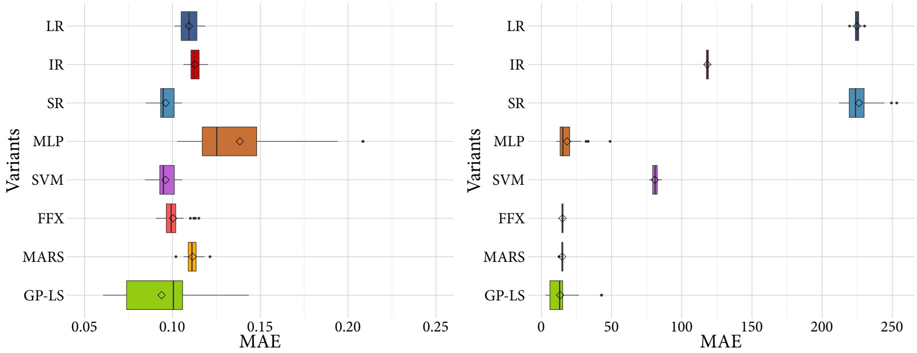
On the other hand, the model corresponding to the Temperature variant is given by

$$\begin{aligned} f_T =&((\theta_1*(\theta_2((log((((((\theta_3((tan((((\theta_4((log((\theta_5(X_2))))))*\\ &(\theta_6((exp((\theta_7(X_2))))))*(\theta_8((exp((\theta_9(X_1))))))))))+\\ &(\theta_{10}((exp((\theta_{11}((sin((\theta_{12}((sin((\theta_{13}(X_1))))))))))))))/\\ &((\theta_{14}((tan((((\theta_{15}((log((\theta_{16}(X_2))))))*(\theta_{17}((exp((\theta_{18}(X_2))))))*\\ &(\theta_{19}((exp((\theta_{20}(X_1)))))))))))/(\theta_{21}((tan((((\theta_{22}(X_2))*\\ &(\theta_{23}((exp((\theta_{24}(X_2)))))))*(\theta_{25}((exp((\theta_{26}(X_1)))))))))))))*\\ &(\theta_{27}(X_2)))/((\theta_{28}((exp((\theta_{29}((sin((\theta_{30}(X_1))))))))))/\\ &(\theta_{31}((tan((((\theta_{32}((log((\theta_{33}(X_2))))))*\\ &(\theta_{34}((exp((\theta_{35}(X_2)))))))*(\theta_{36}((exp((\theta_{37}(X_1))))))))))))))))))+\theta_{38})\,, \end{aligned} \tag{B.6}$$

where f_T is the temperature function, $\theta_1 = 5.7509$, $\theta_2 = 5.2602$, $\theta_3 = 16.2779$, $\theta_4 = 9.2538$, $\theta_5 = 1.0549$, $\theta_6 = -4.4801$, $\theta_7 = -1.2692$, $\theta_8 = 0.6149$, $\theta_9 = -0.3637$, $\theta_{10} = -15.6936$, $\theta_{11} = 2.6504$, $\theta_{12} = 99.5816$, $\theta_{13} = -0.0022$, $\theta_{14} = 9.3046$, $\theta_{15} = 51.3792$, $\theta_{16} = 0.7228$, $\theta_{17} = -179.6383$, $\theta_{18} = -7.6820$, $\theta_{19} = 0.0430$, $\theta_{20} = 1.1433$, $\theta_{21} = 10.0145$, $\theta_{22} = -77.0285$, $\theta_{23} = 562.4549$, $\theta_{24} = -0.3169$, $\theta_{25} = 562.6132$, $\theta_{26} = -2.1891$, $\theta_{27} = -6.0626$, $\theta_{28} = 0.3125$, $\theta_{29} = 3.2894$, $\theta_{30} = 0.4170$, $\theta_{31} = 12.0751$, $\theta_{32} = 254.8610$, $\theta_{33} = 0.7083$, $\theta_{34} = -0.5231$, $\theta_{35} = -7.5917$, $\theta_{36} = 310.6381$, $\theta_{37} = 3.5085$, $\theta_{38} = -2.6837$. $X1$ represents the compressor discharge and X_2 the fuel flow.

B.4.3 *Comparison with other Machine Learning methods*

In order to get a better contextualization of the previous results, we compare the best GP method (GP-LS) with other common techniques for data-driven modeling taken from machine learning literature. In particular the following methods from the Weka open source library (Hall et al., 2009) have been considered: (1) Linear regression (LR); (2) Isotonic Regression (IR); (3) Square Regression (SR); (4) Multilayer Perceptron (MLP); and (5) Support Vector Machines (SVM) with polynomial kernel. In all cases, we use the default configurations for each method except for MLP, in which we performed a parameter sweep of the decay learning rate and the optimal number of hidden nodes. For both problems the learning rate was left in the default one, while the number of neurons in the hidden layer was set to 28 for the Flow problem and to 8 for the Temperature problem performing a greedy hill climber search. Moreover, we also performed a comparison with the Fast Function Extraction algorithm (FFX) (McConaghy, 2011) and Multivariate Adaptive Regression Splines (MARS) (Friedman, 1991). Generally speaking, FFX and MARS build a linear combination of basis functions to construct the model that best describes the training data of a supervised learning problem. On the one hand, FFX was developed as an alternative to the evolutionary search of standard GP but with the same general goals, while MARS is a well established method that continues to be widely



(a) Flow: Test

(b) Temperature: Test

Figure B.7: Experimental results showing the train test performance of all algorithms on each problem, Flow (a) and Temperature (b).

used in difficult real-world domains (Zhang and Goh, 2014; Pramila and Mahesh, 2015; Vijayaraghavan and Veezhinathan, 2015). Both of these techniques are similar to GP in the sense that their goal is to produce a new symbolic expression that best fits the data, so we will also compare these methods with GP-LS based on the size of the models each method produces. For both MARS[2] and FFX[3], we used the freely available toolboxes provided by the authors using their default configuration.

Figure B.7 shows a boxplot comparison between all methods based on the test performance on both problems. Table B.7 compares the test performance of all of the above methods with GP-LS on both problems. Moreover, we apply the same statistical pairwise test as before ans report the results in Table B.8 for the Flow problem and in Table B.9 for the Temperature problem. As before, there is little difference between the methods on the Flow problem, with the statistical tests suggesting that the best performance is achieved by SR, SVM, FFX and GP-LS. On the other hand, on the Temperature problem two methods stand out: MARS and GP-LS achieve the lowest median performance and the statis-

[2] http://www.cs.rtu.lv/jekabsons/regression.html
[3] http://trent.st/ffx/

Algorithm	Flow Test	Temperature Test
LR	0.1095	224.96
IR	0.1122	118.16
SR	0.0947	223.80
MLP	0.1493	15.23
SVM	0.0948	80.98
FFX	0.0992	15.03
MARS	0.1111	14.81
GP-LS	0.1006	12.92

Table B.7: Summary of median performance by each algorithm on each problem, comparing GP-LS with other machine learning methods.

	LR	IR	SR	MLP	SVM	FFX	MARS	GP-LS
LR	-	0.0068*	0.0000*	0.0000*	0.0000*	0.0000*	0.0129	0.0035*
IR	-	-	0.0000*	0.0001*	0.0000*	0.0000*	0.5010	0.0001*
SR	-	-	-	0.0000*	0.1104	0.0046*	0.0000*	1.0000
MLP	-	-	-	-	0.0000*	0.0000*	0.0004*	0.0000*
SVM	-	-	-	-	-	0.0015*	0.0000*	1.0000
FFX	-	-	-	-	-	-	0.0000*	0.3061
MARS	-	-	-	-	-	-	-	0.0000*
GP-LS	-	-	-	-	-	-	-	-

Table B.8: Summary p-values of the pairwise Friedman test with Benjamini-Hochberg correction for the test performance on the Flow problem; an asterisk (*) indicates that the null hypothesis can be rejected at the $\alpha = 0.05$ significance level.

tical test suggesting that there is no significant difference between their median performance. However, if we analyze the boxplots of Figure B.7 we can see that the lowest error was produced by GP-LS.

Finally, it is useful to compare the size of the models produced by FFX, MARS and GP-LS, since these methods are aimed at producing new symbolic expressions that describe the problem data. In this task, it is always preferable to generate the smallest or most parsimonious solutions, for ease of interpretation. Figure B.8 presents boxplots that compare the size of the solutions produced by each method over all runs. For a fair comparison, we express the models produced by MARS and FFX as syntax trees, and define the size of the models as the total number of nodes. On the Flow, we can see that the smallest solutions are produced

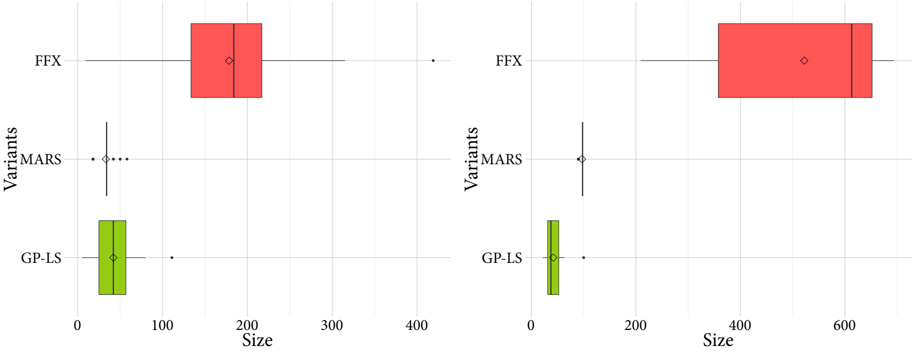
	LR	IR	SR	MLP	SVM	FFX	MARS	GP-LS
LR	-	0.0000*	0.7700	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
IR	-	-	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
SR	-	-	-	0.0000*	0.0000*	0.0000*	0.0000*	0.0000*
MLP	-	-	-	-	0.0000*	1.0000	1.0000	0.0346*
SVM	-	-	-	-	-	0.0000*	0.0000*	0.0000*
FFX	-	-	-	-	-	-	0.5210	0.0346*
MARS	-	-	-	-	-	-	-	0.0792*
GP-LS	-	-	-	-	-	-	-	-

Table B.9: Summary p-values of the pairwise Friedman test with Benjamini-Hochberg correction for the test performance on the Flow problem; an asterisk (*) indicates that the null hypothesis can be rejected at the $\alpha = 0.05$ significance level.

by GP-LS and MARS, but it is important to remember that GP-LS outperforms MARS in terms of test error. Moreover, while FFX achieved similar performance to GP-LS in terms of test error, the models it generates are one order of magnitude larger. Similarly, on the Temperature problem the smallest solutions are generated by GP-LS. This is particularly important since it outperforms all other methods in terms of test MAE, and while MARS achieves similar performance it is evident that it does so at the cost of generating less parsimonious solutions.

B.5 CONCLUSIONS AND FUTURE WORK

This work addresses the problem of modeling key aspects of a GT power system, namely the fuel flow of the system and the temperature of the exhaust gas. Both variables are critical to the proper functioning and performance of the GT power generator. In particular, this paper describes a data-driven approach to generate the models using GP, a paradigm of evolutionary computation for automatic program induction. Recent state-of-the-art variants of GP are tested. These variants integrate a powerful local search mechanism (GP-LS) or are based on geometric semantic concepts (GSGP). To pose the supervised learning problem, real data taken from a power plant in Mexico is used, resulting in a real-world difficult task. Comprehensive experimental valida-



(a) Flow: Size

(b) Temperature: Size

Figure B.8: Experimental results showing the size of the solutions generated by GP-LS, MARS and FFC on both problems: (a) Flow; and (b) Temperature.

tion clearly shows that GP-LS outperforms all other GP variants on both problems, with clear statistical significance in the detected differences among the methods.

While recent literature has emphasized the power of geometric semantic versions of GP, these results show that standard GP algorithms can still be used to address difficult real-world problems with strong results, while also evolving much smaller solutions. This is achievable by integrating local search strategies into the search process, allowing standard GP operators to focus on evolving the structure of the solution and allowing the fine-tuning of these models to be performed by standard numerical methods. These results are consistent with what we showed in previous works using community benchmarks.

Future work will be devoted towards integrating the evolved models into an online monitoring system, to be used by engineers and researchers working at the power plant. The goal is to develop an automatic fault detection system, that can allow for easy monitoring and control to avoid possible damage or losses in power service delivered by the plant.

C

CASE STUDY: RF-PAS MODELING

Based on Cárdenas Valdez, J. R., Z-Flores, E., Núñez Pérez, J. C., and Trujillo, L. (2017). Local Search Approach to Genetic Programming for RF-PAs Modeling Implemented in FPGA. In *NEO 2015*, Studies in Computational Intelligence, pages 67–88. Springer

C.1 INTRODUCTION

An amplifier circuit is designed to increase the power levels of an input signal, and is the most important component of a Radio Frequency (RF) link. There are two types of amplifiers in the RF domain that are: the power amplifier (PA) and low noise amplifier (LNA). The PA is mainly present in the transmission stage and is designed to increase the power level of the signal before to be transmitted by the antenna. This increased capacity is crucial to achieve the desired signal-to-noise ratio in the receiver side. Unfortunately, the PA is intrinsically a nonlinear system that generates four typical unintended consequences: spectral regrowth, memory effects, inter-modulation products and adjacent channel interference. This signal degradation becomes severe when multilevel modulation schemes signal are used (Biglieri et al., 1988).

Due to rapid growth and high demand for data transmission, the RF spectrum availability is becoming increasingly scarce. Additionally, there is a concern related to the requirements obliged to use higher frequencies for signals with digital modulation systems such as wireless broadband communications, for example, Wide Code Division Multiple Access (WCDMA), Worldwide Interoperability for Access Microwave

(WiMAX), and others. In such systems, the transmitter chain is included within a PA as the main element in the amplification process, however, its inherent nonlinearity causes distortion to the operating band intermodulation products and expands the spectrum allocated, infringing adjacent channels (Kenington, 2000).

The development of pre-distorters is now facilitated by the availability of a variety of behavioral models of PAs, mainly based on the Volterra series, which considers undesirable effects such as memory and nonlinearity in order to describe the nonlinearity of a PA. However, due to the high computational complexity of Volterra series it is impractical in some real applications due to the number of parameters to be estimated (Xiaofang et al., 2009; Schetzen, 2006). The memory polynomial model (MPM) is used to overcome these limitations by reducing the processing time for computational modeling compared to a full Volterra series (Bennadji, 2006; Pedro and Maas, 2005; Asbeck et al., 2002). Furthermore, it has been shown that the MPM can reduce internal iterations during the shaping step, without losing the accuracy of the model identification (Ku and Kenney, 2003).

The basic RF-PA modeling techniques involves special truncations of the Volterra Series (Li and Wang, 2010; Moon et al., 2012; Misic et al., 2014), but for very complex nonlinear models with wide data range is required the use of the whole Volterra Series (Staudinger et al., 2010; Zhu and Brazil, 2004), Fuzzy Neural Network (Zhai et al., 2010; Mkadem et al., 2010) or GP (Zhang Sheng et al., 2010; Patelli and Ferariu, 2011). By other hand, GP performs an evolutionary search within the space of possible program syntaxes, achieving the expression that best solves a given model. GP can be viewed as a biologic evolutionary inspired algorithm where a pool of symbolic expressions are built in a synergy fashion upon a target. Each expression competes for survival at each iteration by measuring its fitness value. This is usually expressed by an error metric toward the objective. In general, each symbolic expression consist of a mathematical equation that represent a potential candidate model in the imposed problem. Standard GP can solve complex problems by searching in the syntax space, however accuracy on the solutions can be stagnated through the evolution and expressions might grow in size.

In recent years, a few outsourcing protocols have been proposed for hardware implementation FPGA based (Alasady and Ibnkahla, 2008; Xiong et al., 2008). This work implements in hardware behavioral models based on genetic programming (GP) and MPM. The results are then compared based on normalized mean squared error (NMSE), mean squared error (MSE), mean absolute error (MAE) and Correlation coefficient. The experimental setup is based on the RF-PA Doherty 7W @ 2.11 GHz. It is important to note, that to the authors knowledge this is the first work to use GP to derive models for such a PA. Moreover, we propose the use of a recent variant of GP that incorporates numerical methods as local search methods, which greatly enhance the performance of GP (Z-Flores et al., 2014, 2015).

The remainder of this work proceeds as follows. Section C.2 describes the AM/AM and AM/PM conversions, describing how data was collected to apply the data-driven approach. Section C.3 describes the Volterra Series and MPM as special case of the Volterra Series. Section C.4 provides a general overview of GP, focusing on the specialized variants employed in this work. The experimental work and results are presented in Section C.5. Finally, conclusions and future work are outlined in Section C.6.

C.2 AM/AM AND AM/PM CONVERSIONS

The AM/AM distortion curve of a nonlinear system is the relationship between the output power magnitude of the system and the input magnitude. The AM/PM distortion curve for the linear system is the ratio of the phase shift between the output and the input power magnitude. The figure C.1 shows a general block diagram of a PA based on distortion curves.

A pass-band input signal, can be represented by equation C.1.

$$x(n) = r(n)cos(\omega_c n + \varphi(n)), \tag{C.1}$$

where ω_c is the center carrier frequency, $r(n)$ is the amplitude of the signal, $\varphi(n)$ is the modulated phase. Assuming the previous pass-band

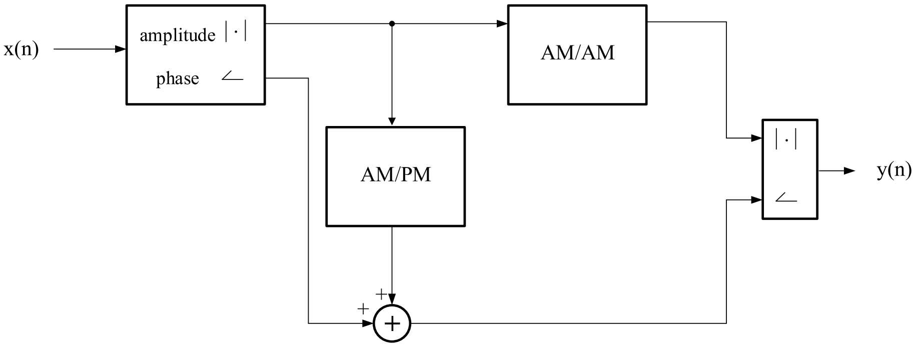


Figure C.1: General overview of a RF-PA based on distortion curves.

input signal, the output of the AM/AM and AM/PM model $Y_{PB}(n)$ can be denoted as equation C.2.

$$Y_{PB} = g(r(n))cos(\omega_0 n + \varphi(n) + f(r(n)), \tag{C.2}$$

where $g(r(n))$ is the amplitude nonlinearity or AM/AM conversions, $f(r(n))$ is the phase nonlinearity or AM-PM conversion.

C.3 VOLTERRA SERIES

Volterra series are a combination of linear convolution and a series of nonlinear powers which can be used to describe the relationship between input/output of a nonlinear causal time invariant system with memory order.

The main disadvantage of Volterra series is the increase in the number of parameters needed regarding the nonlinearity order and memory depth; as a consequence it has a drastic increase of the complexity in the step of identifying parameters. As the bandwidth of the input signal increases, the time span memory PA becomes comparable to variations in time of the envelope of the input signal. The general Volterra series are expressed as following:

$$y(n) = \sum_{m_1=0}^{M} h_1(m_1)x(n-m_1) + \sum_{m_1=0}^{M}\sum_{m_2=0}^{M} h_2(m_1,m_2)x(n-m_1)x(n-m_2)+\ldots$$
$$+\sum_{m_1=0}^{M}\sum_{m_2=0}^{M}\sum_{m_3=0}^{M} h_3(m_1,m_2,m_3)x(n-m_1)x(n-m_2)x(n-m_3)+\ldots, \tag{C.3}$$

where $x(n)$ represents the complex signal input baseband, and $y(n)$ is the complex baseband signal output, $h_n m_n$ the coefficients of the Volterra series, n is the order of the nonlinearity and m is the depth of system memory. As we can see, the number of coefficients of the Volterra series increases exponentially with the size of the memory system and the order of the nonlinearity (Lei, 2004).

C.3.1 *Memory Polynomial Model*

A review of the current research related to the RF-PA modeling considering memory effects, mainly based MPM as a special case of the Volterra series, give us that it is widely understood that MPM is a popular way to make the modeling work reducing coefficients required for this process, mainly obtained by the method of least square error (Rawat et al., 2013; Liu et al., 2014).

The MPM is a subset of the Volterra series, it consists of several stages of delay and nonlinear static functions, representing a truncation of the general Volterra series considering only the diagonal terms of the kernels (Ku and Kenney, 2003). Therefore, the number of parameters is significantly reduced compared to the original series. MPM considers is expressed as:

$$y(n) = \sum_{q=0}^{Q}\sum_{k=1}^{K} a_{2k-1,q}|x(n-q)|^{2(k-1)}x(n-q), \tag{C.4}$$

where $x(n)$ is the input signal, $y(n)$ is the output signal, $a_{k,q}$ are the Volterra Kernels, Q is the memory depth and K the nonlinearity order.

The output system based on memory steps can be as equation C.5:

$$y(n) = \sum_{q=0}^{Q} F_q(n-q) = F_0(n) + F_1(n-1) + \cdots + F_q(n-q) + \cdots F_Q(n-Q), \tag{C.5}$$

where $F_q(n)$ is given by

$$F_q = \sum_{q=0}^{Q} a_{2k-1,q}|x(n-q)|^{2(k-1)}x(n), \tag{C.6}$$

and $F_q(n)$ can be expressed as extended form as

$$F_q(n) = a_{1,q}x(n) + a_{3,q}|x(n)|^2x(n) + a_{5,q}|x(n)|^4x(n) + \cdots + a_{2k-1,q}|x(n)|^{2(k-1)}x(n). \tag{C.7}$$

Each stage is subdivided into MPM terms based on the sampling time of the input signal. The figure C.2 shows the internal structure and the delay made for each step and the overview of this model can be noted that phase shift of the signal during the first cycle is increased.

In the figure C.3, equation C.7 is represented for each F_q stage.

C.4 GENETIC PROGRAMMING WITH LOCAL SEARCH

Being previously stated that the GP goal is to search for syntactic expressions that perform some form of computation, attempting to find the relation between a set of independent variables (inputs) and dependent variables (outputs), but recognized that the canonical version of GP suffers from some shortcomings. That is the search operators perform at syntax level, blind to the effects of the solutions output. For that, the incorporation of LS techniques is beneficial to the search by accelerating the overall quality convergence producing smaller and better solutions. Moreover, in this work we follow the same strategies presented in

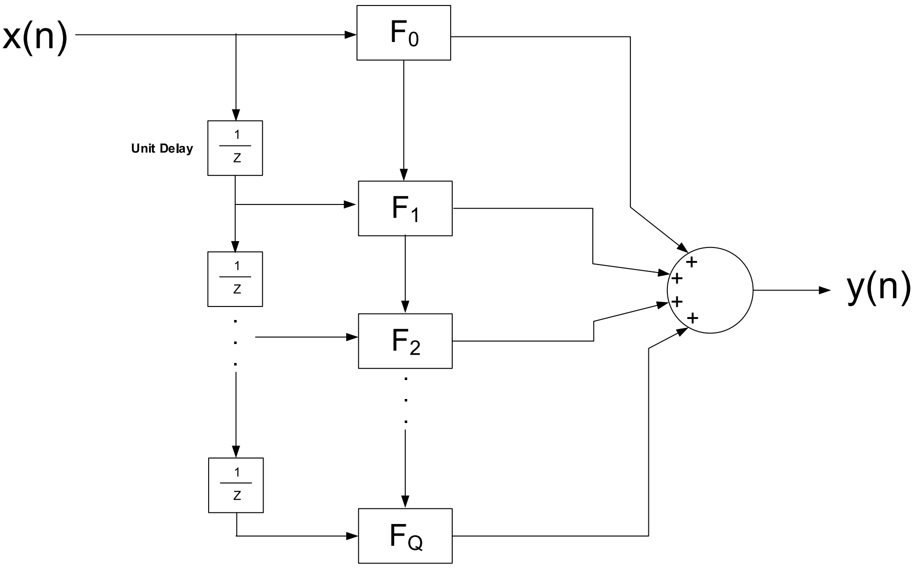


Figure C.2: Overview of $F_q(n)$ as block diagram.

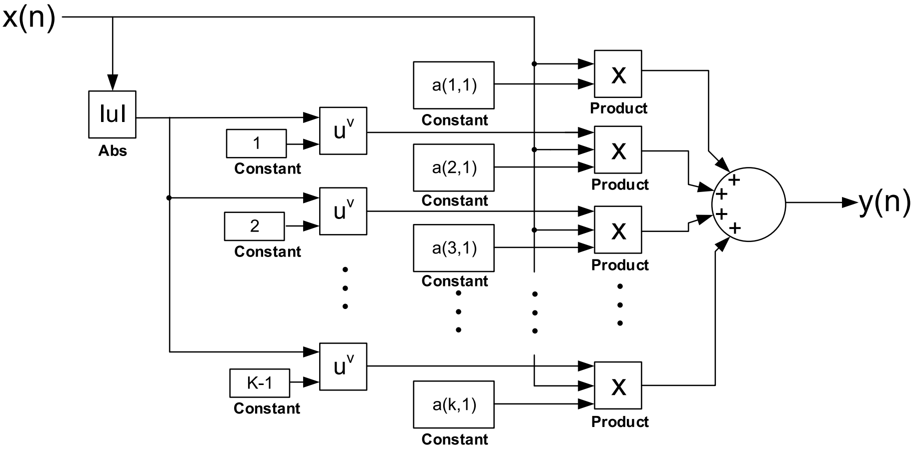


Figure C.3: Description of each stage of F_q with the Volterra Kernels.

appendix A in terms of the methodology for the LS incorporation into GP.

C.5 EXPERIMENTATION

This work will focus on applying MPM and GP-LS to model the behavior of a real RF-PA. Therefore, we will use the datasets based on the table C.1 that describes the behavior of a Doherty PA. The figures C.4 and C.5 describe the AM/AM and AM/PM conversion curves, the AM/AM data was generated by plotting output power versus input power, and AM/PM data by plotting the phase differences between input and output data versus input power. However, the data will be implemented within a FPGA development board, so they must be modified to use an amplitude reference as volts. The experiments will compare MPM with a high nonlinearity order of NL=15 with the best models produced by GP-LS. In this way comparing a common modeling tool, with an approach based on GP, a paradigm that has been used only sparingly in this domain. The data is introduced to the MPM and GP-LS methods. Afterward, the models are used to predict the PA behavior using two different schemes. First, using the raw output of the models; in this case the models will try to fit all of the underlying behavior of the system. We also use a second approach, where we combine the output of the generated models with a high frequency signal. This second method allows the models to focus on the large-scale and low frequency behavior of the system, with high-frequency details assumed to be stationary. While this can be achieved in different ways, in this work we use a white noise signal. Future work will study the effects of using other noise models.

C.5.1 *Experimental Setup*

Table C.1 shows the RF-PA Doherty 7W @2.1 GHz specifications used for this work. This RF-PA has maintains a constant gain for low level input signals. However, at higher input power, the PA goes into

Table C.1: Doherty 7W @2.1 GHz RF-PA Characteristics

Gain	14.5 dB @ 2.11 GHz
P1dB	38 dBm
Maximum Power	7 Watts
Polarization	VDS = 31 V, VGS = -2 V
Bandwidth	2110 - 2170 MHz

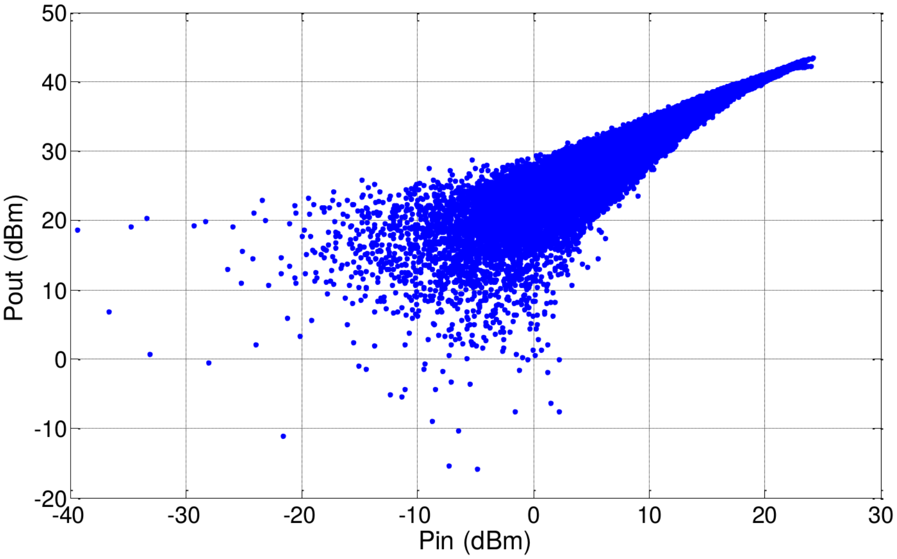


Figure C.4: AM/AM conversion curve for the RF-PA Doherty 7W @ 2.1 GHz.

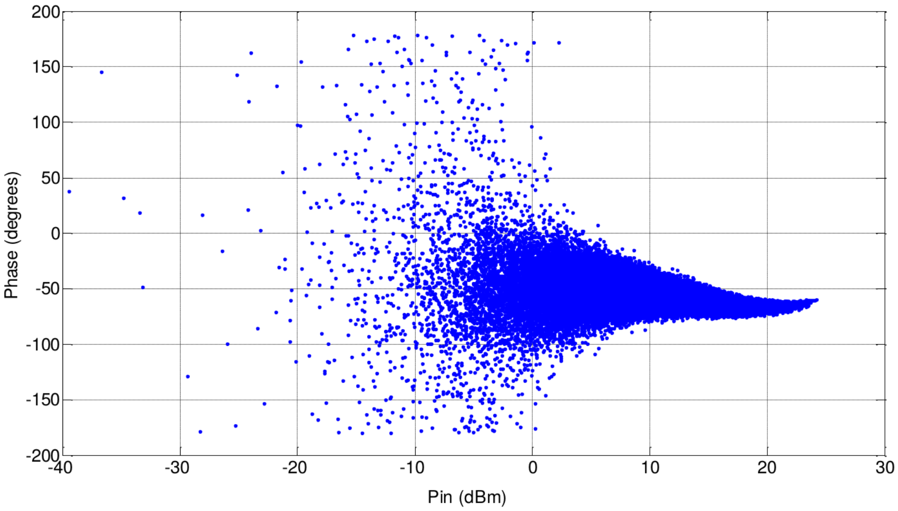


Figure C.5: AM/PM conversion curve for the RF-PA Doherty 7W @ 2.1 GHz.

saturation. The 1 dB compression point (P1dB) is 38 dBm and it indicates the power level reference that causes the gain to drop by 1 dB.

Figure C.4 presents the Doherty 7W @2.1 GHz RF-PA AM/AM and figure C.5 shows the AM/PM conversion curve, these figures illustrate the PA readings obtained.

Figure C.6 presents the Doherty 7W @2.1 GHz RF-PA AM/AM curves as volts. The transformation from dBm to volts is necessary, and the only way to implement a GP or MPM model into a FPGA development board. Figure C.7 plots the AM/PM behavior expressed in degrees against the input voltage.

C.5.1.1 *MPM Experimental Setup*

The MPM model is shown in figure C.8 as it is implemented in Simulink, The structure of the model can be adapted for any memory depth and nonlinearity order. The figure shows the models for both AM/AM and AM/PM. The figure C.9 shows an overview of the DSP

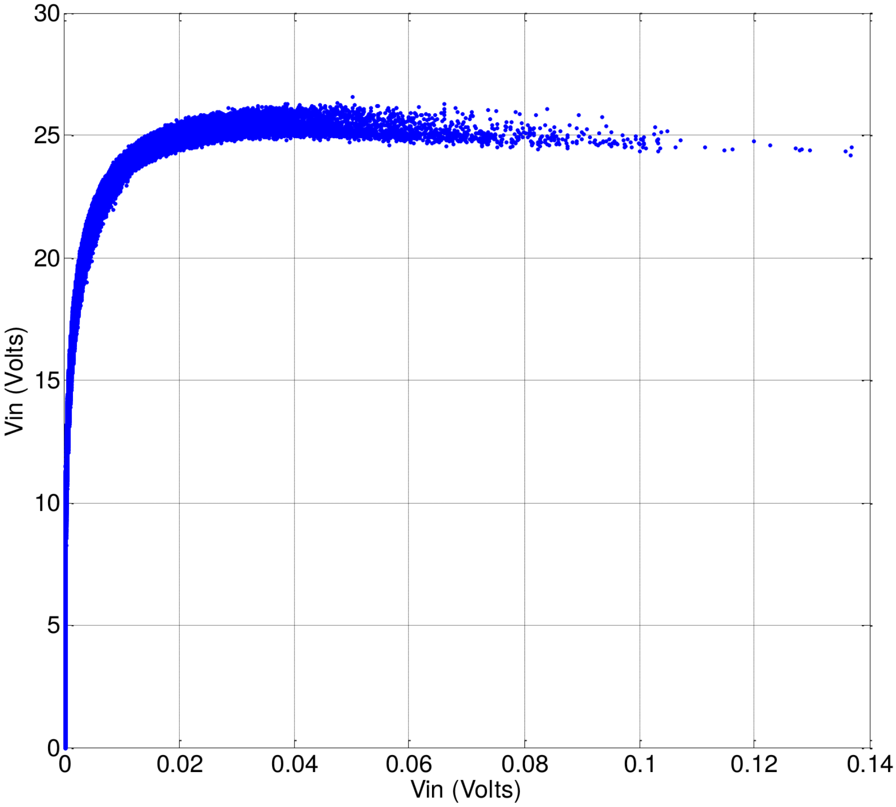


Figure C.6: AM/AM conversion curve expressed as voltage.

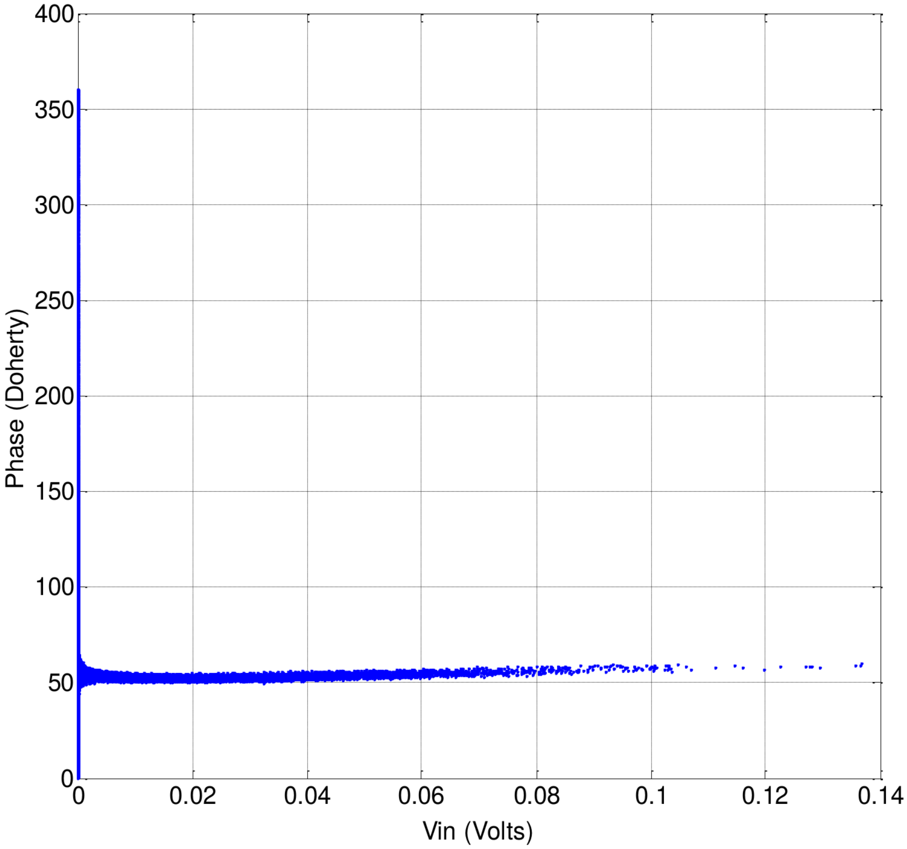


Figure C.7: AM/PM conversion curve expressed as voltage.

Builder stage, these icons represent advanced and standard blockset that allows for high-performance HDL generation algorithms directly from the Simulink environment.

C.5.1.2 *GP-LS Experimental Setup*

Since GP is intrinsically a stochastic search and optimization algorithm, this means that different GP runs will lead to different models (or solutions) produced as output. Therefore, we performed 10 independent runs of GP-LS using the parameters summarized in table C.2. From these runs we choose the best model found for each problem. In particular, we chose the models that achieved the lowest training error. In both cases the evolved models are relatively small, with a total of 17 nodes each. This means that each model is accompanied by 17 real-valued parameters. Informally, we can say that the models are parsimonious and relatively simple, particularly in terms of size. Of course, their efficiency will be evaluated experimentally below, from the perspective of a hardware implementation of each. Figure C.10 depicts the AM/AM model generated by GP-LS, which, as stated before, consists of 17 nodes and contains 17 parameters to reproduce the AM/AM behavior. Similarly, figure C.11 depicts the AM/PM model generated by GP-LS, which has the same size and the same number of parameters.

C.5.2 *Results and Analysis*

Table C.3 summarizes the results numerically for MPM, showing a very high error using the mean squared error (MSE), mean absolute error (MAE) and root mean squared error (RMSE) as metrics, each computed by comparing the known behavior with the predicted behavior of each model. Results are shown for the raw model outputs, and its combination with white noise. What immediately stands out from these results is that MPM does not have the required precision for a complex model such as the one studied here, considering that our data is composed by 122,880 samples.

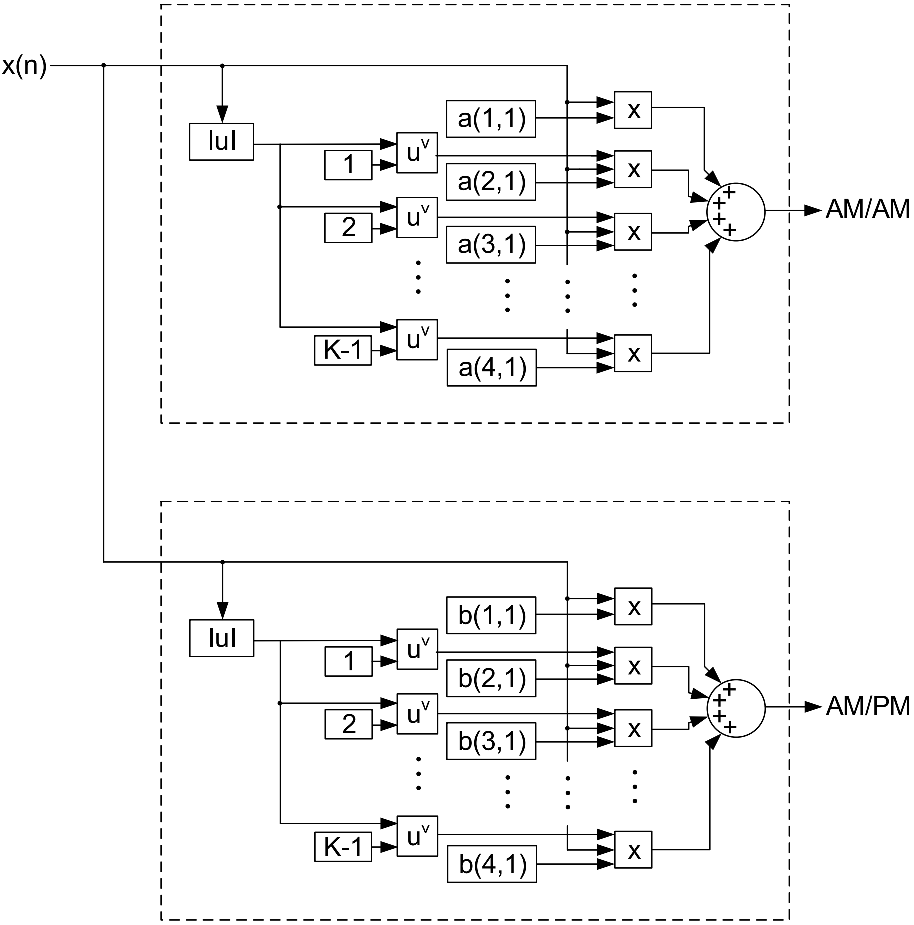


Figure C.8: General MPM structure for RF-PA modeling.

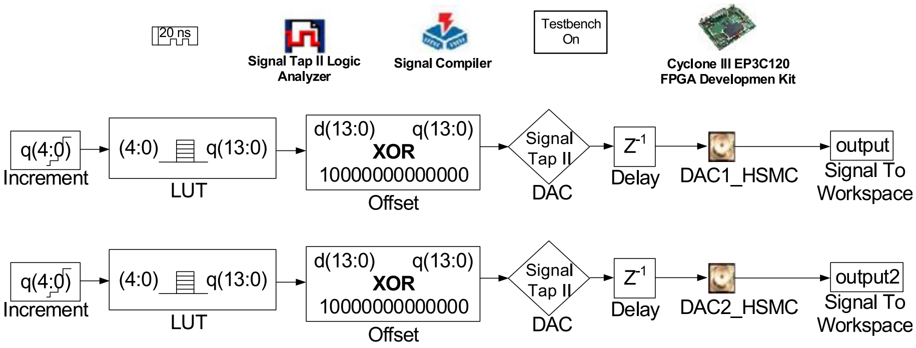


Figure C.9: DSP Builder chain for the specified board.

Table C.2: GP-LS Parameters

Parameter	Value
Population	200
Generations	40
Crossover operator	Standard subtree, 0.9 probability
Mutation operator	0.05 probability
Tree initialization	Ramped Half-and-half, max depth 6
Function set	+,-,x,/,exp,sin,cos,log,sqrt,tan,tanh
Terminal set	Input features, random constants
Selection type	Tournament, size 7
Elitism	Best individual survives
Maximum tree depth	17
Maximum LS iterations	500

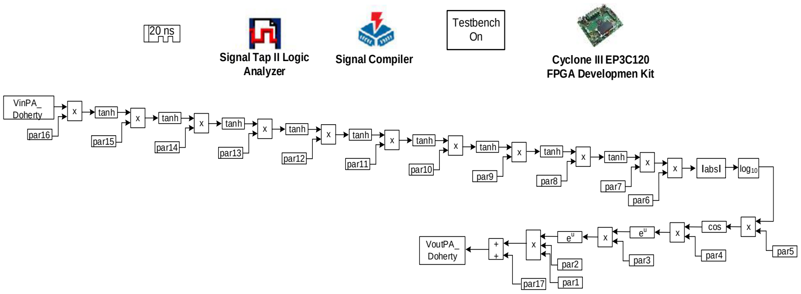


Figure C.10: DSP Builder Tool implementation of the GP-LS model using 17 parameters for the AM/AM conversion curve.

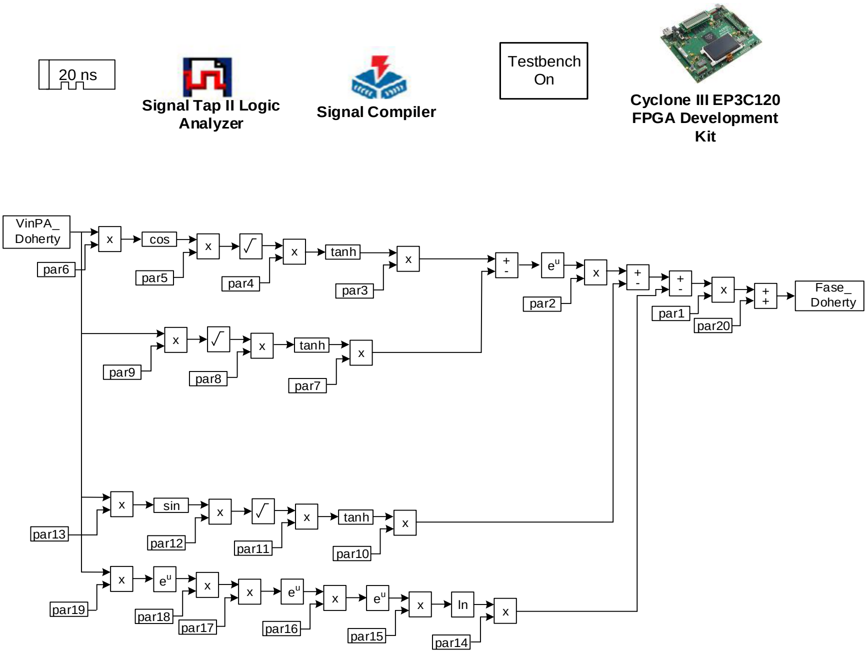


Figure C.11: DSP Builder Tool implementation of the GP-LS model using 20 parameters for the AM/PM conversion curve.

Table C.3: Summary of performance by MPM with NL=15 for AM/AM and AM/PM, with and without white noise added at the output.

Performance Measures	AM/AM with MPM using NL=15	AM/AM with MPM using NL=15 and adding white noise	AM/PM with MPM using NL=15	AM/PM with MPM using NL=15 and adding white noise
MSE	362.6932	342.8695	5012.4	4965.4
MAE	15.8983	15.2569	47.1811	47.0634
RMSE	8949.8	8606.9	2494.1	25414

In figure C.12 we can see the AM/AM results using the MPM with NL=15 and figure C.13 shows the same for AM/AM with white noise added to the output. In both cases it is noticeable that the model does not follow with high precision the real PA behavior. These result demonstrate that for complex RF-PA behaviors alternative modeling methods should be used. Figures C.14 and C.15 show similar analysis for AM/PM behavior, with similar results indicating that MPM cannot provide accurate predictions of amplifier behavior.

Table C.4 summarizes the performance by GP-LS on both modelling problems, showing a similar analysis as table C.3. If we compare both tables it is clear that GP-LS achieves the best overall performance in both cases, AM/AM and AM/PM. Moreover, notice that adding white noise reduces the error, particularly for the AM/AM model.

Figure C.16 depicts the GP-LS behavior for the AM/AM model using the raw output, while figure C.17 shows the behavior for AM/AM after adding white noise to the output. Figures C.18 and C.19 present a similar analysis for the AM/PM model. All four plots are in agreement with the actual PA behavior, confirming the numerical results of Table C.4.

These experimental results shows that the best solutions generated by GP-LS clearly outperforms the standard MPM approach to model the behavior of a complex PA. In terms of accuracy, it is clear that GP-LS is a

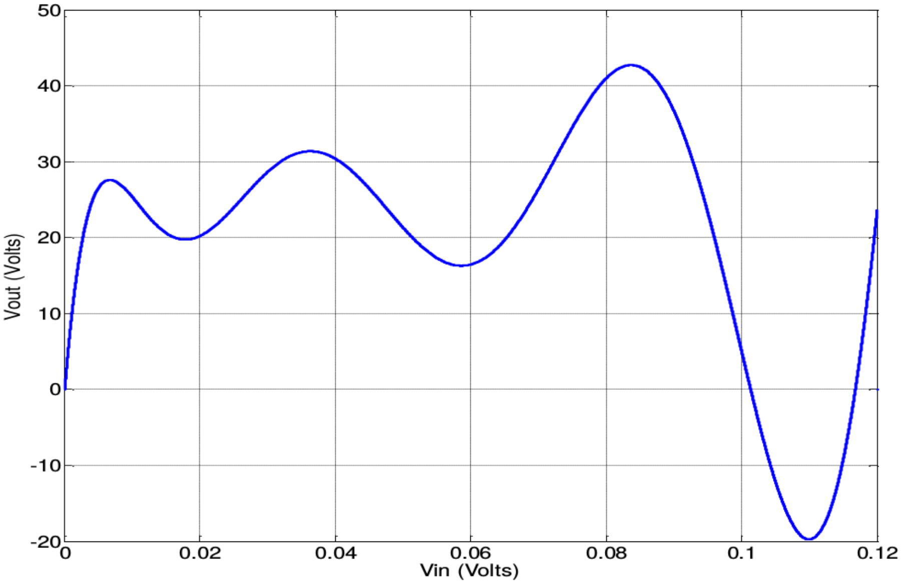


Figure C.12: RF-PA Doherty 7W@2.1 GHz AM/AM using a traditional modeling technique as MPM with NL=15.

Table C.4: Summary of performance by GP-LS for AM/AM and AM/PM, with and without white noise added at the output.

Performance Measures	AM/AM with GP-LS	AM/AM with GP-LS adding white noise	AM/PM with GP-LS	AM/PM with GP-LS adding white noise
MSE	116.1207	78.2696	936.8455	911.2172
MAE	8.9220	7.5702	8.1459	8.9996
RMSE	6881.9	5377.9	18824	19964

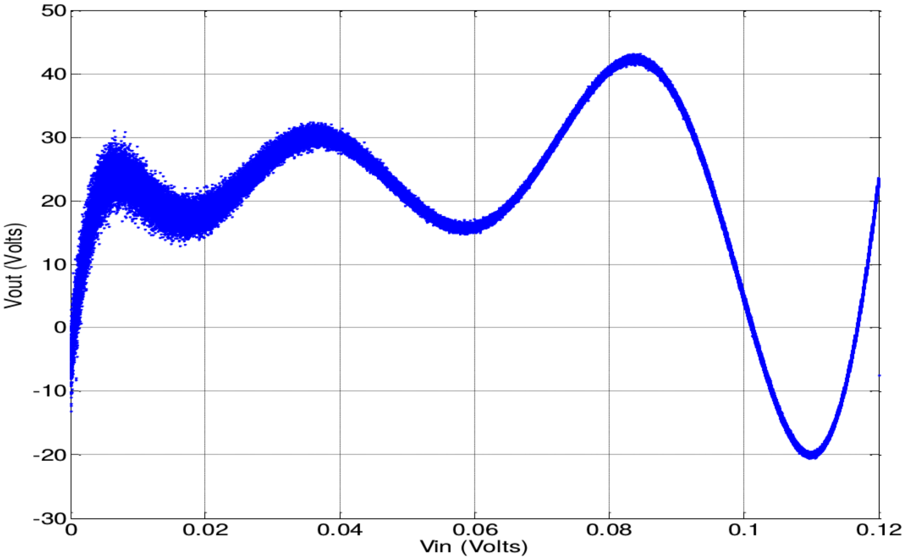


Figure C.13: RF-PA Doherty 7W@2.1 GHz AM/AM using a traditional modeling technique as MPM with NL=15 and white noise.

better option. We now compare the hardware resources required by each mode, this is summarized in Table C.5, considering that all models were implemented using the DSP Development Board Cyclone III-ALTERA. Again, the GP-LS models clearly require less resources on the FPGA board that the standard MPM approach. The AM/AM and AM/PM must be implemented in hardware together, both models represent the RF-PA behavior in amplitude and phase respectively

C.6 CONCLUSIONS

This work tackles the problem of predicting the behavior of a real RF-PA, particularly a RF-PA Doherty 7W@2.1 GHz. To do so, we used a traditional MPM approach for both the AM/AM and AM/PM conversion curves, as special case of the well-known Volterra series. Moreover, we used a recently developed variant of GP search for both modeling problems, GP-LS that integrates a numerical local search method

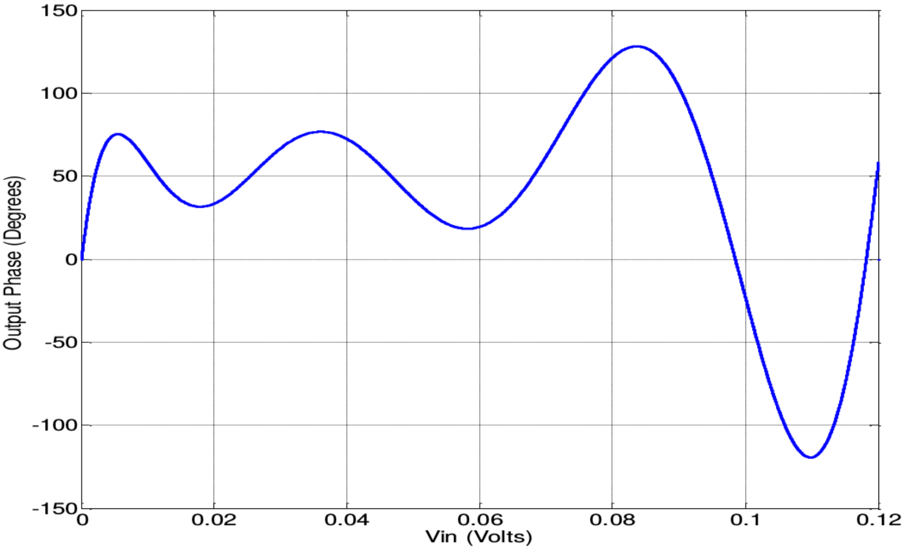


Figure C.14: RF-PA Doherty 7W@2.1 GHz AM/PM using a traditional modeling technique as MPM with NL=15.

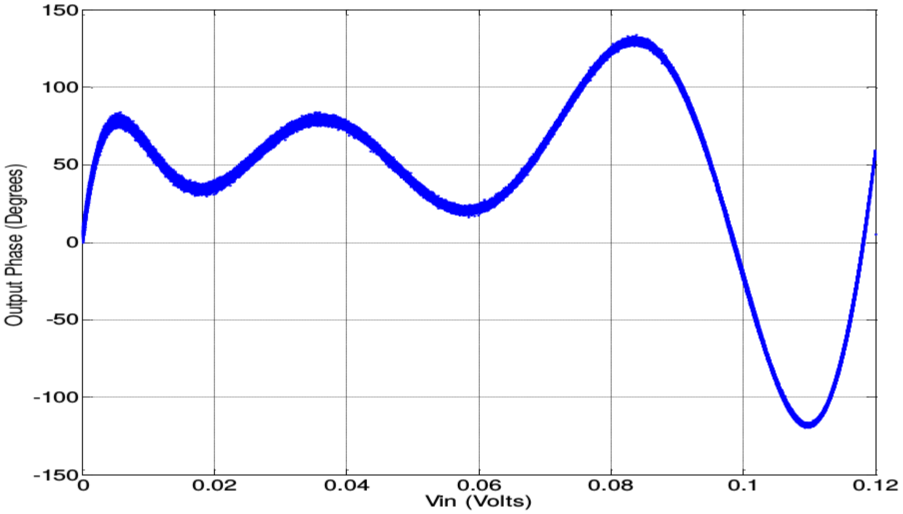


Figure C.15: RF-PA Doherty 7W@2.1 GHz AM/PM using a traditional modeling technique as MPM with NL=15 and white noise.

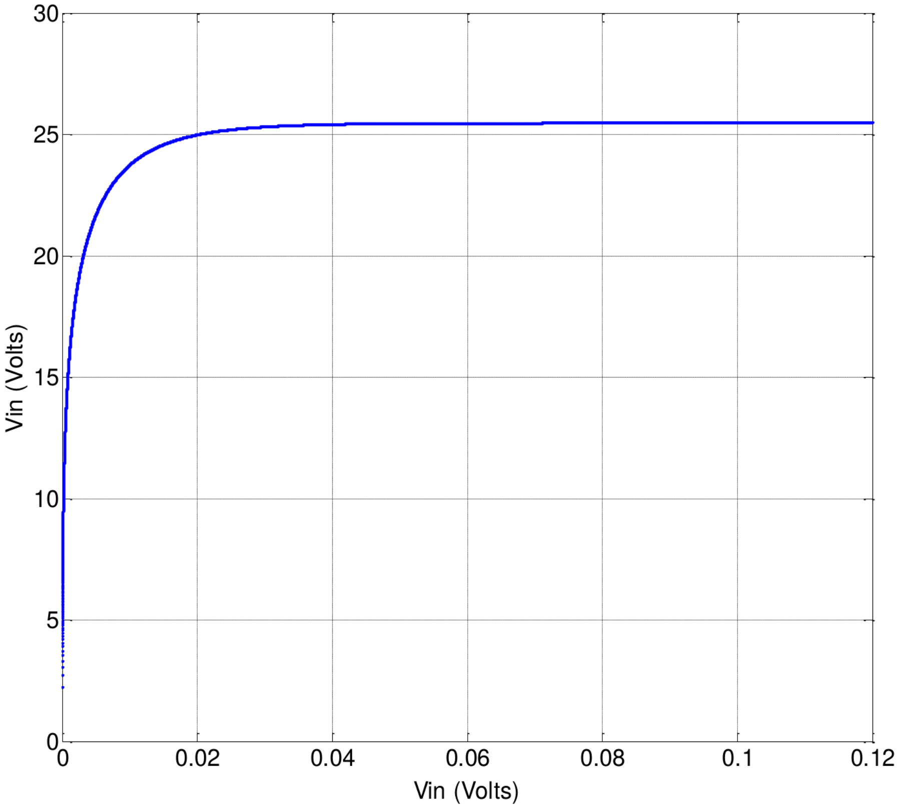


Figure C.16: RF-PA Doherty 7W@2.1 GHz AM/AM using the GP-LS model with 17 parameters.

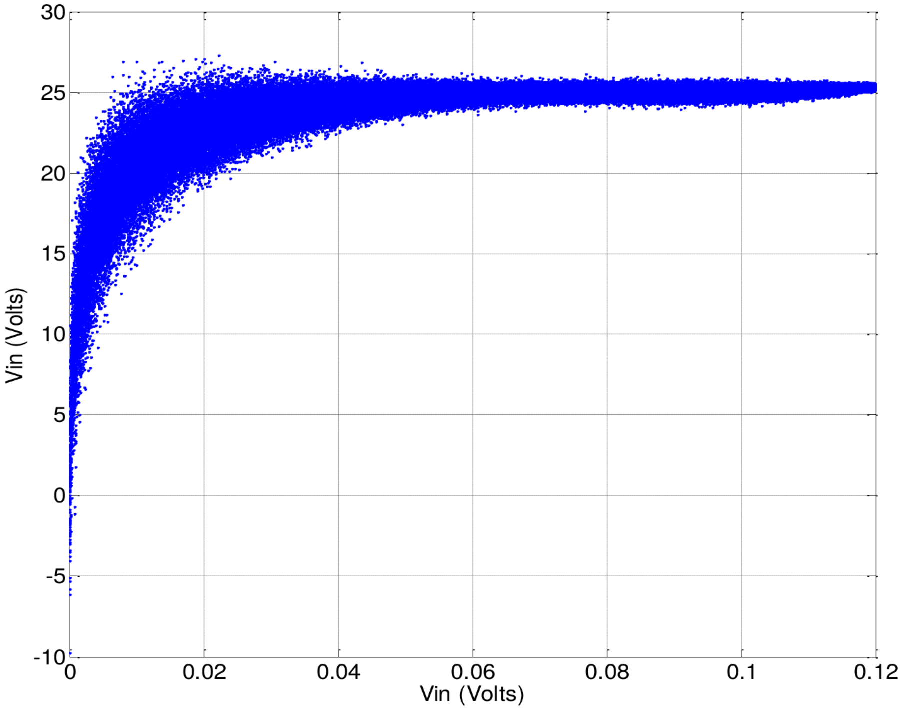


Figure C.17: RF-PA Doherty 7W@2.1 GHz AM/AM using the GP-LS model with 17 parameters, with white noise.

Table C.5: FPGA overall resources for the AM/AM and AM/PM using MPM and GP-LS

	Total Logic Elements	Total combinational functions	Dedicated Logic Registers	Total memory bits
Resources used by GP-LS	862/119088 (<1%)	509/119088 (<1%)	724/119088 (<1%)	211382/ 3981312 (1%)
Resources used by MPM	950/119088 (<1%)	800/119088 (<1%)	850/119088 (<1%)	212890/ 3981312 (1%)

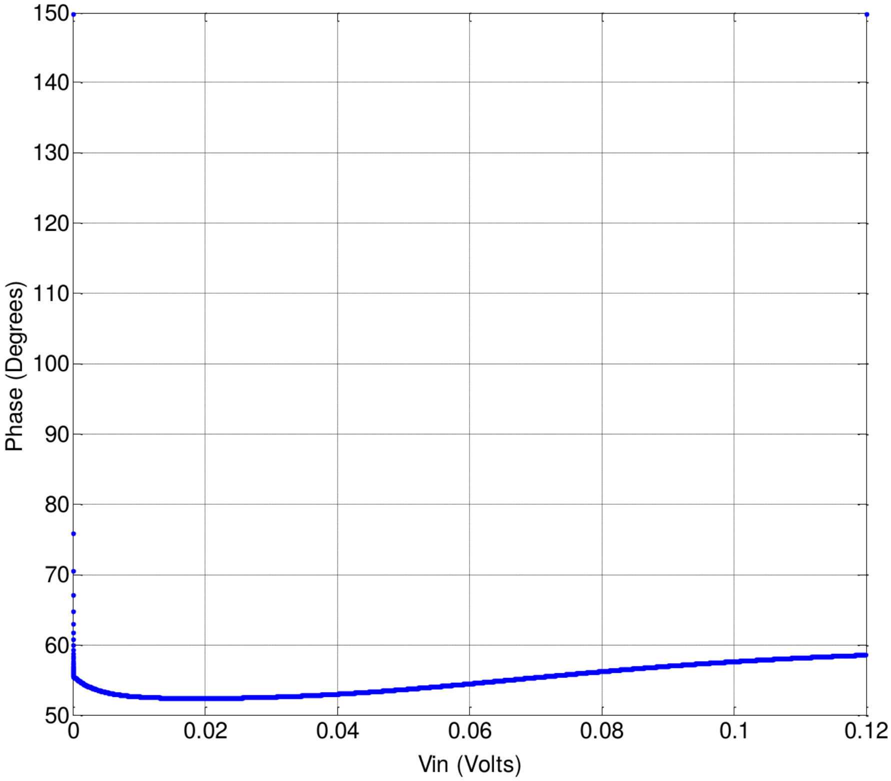


Figure C.18: RF-PA Doherty 7W@2.1 GHz AM/PM using the GP-LS model with 20 parameters.

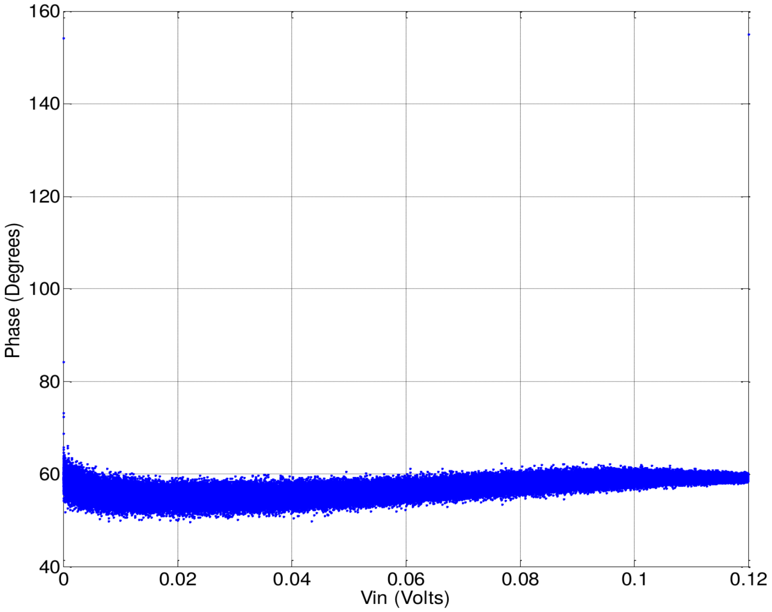


Figure C.19: RF-PA Doherty 7W@2.1 GHz AM/PM using the GP-LS model with 20 parameters, with white noise.

with the standard syntax-based search. The experimental results GP-LS produces much more reliable solutions than using MPM. For instance, for the AM/AM problem a MPM with NL=15 resulted in MSE=362.69, MAE= 25.25 and RMSE= 2494.1, while for GP-LS they were as low as MSE=78.26, MAE= 7.5702 and RMSE= 5377.9. Moreover, model performance was improved by combining the raw output of the evolved models by GP-LS with a white noise component, which provides a better approximation of the true PA behavior. Moreover, the MPM and GP-LS were also compared based on their efficiency, given by the amount of resources they require when they are implemented on an FPGA. Again, the GP-LS models require less resources on the boards than standard MPM. In summary, GP-LS are not only better at approximating PA behavior they do so in a more efficient way. These results are encouraging for real-world testing and deployment, which is left as future work.

D

CASE STUDY: ADSORPTION OF PHENOLS AND NITROPHENOLS MODELING

Based on Z-Flores, E., Abatal, M., Bassam, A., Trujillo, L., Juarez-Smith, P., and El Hamzaoui, Y. (2017). Modeling the Adsorption of Phenols and Nitrophenols by Activated Carbon using Genetic Programming. *Journal of Cleaner Production*

D.1 INTRODUCTION

Currently, the exponential growth of global population and the manner in which industrial societies are developing, have given rise to serious environmental problems. For instance, the intensive development of the chemical industry, that has been required to meet the growing demand, has produced large scale pollution from coal processing (Song et al., 2015) and carbon emissions (Song and Zhou, 2015); global issues with significant economic, social and environmental implications (Lee et al., 2016; Jaromir and Varbanov, 2016). For these reasons, multidisciplinary research and collaboration between stakeholders should be enhanced to stimulate the transition toward a more sustainable society (Niesten et al., 2016).

Regarding our water supply, the excessive applications of phenolic compounds leads to serious repercussions. Water is widely used as a raw material in the process industries for a wide range of products. In particular, phenol and phenolic compounds are common contaminants found in effluents from the production of plastics, leather, paint, textile, paper, printing and other petrochemical processes (Altenor et al., 2009). The wastewaters from these industries are generating considerable amounts of phenolic pollutants (Kumar and Jena, 2016), leading to increasingly stringent environmental regulations on the discharge of effluents (Klemeš et al., 2011). The main reason for this is that these contaminants are extremely toxic, even if they are present in small concentrations their impact on health includes protein degeneration, tissue erosion, paralysis of the central nervous system, vomiting, difficulty in swallowing, anorexia, liver and kidney damage, headache, fainting and various mental disturbances (Qadeer and Rehan, 2002; Srivastava et al., 2006; Takahashi et al., 1994). In fact, phenolic compounds are classified as high priority pollutants due to their carcinogenic effect on human health and their harmful effects on wild life (Altenor et al., 2009). Therefore, it is necessary to remove these phenolic compounds from industrial effluents before discharging them into the water stream.

The US Environmental Protection Agency (EPA) has instituted a regulation for lowering phenol concentration in the wastewater to less than 1.0 mg/dm^3 (Tian et al., 2007). Therefore, some researchers have proposed a variety of methods for their removal from wastewater (Ukrainczyk and McBride, 1992). Among current proposals, the removal of phenols by adsorption technologies is regarded as one of the best methods because it does not require a high operating temperature and can be considered to be a relatively simple process from an implementation standpoint. Indeed, various adsorbent solids have been used to remove phenolic compounds from wastewater, including activated carbon (Nouri et al., 2002), silica (Hanna et al., 2002), polymeric resins (Abburi, 2003), fly ash (Sarkar et al., 2005) and kaolinite (Barhoumi et al., 2003).

D.1.1 *Proposal and Contributions*

This work studies the use of activated carbon for the adsorption of phenols and nitrophenols. In particular, the goal of this work is to find models that capture the nonlinear relationships between several relevant operating conditions, such as the contaminant, initial concentration, pH and contact time, to predict the adsorption efficiency in an aqueous solution. Modeling phenol and nitrophenols adsorption onto activated carbon is very complex, it requires the solution of systems of equations that involve the radiant energy balance, the spatial distribution of the adsorbed radiation, mass transfer and the mechanisms of the adsorption transport problem (Altenor et al., 2009; Kumar and Jena, 2016). The studied adsorption process involves forces near the surface of the molecules, as well as the attractive and repulsive mobility forces between molecules of various types and shapes, which include the effects of molecular dissymmetry on properties of matter, such as evaporation, condensation and reflection. The process depends on several factors and phenomena, exhibiting a nonlinear behavior which is difficult to describe by linear mathematical models, such as those derived by linear regression methods. In similar problems, these considerations have lead researchers to turn towards non-linear data driven modeling methods (Nia et al., 2014; Karimi and Ghaedi, 2014; Dil et al., 2015; Ghaedi et al., 2015b,a).

Appropriate process models could be used to take corrective actions when the adsorption efficiency is predicted to be outside the required levels. The ability to take such actions could help reduce the environmental impact of the effluents when they are discharged into the water stream.

Therefore, the first main contribution of this work is the proposal of a relatively simple data-driven approach to derive models of the adsorption efficiency, based on a real-world experimental study. The derived models can help researchers and practitioners to predict the effect that variations in the operating conditions might have on the adsorption process, without recurring to costly and time-consuming experimental tests. The second contribution of this work relates to the specific algorithmic approach taken in this work, which is based on the use of genetic programming (GP) (Koza, 1992). Basically, we pose a supervised learning problem to automatically derive descriptive models of the adsorption efficiency and solve the problem by means of GP.

In this work, we compare two state-of-the-art GP methods recently proposed by the authors: (1) GP with local search optimization (GP-LS);

and (2) neat-GP, a variant that searches for parsimonious solutions. The GP algorithms are also compared with standard regression techniques, including multivariante linear, quadratic and robust regression. Moreover, GP is also compared with more powerful regression techniques, namely Multivariate Adaptive Regression Splines (Friedman, 1991) and the Fast Function Extraction algorithm (McConaghy, 2011). We present a comprehensive numerical and statistical comparison of the results, showing that GP can be used to construct useful predictive models of the phenol adsorption efficiency by activated carbon, outperforming all other methods.

The remainder of this work is organized as follows. Section D.2 provides a detailed description of the process of adsorption of phenol and nitrophenols by activated carbon. Afterward, section D.3 and section D.4 describe our GP-based modeling algorithms, GP-LS and neat-GP. Section D.5 presents our experimental work, describing how we derive a representative dataset of the adsorption process, the experimental setup and a discussion of the main results. Finally, Section D.6 presents our conclusions and outlines future work.

D.2 PHENOL AND NITROPHENOL ADSORPTION BY ACTIVATED CARBON

The process of adsorption refers to the adhesion of molecules from a mixture in a gaseous or liquid state onto a solid surface. This process creates a film of the adsorbate on the surface of the adsorbent. The present work focuses on activated carbon as an absorbent, which has been widely used in industry for many years, due to its uniform porous structure and appropriate selective adsorptive (Ruthven, 1984; Speight, 1991).

The surface properties of activated carbon used in this work are summarized in Table D.1, which were calculated using a t-plot analysis (Do, 1998; Lippens and de Boer, 1965; Nakai et al., 1993). An average diameter distribution of 0.3 nm is obtained by the method described by Horváth and Kawazoe (Horváth and Kawazoe, 1983). The material is characterized by scanning electronic microscopy (SEM) using a Hitachi

Property	Value
BET Surface Area	232.40 m^2/g
Average Diameter	0.30 nm
Adsorbent Density	2.2460 g/cc
Surface Atom Density	38.45e^{14} $molec/cm^2$
Langmuir Surface Area	1.034e^{03} m^2/g
Surface Tension	8,85 erg/cm^2
Critical Pressure	33.50 atm

Table D.1: Surface properties of activated carbon calculated by t-plot analysis.

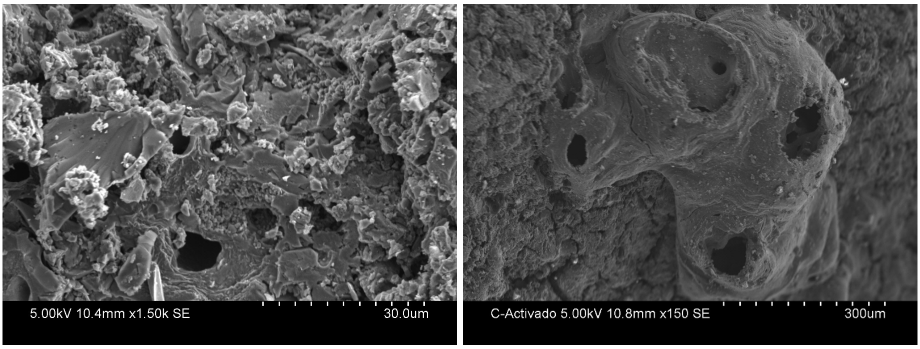
S-3400N instrument operated at 5 kV and the magnification is adjusted to obtain a clear image. Micrographs of activated carbon are shown in Figure D.1, showing that the external surface is fairly homogeneous with cavities of different sizes. We use breakthrough curves to better understand the behavior of activated carbon during the adsorption process (Singha et al., 2014), to study the effect that important operating parameters, such as pH and the initial concentration of the contaminant, have on the adsorption efficiency $W\%$ over time. The adsorption efficiency is given by

$$W\% = \frac{C_i - C_f}{C_i} \times 100 \tag{D.1}$$

where C_i is the initial concentration of the contaminant at the absorbent, and C_f is the final concentration of the contaminant in the absorbent at a particular time instant.

D.3 GENETIC PROGRAMMING WITH LOCAL SEARCH

The most common application of GP is to solve what are known as symbolic regression problems, where a model that best fits a dataset is sought. In particular we adopt the approach proposed in (Z-Flores et al., 2014, 2015) which has been applied to both symbolic regression and classification problems.



(a)

(b)

Figure D.1: Two example of SEM images of activated carbon at 1.50K magnification.

D.4 BLOAT AND NEAT-GP

Despite its well documented success (Koza, 2010), the basic GP approach suffers from a severe drawback that is commonly referred to as bloat (Silva and Costa, 2009; Silva, 2011). When GP is used to evolve solutions of unconstrained size, it will usually produce excessively large solutions.

For this reason, many researchers have proposed bloat-free GP variants, ranging from the very simple to more elaborate methods. In particular, in this work we study the recently proposed neat-GP algorithm (Trujillo et al., 2016), which is based on the NeuroEvolution of Augmenting Topologies (NEAT) algorithm (Stanley and Miikkulainen, 2002), originally developed for the evolution of neural networks. Recently published results show that neat-GP outperforms other bloat control methods, and achieves equal or better performance that the standard GP search (Trujillo et al., 2016). However, while previous results showed that neat-GP can be competitive with the state-of-the-art, this work is the first to apply the method to a real-world data modeling problem.

D.5 EXPERIMENTS

As stated before, the goal of this work is to derive a a model that describes the relationship between operating variables and adsorption efficiency of phenol and nitrophenols onto activated carbon. In particular, we use a data-driven approach, where real experimental data is generated, recorded and then used to pose a supervised-learning problem. Afterward, GP (GP-LS and neat-GP) is used to solve a supervised learning problem. Performance is compared with several baseline and state-of-the art methods to evaluate their predictive accuracy and model complexity.

D.5.1 *Data Generation*

To generate our ground truth data, experiments are performed with the adsorption process in a liquid phase. A constant amount of activated carbon (0.1 g) is mixed with a 10 mL solution of phenol (or 2-nitrophenol, 3-nitrophenol, 4-nitrophenol) in a 15 mL centrifuge tube at a pH range between 2-10. The suspensions are shaken at room temperature from 5 minutes to 6 hours, using an orbital shaker with a constant agitation rate. The phases are separated by centrifugation for 2 min at 1500 rpm. The concentrations of phenol, 2-nitrophenol, 3-nitrophenol and 4-nitrophenol in the supernatant solution after adsorption are determined with the ultraviolet-visible (UV-Vis) spectrophotometer technique. The maximum absorbances wavelength used are 269 nm, 278, 273 and 316 nm, respectively, for phenol, 2-nitrophenol, 3-nitrophenol and 4-nitrophenol. The experimental process is summarized in Figure D.2.

The experimental database is provided by Abatal et al. as published in (Abatal and Olguin, 2012). It contains 975 samples of input/output pairs. The inputs are: (1) the contaminant present in the suspension (phenol, 2-nitrophenol, 3-nitrophenol, 4-nitrophenol); (2) the initial contaminant concentration in the adsorption process (20, 40, 60, 80 and 100 ppm); (3) the pH range (2, 4, 6, 8 and 10); and (4) the adsorption

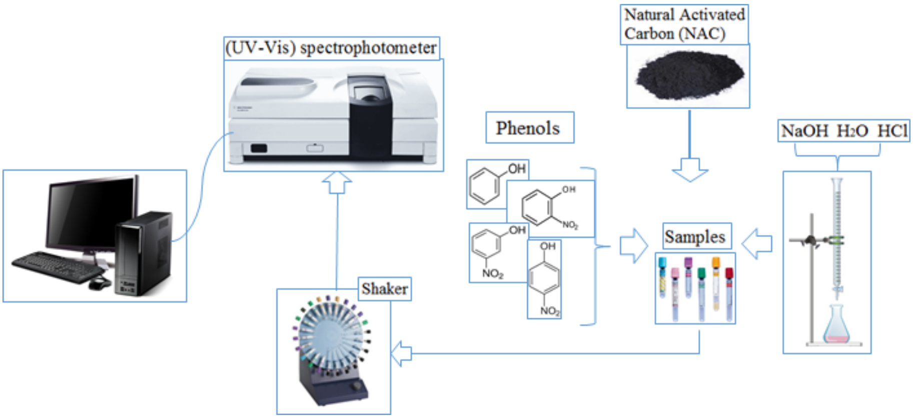


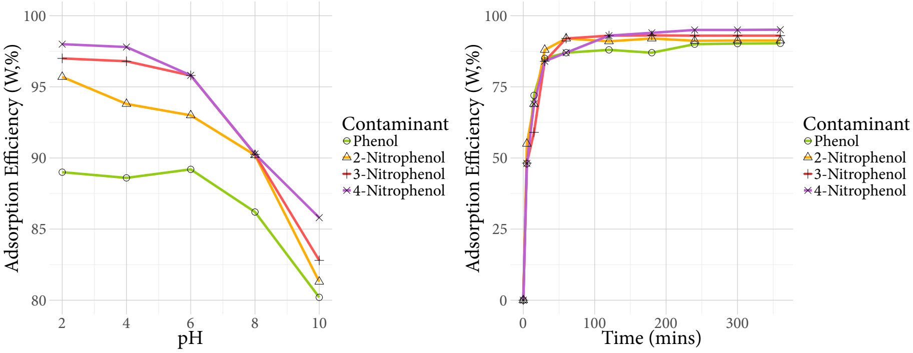
Figure D.2: Schematic diagram of the experimental adsorption process used in this work.

Variable	Type	Operating Range	Units
Contaminant (C)	Input	$\{1, 2, 3, 4\}$	-
Initial concentration (C_i)	Input	$[20, 100]$	ppm
pH	Input	$[2, 10]$	-
Time (t)	Input	$[0, 360]$	minutes
Adsorption efficiency ($W\%$)	Output	$[0, 100]$	[%]

Table D.2: Summary of collected variables in the experimental dataset.

time (0-360 mins.). The output to be approximated is the adsorption efficiency $W\%$. These input and output variables are summarized in Table D.2.

From the collected data, we can observe that the removal of pollutants from wastewater is affected by the pH of the phase where the removal occurs; as shown in Figure D.3(a), which plots the effect of pH on the adsorption of each contaminant. The plot shows that the adsorption of phenolic compounds by activated carbon is reduced with increasing pH values. The pH of the solution has an impact on the surface charge and the degree of ionization of the absorbent, accelerating the internal surface area and increasing the creation of the porous material of activated carbon as an adsorbent solid. This might lead to change the



(a) Effect pH on Phenols $W\%$

(b) Effect of contact time on $W\%$

Figure D.3: Analysis of the experimental data.

kinetics and equilibrium characteristics of the adsorption process that degrade the phenolic compounds over time, as shown in Figure D.3(b).

We have observed that when the experiments are carried out for one to three hours with pH values lower than 7, the adsorption efficiency of phenolic compounds reaches the critical values (maximum). Therefore, we can confirm that the interference of activated carbon on phenolic compounds is affected by the operating conditions considered in this work, namely the pH, contact time, initial contaminant concentration and contaminant.

D.5.2 *Algorithms and Experimental Setup*

The regression problem posed in this work is solved with several methods. First, we use basic methods common in such works to provide a baseline for our experimental study; these are: multivariate linear regression (Linear), multivariate quadratic regression (Quadratic) and iteratively reweighted least squares linear regression (Robust). Second, we use two powerful regression methods, the Fast Function Extraction algorithm (FFX) (McConaghy, 2011) and Multivariate Adaptive Regression Splines (MARS) (Friedman, 1991; Zhang and Goh, 2014; Pramila and Mahesh, 2015; Vijayaraghavan and Veezhinathan, 2015). Generally speaking, FFX and MARS build a linear combination of basis functions to construct the model that best describes the training data. Third, we use the two GP variants presented in Section B.3, neat-GP and GP-LS.

In summary, we use seven algorithms to solve the modeling problem: Linear, Quadratic, Robust, FFX, MARS, neat-GP and GP-LS. For each algorithm we performed 30 independent runs. In each run the problem data was split into a training and testing set, where the former contains 75% of the data samples and the latter 25%. The datasets were randomly partitioned before each run, and while training performance will be reported the critical result will be the prediction accuracy of the evolved models over the test data. In particular, the algorithms are compared using the Mean Squared Error (MSE) and the coefficient of determination (R^2), computed over the ground truth output and the estimated output of each algorithm, for the training and testing data. Table D.3 summarizes all other algorithm parameters for neat-GP and GP-LS. Moreover, we will also report an analysis of the size of the solutions generated by the algorithms that achieved the best performance, as an approximation of model complexity.

For the Linear, Quadratic and Robust regression methods we used the implementation offered in Matlab 2014a with the default configuration. In the case of MARS[1] and FFX[2], we used the freely available toolboxes provided by the authors with their default configuration. GP-LS is implemented using Matlab and the GPLab toolbox (Silva and Almeida, 2005) and neat-GP is implemented using the DEAP framework (De Rainville et al., 2012), which is freely available on our website[3].

D.5.3 *Results and Discussion*

Figure D.4 summarizes the performance of each algorithm using boxplots that depict rank statistics (maximum, minimum, first quar-

[1] http://www.cs.rtu.lv/jekabsons/regression.html

[2] http://trent.st/ffx/

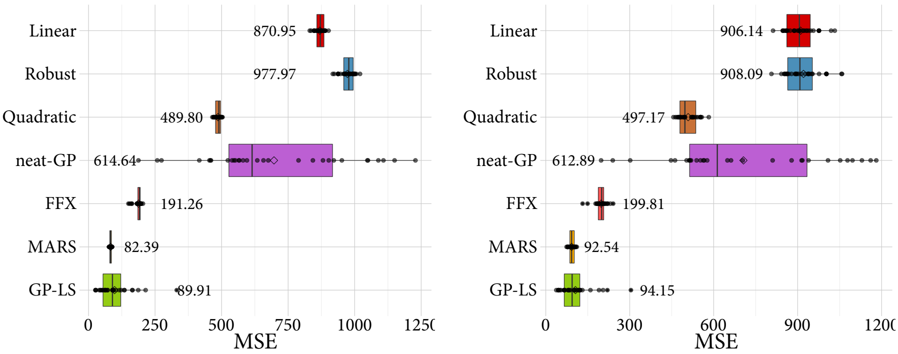
[3] http://www.tree-lab.org/index.php/resources-2/downloads/open-source-tools

Parameter	neat-GP	GP-LS
Population size	500	250
Initialization	Ramped Half & Half	Ramped Half & Half
Max. initial depth	3 levels	6 levels
Crossover rate	0.7	0.9
Mutation rate	0.3	0.1
Function set	+,-,×,÷,sin,exp,cos,log,sqrt,tan,tanh	+,-,×,÷,sin,exp,cos,log,sqrt,tan,tanh
Terminal set	Input features & random constants	Input features & random constants
Selection	Tournament of size 6	Tournament of size 7
Elitism	Best individual survives	Best individual survives
Maximum tree depth	none	17

Table D.3: Parameters for neat-GP and GP-LS.

tile, median, third quartile and outliers) over all 30 runs. Figure D.4(a) presents the MSE of each algorithm on the training data, while Figure D.4(b) shows the same on the test data. Moreover, Table D.4 summarizes these results numerically, reporting the median, first and third quartiles of the train and test MSE. Using only visual inspection of these results, the worst performance is achieved by Linear and Robust regression methods, with the best performance clearly achieved by MARS and GP-LS. Note the poor performance by neat-GP, which is outperformed by Quadratic regression method and also shows the highest variance in performance, suggesting that the method is highly sensitive to the training data on this problem. FFX outperforms all methods, except for MARS and GP-LS. It is also important to note that while there are obvious differences among the compared algorithms, the performance of all methods does not vary substantially when comparing their training and testing MSE, this suggests than none of the methods suffered from overfitting. Among all methods, GP-LS achieves the minimum error on the present problem, while MARS exhibits a slightly lower median error.

To evaluate the significance of these results, we use the Friedman test to perform all pairwise comparisons of the methods, focusing specifically on test performance for both problems. The null hypothesis in each pairwise comparison specifies that both sets of results share a common median, and the p-values are corrected using the Benjamini-Hochberg



(a) Training Performance

(b) Testing Performance

Figure D.4: Boxplots that show the MSE performance of each algorithm on the training (a) and testing (b) data over all thirty runs. Numerical values represent the median.

Algorithm	Train Median	Train 1st Quartile	Train 3rd Quartile	Test Median	Test 1st Quartile	Test 3rd Quartile
Linear	870.95	857.55	884.87	906.14	860.23	946.75
Robust	977.975	958.02	994.55	908.08	863.29	953.21
Quadratic	489.795	476.48	496.22	497.17	478.96	537.63
neat-GP	614.64	523.46	921.64	612.89	513.28	938.92
FFX	191.26	184.35	194.30	199.81	187.22	207.45
MARS	82.3865	79.72	86.05	92.53	85.83	101.86
GP-LS	89.915	52.57	122.78	94.14	65.44	122.11

Table D.4: Summary of MSE by each algorithm on the training and testing sets, showing the median, first and third quartiles.

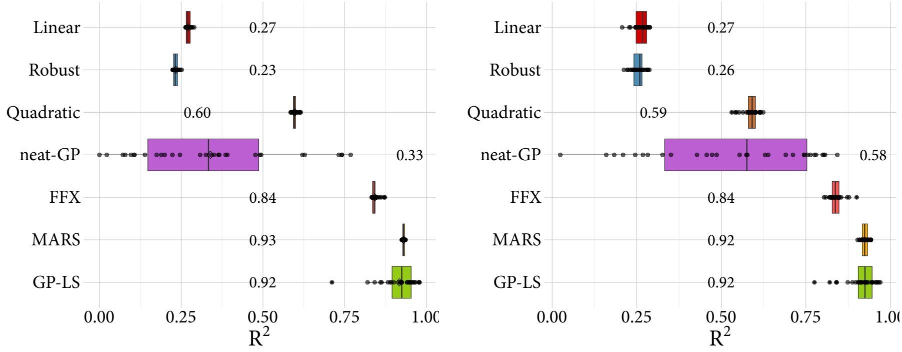
	Linear	Robust	Quadratic	neat-GP	FFX	MARS	GP-LS
Linear	-	0.001*	0.000*	0.0106	0.0000*	0.000*	0.000*
Robust	-	-	0.000*	0.0106	0.000*	0.000*	0.000*
Quadratic	-	-	-	0.0106	0.000*	0.000*	0.000*
neat-GP	-	-	-	-	0.000*	0.000*	0.000*
FFX	-	-	-	-	-	0.000*	0.000*
MARS	-	-	-	-	-	-	1.000
GP-LS	-	-	-	-	-	-	-

Table D.5: Summary p-values of the pairwise Friedman test with Benjamini-Hochberg correction for the test MSE over all runs; an asterisk (*) indicates that the null hypothesis can be rejected at the $\alpha = 0.01$ significance level.

algorithm. The resulting p-values are given in Table D.5. In both tables an asterisk (*) indicates that the null hypothesis can be rejected at the $\alpha = 0.01$ significance level. These results allow us to make some important conclusions. First, that the difference between MARS and GP-LS is not statistically significant in terms of median performance. Second, that the difference between both of these algorithms and the rest of the methods is statistically significant. Finally, neat-GP is clearly outperformed by FFX, MARS and GP-LS.

To provide a deeper analysis of the performance of each method, we analyze the coefficient of determination R^2 of a linear fit between the expected output and the estimated output of each method. Figure D.5 summarizes the R^2 performance of each algorithm using boxplots, showing the training (Figure D.5(a)) and testing performance (Figure D.5(b)). Table D.6 presents a numerical comparison based on R^2, showing the median, first and second quartiles over all runs. These results are consistent with what was shown above, with MARS and GP-LS substantially outperforming all other methods, and neat-GP performing worse. The statistical significance of these results is tested using the same procedure with the corresponding p-values reported in Table D.7. These results lead to the same conclusions discussed above

It is also instructive to compare the methods based on the size of the models found. In this comparison we exclude the more basic regression methods (Linear, Quadratic and Robust) since their size is completely



(a) Training Performance

(b) Testing Performance

Figure D.5: Boxplots that show the R^2 performance of each algorithm on the training (a) and testing (b) data over all thirty runs. Numerical values represent the median.

Algorithm	Train Median	Train 1st Quartile	Train 3rd Quartile	Test Median	Test 1st Quartile	Test 3rd Quartile
Linear	0.26	0.27	0.28	0.27	0.25	0.28
Robust	0.23	0.23	0.24	0.26	0.24	0.27
Quadratic	0.60	0.59	0.60	0.59	0.58	0.60
neat-GP	0.33	0.33	0.49	0.58	0.33	0.76
FFX	0.83	0.84	0.84	0.84	0.83	0.85
MARS	0.93	0.93	0.93	0.92	0.92	0.93
GP-LS	0.92	0.89	0.95	0.92	0.90	0.95

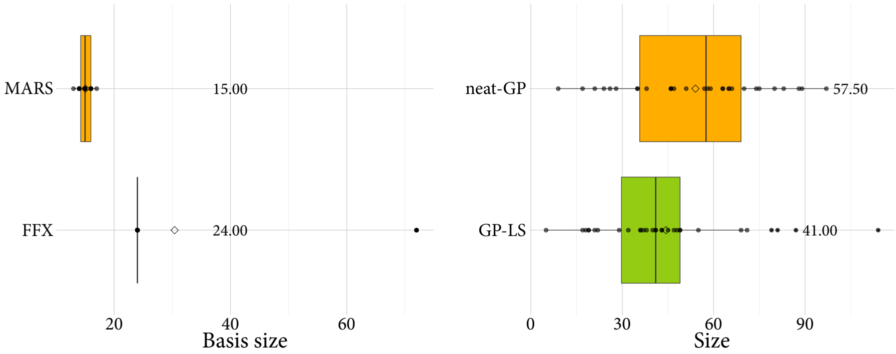
Table D.6: Summary of R^2 by each algorithm on the training and testing sets, showing the median, first and third quartiles.

	Linear	Robust	Quadratic	neat-GP	FFX	MARS	GP-LS
Linear	-	0.0012*	0.0000*	0.4885	0.0000*	0.0000*	0.0000*
Robust	-	-	0.0000*	0.4885	0.0000*	0.0000*	0.0000*
Quadratic	-	-	-	0.0012	0.0000*	0.0000*	0.0000*
neat-GP	-	-	-	-	0.0000*	0.0000*	0.0000*
FFX	-	-	-	-	-	0.0000*	0.0000*
MARS	-	-	-	-	-	-	1.0000
GP-LS	-	-	-	-	-	-	-

Table D.7: Summary p-values of the pairwise Friedman test with Benjamini-Hochberg correction for the test R^2 over all runs; an asterisk (*) indicates that the null hypothesis can be rejected at the $\alpha = 0.01$ significance level.

defined by the number of input variables and their performance was not competitive with the more advanced methods. Moreover, we differentiate between the methods based on a linear combination of basis functions (FFX and MARS) and the GP-based methods (neat-GP and GP-LS). In the former group, size is given based on the total number of basis functions used, while in the latter group we use the total number of tree nodes in the best solution found. These results are summarized and presented as boxplots in Figure D.6, with the following observations. First, MARS produces smaller solutions than FFX, by a substantial margin, using a median of 15 basis functions while FFX uses 24. Second, while GP-LS is not designed to be a bloat control method, it nevertheless produces solutions that are at least as small as those generated by neat-GP, the median solution size is 41 nodes for GP-LS and 57 for neat-GP. These results are particularly noteworthy, since both MARS and GP-LS also outperform FFX and neat-GP based on MSE and R^2. Moreover, it is important to mention that the size of the solutions generated by GP-LS should be considered as smaller than those generated by MARS. If a basis function (as used by either MARS or FFX) is expressed as a tree structure it would require at least 3 nodes, and sometimes as many as 5 or 7. Therefore, taking the conservative estimate of 3 nodes, and counting each of the sum operations in the linear models, MARS would require about 60 nodes to express the models it is generation, higher than the 41 required by GP-LS. Taking both the size and performance of the models into consideration, it is reasonable to state that GP-LS can produce the most efficient models of the adsorption phenomenon.

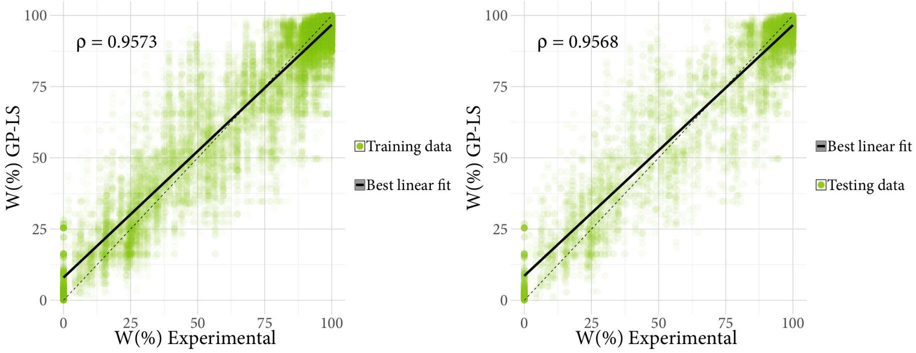
To visualize the quality of the GP-LS models, Figure D.7 shows scatter plots for all thirty models, plotting the ground truth output with respect to the estimated adsorption efficiency $W\%$. Figure D.7(a) shows the behavior on the train data while Figure D.7(b) shows the same on the test data, in both cases the figures specify the Pearson correlation coefficient between both values. To provide a finer level of detail of how the GP-LS perform on this problem, Figures D.8 and D.9 show breakthrough curves for two of the models generated by GP-LS, lets call them models K_A and K_B. The plots compare the predictions made by each model with the ground truth experimental data. The plots show



(a) Basis Functions: FFX and MARS

(b) Nodes: neat-GP and GP-LS

Figure D.6: Size comparison between basis function based methods (a) and GP-based methods (b).



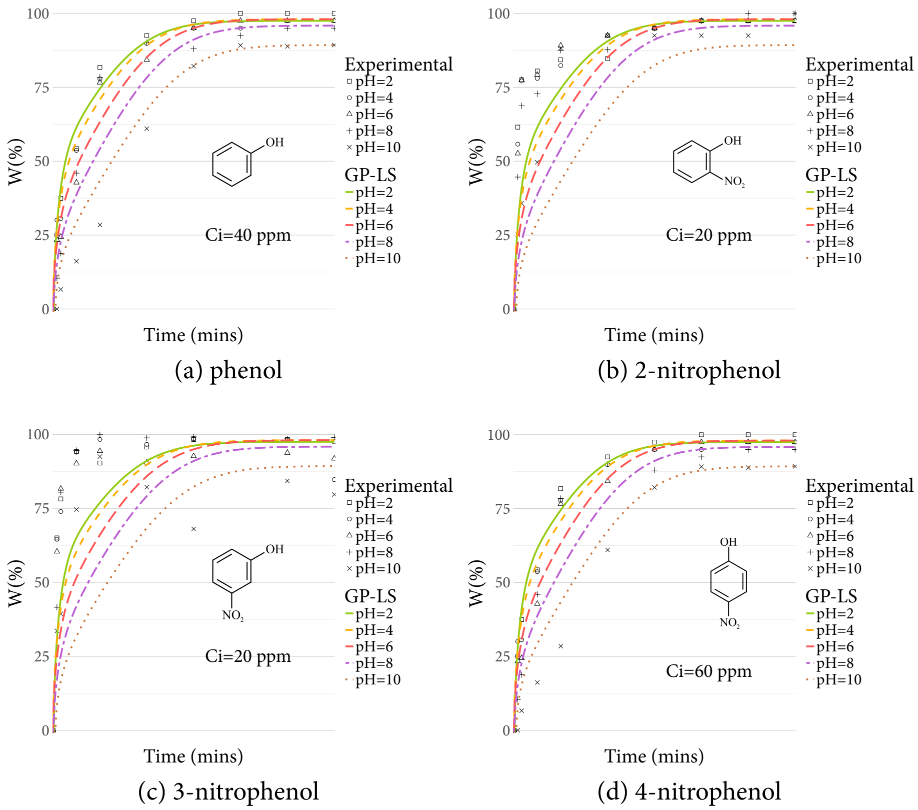
(a)

(b)

Figure D.7: Scatter plots of all 30 GP-LS models, relating the ground truth adsorption efficiency $W\%$ with the estimated value produced by each model, on both the training data (a) and the test data (b); in both cases the figures provides the value of the Pearson correlation coefficient ρ.

how $W\%$ varies as a function of time with respect to each contaminant (respectively, each row is for phenol, 2-nitrophenol, 3-nitrophenol and 4-nitrophenol) at different pH and concentration values. The numerical performance for model K_A on the test data are a MSE of 44.14 and $R^2 = 0.96$, while model K_B gave a MSE of 65.44 and $R^2 = 0.94$. The plots in Figure D.8 and Figure D.9 provide a visual illustration of the ability of the GP models to approximate the underlying contaminant adsorption under different conditions

Finally, it is instructive to mention that the GP process performs an implicit feature subset selection stage during the search. As the GP algorithm generates different candidate models, the number of input variables (terminals) used by each model is decided by the search process. Therefore, some of the input variables might not be included in the final model returned by the search. Nevertheless, it is important to analyze which of the input variables were most often used by the best models found in all of the runs (30 models in total). Figure D.10 analyzes the usage frequency for each of the input variables from the best solutions found in each run. First, Figure D.10(a) shows a bar plot with values between 0 to 30, where the height of each bar represents the number of models that used a particular feature. The most used feature is time, followed by the contaminant, the pH and finally the initial concentration. Figure D.10(b) provides a finer level of detail. In this plot the y-axis represents the proportion that each feature represents of all the inputs to a model (as leaves on the tree representation), ordered on the x-axis based on their performance rank. Moreover, the right side of the plot shows the average proportion over all runs for each feature. Figure D.10(b) emphasizes the fact that time is the most important feature, followed by the contaminant, with the pH and the initial concentration C_i representing the least frequently used features.



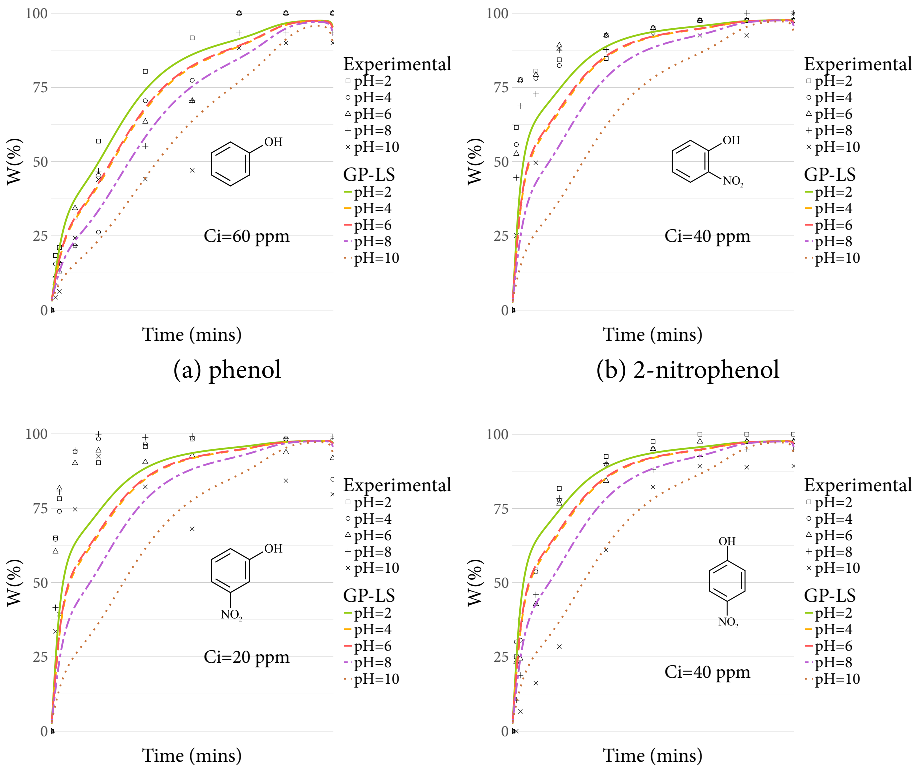
(a) phenol

(b) 2-nitrophenol

(c) 3-nitrophenol

(d) 4-nitrophenol

Figure D.8: Breakthrough curves for model K_A, comparing the ground truth $W\%$ (points) with the model estimates (solid curves).



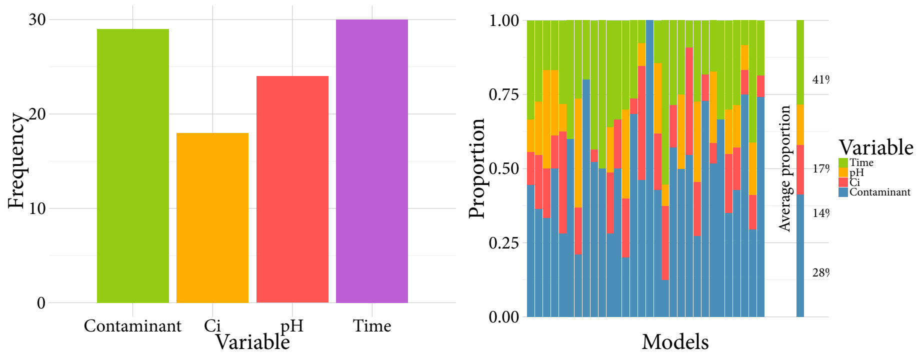
(a) phenol

(b) 2-nitrophenol

(c) M3-nitrophenol

(d) 4-nitrophenol

Figure D.9: Breakthrough curves for model K_B, comparing the ground truth $W\%$ (points) with the model estimates (solid curves).



(a) Usage Frequency

(b) Proportion of tree terminals

Figure D.10: Analysis of use frequency of each input feature: (a) shows how often a feature was used in the best model found; (c) the proportion of terminal elements that each feature represents in each tree.

D.6 SUMMARY, CONCLUSIONS AND FUTURE WORK

The effective planning and management of industrial process that release contaminants into our ecosystem has become a primary concern over recent decades, since contamination and resource scarcity problems have led to a variety of impacts on our well being. However, achieving a reasonable and efficient management strategy is difficult since many conflicting factors have to be balanced, due to complexities of real world problems.

This work presents the first application of GP to model the adsorption of phenols and nitrophenols, one of the most used processes in industrial water treatment applications. The methodology used is to pose a supervised learning problem, after collecting relevant data from real-world experimental tests. The considered input variables are the contaminant, the pH levels, the initial concentration and the elapsed time. The desired output of the model is a prediction of the adsorption efficiency of the process $W\%$.

The experimental work considered two recently proposed GP variants, a bloat free algorithm (neat-GP) and a GP-based search that incorporates a numerical local optimizer (GP-LS). Moreover, the GP algorithms were compared with standard regression techniques (linear, quadratic and robust regression) and with more sophisticated regression methods, namely FFX and MARS. The experimental results show that GP-LS clearly outperforms all other methods based on accuracy (MSE and R^2), except for MARS that achieves equivalent performance. On the other hand, neat-GP performed rather poorly, with median performance similar to that of the simpler linear methods and a larger variance. Moreover, when comparing the derived models based on size, a good indicator of model complexity, GP-LS achieved the best results, generating smaller models than FFX, MARS and neat-GP. Therefore, considering both accuracy and size, we can confidently state that GP-LS can be used to model the adsorption efficiency quite effectively. These observations are further extended by showing breakthrough curves comparing the GP-LS models with the ground truth data.

Future work derived from this research will be centered on the development of an automated wastewater treatment system, sufficiently suitable to directly control the adsorption of phenolic compounds onto activated carbon. Or more generally, to derive data-driven models for other absorbents, to monitor and predict when there might be a hazardous effect caused by the pollutants in real time. The models derived by the GP algorithms have the potential to be implemented in smart sensors systems, to provide accurate predictions and help decision makers and high-level managers of industrial process, that use phenolic compounds, develop cost-effective management policies to remove these harmful contaminants from the water. Therefore, future work will also focus on the managerial implications by examining and assessing the economic, environmental and ecological impacts of using the derived models as part of an online wastewater treatment system.

E

LEAFGP: VISUALIZATION OF REGRESSION MODELING USING GP

E.1 INTRODUCTION

In the field of model generation for the characterization of a system's behavior by data inspection, the morphology of the model (syntax) is just as important as its performance, specially when the system is non-linear and complex architectures are required. Although there is a vast quantity of different model generation algorithms (Morlini et al., 2015), most of them offers a pre-built model structure, where the algorithm searches for the best fit of this structure to input data, however only few focus on model-free approaches; that is, no prior knowledge on a specific model structure is imposed, rather the algorithm builds it at run-time. GP belongs to this specific group of model generation algorithms. The model generation process is reproducible by a supervised learning task, namely regression. Moreover, GP searches for a near-optimal symbolic expression that minimizes or maximizes a cost function adapted to the problem to be solved, by evolving a pool of candidate solutions using syntax search operators. It is noteworthy to mention that two tasks, the model syntax and its parameters, are solved simultaneously during the algorithm iterative process. A symbolic regression problem can be formally defined as

$$K^* \leftarrow \underset{K \in \mathbb{G}}{min} f(K(\boldsymbol{x}_j), \boldsymbol{y}) \text{ with } j = 1, \ldots, p, \tag{E.1}$$

where K^* is a suitable symbolic expression that satisfies the cost function f, $\mathbb{G}$ is the syntactic space defined by the primitive set $\mathbb{P}$ of functions and terminals, $x_j \in \mathbb{R}^p$ is the input data matrix, p is the number of variables in input data, $y \in \mathbb{R}$ is the output data vector and K is the symbolic expression output. Usually, f is defined as an error metric, like $f = \frac{1}{N}\sum_i^N (K(\boldsymbol{x_i}) - y_i)^2$ (Mean Squared Error [MSE]).

Despite that in recent years the use of meta-heuristic algorithms, like GP, has been successfully used in different domains, mainly because its outstanding results in rather complex problems (Koza, 2010)(the work in (Mittman and Cooper, 2014) won the 2014 Human-Competitive Award), the widespread deployment is still in its early stages compared with other more mature techniques. Although one might argue that theoretical formality is still lacking in some areas of Evolutionary Algorithms (EA), in particular for GP (White et al., 2013), the fact is that apart from that, the spread of tools toward a broader audience helps to the overall GP influence. From a research point of view, these tools are essential for experimentation and comparison purposes, however real-world engineering problems should be also benefited from these aforementioned tools. Indeed, a truly efficient model extraction application might not require an expert to solve the problematic at hand.

Specifically for GP, there are several software libraries and applications accessible to both practitioners and developers. Available libraries ranges from the simpler to the most complex ones. Either by just focusing on only GP or on several EAs, however only few are still actively updated. Some examples are:

- BEAGLE[1] - A Evolutionary Computation (EC) C++ framework. Although it includes capabilities for GP, there is a minimalist stripped version specific for GP called Puppy.

- BEAGLE Puppy[2] - Written in compliance of BEAGLE codebase, it is a fast tree-based GP C++ implementation, which this work is based on.

[1] https://github.com/chgagne/beagle

[2] https://code.google.com/archive/p/beagle/wikis/Puppy.wiki

- DEAP[3] - A Python framework that covers a wide range of EAs, including GP.
- DGPF[4] - A Java based GP framework that can easily extend to other EAs as well. It has a distributed workload design built-in.
- ECJ[5] - A vast EC Java based framework with a steeply learning curve, that is still actively developed.
- EO[6] - Another EC framework written in C++ with its main focus being the execution speed. It supports parallel computing through third party software.
- EpochX[7] - A specific GP Java based framework.
- GPLAB[8] - A MATLAB GP toolbox.
- μGP[9] - A C++ GP optimizer, tested in a wide range of distinct problems.
- PushGP[10] - An EA framework that uses a self designed programming language to evolve programs using a clever stack based methodology.
- GPOCL[11] - An OpenCL implementation of GP. It can run in multiple platforms.

Some examples of GP applications with a graphical interface are:

- Discipulus[12] - A commercial product that is capable of building predictive models using GP.

[3] https://github.com/deap/deap

[4] http://dgpf.sourceforge.net/

[5] http://cs.gmu.edu/ eclab/projects/ecj/

[6] http://eodev.sourceforge.net/

[7] http://www.epochx.org/

[8] http://gplab.sourceforge.net/

[9] http://ugp3.sourceforge.net/

[10] http://faculty.hampshire.edu/lspector/push.html

[11] http://gpocl.sourceforge.net

[12] http://www.rmltech.com/

- DataModeler[13] - A commercial product using GP as model generation.
- HeuristicLab[14] - An open source software that handles a wide range of EAs through a visual interface. Actively in development, is one of the few applications that provide several tools varying from data management to results reporting.
- Eureqa[15] - A commercial product focused on simple data model extraction and its deployment.

All of these tools offers different layers of functionality and might not be useful for all kind of tasks. Only few of them focuses on the user interaction, being either an expert on the field or just a practitioner. This work pretends to close more that gap by proposing an open source visual tool (LeafGP) for model extraction that it is intuitive and easy to use by both, a casual and an experienced user.

E.2 THE GRAPHICAL USER INTERFACE

The Graphical User Interface (GUI) was implemented in Qt and programmed in C++. Qt is a cross-platform framework for the development of software, specially for rapid graphical interface deployment across multiple hardware. Currently in its 5.7 version, it is widely used in many different scenarios, ranging from purpose-specific applications to full OS desktop environments. Since GP is the sole algorithm for model generation, LeafGP is built around a modified version of GPOCL, providing very efficient properties given the parallel nature of its design. In the following subsections we will explore LeafGP throughly; first, we will briefly describe some characteristics of the GPOCL implementation and how it is utilized by LeafGP; and then, we will outline the different functionalities from the user interaction point of view.

[13] http://www.evolved-analytics.com/

[14] http://dev.heuristiclab.com/

[15] http://www.nutonian.com/products/eureqa/

E.2.1 *GPOCL*

GPOCL is a freely available high-performance implementation written in C++ and OpenCL that implements a canonical GP system using a prefix linear tree representation (Augusto and Barbosa, 2013). OpenCL is an open standard for uniform and portable parallel programming across heterogeneous computing platforms. It aims to provide low and high level access to data- or task-parallel devices, either individually or simultaneously. Besides the conventional multi-core CPUs and GPUs from multiple vendors, OpenCL also supports some other parallel devices, such as FPGA, Digital Signal Processors (DSP), IBM's Cell Broadband engine, and more.

E.2.1.1 *Programming model*

The OpenCL's Application Program Interface (API) exposes the access to the parallel computing devices by querying the host system to determine which platforms are present. A platform is a particular OpenCL implementation installed on the system. From the platform it is possible to discover which compatible computing devices are available and then select the ones according to some criteria defined by the programmer; a criterion, for instance, may be a particular architecture, like CPU or GPU. Given one or more devices, a memory buffer can be allocated on them, and a command queue, responsible for kernel submissions to the devices, can be set up.

E.2.1.2 *General structure*

The main routine of GPOCL performs the following high-level procedures in the given order.

1. *OpenCL initialization*: this is the step where the general OpenCL-related tasks are initialized. An OpenCL platform is discovered, a device representation is created and its specifications are extracted, and a command queue is created.

2. *Calculating n-dimensional ranges*: defines how much parallel work there will be and how they are distributed among the compute units; in other words, *local size* and *global size* parameters are calculated.

3. *Memory buffers creation*: in this phase all global memory regions accessed by the OpenCL kernels are allocated on the device and possibly initialized. The fitness cases are transferred and enough space is reserved for the population and error vectors.

4. *Kernel building*: an OpenCL kernel, relative to a given strategy of parallelization, is compiled just-in-time, targeting the compute device. Before the actual compilation, an optimized GP interpreter is assembled and helper functions and runtime parameters/flags are embodied in the kernel source.

5. *Evolving*: this iterative routine implements the actual GP dynamics.

E.2.1.3 *Evolutionary algorithm*

Most of the steps in the GP cycle are performed sequentially and performed by the *host* machine, but the costly procedure of *evaluation* of programs are done in parallel by the target computing device, be it the CPU or GPU. Moreover, LeafGP extends this behavior by implementing also the evaluation in a sequential procedure. This provides to the user further flexibility in the deployment stage, specifically when the parallel hardware is not available, thus allowing GP to be executed in a single CPU core.

The evaluation step itself does not do much, because the hard work is done mostly by the OpenCL kernels. Basically, three things happen within `Evaluate`(P):

1. *Population transfer*: all programs of P are transferred to the target compute device. For the GPU, due to the physical separation between the host memory and the device memory, an explicit copy or synchronization is required.

2. *Kernel execution*: for any non-trivial problem, this is the most demanding phase. Here, the entire recently transferred population is evaluated – by interpreting each program over each fitness case – on the compute device. Fortunately, this step can be done both in parallel as well accelerated by GPUs.

3. *Error retrieva*l: after being computed and accumulated in the previous step, the prediction errors of the whole population need to be transferred to the host so that this information is available to the evolutionary process. Again, if the compute devices and the host are not the same, an explicit transfer is necessary.

E.2.1.4 *GP program interpreter*

The GP tree interpreter is a function that takes a GP program and its inputs, and then executes the instructions stored on the tree's nodes in a flow given by the tree's structure. An interpreter is commonly implemented as a stack-based procedure in which the partial results coming from the children nodes are pushed into a stack and pulled back when a parent operator needs the accumulated result.

A pseudo-code for a GP tree interpreter is presented by the function `Interpreter` (*program*, *n*), described in Algorithm E.2.1. The argument *program* is an array of integers encoding a tree (the structure, operators, operands and variables) via a prefix linear representation, whereas *n* is the index of the training data point currently being evaluated. The function `INDEX()` extracts the operator/operand code, `VALUE()` returns the actual numerical value of a constant or the index if the argument is a variable, $\mathrm{X_n}$ is the *n*-th row of the training dataset, `PUSH()` puts a number into the stack and, finally, `POP` pulls back a value from the top of the stack. At the end of the interpretation process, the program's predicted output is left on the top of the stack, so this value is returned by the interpreter and then used to compute the program's prediction error.

LeafGP extends the arithmetic functions shown in Algorithm E.2.1 with a collection of conditional, logical, relational and trigonometric operators.

Algorithm E.2.1 Function `Interpreter` (*program*, *n*)

```
for $op \leftarrow program_{\text{size}} - 1$ to 0 do
    switch INDEX ($program[op]$) do
        case ADD
            PUSH ( POP + POP );
        case SUB
            PUSH ( POP - POP );
        case MUL
            PUSH ( POP × POP );
        case DIV
            PUSH ( POP ÷ POP );
        case CONSTANT
            PUSH ( VALUE ( program[op] ) );
        otherwise
            PUSH ( $X_n$[ VALUE ( program[op] ) ] );
end for
return POP ;
```

E.2.2 *LeafGP*

In general, LeafGP has four different functional areas: data loading and exploration, GP settings, GP execution and results analysis. Since LeafGP is intended to cover two main audiences, each area is divided in two subareas: the express, for the casual and inexperienced user that does not have to worry about changing parameters; and the advanced, for the experienced user with knowledge of GP, where every possible parameter is tunable.

LeafGP is designed to be executed over two threads. The first thread is responsible of the GUI responsiveness, while the second thread executes complete GP cycles. This scheme is quite common in graphical interfaces that requires heavy calculations underneath. The main GUI is enclosed in the parent class `MainWidget` and is connected through the Qt system of signal and slots to communicate among threads. This mechanism allows the asynchronous transfer of data between the GP cycle and the presentation of information. For example, once GP is started, it sends all sort of information relevant to the solutions fitness, population statistics, consumed resources, etc. to the user so the algorithm's performance can be reviewed in real-time.

A parent class is used to enclose all the data used during the execution of LeafGP, namely `gpExperiment`. The data in `gpExperiment` is organized as follows:

- Performance information of a GP run.
- Processing of input data through the `Dataset` class.
- GP parameters used during an experiment.
- In a hierarchical scheme, information of all GP populations is stored in this order: 1) run, 2) generation, 3) population, 4) individual.

This data is shared along all the different areas in LeafGP, available for reading or writing. In the data loading section, samples are fed to a `Dataset` object. Once the user has finished manipulating the data, then it is transfered to a `gpExperiment` object. GP parameters are also stored in the aforementioned object.

The main GP cycle is implemented in the `GP` class, where the second thread is executed, namely the `workerGP` class, providing responsiveness to the GUI. `GP` includes a series of signal and slots to pass information to the main thread. This mechanism has two objectives: first, to give a visual feedback to the user; and second, to store into `gpExperiment` partial information related to the GP run. If for any reason the GP cycle is interrupted, the user can still visualize and explore the partial run.

Once a GP run has been completed, then all sort of information is available either for analysis or for generation of reports.

In the following subsection, each part of LeafGP is described and can be used as a quick user guide.

E.2.2.1 *Application navigation*

As briefly mentioned before, LeafGP contains four distinctive parts, which will be explained below:

1. Data exploration. Input data can be analyzed and pre-processed before the algorithm is executed.

    - Express (Fig. E.1). New or existing projects can be created or loaded, respectively. A project is the easiest way to quickly retrieve a previously executed experiment, which could contain any combination of the dataset, the GP settings, the population information or the summary from results. If a new project is created the two options can be chosen: load a prebuilt dataset from a list of benchmark problems or load a stored dataset. In the current version, the latter supports Comma Separated Values (.csv) and Matlab Uncompressed files (.mat). Once the data has been loaded, a quick summary from the data is shown.
    - Advanced (Fig. E.2). Three different pages are exposed. *Summary*, where data is displayed in a spreadsheet format; the user can edit or explore data value statistics. *Metrics*, with

three plots available depending on the selection of the variable in the dataset: pearson correlation, multiple linear regression and quantile-quantile (Q-Q). *Processing*, at the moment is being developed.

2. Algorithm settings. Parameters heavily influence on performance and solutions quality. These can be tweaked to better fit algorithm expectations.

    - Express (Fig. E.3). GP parameters are automatically defined without the intervention of the user, either from an analysis of the data or hyper-parameterization.
    - Advanced (Fig. E.4). Eight pages are included in this part. *Experiment*, where the settings related to the run are defined here, like the number of runs or the type of the data partition. *Population*, the size of population or type of initialization is specified here; a pre-built population can be loaded as well. *Selection*, where the GP selection schemes are defined. *Functions*, by using a drag-and-drop the user can select the function set. *Terminals*, default terminals are set according with the loaded dataset, but the user can remove or add new ones. *Operators*, the canonical search operators are tuned here by changing probabilities. *Fitness*, in the current version it is still in development. *Others*, accelerated processing is available depending on the current hardware where LeafGP is executed; the actual support follows closely to the GPOCL implementation, that is either CPU or GPU.

3. GP Launcher. GP launcher based on predefined settings.

    - Express (Fig. E.5). The main information from the GP run is displayed here, together with data related to the best solution of the evolution, like a tree representation or a symbolic representation in a prefix syntax.
    - Advanced (Fig. E.6). Two different types of information can be explored. *Performance*, convergence plots of quality and

size are displayed. *Population*, an interactive population interface, where a dot-matrix graph represents all the individuals during the run; the user can review origin of operator, size and quality for any chosen individual. The colormap can be changed according with the relative fitness, operator or relative size. The genealogy for any individual can be analyzed as well by selecting a dot and pressing the ancestor button.

4. Results. Evolved solutions can be explored and analyzed.

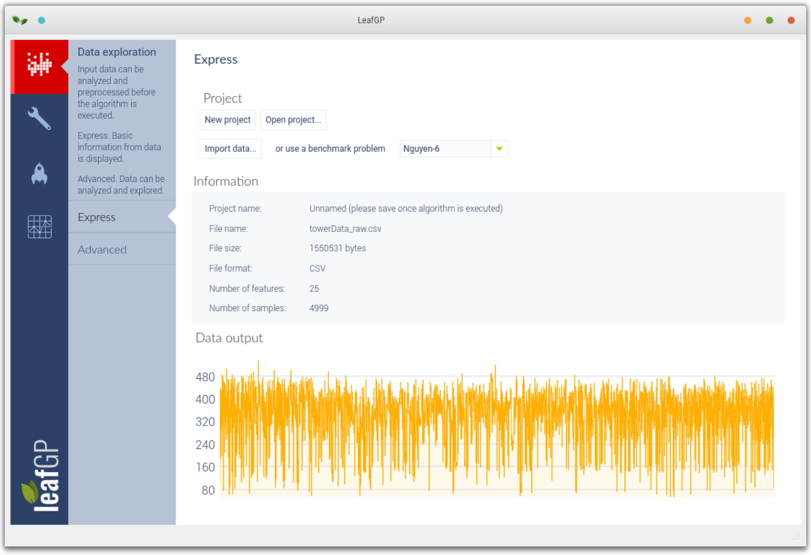
    - Express (Fig. E.7). The best solution for the last run is displayed here, with its error and the correlation with the expected data.
    - Advanced (Fig. E.8). Four different sections can be selected by the user. *Summary*, where a quick condensation of the last run is displayed including an evolvability plot. *Population*, here single individuals from full populations in multiple runs can be reviewed. *Statistics*, in case that three or more runs are executed, then boxplots for size and quality are displayed. *Saving*, where the results can be stored in a single file, with the flexibility of choosing different level of details.
    - Report (Fig. E.9). In the current version, a PDF file containing user selected information can be generated.

E.2.2.2 *Implementation details*

The design of LeafGP reduces external dependencies as much as possible. In the current version, it contains the following dependencies:

- OpenCL SDK. Depending in the hardware vendor, a SDK is required in order to be compiled, this being either CPU or GPU hardware.

- OpenBLAS. Any variant of the linear algebra library is required, however an optimized version is preferred.



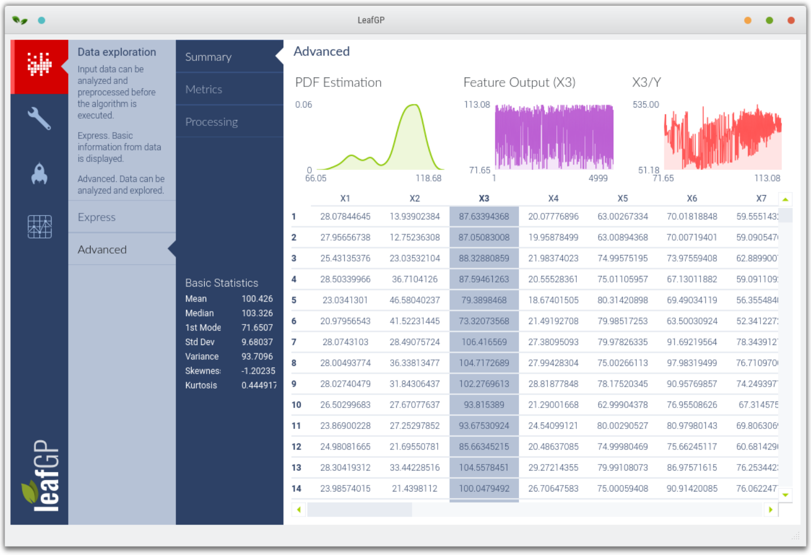
Input data can be analyzed and preprocessed before the algorithm is executed.

Express. Basic information from data is displayed.

Advanced. Data can be analyzed and explored.

Project name:	Unnamed (please save once algorithm is executed)
File name:	towerData_raw.csv
File size:	1550531 bytes
File format:	CSV
Number of features:	25
Number of samples:	4999

Figure E.1: Data exploration. Express page.



Input data can be analyzed and preprocessed before the algorithm is executed.

Express. Basic information from data is displayed.

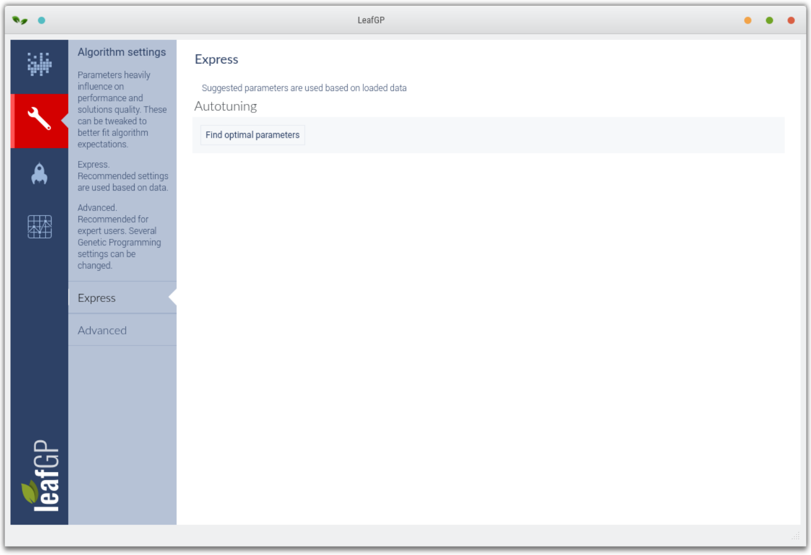
Advanced. Data can be analyzed and explored.

Basic Statistics

Mean	100.426
Median	103.326
1st Mode	71.6507
Std Dev	9.68037
Variance	93.7096
Skewnes:	-1.20235
Kurtosis	0.444917

	X1	X2	X3	X4	X5	X6	X7
1	28.07844645	13.93902384	87.63394368	20.07776896	63.00267334	70.01818848	59.555143
2	27.95656738	12.75236308	87.05083008	19.95878499	63.00894368	70.00719401	59.090547
3	25.43135376	23.03532104	88.32880859	21.98374023	74.99575195	73.97559408	62.889900
4	28.50339966	36.7104126	87.59461263	20.55528361	75.01105957	67.13011882	59.091109
5	23.0341301	46.58040237	79.3898468	18.67401505	80.31420898	69.49034119	56.355484
6	20.97956543	41.52231445	73.32073568	21.49192708	79.98517253	63.50030924	52.341227
7	28.0743103	28.49075724	106.416569	27.38095093	79.97826335	91.69219564	78.343912
8	28.00493774	36.33813477	104.7172689	27.99428304	75.00266113	97.98319499	76.710970
9	28.02740479	31.84306437	102.2769613	28.81877848	78.17520345	90.95769857	74.249397
10	26.50299683	27.67077637	93.815389	21.29001668	62.99904378	76.95508626	67.314575
11	23.86900228	27.25297852	93.67530924	24.54099121	80.00290527	80.97980143	69.806306
12	24.98081665	21.69550781	85.66345215	20.48637085	74.99980469	75.66245117	60.681429
13	28.30419312	33.44228516	104.5578451	29.27214355	79.99108073	86.97571615	76.253442
14	23.98574015	21.4398112	100.0479492	26.70647583	75.00059408	90.91420085	76.062247

Figure E.2: Data exploration. Advanced page.



Algorithm settings

Parameters heavily influence on performance and solutions quality. These can be tweaked to better fit algorithm expectations.

Express. Recommended settings are used based on data.

Advanced. Recommended for expert users. Several Genetic Programming settings can be changed.

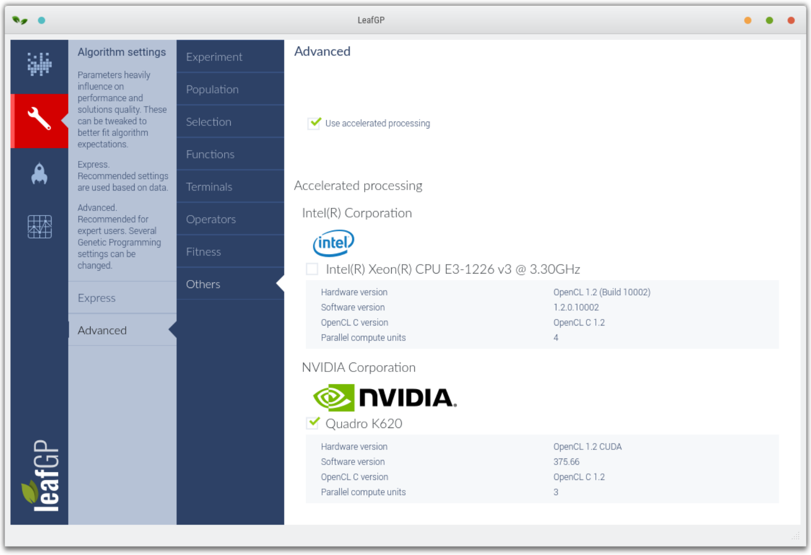
Express

Suggested parameters are used based on loaded data

Autotuning

Find optimal parameters

Figure E.3: Algorithm settings. Express page.



Algorithm settings

Parameters heavily influence on performance and solutions quality. These can be tweaked to better fit algorithm expectations.

Express. Recommended settings are used based on data.

Advanced. Recommended for expert users. Several Genetic Programming settings can be changed.

Advanced

☑ Use accelerated processing

Accelerated processing

Intel(R) Corporation

□ Intel(R) Xeon(R) CPU E3-1226 v3 @ 3.30GHz

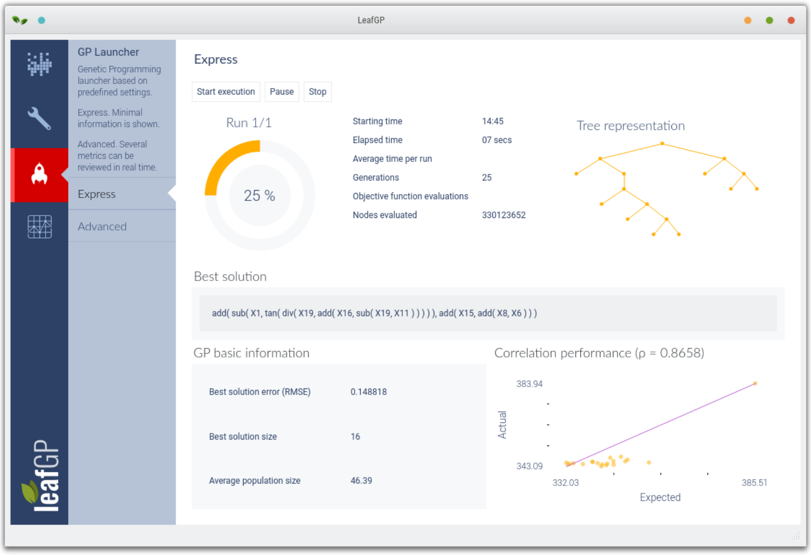
Hardware version	OpenCL 1.2 (Build 10002)
Software version	1.2.0.10002
OpenCL C version	OpenCL C 1.2
Parallel compute units	4

NVIDIA Corporation

☑ Quadro K620

Hardware version	OpenCL 1.2 CUDA
Software version	375.66
OpenCL C version	OpenCL C 1.2
Parallel compute units	3

Figure E.4: Algorithm settings. Advanced page.



GP Launcher

Genetic Programming launcher based on predefined settings.

Express. Minimal information is shown.

Advanced. Several metrics can be reviewed in real time.

Express

Starting time	14:45
Elapsed time	07 secs
Average time per run	
Generations	25
Objective function evaluations	
Nodes evaluated	330123652

Best solution

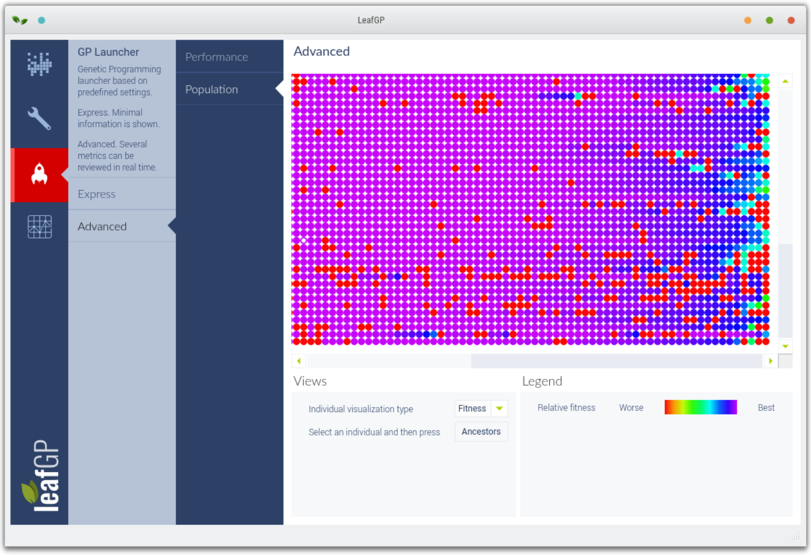
```
add( sub( X1, tan( div( X19, add( X16, sub( X19, X11 ) ) ) ) ), add( X15, add( X8, X6 ) ) )
```

GP basic information

Best solution error (RMSE)	0.148818
Best solution size	16
Average population size	46.39

Correlation performance (ρ = 0.8658)

Figure E.5: GP Launcher. Express page.



GP Launcher

Genetic Programming launcher based on predefined settings.

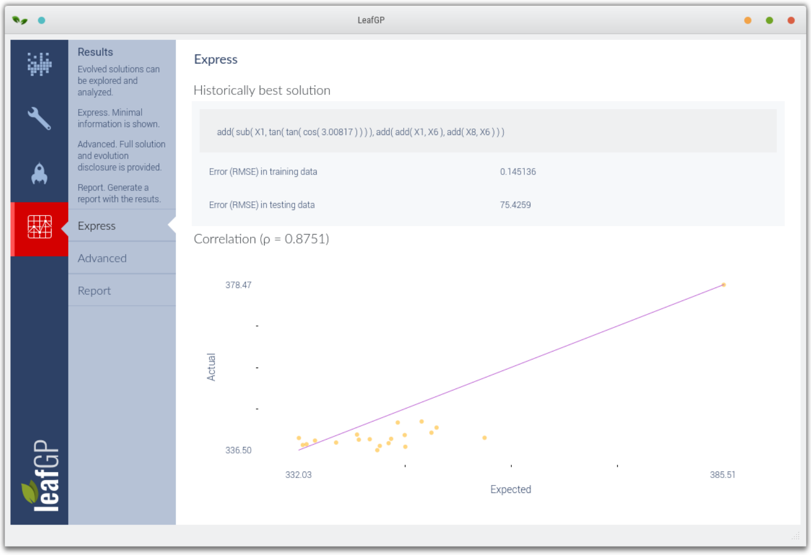
Express. Minimal information is shown.

Advanced. Several metrics can be reviewed in real time.

Individual visualization type: Fitness

Select an individual and then press Ancestors

Figure E.6: GP Launcher. Advanced page.



Results

Evolved solutions can be explored and analyzed.

Express. Minimal information is shown.

Advanced. Full solution and evolution disclosure is provided.

Report. Generate a report with the resuts.

Express

Advanced

Report

Express

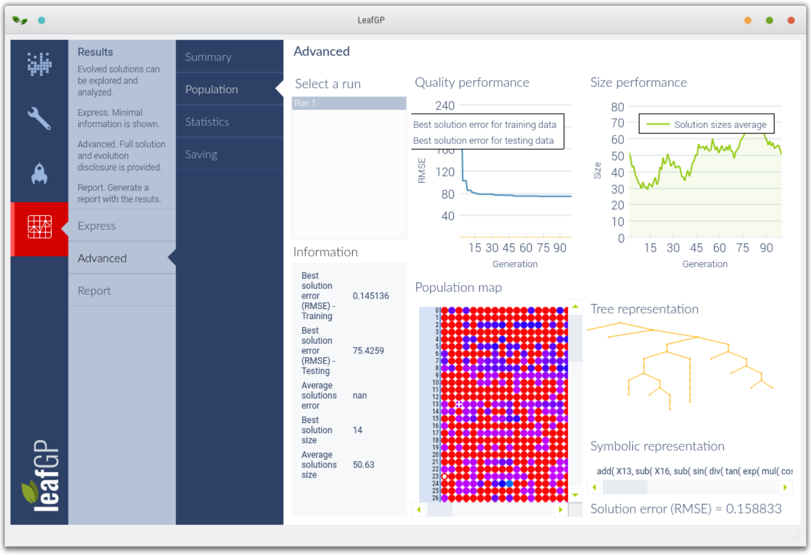
Historically best solution

add(sub(X1, tan(tan(cos(3.00817)))), add(add(X1, X6), add(X8, X6)))

Error (RMSE) in training data	0.145136
Error (RMSE) in testing data	75.4259

Correlation (ρ = 0.8751)

Figure E.7: Results. Express page.



Results

Evolved solutions can be explored and analyzed.

Express. Minimal information is shown.

Advanced. Full solution and evolution disclosure is provided.

Report. Generate a report with the resuts.

Express

Advanced

Report

Summary

Population

Statistics

Saving

Advanced

Select a run

Run 1

Information

Best solution error (RMSE) - Training	0.145136
Best solution error (RMSE) - Testing	75.4259
Average solutions error	nan
Best solution size	14
Average solutions size	50.63

Quality performance

240
Best solution error for training data
Best solution error for testing data
160
120
80
40
RMSE
15 30 45 60 75 90
Generation

Size performance

Population map

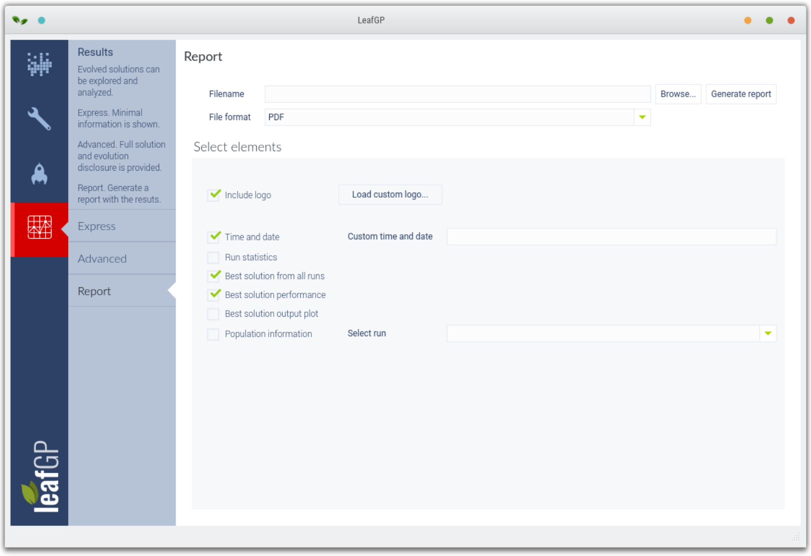
Tree representation

Symbolic representation

add(X13, sub(X16, sub(sin(div(tan(exp(mul(cos

Solution error (RMSE) = 0.158833

Figure E.8: Results. Advanced page.



Results

Evolved solutions can be explored and analyzed.

Express. Minimal information is shown.

Advanced. Full solution and evolution disclosure is provided.

Report. Generate a report with the resuts.

Report

Filename

File format PDF

Select elements

☑ Include logo

☑ Time and date — Custom time and date

□ Run statistics

☑ Best solution from all runs

☑ Best solution performance

□ Best solution output plot

□ Population information — Select run

Figure E.9: Results. Report page.

The source code has been licensed as LGPL 2.1. LeafGP is freely available in a github repository [16].

Because of the nature of Qt framework cross-platform scope, the code can be compiled in Linux, OSx and Windows. However, only the deployment in Linux environment has been tested. The source code has been compiled in a Intel Xeon@3.3GHz CPU with 4-cores, running Kubuntu 17.04 64-bits, Qt version 5.7.

E.3 CONCLUSIONS

In this work, a graphical open source GP tool for model extraction has been presented, which is based on Qt and GPOCL. Although this software is intended as a mid-long term development, it currently packs a number of basic features that make it user friendly. It contains data exploration, GP parameterization, real-time plotting and graphical results

[16] https://github.com/emigdioz/LeafGP

capabilities. Parallel processing is available depending on the installed hardware. Further development will require more resources, and given the open source characteristic of LeafGP, the authors actively encourage that any enthusiastic developer join the project and contribute, either by providing new ideas or simply by coding.

BIBLIOGRAPHY

Abásolo, D., Hornero, R., Espino, P., Poza, J., Sánchez, C. I., and de la Rosa, R. (2005). Analysis of regularity in the {EEG} background activity of alzheimer's disease patients with approximate entropy. *Clinical Neurophysiology*, 116(8):1826–1834.

Abatal, M. and Olguin, M. T. (2012). Comparative adsorption behavior between phenol and p-nitrophenol by na- and hdtma-clinoptilolite-rich tuff. *Environmental Earth Sciences*, 69(8):2691–698.

Abburi, K. (2003). Adsorption of phenol and p-chlorophenol from their single and bisolute aqueous solutions on amberlite xad-16 resin. *Journal of Hazardous Materials*, 105(1–3):143–156.

Abdulkader, S. N., Atia, A., and Mostafa, M. S. M. (2015). Brain computer interfacing: Applications and challenges. *Egyptian Informatics Journal*, 16(2):213–230.

Acharya, U. R., Fujita, H., Sudarshan, V. K., Bhat, S., and Koh, J. E. (2015). Application of entropies for automated diagnosis of epilepsy using EEG signals: A review. *Knowledge-Based Systems*, 88:85–96.

Acharya, U. R., Molinari, F., Sree, S. V., Chattopadhyay, S., Ng, K.-H., and Suri, J. S. (2012a). Automated diagnosis of epileptic EEG using entropies. *Biomedical Signal Processing and Control*, 7(4):401–408.

Acharya, U. R., Sree, S. V., Ang, P. C. A., Yanti, R., and Suri, J. S. (2012b). Application of Non-Linear and Wavelet Based Features for the Automated Identification of Epileptic Eeg Signals. *International Journal of Neural Systems*, 22(02):1–12.

Acharya, U. R., Vinitha Sree, S., Swapna, G., Martis, R. J., and Suri, J. S. (2013). Automated EEG analysis of epilepsy: A review. *Knowledge-Based Systems*, 45:147–165.

Aftanas, L. I. and Golocheikine, S. A. (2001). Human anterior and frontal midline theta and lower alpha reflect emotionally positive state and internalized attention: High-resolution EEG investigation of meditation. *Neuroscience Letters*, 310(1):57–60.

Ahammad, N., Fathima, T., and Joseph, P. (2014). Detection of epileptic seizure event and onset using EEG. *BioMed research international*, 2014.

Alam, S. M. S. and Bhuiyan, M. I. H. (2013). Detection of seizure and epilepsy using higher order statistics in the EMD domain. *IEEE Journal of Biomedical and Health Informatics*, 17(2):312–318.

Alasady, H. and Ibnkahla, M. (2008). Design and hardware implementation of look-up table predistortion on ALTERA Stratix DSP board. In *Canadian Conference on Electrical and Computer Engineering*, pages 1535–1538.

Alhola, P. and Plo-Kantola, P. (2007). Sleep deprivation: Impact on cognitive performance. *Neuropsychiatric disease and treatment*, 3(5):553–567.

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2nd edition.

Altenor, S., Carene, B., Emmanuel, E., Lambert, J., Ehrhardt, J.-J., and Gaspard, S. (2009). Adsorption studies of methylene blue and phenol onto vetiver roots activated carbon prepared by chemical activation. *Journal of Hazardous Materials*, 165(1–3):1029 – 1039.

Amiri, S., Fazel-rezai, R., and Asadpour, V. (2013). A Review of Hybrid Brain-Computer Interface Systems. *Advances in Human-Computer Interaction - Special issue on Using Brain Waves to Control Computers and Machines*, 2013.

Andrzejak, R. G., Lehnertz, K., Mormann, F., Rieke, C., David, P., and Elger, C. E. (2001). Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 64(6 Pt 1).

Arellano, G. M., Cant, R., and Nolle, L. (2014). Prediction of jet engine parameters for control design using genetic programming. In *Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on*, pages 45–50.

Arnaldo, I., Krawiec, K., and O'Reilly, U.-M. (2014). Multiple regression genetic programming. *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14*, pages 879–886.

A.Saidatul, Paulraj, S. Y. (2013). Mental Stress Level Classification Using Eigenvector Features and Principal Component Analysis. *Communications in Information Science and Management Engineering*, 3(5):254–261.

Asbeck, P., Kobayashi, H., Iwamoto, M., Hanington, G., Nam, S., and Larson, L. (2002). Augmented behavioral characterization for modeling the nonlinear response of power amplifiers. In *2002 IEEE MTT-S International Microwave Symposium Digest*, pages 135–138.

Augusto, D. A. and Barbosa, H. J. C. (2013). Accelerated parallel genetic programming tree evaluation with OpenCL. *Journal of Parallel and Distributed Computing*, 73(1):86–100.

Ayaz, H., Shewokis, P. A., Bunce, S., and Onaral, B. (2011). An optical brain computer interface for environmental control. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 025025, pages 6327–6330. IEEE.

Azad, R. and Ryan, C. (2014). A Simple Approach to Lifetime Learning in Genetic Programming-Based Symbolic Regression. *Evolutionary computation*, 22(2):287–317.

Baars, B. (1988). *A Cognitive Theory of Consciousness*. Cambridge University Press.

Babaelahi, M. and Sayyaadi, H. (2016). Analytical closed-form model for predicting the power and efficiency of stirling engines based on a comprehensive numerical model and the genetic programming. *Energy*, 98:324 – 339.

Babiloni, C., Pizzella, V., Gratta, C. D., Ferretti, A., and Romani, G. L. (2009). *Fundamentals of Electroencefalography, Magnetoencefalography, and Functional Magnetic Resonance Imaging*, volume 86. Elsevier Inc., 1 edition.

Baillet, S., Mosher, J. C., and Leahy, R. M. (2001). Electromagnetic brain mapping. *IEEE Signal Processing Magazine*, 18(6):14–30.

Bajaj, V. and Pachori, R. (2012). EEG signal classification using empirical mode decomposition and support vector machine. *Proceedings of the International Conference on Soft Computing*, pages 581–592.

Ball, T., Kern, M., Mutschler, I., Aertsen, A., and Schulze-Bonhage, A. (2009). Signal quality of simultaneously recorded invasive and non-invasive EEG. *NeuroImage*, 46(3):708–716.

Barhoumi, M., Beurroies, I., Denoyel, R., Saïd, H., and Hanna, K. (2003). Coadsorption of alkylphenols and nonionic surfactants onto kaolinite. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 219(1–3):25 – 33.

Bénar, C. G., Papadopoulo, T., Torrésani, B., and Clerc, M. (2009). Consensus Matching Pursuit for multi-trial EEG signals. *Journal of Neuroscience Methods*, 180(1):161–170.

Bennadji, A. (2006). *Implémentation de modèles comportementaux d'amplificateurs de puissance dans des environnements de simulation système et co-simulation circuit-système*. PhD thesis, Universite de Limoges. Thèse de doctorat Électronique des hautes fréquences et optoélectronique. Communications optiques et micro-ondes Limoges 2006.

Bhardwaj, A., Tiwari, A., Krishna, R., and Varma, V. (2016). A novel genetic programming approach for epileptic seizure detection. *Computer Methods and Programs in Biomedicine*, 124:2–18.

Bhardwaj, A., Tiwari, A., Varma, M. V., and Krishna, M. R. (2014). Classification of EEG signals using a novel genetic programming approach. *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14*, pages 1297–1304.

Bhowan, U., Johnston, M., and Zhang, M. (2012). Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Trans. on Systems, Man, and Cybernetics*, 42(2):406–21.

Biglieri, E., Barberis, S., and Catena, M. (1988). Analysis and compensation of nonlinearities in digital transmission systems. *IEEE Journal on Selected Areas in Communications*, 6(1):42–51.

Binnie, C. D. and Prior, P. F. (1994). Electroencephalography. *Journal of neurology, neurosurgery, and psychiatry*, 57(11):1308–1319.

Black, A. (1972). The Operant Conditioning of Central Nervous System Electrical Activity. In *Psychology of Learning and Motivation*, volume 6, pages 47–95. Academic Press.

Blankertz, B., Tomioka, R., Lemm, S., Kawanabe, M., and Muller, K.-r. (2008). Optimizing Spatial filters for Robust EEG Single-Trial Analysis. *IEEE Signal Processing Magazine*, 25(1):41–56.

Breiman, L. (2001). Random forests. *Machine learning*, pages 5–32.

Brown, P., Salenius, S., Rothwell, J. C., and Hari, R. (1998). Cortical correlate of the piper rhythm in humans. *Journal of Neurophysiology*, 80(6):2911–2917.

Cao, H., Kang, L., Chen, Y., and Yu, J. (2000). Evolutionary modeling of systems of ordinary differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, 1(4):309–337.

Caplan, J. B., Madsen, J. R., Raghavachari, S., Michael, J., Herz, D. M., Florin, E., Christensen, M. S., Reck, C., Thomas, M., Tscheuschler, M. K., Tittgemeyer, M., Siebner, H. R., Marzano, C., Ferrara, M., Mauro, F., Moroni, F., Gorgoni, M., Cipolli, C., Gennaro, L. D., and Kahana, M. J. (2001). Distinct Patterns of Brain Oscillations Underlie Two Basic Parameters of Human Maze Learning. *J Neurophysiol*, 86:368–380.

Cárdenas Valdez, J. R., Z-Flores, E., Núñez Pérez, J. C., and Trujillo, L. (2017). Local Search Approach to Genetic Programming for RF-PAs Modeling Implemented in FPGA. In *NEO 2015*, Studies in Computational Intelligence, pages 67–88. Springer.

Carruthers, M. (1979). Autogenic training. *Journal of Psychosomatic Research*, 23(6):437 – 440.

Castelli, M., Silva, S., and Vanneschi, L. (2015a). A c++ framework for geometric semantic genetic programming. *Genetic Programming and Evolvable Machines*, 16(1):73–81.

Castelli, M., Trujillo, L., and Vanneschi, L. (2015b). Energy consumption forecasting using semantic-based genetic programming with local search optimizer. *Intell. Neuroscience*, 2015.

Castelli, M., Trujillo, L., Vanneschi, L., and Popovič, A. (2015c). Prediction of energy performance of residential buildings: A genetic programming approach. *Energy and Buildings*, 102:67–74.

Castelli, M., Trujillo, L., Vanneschi, L., and Popovič, A. (2015d). Prediction of relative position of CT slices using a computational intelligence system. *Applied Soft Computing*.

Castelli, M., Trujillo, L., Vanneschi, L., Silva, S., Z-Flores, E., and Legrand, P. (2015e). Geometric Semantic Genetic Programming with Local Search. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 999–1006. ACM.

Castelli, M., Vanneschi, L., and Felice, M. D. (2015f). Forecasting short-term electricity consumption using a semantics-based genetic programming framework: The south italy case. *Energy Economics*, 47(0):37 – 41.

Castelli, M., Vanneschi, L., and Popovič, A. (2015g). Parameter evaluation of geometric semantic genetic programming in pharmacokinetics. *International Journal of Bio-Inspired Computation*, pages 1–9. To appear.

Castelli, M., Vanneschi, L., and Silva, S. (2013). Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators. *Expert Systems with Applications*, 40(17):6856–6862.

Cava, W. L., Danai, K., Spector, L., Fleming, P., Wright, A., and Lackner, M. (2016). Automatic identification of wind turbine models using evolutionary multiobjective optimization. *Renewable Energy*, 87, Part 2:892 – 902.

CFE (1998). Introducción a centrales turbogas. Technical report, Central Escuela Celaya, Comisión Federal de Electricidad, Celaya, Mexico.

Chaquet, J. M., Carmona, E. J., and Corral, R. (2012). Using genetic algorithms to improve the thermodynamic efficiency of gas turbines designed by traditional methods. *Appl. Soft Comput.*, 12(11):3627–3635.

Chen, G. (2014). Automatic EEG seizure detection using dual-tree complex wavelet-Fourier features. *Expert Systems with Applications*, 41(5):2391–2394.

Chen, S., Donoho, D., and Saunders, M. (1998). Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, 43(1):129–159.

Chen, W., Wang, Y., Cao, G., Chen, G., and Gu, Q. (2014). A random forest model based classification scheme for neonatal amplitude-integrated EEG. *BioMedical Engineering OnLine*, 13(Suppl 2):1–13.

Chen, X., Ong, Y.-S., Lim, M.-H., and Tan, K. C. (2011). A multi-facet survey on memetic computation. *Evolutionary Computation, IEEE Transactions on*, 15(5):591–607.

Chuckravanen, D. (2014). Approximate Entropy as a Measure of Cognitive Fatigue: An EEG Pilot Study. *International Journal of Emerging Trends in Science and Technology*, pages 1036–1042.

Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Coleman, T. F. and Li, Y. (1992). On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds.

Coleman, T. F. and Li, Y. (1993). An interior trust region approach for nonlinear minimization subject to bounds. Technical report, Office of Energy Research, Ithaca, NY, USA.

Coyle, S. M., Ward, T. E., Markham, C. M., Ward, E., and Markham, C. M. (2007). Brain-computer interface using a simplified functional near-infrared spectroscopy system. *Journal of neural engineering*, 4(3):219–26.

Crone, N. E., Miglioretti, D. L., Gordon, B., Lesser, R. P., and Crone, N. (1998). Functional mapping of human sensorimotor cortex with electrocorticographic spectral analysis II. Event-related synchronization in the gamma band. *Brain*, 121(August 2016):2301–2315.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 2933–2941. MIT Press.

De Jong, K. (2006). *Evolutionary Computation: A Unified Approach*. Bradford Book. Mit Press.

De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). Deap: A python framework for evolutionary algorithms. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 85–92. ACM.

Delgadillo, M. and Hernandez, M. (2002). Modelling and dynamic simulation of gas turbine. In *Proceedings 45th Annual ISA-POWID Conference*, ISA-POWID 2002.

DePold, H. R. and Gass, F. D. (1999). The application of expert systems and neural networks to gas turbine prognostics and diagnostics. *J. Eng. Gas Turbines Power*, 121(4):607–612.

Dick, O. E. and Svyatogor, I. a. (2012). Potentialities of the wavelet and multifractal techniques to evaluate changes in the functional state of the human brain. *Neurocomputing*, 82:207–215.

Dil, E. A., Ghaedi, M., Ghaedi, A., Asfaram, A., Jamshidi, M., and Purkait, M. K. (2015). Application of artificial neural network and response surface methodology for the removal of crystal violet by zinc oxide nanorods loaded on activate carbon: kinetics and equilibrium study. *Journal of the Taiwan Institute of Chemical Engineers*, pages –. in press.

Divya, S. (2015). Classification of EEG Signal for Epileptic Seizure Detection using EMD and ELM. *International Journal for Trends in Engineering and Technology*, 3(2):68–74.

Do, D. (1998). *Adsorption analysis: equilibria and kinetics*. Imperial College Press.

Dong, Y., Hu, Z., Uchimura, K., and Murayama, N. (2011). Driver inattention monitoring system for intelligent vehicles: A review. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):596–614.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience.

Dufourq, E. and Pillay, N. (2013). A comparison of genetic programming representations for binary data classification. In *Information and Communication Technologies (WICT), 2013 Third World Congress on*, pages 134–140.

Dunn, E., Olague, G., and Lutton, E. (2006). Parisian camera placement for vision metrology. *Pattern Recogn. Lett.*, 27(11):1209–1219.

Durka, P. (2004). Adaptive time-frequency parametrization of epileptic spikes. *Physical Review E.*

Durka, P. and Blinowska, K. (1995). Analysis of eeg transients by means of matching pursuit. *Annals of Biomedical Engineering*, 23(5):608–611.

Durka, P., Ircha, D., and Blinowska, K. (2001). Stochastic time-frequency dictionaries for matching pursuit. *IEEE Transactions on Signal Processing*, 49(3):507–510.

Durka, P. J., Matysiak, A., Montes, E. M., Sosa, P. V., and Blinowska, K. J. (2005). Multichannel matching pursuit and EEG inverse solutions. *Journal of Neuroscience Methods*, 148(1):49–59.

Eadie, M. J. (2012). Shortcomings in the current treatment of epilepsy. *Expert Review of Neurotherapeutics*, 12(12):1419–1427.

Eggermont, J., Eiben, A. E., and Hemert, J. I. (1999). *Genetic Programming: Second European Workshop, EuroGP'99 Göteborg, Sweden, May 26–27, 1999 Proceedings*, chapter Adapting the Fitness Function in GP for Data Mining, pages 193–202. Springer Berlin Heidelberg.

Eggermont, J., Kok, J. N., and Kosters, W. A. (2004). Genetic programming for data classification: Partitioning the search space. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, pages 1001–1005. ACM.

Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing, 2nd Edition.* Springer Verlag.

Else, T. and Hammer, G. D. (2013). *Disorders of the Adrenal Cortex*, chapter 21. McGraw-Hill Education, New York, NY.

Elton, D., Burrows, G. D., and Stanley, G. V. (1978). Relaxation Theory and Practice. *Australian Journal of Physiotherapy*, 24(3):143–149.

Emmerich, M., Grötzner, M., and Schütz, M. (2001). Design of graph-based evolutionary algorithms: A case study for chemical process networks. *Evol. Comput.*, 9(3):329–354.

Enríquez-Zárate, J., Trujillo, L., de Lara, S., Castelli, M., Z-Flores, E., Muñoz, L., and Popovič, A. (2017). Automatic modeling of a gas turbine using genetic programming: An experimental study. *Applied Soft Computing*, 50:212–222.

Erguzel, T. T., Ozekes, S., Gultekin, S., and Tarhan, N. (2014). Ant colony optimization based feature selection method for QEEG data classification. *Psychiatry Investigation*, 11(3):243–250.

Fan, S., Yeh, J., Chen, B., and Shieh, J. (2011). Comparison of eeg approximate entropy and complexity measures of depth of anaesthesia during inhalational general anaesthesia. *Journal of Medical and Biological Engineering*, 31(5):359–366.

Fang, C., Li, H., and Ma, L. (2013). EEG Signal Classification Using the Event-Related Coherence and Genetic Algorithm. In *Advances in Brain Inspired Cognitive Systems*, pages 92–100. Springer.

Faust, O., Acharya, U. R., Adeli, H., and Adeli, A. (2015). Wavelet-based EEG Processing for Computer-aided Seizure Detection and Epilepsy Diagnosis. *Seizure*, 26:56–64.

Fernández, T., Harmony, T., Rodríguez, M., Bernal, J., Silva, J., Reyes, A., and Marosi, E. (1995). EEG activation patterns during the performance of tasks involving different components of mental calculation. *Electroencephalography and Clinical Neurophysiology*, 94(3):175–182.

Fernández-Blanco, E., Rivero, D., Gestal, M., and Dorado, J. (2013). Classification of signals by means of Genetic Programming. *Soft Computing*, 17(10):1929–1937.

Figliola, A. and Serrano, E. (2007). Study of EEG brain maturation signals with multifractal detrended fluctuation analysis. *XV Conference on Nonequilibrium Statistical Mechanics and Nonlinear Physics*, (6 mm):190–195.

Firpi, H., Goodman, E., and Echauz, J. (2006). On prediction of epileptic seizures by means of genetic programming artificial features. *Annals of Biomedical Engineering*, 34(3):515–529.

Fisher, R. S., Acevedo, C., Arzimanoglou, A., Bogacz, A., Cross, J. H., Elger, C. E., Engel, J., Forsgren, L., French, J. a., Glynn, M., Hesdorffer, D. C., Lee, B. I., Mathern, G. W., Moshé, S. L., Perucca, E., Scheffer, I. E., Tomson, T., Watanabe, M., and Wiebe, S. (2014). ILAE official report: a practical clinical definition of epilepsy. *Epilepsia*, 55(4):475–82.

Fisher, R. S. and Schachter, S. C. (2000). The postictal state: a neglected entity in the management of epilepsy. *Epilepsy & Behavior*, 1(1):52–59.

Fonseca, C., Silva Cunha, J. P., Martins, R. E., Ferreira, V. M., Marques De Sá, J. P., Barbosa, M. A., and Martins Da Silva, A. (2007). A novel dry active electrode for EEG recording. *IEEE Transactions on Biomedical Engineering*, 54(1):162–165.

Franaszczuk, P. J., Bergey, G. K., Durka, P. J., and Eisenberg, H. M. (1998). Time–frequency analysis using the matching pursuit algorithm applied to seizures originating from the mesial temporal lobe. *Electroencephalography and Clinical Neurophysiology*, 106(6):513–521.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–67.

Gandhi, T. K., Chakraborty, P., Roy, G. G., and Panigrahi, B. K. (2012). Discrete harmony search based expert model for epileptic seizure detection in electroencephalography. *Expert Systems with Applications*, 39(4):4055–4062.

Garcés Correa, A., Orosco, L., and Laciar, E. (2014). Automatic detection of drowsiness in EEG records based on multimodal analysis. *Medical Engineering and Physics*, 36(2):244–249.

Ghaedi, M., Ansari, A., Bahari, F., Ghaedi, A., and Vafaei, A. (2015a). A hybrid artificial neural network and particle swarm optimization for prediction of removal of hazardous dye brilliant green from aqueous solution using zinc sulfide nanoparticle loaded on activated carbon. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 137:1004 – 1015.

Ghaedi, M., Daneshfar, A., Ahmadi, A., and Momeni, M. (2015b). Artificial neural network-genetic algorithm based optimization for the adsorption of phenol red (pr) onto gold and titanium dioxide nanoparticles loaded on activated carbon. *Journal of Industrial and Engineering Chemistry*, 21:587 – 598.

Ghazvini, A., Awwalu, J., and Bakar, A. A. (2014). Comparative Analysis of Algorithms in Supervised Classification : A Case study of Bank Notes Dataset. *International Journal of Computer Trends and Technology*, 17(1):39–43.

Gill, P. E., Murray, W., and Wright, M. H. (1981). *Practical optimization*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

Graff, M., Peña, R., and Medina, A. (2013). Wind speed forecasting using genetic programming. In *2013 IEEE Congress on Evolutionary Computation*, pages 408–415.

Grizou, J., Iturrate, I., Montesano, L., Oudeyer, P.-Y., and Lopes, M. (2014). Calibration-Free BCI Based Control. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1–8, Quebec, Canada.

Guler, I. and Ubeyli, E. (2007). Multiclass Support Vector Machines for EEG-Signals Classification. *IEEE Transactions on Information Technology in Biomedicine*, 11(2):117–126.

Güler, I. and Ubeyli, E. D. (2005). Adaptive neuro-fuzzy inference system for classification of EEG signals using wavelet coefficients. *Journal of neuroscience methods*, 148:113–121.

Guler, N., Ubeyli, E., and Guler, I. (2005). Recurrent neural networks employing Lyapunov exponents for EEG signals classification. *Expert Systems with Applications*, 29(3):506–514.

Guo, L., Rivero, D., Dorado, J., Munteanu, C. R., and Pazos, A. (2011). Automatic feature extraction using genetic programming: An application to epileptic EEG classification. *Expert Systems with Applications*, 38(8):10425–10436.

Guo, L., Rivero, D., and Pazos, A. (2010). Epileptic seizure detection using multiwavelet transform based approximate entropy and artificial neural networks. *Journal of neuroscience methods*, 193(1):156–63.

Gupta, A., Agrawal, R. K., and Kaur, B. (2015). Performance enhancement of mental task classification using EEG signal: a study of multivariate feature selection methods. *Soft Computing*, 19(10):2799–2812.

Gusel, L. and Brezocnik, M. (2011). Application of genetic programming for modelling of material characteristics. *Expert Systems with Applications*, 38(12):15014 – 15019.

Hajinoroozi, M., Mao, Z., and Huang, Y. (2016). Prediction of driver's drowsy and alert states from EEG signals with deep learning. *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, CAMSAP 2015*, pages 493–496.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Hanna, K., Beurroies, I., Denoyel, R., Desplantier-Giscard, D., Galarneau, A., and Renzo, F. D. (2002). Sorption of hydrophobic molecules by organic/inorganic mesostructures. *J Colloid Interface Sci.*, 252(2):276–283.

Hassani, K. and Lee, W.-s. (2014). An Incremental Framework for Classification of EEG Signals Using Quantum Particle Swarm Optimization. *IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pages 40–45.

Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., and Tibshirani, R. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer.

Hirtz, D., Thurman, D. J., Gwinn-Hardy, K., Mohamed, M., Chaudhuri, a. R., and Zalutsky, R. (2007). How common are the ”common” neurologic disorders? *Neurology*, 68(5):326–337.

Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D., and Donoghue, J. P. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–71.

Hongxia, L., Hongxi, D., Jian, L., and Shuicheng, T. (2016). Research on the application of the improved genetic algorithm in the electroencephalogram-based mental workload evaluation for miners. *Journal of Algorithms & Computational Technology*, 0(0):1–10.

Hornby, G. S., Lohn, J. D., and Linden, D. S. (2011). Computer-automated evolution of an x-band antenna for nasa’s space technology 5 mission. *Evol. Comput.*, 19(1):1–23.

Horváth, G. and Kawazoe, K. (1983). Method for calculation of effective pore size distribution in molecular sieve carbon. *Journal of Chemical Engineering of Japan*, 16(6):470–475.

Hussain, Z. and Shawe-taylor, J. (2009). Theory of matching pursuit. *Advances in Neural Information Processing Systems*, pages 1–8.

Jaffard, S. (2004). Wavelet techniques in multifractal analysis. *Proceedings of symposia in pure mathematics*.

Jaffard, S. and Meyer, Y. (1996). *Wavelet methods for pointwise regularity and local oscillations of functions*. Providence, R.I. : American Mathematical Society.

Jaiantilal, A. (2012). *RF Matlab interface, Version 0.02*.

Jain, A., Duin, R. P. W., and Mao, J. (2000). Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37.

Jaromir, J. and Varbanov, P. (2016). *Process integration: pinch analysis and mathematical programming- directions for future development*, volume 38, pages 2405–2406.

Jayadeva (2015). Learning a hyperplane classifier by minimizing an exact bound on the {VC} dimension1. *Neurocomputing*, 149, Part B:683 – 689.

Jurcak, V., Tsuzuki, D., and Dan, I. (2007). 10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems. *NeuroImage*, 34(4):1600–1611.

Kamath, C. (2015). Analysis of EEG Dynamics in Epileptic Patients and Healthy Subjects Using Hilbert Transform Scatter Plots. *OALib*, 02:1–14.

Kantelhardt, J. W., Zschiegner, S. a., Koscielny-Bunde, E., Havlin, S., Bunde, A., and Stanley, H. E. (2002). Multifractal detrended fluctuation analysis of nonstationary time series. *Physica A: Statistical Mechanics and its Applications*, 316(1-4):87–114.

Karimi, H. and Ghaedi, M. (2014). Application of artificial neural network and genetic algorithm to modeling and optimization of removal of methylene blue using activated carbon. *Journal of Industrial and Engineering Chemistry*, 20(4):2471 – 2476.

Keijzer, M. (2003). Improving symbolic regression with interval arithmetic and linear scaling. In *Proceedings of the 6th European Conference on Genetic Programming*, EuroGP'03, pages 70–82, Berlin, Heidelberg. Springer-Verlag.

Kenington, P. B. (2000). *High Linearity RF Amplifier Design*. Artech House, Inc., Norwood, MA, USA, 1st edition.

Klem, G., Luders, H., Jasper, H., and Elger, C. (1958). The ten-twenty electrode system of the International Federation. *Electroencephalography and Clinical Neurophysiology*, 10(2):371–375.

Klemeš, J. J., Lam, H. L., and Foo, D. C. Y. (2011). *Water Integration for Recycling and Recovery in Process Industry*, pages 1–12. Springer Netherlands, Dordrecht.

Klimesch, W. (1997). EEG-alpha rhythms and memory processes. *International Journal of Psychophysiology*, 26(1-3):319–340.

Klimesch, W., Doppelmayr, M., Yonelinas, A., Kroll, N. E. A., Lazzara, M., Röhm, D., and Gruber, W. (2001). Theta synchronization during episodic retrieval: Neural correlates of conscious awareness. *Cognitive Brain Research*, 12(1):33–38.

Kohsaka, S., Mizukami, S., Kohsaka, M., Shiraishi, H., and Kobayashi, K. (2002). Widespread activation of the brainstem preceding the recruiting rhythm in human epilepsies. *Neuroscience*, 115(3):697–706.

Kommenda, M., Kronberger, G., Winkler, S., Affenzeller, M., and Wagner, S. (2013). Effects of constant optimization by nonlinear least squares minimization in symbolic regression. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, pages 1121–1128. ACM.

Korns, M. F. (2011). Accuracy in symbolic regression. In Riolo, R., Vladislavleva, E., and Moore, J. H., editors, *Genetic Programming Theory and Practice IX*, Genetic and Evolutionary Computation, chapter 8, pages 129–151. Springer, Ann Arbor, USA.

Koshiyama, A. S., Escovedo, T., Dias, D. M., Vellasco, M. M. B. R., and Tanscheit, R. (2013). Gpf-class: A genetic fuzzy model for classification. In *2013 IEEE Congress on Evolutionary Computation*, pages 3275–3282.

Koubeissi, M. Z., Jouny, C. C., Blakeley, J. O., and Bergey, G. K. (2009). Analysis of dynamics and propagation of parietal cingulate seizures with secondary mesial temporal involvement. *Epilepsy and Behavior*, 14(1):108–112.

Kovačič, M. and Dolenc, F. (2016). Prediction of the natural gas consumption in chemical processing facilities with genetic programming. *Genetic Programming and Evolvable Machines*, pages 1–19.

Kovacs, P., Samiee, K., and Gabbouj, M. (2014). On application of rational discrete short time fourier transform in epileptic seizure classification. *Acoustics, Speech and Signal*, pages 5880–5884.

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

Koza, J. R. (2010). Human-competitive Results Produced by Genetic Programming. *Genetic Programming and Evolvable Machines*, 11(3-4):251–284.

Ku, H. and Kenney, J. S. (2003). Behavioral modeling of nonlinear RF power amplifiers considering memory effects. *IEEE Transactions on Microwave Theory and Techniques*, 51(12):2495–2504.

Kübler, A., Kotchoubey, B., Kaiser, J., Wolpaw, J. R., and Birbaumer, N. (2001). Brain-computer communication: unlocking the locked in. *Psychological bulletin*, 127(3):358–375.

Kumar, A. and Jena, H. M. (2016). Removal of methylene blue and phenol onto prepared activated carbon from fox nutshell by chemical activation in batch and fixed-bed column. *Journal of Cleaner Production*, 137:1246 – 1259.

Kumar, Y., Dewal, M., and Anand, R. (2014). Epileptic seizure detection using DWT based fuzzy approximate entropy and support vector machine. *Neurocomputing*, 133:271–279.

Kumari, S. and Prabin, J. (2011). Seizure Detection in EEG Using Time Frequency Anlysis and SVM. *2011 International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT)*, pages 626–630.

Langdon, W. and Poli, R. (2001). *Foundations of Genetic Programming*. Springer, Berlin, Heidelberg, New York.

Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming*. Springer-Verlag.

Laureys, S. (2005). The neural correlate of (un)awareness: Lessons from the vegetative state. *Trends in Cognitive Sciences*, 9(12):556–559.

Laureys, S., Boly, M., and Tononi, G. (2009). Functional neuroimaging. In *The Neurology of Consciousness*, pages 31 – 42. Academic Press, San Diego.

Lawson, C. L. and Hanson, R. J. (1995). *Solving least squares problems*, volume 15 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM).

Leach, W. M. (1994). Fundamentals of Low-Noise Analog Circuit Design. *Proceedings of the IEEE*, 82(10):1515–1538.

Lee, C. T., Hashim, H., Ho, C. S., Fan, Y. V., and Klemeš, J. J. (2016). Sustaining the low-carbon emission development in asia and beyond: Sustainable energy, water, transportation and low-carbon emission technology. *Journal of Cleaner Production*, pages –.

Lee, K. H., Williams, L. M., Breakspear, M., and Gordon, E. (2003). Synchronous Gamma activity: A review and contribution to an integrative neuroscience model of schizophrenia. *Brain Research Reviews*, 41(1):57–78.

Legrand, P. (2004). *Débruitage et interpolation par analyse de la régularité Hölderienne. Application à la modélisation du frottement pneumatique-chaussée.* Theses, Ecole Centrale de Nantes (ECN); Université de Nantes.

Legrand, P. and Vehel, J. (2003). Local regularity-based image denoising. *Proceedings 2003 International Conference on Image Processing*, 3(1):0–3.

Lei, D. (2004). *Digital predistortion of power amplifiers for wireless applications.* PhD thesis, Georgia Institute of Technology.

Li, P. H. and Wang, P. (2010). Wiener-saleh modeling of nonlinear RF power amplifiers considering memory effects. In *2010 International Conference on Microwave and Millimeter Wave Technology, ICMMT 2010*, pages 1447–1449.

Li, Y.-H. and Wei, X.-k. (2006). Linear-in-parameter models based on parsimonious genetic programming algorithm and its application to aero-engine start modeling. *Chinese Journal of Aeronautics*, 19(4):295 – 303.

Lichman, M. (2013). UCI machine learning repository.

Lima, C. a. M., Coelho, A. L. V., and Eisencraft, M. (2010). Tackling EEG signal classification with least squares support vector machines: A sensitivity analysis study. *Computers in Biology and Medicine*, 40:705–714.

Lin, C.-H., Chen, H.-Y., and Wu, Y.-S. (2014). Study of image retrieval and classification based on adaptive features using genetic algorithm feature selection. *Expert Systems with Applications*, 41(15):6611 – 6621.

Lin, C.-J. and Hsieh, M.-H. (2009). Classification of mental task from EEG data using neural networks based on particle swarm optimization. *Neurocomputing*, 72(4-6):1121–1130.

Lippens, B. and de Boer, J. (1965). Studies on pore systems in catalysts: V. the t method. *Journal of Catalysis*, 4(3):319––323.

Little, M., McSharry, P., Roberts, S., Costello, D., and Moroz, I. (2007). Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine*, 6(1).

Liu, Y. J., Zhou, J., Chen, W., and Zhou, B. H. (2014). A robust augmented complexity-reduced generalized memory polynomial for wideband RF power amplifiers. *IEEE Transactions on Industrial Electronics*, 61(5):2389–2401.

Lohmann, R. (1991). Proceedings of parallel problem solving from nature (ppsn i) – first workshop. In *Proceedings from the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 496 of *LNCS*, pages 198–208. Springer-Verlag.

Lotte, F., Congedo, M., Lécuyer, A., Lamarche, F., and Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain-computer interfaces. *Journal of neural engineering*, 4(2):R1–R13.

Luke, S. (2013). *Essentials of Metaheuristics*. Lulu, second edition. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

Lutzenberger, W., Pulvermüller, F., Elbert, T., and Birbaumer, N. (1995). Visual stimulation alters local 40-Hz responses in humans: an EEG-study. *Neuroscience Letters*, 183(1-2):39–42.

Ma, C., OuYang, J., Chen, H.-L., and Zhao, X. (2014). An efficient diagnosis system for parkinson's disease using kernel-based extreme learning machine with subtractive clustering features weighting approach. *Comp. Math. Methods in Medicine*, 2014:985789:1–985789:14.

Ma, Q., Ning, X., Wang, J., and Li, J. (2006). Sleep-stage characterization by nonlinear EEG analysis using wavelet-based multifractal formalism. *IEEE Engineering in Medicine and Biology Society*, pages 4526–4529.

Mallat, S. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415.

Mallat, S. (2008). *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd edition.

Mallat, S. and Hwang, W. L. (1992). Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38(2 pt II):617–643.

Mallikarjun, H. M., Suresh, H. N., and Manimegalai, P. (2016). Mental State Recognition by using Brain Waves. *Indian Journal of Science and Technology*, 9(33):2–6.

Martis, R. J., Acharya, U. R., Tan, J. H., Petznick, A., Tong, L., Chua, C. K., and NG, E. Y. K. (2013). Application of Intrinsic Time-Scale Decomposition (Itd) To Eeg Signals for Automated Seizure Prediction. *International Journal of Neural Systems*, 23(05):1–13.

Mathuvanesan, C. and Jayasankar, T. (2013). Performance Analysis Of Singularity And Irregular Detection In Human Health Monitoring Using Lipschitz Exponent Function. 2(6):414–418.

MATLAB (2014). *version 8.3 (R2014a)*. The MathWorks Inc., Natick, Massachusetts.

MATLAB (2015). *version 8.5 (R2015a)*. The MathWorks Inc., Natick, Massachusetts.

McConaghy, T. (2011). *FFX: Fast, Scalable, Deterministic Symbolic Regression Technology*, chapter Genetic Programming Theory and Practice IX, pages 235–260. Springer New York.

McDermott, J., White, D. R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., and O'Reilly, U.-M. (2012). Genetic programming needs better benchmarks. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pages 791–798, New York, NY, USA. ACM.

Mikaili, M. and Golpayegani, S. M. R. H. (2002). Assessment of the complexity/regularity of transient brain waves (eeg) during sleep, based on wavelet theory and the concept of entropy. *Iranian Journal of Science and Technology*, 26(B4):639–646.

Mima, T., Simpkins, N., Oluwatimilehin, T., and Hallett, M. (1999). Force level modulates human cortical oscillatory activities. *Neuroscience Letters*, 275(2):77–80.

Misic, J., Markovic, V., and Marinkovic, Z. (2014). Volterra kernels extraction from neural networks for amplifier behavioral modeling. In *2014 X International Symposium on Telecommunications (BIHTEL)*, pages 1–6. IEEE.

Mittman, C. B. and Cooper, D. W. (2014). Computer Chess Programs. *ACM Computing Surveys*, 18(5):779–789.

Mkadem, F., Ayed, M. B., Boumaiza, S., Wood, J., and Aaen, P. (2010). Behavioral modeling and digital predistortion of Power Amplifiers with memory using Two Hidden Layers Artificial Neural Networks. In *2010 IEEE MTT-S International Microwave Symposium*, pages 656–659.

Moon, J., Saad, P., Son, J., Fager, C., and Kim, B. (2012). 2-d enhanced hammerstein behavior model for concurrent dual-band power amplifiers. In *Microwave Conference (EuMC), 2012 42nd European*, pages 1249–1252.

Moraglio, A., Krawiec, K., and Johnson, C. G. (2012). Geometric semantic genetic programming. *Lecture Notes in Computer Science (includ-*

ing subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7491 LNCS(PART 1):21–31.

Moré, J. J. and Sorensen, D. C. (1983). Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572.

Morlini, I., Minerva, T., and Vichi, M. (2015). *Advances in statistical models for data analysis*, volume 50. Kluwer Academic Publishers.

Muñoz, L., Silva, S., and Trujillo, L. (2015). *M3GP – Multiclass Classification with GP*, pages 78–91. Springer International Publishing.

Murugavel, A. S. M. and Ramakrishnan, S. (2014). An Optimized Extreme Learning Machine for Epileptic Seizure Detection. *International Journal of Computer Science*, (November):212–221.

Myrden, A. and Chau, T. (2015). Effects of user mental state on EEG-BCI performance. *Frontiers in Human Neuroscience*, 9(June):308.

Naitoh, P., Johnson, L. C., and Lubin, A. (1971). Modification of surface negative slow potential (CNV) in the human brain after total sleep loss. *Electroencephalography and Clinical Neurophysiology*, 30(1):17–22.

Nakai, K., Sonoda, J., Kondo, S., and Abe, I. (1993). The analysis of surface and pores of activated carbons by the adsorption of various kinds of gases. In Suzuki, M., editor, *Fundamentals of Adsorption, Proceedings of the Fourth International Conference on Fundamentals of Adsorption*, volume 80 of *Studies in Surface Science and Catalysis*, pages 461–466. Elsevier.

Naredo, E., Trujillo, L., Legrand, P., Silva, S., and Muñoz, L. (2016). Evolving genetic programming classifiers with novelty search. *Information Sciences*, 369:347–367.

Natarajan, K., Acharya U, R., Alias, F., Tiboleng, T., and Puthusserypady, S. K. (2004). Nonlinear analysis of EEG signals at different mental states. *Biomedical engineering online*, 3(1):1–11.

Nia, R. H., Ghaedi, M., and Ghaedi, A. (2014). Modeling of reactive orange 12 (ro 12) adsorption onto gold nanoparticle-activated carbon using artificial neural network optimization based on an imperialist competitive algorithm. *Journal of Molecular Liquids*, 195:219 – 229.

Nicolaou, N. and Georgiou, J. (2012). Detection of epileptic electroencephalogram based on Permutation Entropy and Support Vector Machines. *Expert Systems with Applications*, 39(1):202–209.

Nicolas-Alonso, L. F. and Gomez-Gil, J. (2012). Brain computer interfaces, a review. *Sensors*, 12(2):1211–1279.

Niesten, E., Jolink, A., de Sousa Jabbour, A. B. L., Chappin, M., and Lozano, R. (2016). Sustainable collaboration: The impact of governance and institutions on sustainable performance. *Journal of Cleaner Production*, pages –. in press.

Niknazar, M., Mousavi, S. R., Vahdat, B. V., and Sayyah, M. (2013). A new framework based on recurrence quantification analysis for epileptic seizure detection. *IEEE journal of biomedical and health informatics*, 17(3):572–8.

Nouri, S., Haghseresht, F., and Lu, G. (2002). Comparison of adsorption capacity of p-cresol and p-nitrophenol by activated carbon in single and double solute. *Adsorption*, 8(3):215–223.

Nunes, T. M., Coelho, A. L., Lima, C. A., Papa, J. P., and de Albuquerque, V. H. C. (2014). EEG signal classification for epilepsy diagnosis via optimum path forest – A systematic assessment. *Neurocomputing*, 136:103–123.

Olague, G. and Trujillo, L. (2011). Evolutionary-computer-assisted design of image operators that detect interest points using genetic programming. *Image Vision Comput.*, 29(7):484–498.

O'Neill, M., Vanneschi, L., Gustafson, S., and Banzhaf, W. (2010). Open issues in Genetic Programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363.

Orhan, U., Hekim, M., and Ozer, M. (2011). EEG signals classification using the K-means clustering and a multilayer perceptron neural network model. *Expert Systems with Applications*, 38(10):13475–13481.

Ortigosa, I., López, R., and García, J. (2007). A neural networks approach to residuary resistance of sailing yachts prediction. *Proceedings of the international conference on marine engineering MARINE*, 2007:250.

Ozcift, A. and Gulten, A. (2011). Classifier ensemble construction with rotation forest to improve medical diagnosis performance of machine learning algorithms. *Computer Methods and Programs in Biomedicine*, 104(3):443–451.

Pagie, L. and Hogeweg, P. (1998). Evolutionary consequences of coevolving targets. *Evolutionary Computation*, 5:401–418.

Patelli, A. and Ferariu, L. (2011). A regressive schema theory based tool for gp evolved nonlinear models. In *Automation and Computing (ICAC), 2011 17th International Conference on*, pages 201–206.

Pedro, J. C. and Maas, S. A. (2005). A comparative overview of microwave and wireless power-amplifier behavioral modeling approaches. *IEEE Transactions on Microwave Theory and Techniques*, 53(4 I):1150–1163.

Peng, H., Hu, B., Qi, Y., Zhao, Q., and Ratcliffe, M. (2011). An improved EEG de-noising approach in electroencephalogram (EEG) for home care. *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 469–474.

Pfurtscheller, G., Brunner, C., Schlögl, A., and Lopes da Silva, F. H. (2006). Mu rhythm (de)synchronization and EEG single-trial classification of different motor imagery tasks. *NeuroImage*, 31(1):153–159.

Pfurtscheller, G. and Neuper, C. (2001). Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134.

Pfurtscheller, G., Neuper, C., Flotzinger, D., and Pregenzer, M. (1997). EEG-based discrimination between imagination of right and left hand movement. *Electroencephalography and Clinical Neurophysiology*, 103(6):642–651.

Picot, A., Whitmore, H., and Chapotot, F. (2012). Detection of cortical slow waves in the sleep EEG using a modified matching pursuit method with a restricted dictionary. *IEEE Transactions on Biomedical Engineering*, 59(10):2808–2817.

Pineda, J. A. (2005). The functional significance of mu rhythms: Translating ”seeing” and ”hearing” into ”doing”. *Brain Research Reviews*, 50(1):57–68.

Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd.

Poli, R., Salvaris, M., and Cinel, C. (2011). Evolution of a brain-computer interface mouse via genetic programming. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6621 LNCS:203–214.

Popivanov, D., Stomonyakov, V., Minchev, Z., Jivkova, S., Dojnov, P., Jivkov, S., Christova, E., and Kosev, S. (2006). Multifractality of decomposed EEG during imaginary and real visual-motor tracking. *Biological Cybernetics*, 94(2):149–156.

Pramila, P. V. and Mahesh, V. (2015). Comparison of multivariate adaptive regression splines and random forest regression in predicting forced expiratory volume in one second. *International Journal of Medical, Health, Biomedical, Bioengineering and Pharmaceutical Engineering*, i(4):338–342.

Qadeer, R. and Rehan, A. (2002). A study of the adsorption of phenol by activated carbon from aqueous solutions. *Turk. J. Chem.*, 26:357–361.

Rajendra Acharya, U., Vinitha Sree, S., Alvin, A. P. C., and Suri, J. S. (2012). Use of principal component analysis for automatic classification of epileptic EEG activities in wavelet framework. *Expert Systems with Applications*, 39(10):9072–9078.

Ramgopal, S., Thome-Souza, S., Jackson, M., Kadish, N. E., Sánchez Fernández, I., Klehm, J., Bosl, W., Reinsberger, C., Schachter, S., and Loddenkemper, T. (2014). Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy. *Epilepsy & Behavior*, 37:291–307.

Ramoser, H., Müller-Gerking, J., and Pfurtscheller, G. (2000). Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE Transactions on Rehabilitation Engineering*, 8(4):441–446.

Rawat, M., Member, S., and Ghannouchi, F. M. (2013). Three-Layered Biased Memory Polynomial for Dynamic Modeling and Predistortion of Transmitters With Memory. 60(3):768–777.

Rezaee, A. (2016). Applying Genetic Algorithm to EEG Signals for Feature Reduction in Mental Task Classification. *International Journal of Smart Electrical Engineering*, 5(1):4–7.

Rokach, L. and Maimon, O. (2008). *Data Mining with Decision Trees: Theroy and Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Roland, J. L., Hacker, C. D., Breshears, J. D., Gaona, C. M., Hogan, R. E., Burton, H., Corbetta, M., and Leuthardt, E. C. (2013). Brain mapping in a patient with congenital blindness - a case for multimodal approaches. *Frontiers in human neuroscience*, 7(July):431.

Ruthven, D. (1984). *Principles of adsorption and adsorption processes*. Wiley Interscience.

Sabeti, M., Katebi, S., and Boostani, R. (2009). Entropy and complexity measures for EEG signal classification of schizophrenic and control participants. *Artificial Intelligence in Medicine*, 47(3):263–274.

Sarkar, M., Acharya, P. K., and Bhattacharya, B. (2005). Removal characteristics of some priority organic pollutants from water in a fixed bed fly ash column. *Journal of Chemical Technology and Biotechnology*, 80(12):1349––1355.

Sarwar, U., Muhammad, M. B., and Karim, Z. A. (2014). Time series method for machine performance prediction using condition monitoring data. In *IEEE 2014 International Conference on Computer, Communication, and Control Technology*, I4CT 2014.

Schetzen, M. (2006). *The Volterra and Wiener Theories of Nonlinear Systems*. Krieger Publishing Co., Inc., Melbourne, FL, USA.

Schmidt, E., Kincses, W., and Schrauf, M. (2007). Assessing driver's vigilance state during monotonous driving. *Driving Assessment 2007: 4th International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design*, pages 138–145.

Schultze-Kraft, M., Dähne, S., Gugler, M., Curio, G., and Blankertz, B. (2016). Unsupervised classification of operator workload from brain signals. *Journal of Neural Engineering*, 13(3):036008.

Sculley, D. (2011). Results from a Semi-Supervised Feature Learning Competition. *NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning*, pages 1–9.

Sekhon, R., Bassily, H., Wagner, J., and Gaddis, J. (2006). Stationary gas turbines – a real time dynamic model with experimental validation. In *Proceedings of the 2006 American Control Conference*.

Selye, H. (1965). The stress syndrome. *The American Journal of Nursing*, 65(3):97–99.

Shen, K. Q., Li, X. P., Ong, C. J., Shao, S. Y., and Wilder-Smith, E. P. V. (2008). EEG-based mental fatigue measurement using multi-class support vector machines with confidence estimate. *Clinical Neurophysiology*, 119(7):1524–1533.

Shultz, G., Schnabel, R., and Byrd, R. (1982). *A Family of Trust Region Based Algorithms for Unconstrained Minimization with Strong Global Convergence Properties*. Defense Technical Information Center.

Silva, S. (2011). Reassembling operator equalisation: a secret revealed. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1395–1402, New York, NY, USA. ACM.

Silva, S. and Almeida, J. (2005). Gplab-a genetic programming toolbox for matlab. In *In Proc. of the Nordic MATLAB Conference (NMC-2003)*, pages 273–278.

Silva, S. and Costa, E. (2009). Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10(2):141–179.

Sinclair, C. M., Gasper, M. C., and Blum, A. S. (2007). Basic Electronics in Clinical Neurophysiology. In *The Clinical Neurophysiology Primer*, pages 3–18. Humana Press, Totowa, NJ.

Singha, B., Bar, N., and Das, S. K. (2014). The use of artificial neural networks (ann) for modeling of adsorption of cr(vi) ions. *Desalination and Water Treatment*, 52(1-3):415–425.

Siuly, S., Li, Y., and Zhang, Y. (2017). *EEG Signal Analysis and Classification: Techniques and Applications*. Health Information Science. Springer International Publishing.

Song, I.-h. and Lee, D.-s. (2005). Fluctuation Dynamics in Electroencephalogram Time Series. In *First International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 195–202.

Song, M., Wang, S., and Cen, L. (2015). Comprehensive efficiency evaluation of coal enterprises from production and pollution treatment process. *Journal of Cleaner Production*, 104:374 – 379.

Song, M.-L. and Zhou, Y.-X. (2015). Analysis of carbon emissions and their influence factors based on data from anhui of china. *Comput. Econ.*, 46(3):359–374.

Sorensen, D. (1982). *Newton's Method with a Model Trust Region Modification*. Defense Technical Information Center.

Sotelo, A., Guijarro, E., Trujillo, L., Coria, L. N., and Martínez, Y. (2013). Identification of epilepsy stages from ECoG using genetic programming classifiers. *Computers in Biology and Medicine*, 43(11):1713–1723.

Sotelo, A., Guijarro, E. D., and Trujillo, L. (2015). Seizure states identification in experimental epilepsy using gabor atom analysis. *Journal of Neuroscience Methods*, 241:121–131.

Spector, L. (2006). *Automatic Quantum Computer Programming: A Genetic Programming Approach (Genetic Programming)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Speight, J. (1991). *The chemistry and technology of petroleum*. MarcelDekker Inc., 2nd edition.

Srivastava, V. C., Swamy, M. M., Mall, I. D., Prasad, B., and Mishra, I. M. (2006). Adsorptive removal of phenol by bagasse fly ash and activated carbon: Equilibrium, kinetics and thermodynamics. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 272(1–2):89 – 104.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127.

Staudinger, J., Nanan, J. C., and Wood, J. (2010). Memory fading volterra series model for high power infrastructure amplifiers. In *2010 IEEE Radio and Wireless Symposium, RWW 2010 - Paper Digest*, pages 184–187.

Steihaug, T. (1983). The Conjugate Gradient Method and Trust Regions in Large Scale Optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637.

Sun, J., Zuo, H., Wang, W., and Pecht, M. G. (2012). Application of a state space modeling technique to system prognostics based on a health index for condition-based maintenance. *Mech. Syst. Signal Process*, 28:585–596.

Sun, Z., Bebis, G., and Miller, R. (2004). Object detection using feature subset selection. *Pattern Recogn.*, 37(11):2165–2176.

Takahashi, N., Nakai, T., Satoh, Y., and Katoh, Y. (1994). Variation of biodegradability of nitrogenous organic compounds by ozonation. *Water Research*, 28(7):1563–1570.

Tan, P. J. and Dowe, D. L. (2004). MML Inference of Oblique Decision Trees. *Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, pages 1082–1088.

Teplan, M. (2002). Fundamentals of EEG measurement. *Measurement Science Review*, 2(2):1–11.

Thurman, D. J., Beghi, E., Begley, C. E., Berg, A. T., Buchhalter, J. R., Ding, D., Hesdorffer, D. C., Hauser, W. A., Kazis, L., Kobau, R., Kroner, B., Labiner, D., Liow, K., Logroscino, G., Medina, M. T., Newton, C. R., Parko, K., Paschal, A., Preux, P.-M., Sander, J. W., Selassie, A., Theodore, W., Tomson, T., and Wiebe, S. (2011). Standards for epidemiologic studies and surveillance of epilepsy. *Epilepsia*, 52 Suppl 7(1):2–26.

Tian, M., Linda, B., and Chen, A. (2007). Kinetics of the electrochemical oxidation of 2-nitrophenol and 4-nitrophenol studied by in situ uv spectroscopy and chemometrics. *Electrochimica acta*, 52(23):6517–6524.

Togun, N. and Baysec, S. (2010). Genetic programming approach to predict torque and brake specific fuel consumption of a gasoline engine. *Applied Energy*, 87(11):3401 – 3408.

Topchy, A. and Punch, W. F. (2001). Faster genetic programming based on local gradient search of. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'01*, pages 155–162. Morgan Kaufmann.

Trejo, L. J., Kubitz, K., Rosipal, R., Kochavi, R. L., and Montgomery, L. D. (2015). EEG-Based Estimation and Classification of Mental Fatigue. *Psychology*, 06(05):572–589.

Trujillo, L., Legrand, P., Olague, G., and LéVy-VéHel, J. (2012). Evolving estimators of the pointwise hölder exponent with genetic programming. *Inf. Sci.*, 209:61–79.

Trujillo, L., Muñoz, L., Naredo, E., and Martínez, Y. (2014). Neat, there's no bloat. In *Genetic Programming*, volume 8599 of *Lecture Notes in Computer Science*, pages 174–185. Springer Berlin Heidelberg.

Trujillo, L., Muñoz, L., Galván-López, E., and Silva, S. (2016). neat genetic programming: Controlling bloat naturally. *Information Sciences*, 333:21 – 43.

Trujillo, L., Naredo, E., and Martínez, Y. (2013). Preliminary study of bloat in genetic programming with behavior-based search. In *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*, volume 227 of *Advances in Intelligent Systems and Computing*, pages 293–305. Springer International Publishing.

Trujillo, L., Z-Flores, E., Ju, P., Legrand, P., Castelli, M., Vanneschi, L., Sch, O., and Mu, L. (2017). Local Search is Underused in Genetic Programming. In *Genetic Programming Theory and Practice XIV*, number January.

Tsai, H. C. (2011). Using weighted genetic programming to program squat wall strengths and tune associated formulas. *Engineering Applications of Artificial Intelligence*, 24(3):526–533.

Tsanas, A., Little, M., Fox, C., and Ramig, L. (2014). Objective automatic assessment of rehabilitative speech treatment in parkinson's disease. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 22(1):181–190.

Tuson, A. and Ross, P. (1998). Adapting operator settings in genetic algorithms. *Evolutionary computation*, 6(2):161–84.

Tzallas, a. T., Tsipouras, M. G., and Fotiadis, D. I. (2007). Automatic seizure detection based on time-frequency analysis and artificial neural networks. *Computational intelligence and neuroscience*, 2007.

Tzallas, A. T., Tsipouras, M. G., and Fotiadis, D. I. (2009). Epileptic seizure detection in EEGs using time-frequency analysis. *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, 13(5):703–10.

Übeyli, E. D. and Güler, I. (2007). Features extracted by eigenvector methods for detecting variability of EEG signals. *Pattern Recognition Letters*, 28:592–603.

Ukrainczyk, L. and McBride, M. (1992). Oxidation of phenol in acidic aqueous suspensions of manganese oxide. *Clays and Clay Minerals*, 40:157–166.

Usakli, A. B. (2010). Improvement of EEG signal acquisition: An electrical aspect for state of the Art of front end. *Computational Intelligence and Neuroscience*, 2010.

Uy, N. Q., Hoai, N. X., O'Neill, M., Mckay, R. I., and Galván-López, E. (2011). Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119.

Vanneschi, L., Castelli, M., and Silva, S. (2014a). A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines*, 15(2):195–214.

Vanneschi, L., Silva, S., Castelli, M., and Manzoni, L. (2014b). Geometric semantic genetic programming for real life applications. In *Genetic Programming Theory and Practice XI*, pages 191–209. Springer New York.

Véhel, J. and Legrand, P. (2004). Signal and Image processing with FracLab. *Thinking in Patterns, World Scientific.*

Venables, L. and Fairclough, S. H. (2009). The influence of performance feedback on goal-setting and mental effort regulation. *Motivation and Emotion*, 33(1):63–74.

Venthur, B., Blankertz, B., Gugler, M. F., and Curio, G. (2010). Novel applications of BCI technology: Psychophysiological optimization of working conditions in industry. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pages 417–421.

Vézard, L., Chavent, M., Legrand, P., Faïta-Aïnseba, F., and Trujillo, L. (2013). Detecting mental states of alertness with genetic algorithm variable selection. *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pages 1247–1254.

Vezard, L., Legrand, P., Chavent, M., Faita-Ainseba, F., Clauzel, J., and Trujillo, L. (2014). Classification of EEG signals by evolutionary algorithm. In Guillet, F., Pinaud, B., Venturini, G., and Zighed, D., editors, *Advances in Knowledge Discovery and Management Volume 4*, volume 527 of *Studies in Computational Intelligence*, pages 133–153. Springer.

Vézard, L., Legrand, P., Chavent, M., Faïta-Aïnseba, F., and Trujillo, L. (2015). EEG classification for the detection of mental states. *Applied Soft Computing*, 32:113–131.

Vijayaraghavan, P. and Veezhinathan, M. (2015). Multivariate adaptive regression splines based prediction of peak expiratory flow with spirometric data. *Technology and Health Care*, 24(s1):S253–S260.

Vladislavleva, E. J., Smits, G. F., and Den Hertog, D. (2009). Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *Trans. Evol. Comp*, 13(2):333–349.

Waldert, S., Pistohl, T., Braun, C., Ball, T., Aertsen, A., and Mehring, C. (2009). A review on directional information in neural signals for brain-machine interfaces. *Journal of Physiology Paris*, 103(3-5):244–254.

Walter, W. G., Cooper, R., Aldridge, V. J., McCallum, W. C., Winter, and A. L. (1964). Contingent Negative Variation : An Electric Sign of Sensori-Motor Association and Expectancy in the Human Brain. *Nature*, 203(4943):380–384.

Wang, P., Tang, K., Weise, T., Tsang, E., and Yao, X. (2014). Multiobjective genetic programming for maximizing ROC performance. *Neurocomputing*, 125:102–118.

White, D. R., McDermott, J., Castelli, M., Manzoni, L., Goldman, B., Kronberger, G., Ja'skowski, W., O'Reilly, U.-M., and Luke, S. (2013). Better gp benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14(1):3–29.

Winkler, S., Affenzeller, M., and Wagner, S. (2007). Advanced genetic programming based machine learning. *Journal of Mathematical Modelling and Algorithms*, 6(3):455–480.

Woodman, G. F. (2010). A brief introduction to the use of event-related potentials (ERPs) in studies of perception and attention. *Attention and Perceptual Psychophysiology*, 72(8):1–29.

Worm, T. and Chiu, K. (2013). Prioritized grammar enumeration: Symbolic regression by dynamic programming. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 1021–1028. ACM.

Wu, S. C. and Swindlehurst, a. L. (2013). Matching pursuit and source deflation for sparse EEG/MEG dipole moment estimation. *IEEE Transactions on Biomedical Engineering*, 60(8):2280–2288.

Xiaofang, W., Jianghong, S., and Huihuang, C. (2009). On the numerical stability of RF power amplifier's digital predistortion. In *Communications, 2009. APCC 2009. 15th Asia-Pacific Conference on*, volume 3, pages 430–433.

Xie, S. and Krishnan, S. (2014). Dynamic principal component analysis with nonoverlapping moving window and its applications to epileptic EEG classification. *TheScientificWorldJournal*, 2014(1).

Xiong, G., Zhou, X., and Ji, P. (2008). Implementation of the Quadrature Waveform Generator Based on DSP Builder. In *2008 International Symposium on Intelligent Information Technology Application Workshops*, pages 773–776.

Yadav, R., Shah, a. K., Loeb, J. a., Swamy, M. N. S., and Agarwal, R. (2012). Morphology-based automatic seizure detector for intracerebral EEG recordings. *IEEE Transactions on Biomedical Engineering*, 59(7):1871–1881.

Yuan, J. Y. (1996). Numerical methods for generalized least squares problems. *Journal of Computational and Applied Mathematics*, 66(1–2):571 – 584.

Yuen, T. G., Agnew, W. F., and Bullara, L. A. (1987). Tissue response to potential neuroprosthetic materials implanted subdurally. *Biomaterials*, 8(2):138 – 141.

Z-Flores, E., Abatal, M., Bassam, A., Trujillo, L., Juarez-Smith, P., and El Hamzaoui, Y. (2017). Modeling the Adsorption of Phenols and Nitrophenols by Activated Carbon using Genetic Programming. *Journal of Cleaner Production*.

Z-Flores, E., Trujillo, L., Schütze, O., and Legrand, P. (2014). Evaluating the Effects of Local Search in Genetic Programming. In *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, volume 288, pages 213–228.

Z-Flores, E., Trujillo, L., Schütze, O., and Legrand, P. (2015). A Local Search Approach to Genetic Programming for Binary Classification. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO'15, pages 1151–1158. ACM.

Z-Flores, E., Trujillo, L., Sotelo, A., Legrand, P., and Coria, L. N. (2016). Regularity and Matching Pursuit feature extraction for the detection of epileptic seizures. *Journal of Neuroscience Methods*, 266:107–125.

Zainuddin, Z., Huong, L. K., and Pauline, O. (2013). Reliable epileptic seizure detection using an improved wavelet neural network. *The Australasian Medical Journal*, 6(5):308–314.

Zarjam, P., Epps, J., and Lovell, N. H. (2015). Beyond Subjective Self-Rating: EEG Signal Classification of Cognitive Workload. *IEEE Transactions on Autonomous Mental Development*, 7(4):301–310.

Zhai, J., Zhou, J., Zhang, L., and Hong, W. (2010). Behavioral Modeling of Power Amplifiers with Dynamic Fuzzy Networks. In *IEEE Microwave and Wireless Components Letters*, volume 20, pages 528–530.

Zhang, L., He, W., He, C., and Wang, P. (2010). Improving mental task classification by adding high frequency band information. *Journal of Medical Systems*, 34(1):51–60.

Zhang, M. and Smart, W. (2004). Genetic programming with gradient descent search for multiclass object classification. In Keijzer, M., O'Reilly, U.-M., Lucas, S. M., Costa, E., and Soule, T., editors, *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, volume 3003 of *LNCS*, pages 399–408, Coimbra, Portugal. Springer-Verlag.

Zhang, W. and Goh, A. T. (2014). Multivariate adaptive regression splines and neural network models for prediction of pile drivability. *Geoscience Frontiers*, 7(1):45–52.

Zhang Sheng, Shang Xiuyu, and Wang Wei (2010). An ANN model of optimizing activation functions based on constructive algorithm and GP. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, number Iccasm, pages V1–420–V1–424. IEEE.

Zhu, A. and Brazil, T. (2004). Behavioral modeling of RF power amplifiers based on pruned volterra series. *IEEE Microwave and Wireless Components Letters*, 14(12):563–565.

Zorick, T. and Mandelkern, M. a. (2013). Multifractal Detrended Fluctuation Analysis of Human EEG: Preliminary Investigation and Comparison with the Wavelet Transform Modulus Maxima Technique. *PLoS ONE*, 8(7).