Diseño Evolutivo Multi-objetivo de Redes Neuronales de Tercera Generación aplicadas a Reconocimiento de Patrones

Tesis

que para obtener el grado de:

Maestro en Ciencias de la Computación

Presenta:

Lic. Carlos Alberto Juárez Santini

con la asesoría de:

Dr. Manuel Ornelas Rodríguez

con la co-asesoría de:

Dr. Jorge Alberto Soria Alcaraz

Revisores:

Dr. Alfonso Rojas Domínguez

Dr. Héctor José Puga Soberanes

León, Guanajuato. Enero 2021

""2021: Año de la Independencia"

León, Gto. 15/enero/2021

C. LIC. CARLOS ALBERTO JUÁREZ SANTINI
PRESENTE

De acuerdo al fallo emitido por la Comisión Revisora, integrada por los: **Dr. Manuel Ornelas Rodríguez, Dr. Jorge Alberto Soria Alcaraz, Dr. Héctor José Puga Soberanes y Dr. Alfonso Rojas Domínguez**, y considerando que llena todos los requisitos establecidos en los Lineamientos Generales para la Operación del Posgrado del Tecnológico Nacional de México, **se autoriza la impresión** del trabajo de tesis titulado: **"Diseño evolutivo Multi-objetivo de redes neuronales de tercera generación aplicadas al reconocimiento de patrones"**. Lo que hacemos de su conocimiento para los efectos y fines correspondientes.

A T E N T A M E N T E

Excelencia en Educación Tecnológica®
Ciencia Tecnología y Libertad

SEP SECRETARÍA DE EDUCACIÓN PÚBLICA | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LEÓN
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

DR. DAVID ASAEL GUTIÉRREZ HERNÁNDEZ
JEFE DE LA DEPI

C.c.p. Archivo

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

León, Gto., a 8 de Enero de 2021

C. ING. LUIS ROBERTO GALLEGOS MUÑOZ
JEFE DE SERVICIOS ESCOLARES
P R E S E N T E

Por este medio hacemos de su conocimiento que la tesis titulada **"Diseño Evolutivo Multi-objetivo de Redes Neuronales de Tercera Generación aplicadas a Reconocimiento de Patrones"**, ha sido leída y aprobada por los miembros del Comité Tutorial para su evaluación por el jurado del acto de examen de grado al alumno (a) **C. Carlos Alberto Juárez Santini,** con número de control **M19240008** como parte de los requisitos para obtener el grado de Maestro(a) en Ciencias de la Computación (MCCOM-2011-05).

Sin otro particular por el momento, quedamos de Usted.

A T E N T A M E N T E
COMITÉ TUTORIAL

Dr. Manuel Ornelas Rodríguez
DIRECTOR

Dr. Jorge Alberto Soria Alcaraz
CODIRECTOR

Dr. Héctor José Puga Soberanes
REVISOR

Dr. Alfonso Rojas Domínguez
REVISOR

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

DECLARACION DE AUTENTICIDAD Y DE NO PLAGIO

Yo, C. Carlos Alberto Juárez Santini identificado con No. Control M19240008, alumno (a) del programa de la **Maestría en Ciencias de la Computación**, autor (a) de la Tesis titulada: "Diseño Evolutivo Multi-objetivo de Redes Neuronales de Tercera Generación aplicadas a Reconocimiento de Patrones" DECLARO QUE:

1.- El presente trabajo de investigación, tema de la tesis presentada para la obtención del título de **MAESTRO (A) EN CIENCIAS DE LA COMPUTACIÓN** es original, siendo resultado de mi trabajo personal, el cual no he copiado de otro trabajo de investigación, ni utilizado ideas, fórmulas, ni citas completas "stricto sensu", así como ilustraciones, fotografías u otros materiales audiovisuales, obtenidas de cualquier tesis, obra, artículo, memoria, etc. en su versión digital o impresa.

2.- Declaro que el trabajo de investigación que pongo a consideración para evaluación no ha sido presentado anteriormente para obtener algún grado académico o título, ni ha sido publicado en sitio alguno.

3.- Declaro que las pruebas o experimentos derivados de esta investigación fueron realizados bajo el consentimiento de los involucrados y con fines estrictamente académicos conforme a criterios éticos de confidencialidad.

Soy consciente de que el hecho de no respetar los derechos de autor y hacer plagio, es objeto de sanciones universitarias y/o legales por lo que asumo cualquier responsabilidad que pudiera derivarse de irregularidades den la tesis, así como de los derechos sobre la obra presentada.

Asimismo, me hago responsable ante el Tecnológico Nacional de México/Instituto Tecnológico de León o terceros, de cualquier irregularidad o daño que pudiera ocasional por el incumplimiento de lo declarado.

De identificarse falsificación, plagio, fraude, o que el trabajo de investigación haya sido publicado anteriormente; asumo las consecuencias y sanciones que de mi acción se deriven, responsabilizándome por todas las cargas pecuniarias o legales que se deriven de ello sometiéndome a las normas establecidas en los Lineamientos y Disposiciones de la Operación de Estudios de Posgrado en el Tecnológico Nacional de México.

León, Guanajuato a 08 del mes de Enero de 2021.

Carlos Alberto Juárez Santini

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

ACUERDO PARA USO DE OBRA (TESIS DE GRADO)

A QUIEN CORRESPONDA

PRESENTE

Por medio del presente escrito, C. Carlos Alberto Juárez Santini (en lo sucesivo el AUTOR) hace constar que es titular intelectual de la obra denominada: "Diseño Evolutivo Multi-objetivo de Redes Neuronales de Tercera Generación aplicadas a Reconocimiento de Patrones", (en lo sucesivo la OBRA) en virtud de lo cual autoriza al Tecnológico Nacional de México/Instituto Tecnológico de León (en lo sucesivo TECNM/ITLeón) para que efectúe resguardo físico y/o electrónico mediante copia digital o impresa para asegurar su disponibilidad, divulgación, comunicación pública, distribución, transmisión, reproducción, así como digitalización de la misma con fines académicos y sin fines de lucro como parte del Repositorio Institucional del TECNM/ITLeón.

De igual manera, es deseo del AUTOR establecer que esta autorización es voluntaria y gratuita, y que de acuerdo a lo señalado en la Ley Federal del Derecho de Autor y la Ley de Propiedad Industrial el TECNM/ITLeón cuenta con mi autorización para la utilización de la información antes señalada, estableciendo que se utilizará única y exclusivamente para los fines antes señalados. El AUTOR autoriza al TECNM/ITLeón a utilizar la obra en los términos y condiciones aquí expresados, sin que ello implique se le conceda licencia o autorización alguna o algún tipo de derecho distinto al mencionada respecto a la "propiedad intelectual" de la misma OBRA; incluyendo todo tipo de derechos patrimoniales sobre obras y creaciones protegidas por derechos de autor y demás formas de propiedad intelectual reconocida o que lleguen a reconocer las leyes correspondientes. Al reutilizar, reproducir, transmitir y/o distribuir la OBRA se deberá reconocer y dar créditos de autoría de la obra intelectual en los términos especificados por el propio autor, y el no hacerlo implica el término de uso de esta licencia para los fines estipulados. Nada de esta licencia menoscaba o restringe los derechos patrimoniales y morales del AUTOR.

De la misma manera, se hace manifiesto que el contenido académico, literario, la edición y en general de cualquier parte de la OBRA son responsabilidad de AUTOR, por lo que se deslinda al (TECNM/ITLeón) por cualquier violación a los derechos de autor y/o propiedad intelectual, así como cualquier responsabilidad relacionada con la misma frente a terceros. Finalmente, el AUTOR manifiesta que estará depositando la versión final de su documento de Tesis, OBRA, y cuenta con los derechos morales y patrimoniales correspondientes para otorgar la presente autorización de uso.

En la ciudad de León, del estado de Guanajuato a los 08 días del mes de Enero de 2021.

Atentamente,

Nombre y firma autógrafa de EL AUTOR

Carlos Alberto Juárez Santini

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support and assistance. I extend my gratitude towards a number of people whose help was very valuable in this work.

I would first like to thank my supervisors, Dr. Manuel Ornelas Rodríguez, Dr. Jorge Alberto Soria Alcaraz, and Dr. Andres Espinal Jimenez; their expertise was invaluable formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. Thank you for your patient support and for all of the opportunities I was given to further my research, and for their valuable guidance throughout my studies. You provided me with the tools that I needed to choose the right direction and successfully completed my dissertation. My immeasurable appreciation and most profound gratitude for your help and support.

To members of the committee, for your detailed evaluation of my work, whose skills and talents in their field shared their corrections to improve this work.

To my country, to allow me of scholarship to keep studying and growing professionally.

Besides, I would like to thank my family for their wise counsel and sympathetic ear. You are always there for me. My Mom, who ever had words of encouragement. My Dad, for share me precious advice and support. My Sister, for help me in challenging moments.

I could not have completed this dissertation without the support of my girlfriend, Marisol, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research, besides my grateful to her family for its total support.

You know how much I love you.

Abstract

Artificial Neural Networks (ANNs) have been developed to mimic the dynamical biological behavior of the brain. ANNs have been implemented to solve different kinds of problems, in specific to solve classification problems. The study of ANNs has improved towards new models, even grouping them in three generations. During the last decade, different techniques have been proposed to train ANNs, from propagating a classification error, tuning learning rules or focus on their plasticity to using algorithms based on Darwin's theory to learn their parameters. Therefore, many options have been developed to improve the ANN's performance.

In this thesis, Spiking Neural Networks (SNNs) (third generation of ANNs) are evolved by means of a multi-objective evolutionary approach. Focus on optimizing SNN's parameters by aiming to improve its performance in solving pattern classification problems. Firstly, evolving a single spiking neuron's parameters is presented to implement a Leaky Integrate-and-Fire (LIF) model through a comparative between multi-objective and mono-objective algorithms. These algorithms are Optimized Multi-objective Particle Swarm Optimization (OMOPSO) and Particle Swarm Optimization (PSO), respectively.

Afterward, Liquid States Machines (LSMs) are introduced since they internally implement RNNs, which are SNNs with recurrent connections instead of a feed-forward process. A multi-objective approach is proposed to optimize synaptic connections generated into the RNN formed by Spike Response Model (SRM) neurons by Multi-objective Evolutionary Algorithm based on Decomposition with Dynamical Resource Allocation (MOEA/D-DRA). Our studies show a perspective to optimize SNNs compared with work in the state of the art. This perspective involves applying multi-objective algorithms, and objective functions that guide solutions over the search space towards finding a set of optimal solutions.

These two studies were evaluated on a series of well-known benchmark classification problems. The multi-objective approach is the study's object to optimize SNNs achieving better results than the mono-objective approach, even applying the reduction of dimensions from input data. Also, compared against work in the state of the art, our proposal uses lower computational power but cannot improve performance in classification tasks. Nonetheless, our proposal provides an alternative to optimize an LSM taken a new proposal to make its state vectors utilized by the multi-objective evolutionary approach.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ANN	**A**rtificial **N**eural **N**etwork
ANNs	**A**rtificial **N**eural **N**etworks
AP	**A**pproximation **P**roperty
CMA-ES	**C**ovariance **M**atrix **A**daption **E**volution **S**trategy
CPSO	**C**ooperative **P**article **S**warm **O**ptimization
CS	**C**uckoo **S**earch
DE	**D**ifferential **E**volution
EAs	**E**volutionary **A**lgorithms
EO	**E**volutionary **O**ptimization
EONS	**E**volutionary **O**ptimization for **N**euromorphic **S**ystems
ES	**E**volutionary **S**trategy
eSNN	**e**volving **S**piking **N**eural **N**etwork
ESNNs	**E**volutionary **S**piking **N**eural **N**etworks
GA	**G**enetic **A**lgorithm
I&F	**I**ntegrate-and-**F**ire
IFB	**I**ntegrate-and-**F**ire-or-**B**urst
LIF	**L**eaky **I**ntegrate-and-**F**ire
LSM	**L**iquid **S**tate **M**achine
LSMs	**L**iquid **S**tate **M**achines
MODE	**M**ulti-**o**bjective hybrid of **D**ifferential **E**volution
MOEA/D	**M**ulti-**o**bjective **E**volutionary **A**lgorithm based on **D**ecomposition
MOEA/D-DRA	**MOEA/D** with **D**ynamical **R**esource **A**llocation
MOP	**M**ulti-objective **O**ptimization **P**roblem
ms	**m**ili**s**econd
mV	**m**ili**V**olt
MWO	**M**ussels **W**andering **O**ptimization
NSGA-II	**N**on-dominated **S**orting **G**enetic **A**lgorithm-II
OMOPSO	**O**ptimized **M**ulti-**o**bjective **P**article **S**warm **O**ptimization
PARDE	**P**arallel **D**ifferential **E**volution
PF	**P**areto **F**ront
PSO	**P**article **S**warm **O**ptimization
SDSP	**S**pike-**D**riven **Synaptic Plasticity**
RNN	**R**ecurrent **N**eural **N**etwork
RNNs	**R**ecurrent **N**eural **N**etworks
reSNN	**r**eservoir-based **e**volving **S**piking **N**eural **N**etwork
SDSM	**S**eparation **D**riven **S**ynaptic **M**odification (SDSM)
SGD	**S**tochastic **G**radient **D**escent
SNN	**S**piking **N**eural **N**etwork
SNNs	**S**piking **N**eural **N**etworks
SP	**S**eparation **P**roperty
SRM	**S**pike **R**esponse **M**odel
UCI	**U**niversity of **C**alifornia, **I**rvine

Chapter 1

Introduction

The main objective in neurocomputing is to develop mathematical algorithms that will enable Artificial Neural Networks (ANNs) to learn as from information processing and knowledge acquisition in the human brain [1]. They try to simulate the brain's behaviour when it generate, stores, processes and transforms information. An ANN is a system composed of simple processing units, which gives the capacity and property to mapping input-output. This process is developed in two phases: a weighted sum and a non-linear function, that it allows doing training with the input data [2]. Since the 1960s, ANNs have emerged as a novel way to imitate the human brain. ANNs' learning capability converts them into a powerful tool in different approaches, for example, to solve pattern recognition, classification, clustering, vision, control systems and forecasting tasks [3]. Besides, ANNs have been employed to trade with complex constraint (non-linear and discontinuous) optimization problems. ANNs are made up of layers each composed by neurons. Usually, these layers are categorized as: the input layer, hidden layers and output layer. They are interconnected with the previous layer. Neurons interconnected between layers generate synapses. During ANN's learning process, synapses change until the acquired knowledge is enough, i.e., its performance is evaluated, such as an error measure correlated to the network's objective. Once the learning process is over, that knowledge is evaluated employing a sample of data from the problem different from those used in the learning process. With this, it is expected that ANNs will be able to classify patterns from a particular problem with feasible performance [4]. ANNs models are separated in three different models according to their computational units. To begin with the first generation of ANNs which is based on McCulloch-Pitts neurons to process digital data [5]. Afterwards, the second generation of ANNs, is characterized by topologies of neurons connected between layers, i.e., there are cataloged as input, intermediate and output layers with computing units (neurons) that apply activation functions such as sigmoid or hyperbolic tangent [6]. Eventually, Spiking Neural Networks (SNNs) are known as the third generation of artificial neural networks [5]. Neurons within this generation are spiking neurons, which process time encoded data simulating spikes instead of a threshold firing rate [7]. Learning from these models, in this generation, works through correlation between neural firing

rates, which reflects the mean activity of a neuron. That activity is described by action potentials or spikes, which are sequences of short electrical pulses [8]. Spiking neurons can approximate any continuous function of several variables through regard to temporal coding. This gives them the name of "universal approximators," which means that these spiking neural networks can approximate temporal coding any given continuous function of numerous variables [9].

The present research proposes a multi-objective optimization scheme for automatic design of Spiking Neural Networks, which allows solving pattern recognition problems and whose performance is comparable with other works in the state of the art. The rest of this thesis is organized as follows: Chapter 2 studies the state of the art, where ANNs and SNNs concepts related to this thesis are discussed. Chapter 3 introduces the theoretical background, in which the fundamental aspects are explained. Chapter 4 details the proposed methodology employed during this work. Design experimentation as well as results are detailed in Chapter 5, where the results from the methodology are statistically compared. Finally, the conclusions of this thesis are discussed, and future work is proposed.

1.1 Problem Statement

The development in the investigation of the Artificial Neural Networks (ANNs) in the middle of the past century was slow-growing, which affected implementation. After years, the researchers took up the investigations achieving a new paradigm with ANNs to develop the Backpropagation algorithm to train them. Several years later, an innovation emerged from training ANNs: the addition of a paradigm about Evolutionary Algorithms (EAs). This new approach opened new fields of research, which is known as Neuroevolution.

Neuroevolution has had a great acceptance by the advantage that brings this process to find optimal parameters and optimal architectures of an ANN by applying the evolutionary algorithms, most of them based on Darwin's theory. Through this approach, the aim is to reduce the researcher's total participation at the time to fix the parameters of the ANNs manually. Many works have adopted this approach to training neural networks; most of them are in the second generation of Artificial Neural Networks, and recently with the third generation known as Spiking Neural Networks (SNNs). The aim is to optimize parameters such as synaptic weights, delays, and the neural network architecture to evaluate its performance through generations or an evolutionary process employing a mono-objective paradigm, which weighs only one parameter from the unification of parameters as a weighted sum.

Thus, instead of the mono-objective paradigm, the development of multi-objective schemes for the evolutionary design of SNNs can be seen as an exciting area in which it is possible to research with considerable detail and propose design schemes attractive to the scientific community. Through statistical analyses the methodologies

presented in this work are compared against other works of the area. In addition to looking for an optimization scheme to solve pattern classification problems.

1.2 Justification

During the previous decade, different Spiking Neural Networks (SNNs) models have been proposed, which have demonstrated that are more attached to the biological simulation process of the neural networks of the brain, i.e., there are more plausible. In the neurocomputing area, the researchers have designed various ways to train a neural network, such as the Backpropagation algorithm or have implemented evolutionary algorithms based on the theory of Darwin for the resolution of pattern classification problems. In designing SNNs, it is necessary to optimize several parameters in order to achieve an optimal performance of the neural network, for example, calibration of synaptic weights and delays as well as the architecture itself.

This work proposal will be carried out to implement evolutionary algorithms viewed with a multi-objective approach to the design of SNNs, where they will be sought to optimize it by some objective functions so that their performance will be comparable against with other works in the state of the art. Therefore, we search to propose alternatives for the evolutionary design of SNNs.

1.3 Hypothesis

It is possible to determine the design of the topology and the value of the parameters of Spiking Neural Networks through an optimization scheme using the multi-objective approach to obtain an optimal performance to solve pattern classification problems comparable to other models of neural networks detailed in state of the art.

1.4 Objectives

1.4.1 General

To develop an optimization framework based on multi-objective evolutionary algorithms applied to the Spiking Neural Network design, seeking that its performance solving pattern classification problems could be compared with the state of the art.

1.4.2 Specifics

- To carry out documentary research about the state of the art and theoretical background about Spiking Neural Networks and Multi-objective Evolutionary Algorithms.

- Identify architectonic and parametric aspects, spiking neural models, and objective functions to consider in this work.
- Determine the Multi-objective Evolutionary Algorithms to optimize the parameters of the Spiking Neural Networks.
- Implement and test the methodology proposed over datasets about pattern classification problems known in state of the art.
- Make a statistical analysis of the results obtained from the implementation.
- Write a scientific paper about the methodology proposed and its results.
- Write a thesis work to explain the proposed work.

1.5 Scope

While the Artificial Neural Networks are employed in different fields of research, the Spiking Neural Networks have won popularity in the field of neuroscience with the development of a variety of spiking neural models which are more plausible, for example, to resolve problems of pattern classification.

On the other hand, the Multi-objective Evolutionary Algorithm approach has not been much focused on optimizing the parameters of Spiking Neural Networks to get the best performance in the resolution of pattern classification problems. An approach with these topics will be implemented to provide an alternative view to solve such kinds of problems. Finally, from Reservoir Computing, Liquid State Machines will be implemented to solve pattern classification problems since they mainly use spike models.

1.6 Contributions of academic research

Some contributions have been achieved during this thesis along the corresponding time. These contributions are:

- The research paper entitled *Single Spiking Neuron Multi-Objective Optimization for Pattern Classification* presented in the *Workshop on Computational Intelligence (WCI 2019)*, held in Tijuana city, México on August 26-27, 2019. This work was published in the Journal of Automation, Mobile Robotics and Intelligent Systems (14)1, 2020.

Chapter 2

State of the art

This chapter describes a view about the state of the art related to this thesis. It is denoted a historical insight of Artificial Neural Networks, their concepts, and generations in which are classified. A review of the third generation of neural networks is also covered, and an evolutionary approach to improve its performance. Finally, the usability of Spiking Neural Networks is detailed in this chapter.

Concepts of the Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models based on behaviour of biological neurons from the brain [10]. ANNs consist of simple processing units of information linked by weighted connections [11]. They try to simulate the behavior of the brain when they generate, process, or transform information [2]; in fact, they have been implemented to solve different kinds of problems such as pattern classification [12, 13], forecasting [14, 15], robotic control [16, 17, 18], clustering[19, 20], among others. ANNs are formed by organized processing units called *neurons*, which are based on the analogy to the connections between dendrites and axons from realistic biological neurons. Such connections are known as synapses. In artificial neural networks, that connection is defined by the connection weights [1]. Internally, neurons compute some operations, and then its result is compared against a certain threshold to fire a signal. Neurons are distributed in layers; the most basic topology is organized as the input layer, hidden layer, and output layer [4], therefore, neurons are interconnected between layers.

Spiking Neural Networks: Definitions, Concepts and Approaches

Artificial Neural Networks (ANNs) consisting of spiking neurons [21], computing units capable naturally of handling spatio-temporal data, are known as Spiking Neural Networks (SNNs), and they are considered as the third generation of ANNs [22]. Nowadays, SNNs are increasingly being applied to solve real-life problems from engineering and industry domains [23]. Mainly due to the interesting characteristics and computational power of spiking neurons [24, 25] such as a temporal coding in contrast to other generations that cannot encode information continuously resulting

in less efficiency since they require computation from whole neurons in previous layers [26] to obtain a result. SNNs have had several successful implementations; they require care to define their topology (also referred to as architecture), which can be defined around the neuron model, connectivity pattern, and data type of messages [27].

Neuron models have been developed inspiring on the realism of neural activity in the brain with aiming to the measurement of information transferred among neurons [28]. A wide variety of adaptions have been launched showing that by precise spike timing, the information is shared between them.

Firstly, Hodgkin–Huxley model [29] which is one of the models with the most relevance in the neuroscience area since it is related to the models based on conductance involving sodium, potassium and leak current mainly chloride.

Secondly, Integrate-and-Fire (I&F) model, of which there are many proposals,and they are compared in [30], details the action potentials as events [31]. A considerable improvement has arisen from I&F, some of them are Leaky Integrate-and-Fire (LIF) which is one of the most used models in the area [32], Integrate-and-Fire-or-Burst (IFB) is proposed an improvement adding the inactivation of the calcium [33]. Spike Response Model (SRM) [34] is inspired from activity in Integrate-and-Fire. Izhikevich model [35] where is a combination of the dynamic of Hodgkin-Huxley and efficiency of I&F. Thorpe model [36] was derived from the LIF model as a simpler version. A review of these models is detailed in [37].

Therefore, SNNs have been used in many areas to solve different tasks employing these spiking neuron models achieving results in real problems. Also, there is not a unified base spiking neuron model yet. Thence, the choice of a model depends on biological plausibility, the computational costs that require a particular model, and the researcher's choice [38]. To continue some works are presented where the previous models are used.

To achieve a good learning SNNs are trained through different approaches to reach and improve biological plausibility and computational potential, these learning base its algorithms adjusting synaptic weights and delays [39].

Spike-Prop [40] the back-propagation algorithm is adopted to minimize the error between the output and desired spike times. As said before, in [41, 42] the authors minimize the error employing an Evolutionary Strategy fitting the synaptic weights and delays for classification tasks. This approach achieves better results than Spike-Prop.

SpikeComp is detailed in online learning where it evolves the connection between neurons in the hidden layer [43] toward output neurons [44] that are added as they are required with a center represented by the time-to-first-spike to tune the weights of the synapses or add a new output neuron when a new sample is fed to the SNN.

Spike-Driven Synaptic Plasticity (SDSP) employed in a wide variety of learning algorithms given that synaptic weights are modified according to arrival presynaptic spikes to post-synaptic neuron considering its potential and its most recent spike [45].

Evolutionary Algorithms (EAs) are used to optimize the neuron's synapses. These algorithms need a population of individuals; each of them contains whole parameters from the SNNs topology (synaptic weights and delays); besides, these algorithms explore the search space through a single objective function. EAs give the advantage of automatized evolution of neural networks, i.e. to determine a complete or partial topology employing the number of neurons on each layer and their synapses [46]. Formerly, a single spiking neuron was trained to adjust its synaptic weights through Differential Evolution (DE) algorithm, employing LIF and Izhikevich neural model, respectively [47, 48]. Both show competitive performance against a feed-forward neural network of the second generation. Later, SNNs were trained using EAs. Parallel Differential Evolution (PARDE) algorithm to train spiking neural networks [49], evolving their synaptic weights having a comparable performance against multilayer perceptrons trained with Back-prorogation algorithm.

In [48], Differential Evolution (DE) was employed to evolve just synaptic weights according to Izhikevich model. Whereas in [50] Cooperative Particle Swarm Optimization (CPSO), which is a version of PSO is used, where synaptic weights and delays are adjusted preserving as negative as well as positive weights. Cuckoo Search (CS), as CPSO are algorithms based on the behaviour of animals, applied [51] to train SNNs topologies to solve pattern classification tasks.

Evolutionary Strategy (ES) algorithm is used to tune the SNN's parameters since SRM model in [7] and compared against CPSO's performance, being ES better algorithm to tune it. Mussels Wandering Optimization (MWO) is a meta-heuristic similar to CPSO, which is based on a population and animal behaviour. To achieve an acceptable performance to solve pattern classification problems, Enhanced-MWO was proposed in [52] to reduce the SNNs training time using SRM model. Particle Swarm Optimization (PSO) was used in [53] to learn to generate a target spike sequence optimizing an SNN with dynamic synapses of LIF neurons connected from hidden layer to output layer.

Moreover, instead of just employing a single objective function, for example, minimize the classification error with EAs. Multi-objective optimization has been used to find a set of optimal solutions instead of one optimal solution with Mono-objective optimization; therefore, several objective functions can be used to improve learning performance and connectivity. [54]. In [55] a Multi-objective hybrid of Differential Evolution (MODE) is employed to find better results as accuracy and a network's topology. SNNs conformed by SRM neurons have their synapses optimized through Multi-objective Genetic Algorithm (MOGA) [54].

Evolutionary Approach to evolve SNNs in solving real problems

Through advantages that evolutionary algorithms offer, it has been combined with SNNs to learn and optimize their characteristics. Then, its combination has been used to solve real problems.

In [56], a robot "brain", which is an SNN simulator, is optimized by a genetic algorithm (GA) to tune its parameters. An SNN was evolved in an imitation learning way to optimize two sensor modalities from the mobile robot; that is, the potentially competing objectives of light-seeking and obstacle avoidance.

An SNN in [57] is evolved through three algorithms; some of them are Differential Evolution. It aims to optimize the SNN's parameters to produce robust and reliable responses given some inputs recorded from a DVS camera mounted on an aerial vehicle to lead fast and robust responses for robotic applications.

A Genetic Algorithm combined with Differential Evolution algorithm was employed in [58], to optimize synapse weights as well as neuron types (each one described as chromosomes) in an SNN with a reduced number of neurons. They seek to optimize SNN's parameters to imitate to do the control of non-linear dynamic systems (the behaviour of DC motor and arm movement control).

SpiNNaker is neuromorphic hardware to simulate a larger quantity of spiking neurons. Through optimizing a SNN with that hardware. In [46] the authors play Pac-Man using NEAT (is an evolutionary algorithm to create neural networks) algorithm to evolve SNN's topology and its synapse weights employing a SpiNNaker.

Some researchers have implemented multi-objective optimization allowing explore the trade-off among all objectives.

Automatic tuning of a retina model is optimized by means of NSGA-II algorithm [59, 60], where a retina is a neural circuit formed by spiking neurons (the aim is tuning a set of parameters from an ample search space appropriately). In that way, they generate retinal models in an effort to approximate real biological behaviour.

Neuromorphic computing architectures model SNNs in silicon so, it is necessary to train an SNN in order to have small neuromorphic architectures, low power chips with the ability to perform machine learning tasks. Therefore in [61] a genetic algorithm-based training approach called Evolutionary Optimization for Neuromorphic Systems (EONS) is used to evolve the synaptic weights and thresholds of the neurons, and the objective functions minimize the SNN size and maximize its performance.

A multi-objective Immune Genetic Algorithm is proposed to optimize SNN hardware systems to map solutions, whose aim is to reduce energy consumption and communication delays[62].

These are some works in the state of the art where the SNN's topology or its parameters are evolved through Evolutionary Algorithms aiming to solve real problems. Most of them involve learning parameters from SNNs and reduce the error of

classification.

SNNs applied in complex models such as the Liquid State Machines

Reservoir Computing is a general term to cover the following models: Liquid State Machines (LSM) and Echo State Networks. Both models have in common that its structure is formed by Recurrent Neural Networks (RNNs) (liquid or reservoir) and developed independently [63, 64, 37]. Liquid State Machines was introduced in [65], in which the author tries to understand the diversity of how are computed and learnt tasks solved by the brain represented as complex neural circuits or cortical columns. Use of RNNs has a big difference compared to SNNs with feed-forward topology. In RNNs, their connections can be cycles and this gives some advantages: Formerly, it is a dynamical system, i.e., can develop a self-sustained temporal activation dynamics, and more recently it has a dynamical memory which preserves a non-linear transformation of the input history in its internal state [64]. As RNNs and SNNs project the input onto a high-dimensional space (viewed as the internal state) allowing that the readout function to learn in a simple way [66]. The internal states (or liquid states) result from the post-synaptic activity of the neurons in the liquid [67] at a certain time. These are projected to a readout function to compute the desired output from an input [68]. Hence, liquids can be implemented as RNNs, and the readout functions (simple linear functions) can be trained to solve tasks [65, 68, 69].

In the first works developed, liquids are not trained; instead, their topology is defined a priori just creating the internal connections randomly so only the readout was adjusted. In that way, LSMs solve pattern classification problems [70]. In [65], the readout is formed by LIF neurons, and the training phase consists in adjusting the strengths of synapses generated from neurons in the liquid and neurons in readout using a perceptron (also in [71]) to evaluate the liquid's performance. In contrast in [72], an ANN of Hodgkin-Huxley neurons is implemented to analyze the readout responses. Long-term synaptic plasticity is suggested [69] to modify the synaptic synapses within the liquid to achieve separation between input data. Linear least-squares regression is the algorithm used to train the readout function in [66] where the authors prove different parameters configuration related to the synapses between neurons, one of the most interesting configuration is the connection probability between input neurons and liquid neurons where all input neurons are connected into the liquid neurons. Most of the previous works solve artificial pattern classification tasks.

Through an LSM it has been proposed to predict real-world time series. In [73] it is proposed to predict ball trajectories from input data since a video recorded with a robot, and output neurons' connections are trained by linear regression.

A study in [74] shows that there exists a deficiency in the approximation property in an LSM (readout function), producing a poor performance in TIMIT dataset, so in

that work, the authors propose a metric using the separability property of an LSM that can be used as a fitness function in an evolutionary process; also a multi-layer perceptron is proposed as readout function.

A reservoir-based evolving Spiking Neural Network (reSNN) is proposed in [75] where an evolving Spiking Neural Network (eSNN) is used as readout function being the learning process to create a repository of output neurons trained by training dataset.

Another approach to train the readout function is presented in[76], where it is combined from liquid's neurons as the firing rate of each as their membrane potentials. It results in an enhancement of spatiotemporal data recognition.

In contrast, where traditional liquids are not trained, some works propose to train to enhance the ability of LMS's separability. Therefore, different approaches to ameliorate the LSM's performance to solve problems are reviewed. Liquids formed by different spiking neurons are detailed in [77] to observe the effect of connection density on the separability ability with these models. "Liquid" is the term used to name a simple model which represents the mammalian visual system, and the entropy is measured to evaluate the capability of the liquid to suitable represent the information coming from inputs.

A work detailed in [78] looks to improve the separability property changing LIF neurons to Hodgkin-Huxley neurons into the liquid. A goal is to investigate if the liquid's separability is sensitive to employing this kind of neurons modifying their parameters such as soma capacitance and time constants.

Another work [79] employs a different kind of neurons in the liquid; in this case, the liquid is formed by SRM neurons. The readout function is trained with a learning rule based on polychronization [80], which affects the delays to readout neurons and adjusts the synaptic weights in the liquid.

Hebbian learning is investigated to see the effects in the liquid [81] evaluating its performance in a real-world speech problem. Separation Driven Synaptic Modification (SDSM) in [82, 83] is used to optimize the synaptic weights as from separation metric that evaluates the liquid's state vectors.

A dynamic firing threshold is employed [84] in order that the liquid controls its entire internal activity concerning inputs, thus achieving increases in the classification performance by affecting the separability property.

A Genetic Algorithm (GA) is used in [85, 86] to evolve the liquid's synapses. A gene represents a synapse and is formed by its synaptic weight and indexes of presynaptic and post-synaptic neurons it connects. The fitness function involves the mean of each state vector class, and the covariance of each class, besides the within-class and between-class scatter are computed. Then, a Fisher discrimination ratio is maximized. It results from maximizing the distance between the means of classes and minimize the variance of each class. In that way a fitness value for each gene is assigned during evolution. Through a Covariance Matrix Adaption Evolution Strategy (CMA-ES) it is proposed in [87, 88] to optimize the topology and parameters

of an LSM. Three parameters are evolved, a parameter λ to control the density of connections in the liquid, f to distribute the synaptic strength, and τ_n the membrane time constant of the neuron.

There is a review of works where some characteristics of SNNs and RNNs are taken into LSMs to explain how they are trained, and how to improve their performance in pattern classification problems with spatio-temporal data inputs.

Once the state of art has been studied, this thesis proposes a methodology to optimize the parameters of Spiking Neural Networks through Multi-objective algorithms. In the interest of checking if there exists an improvement in training SNNs by means of Multi-objective approach, it is compared against a Mono-objective algorithm as Evolutionary Spiking Neural Networks (ESNNs). Both are designed to solve a specific problem. The methodology considers two phases. In the first phase, a comparative performance between a Single Spiking Neuron optimized by Mono-objective and Multi-objective approaches, including a synapses reduction is explored. In the second phase, a comparison between a recurrent spiking neural network performance trained by a Multi-objective algorithm and a work in the state of art evolved by Mono-objective algorithm is carried out. Both phases aim to automatically find the optimal parameters of an SNN to solve pattern classification tasks.

Chapter 3

Theoretical Background

The topics in this chapter detail theoretical aspects based on which this thesis is developed. Firstly, the main aspects about Artificial Neural Networks (ANNs) are covered. Secondly, the characteristics about Third Generation of ANNs and two well-known spike models are reviewed. Next, Liquid State Machines and a randomly structure for them are described. Finally, a mono-objective algorithm (Particle Swarm Optimization) and two multi-objective algorithms (Optimized Particle Swarm Optimization and Multi-objective Evolutionary Algorithm based on Decomposition by Dynamical Resource Allocation) are described.

3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are inspired by the neural network's biological concept from the human brain. ANNs are designed, as mathematical models of human recognition or biological neurology [89]. ANNs are systems motivated by the distributed, densely parallel computation in the human brain that enables them to resolve successfully complex control and recognition or classification problems. The biological neural network can be mathematically modeled by a weighted graph of immensely linked artificial nodes (neurons) to resolve these problems. Neurons are almost always functions where arguments are conformed by a weighted sum of inputs that arrive at the node [90]. The way that ANNs are connected is by connections between pairs of nodes. Each connection dispatches a signal from one node to another, labeled by a number called "weight," hinting the extent to which signal is amplified or dwindled by a connection [91].

Some characteristics are desirable in these artificial models, chiefly: a) *Non-linearity* allows a better fit to the data; b) *Noise-insensitivity* provides accurate predictions over uncertain data and measurement errors; c) *High Parallelism* entails fast processing; d) *Learning and Adaptivity*, allows the system to update (modify) its internal structure (architecture) in response to changing the environment; and e) *Generalization* enables the application of the model to unlearned data [1].

ANNs can be distinguished in three different generations [5]:

- *First Generation*: based on McCulloch-Pitts [92] neuron as computational units that can handle digital data, and known as perceptrons or threshold gates, and

each boolean function can be computed by linked some perceptrons to build a multilayer perceptron with a single hidden layer.

- *Second Generation*: principally characterized by a multilayer architecture, connectivity separating input, intermediate, and output units. These ANNs are trained by an iterative process with input data to fix synaptic weights to optimal values [6]. Usual examples of networks are feedforward networks, recurrent sigmoidal nets, and networks of radial basis function units, and they are universal for analog computations because any continuous function can be estimated arbitrary well by a network with a single hidden layer [5].

- *Third Generation*: Spiking neurons are models based on computational units in biological neural systems where information to be encoded mainly in temporal patterns of their activity [22]. The spiking neuron has an inherent dynamic nature, mostly defined by an internal state with the characteristic that changes with time. Each postsynaptic neuron fires an action potential or "spike" when its internal state (potential membrane) exceeds a certain threshold [25]. This generation is known as Spiking Neural Networks (SNNs).

3.2 Spiking Neuron Models

Spiking neurons emulate biological neurons closely by transmitting signals in spike trains where each spike has constant amplitude [81]. An ideal spiking neuron can be separated into three functionally distinct parts: a) *dendrites*, b) *soma* and c) *axon*, as seen in Figure 3.1. The **dendrites** work as a collector of signals from other neurons and carry them on to the soma. The **soma** has the functionality of a central processing unit of the arriving information. In this unit, an important fact is that if the total arriving input explodes at a certain threshold, then the output signal is generated. The output signal generated is driven across the **axon** as an output device, which transports it to other neurons. The **synapse** is called the junction between a pair of neurons. Many axonal branches of this neuron end in the nearby neighborhood, but the axon can be wired over several centimeters to reach neurons in other areas of the brain to make a synapse. The action potential or spike is a short voltage pulse of 1-2 ms duration and an amplitude of about 100 mV. This neural signal consists of short electrical pulses and it is represented as a fine electrode too close to the soma or axon. A sequence of action potentials produced by a single neuron is known as a spike train, it means a series of events that happens at regular or irregular intervals, and usually, the action potentials are well separated between them. Therefore, the signal into a spike train is well separated [21].

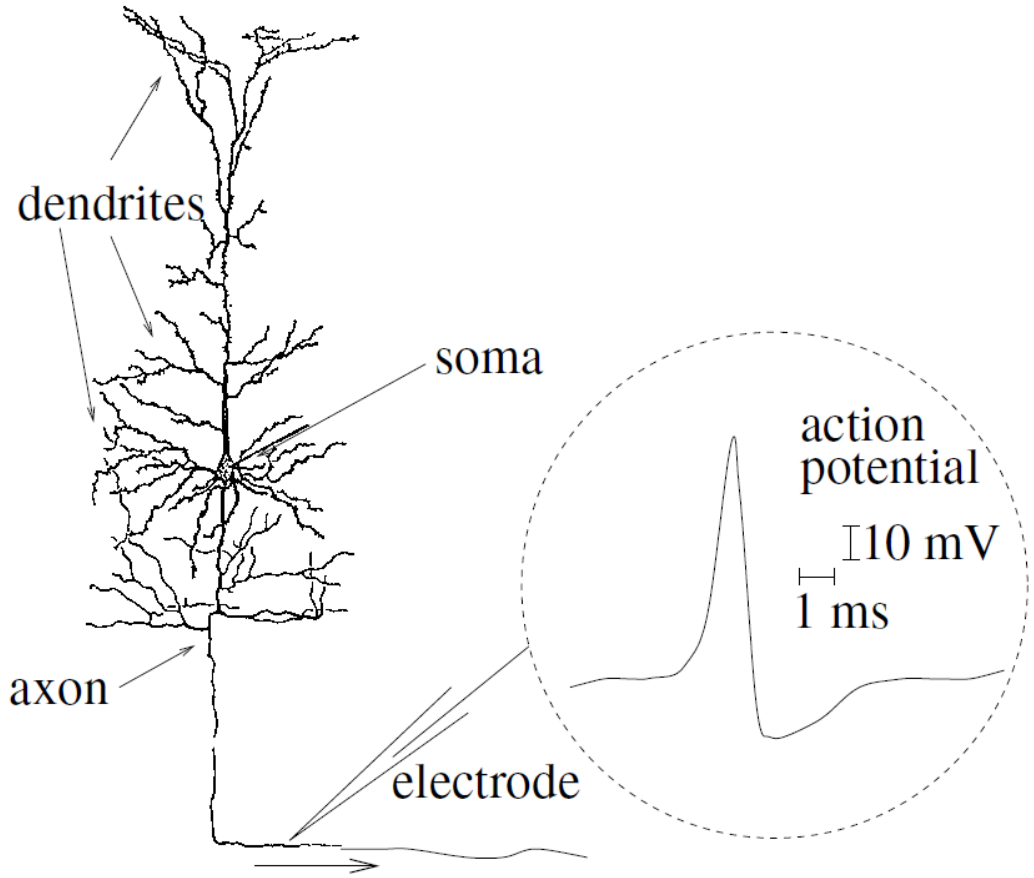

FIGURE 3.1: Drawing of a single biological neuron. Taken from [21]

The successful work by Hodgkin and Huxley in 1952, which describes action potential in a giant axon, leads a whole series of investigations about modeling that try to describe in detail the central aspect of the dynamics of various channels on the soma and dendrites during spike reception and spike emission [93].

Spiking neuron models concern a spectrum from complex and biologically plausible models based on biophysical cortical network architecture to simpler, less realistic models [94], [95]. Some of these models are *Integrate-and-Fire Model* (I&F) [96], *Leaky Integrate-and-Fire Model* (LIF) [47], [32], *Spike Response Model* (SRM) [21], [30], *Hodgkin-Huxley Model* [29], *Izhikevich Model* [35].

Aiming to build a phenomenological model of neural dynamics, it describes the critical voltage for spike initiation by a formal threshold ϑ. Then if $u_i(t)$ exceeds a certain ϑ, the neuron i fires a spike. The action of threshold crossing indicates the firing time $t_i^{(f)}$ [21].

3.2.1 Leaky Integrate and Fire Model

Leaky Integrate-and-Fire model (LIF) is one of the most used in computational neuroscience, given that this model has a more straightforward implementation and a

lower computational cost compared with other spiking neuron models [47]. The mathematical representation for LIF is adapted from [47], [97] and the potential dynamic of the membrane is given by equation 3.1.

$$\tau \frac{dv_i}{dt} = -g_{leak}(v_i - E_{leak}) + I(t) \tag{3.1}$$

Where g_{leak} and E_{leak} are the conductance and the reversal potential of the leak current, τ is the membrane time constant, and $I(t)$ is a current injected into the neuron.

Particularly, the representation of the Leaky Integrate-and-Fire Model (LIF) taken from [32, 47] is of interest in this work, and it is described in equation 3.2.

$$\begin{gathered} v' = I + a - bv \\ \text{if } v \geq v_{threshold}, \text{ then } v \leftarrow c \end{gathered} \tag{3.2}$$

where I is the input current of the neuron, v denotes the membrane potential, a and b are parameters to configure the behavior of the neuron, c is the rest state voltage and $v_{threshold}$ is the threshold for the spike (firing) of the neuron. Besides, an initial condition v_0 is necessary to solve the differential equation by numerical methods.

Since the LIF neuron cannot directly process the input patterns, they must be transformed to input currents through equation 3.3.

$$I = \bar{x} \cdot \bar{w} \cdot \theta \tag{3.3}$$

where $\bar{x} \in \mathbb{R}^n$ is the input pattern vector, $\bar{w} \in \mathbb{R}^n$ is the set of synaptic weights, and θ is a gain factor. Figure 3.2 shows the representation of a LIF neuron. When I is computed, it solves equation 3.2 to obtain the output spike train belonging to the input pattern.

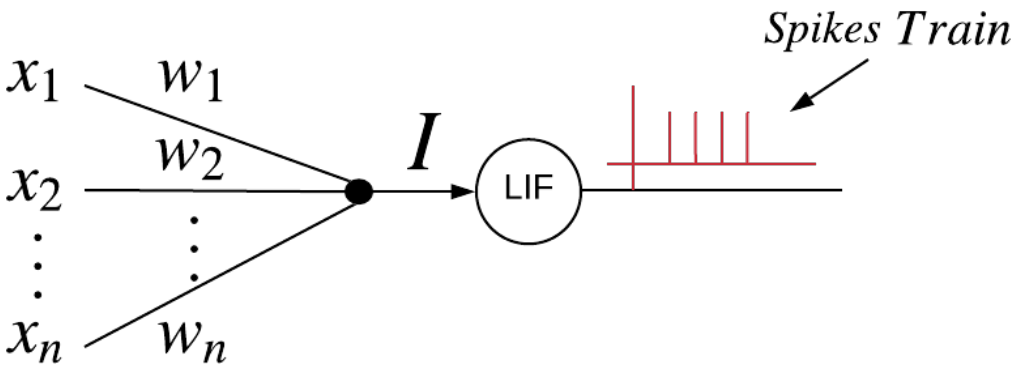

FIGURE 3.2: Representation of a LIF neuron

3.2.2 Spike Response Model

Spike Response Model (SRM) is an approximation of the dynamics of the I&F neuron, in contrast with the idea of Integrate-and-Fire models, which are usually written in differential equations. The idea behind SRM is described by the integrated effect of spike reception or spike emission on the membrane potential [30]. In this thesis, the spiking neurons use the time-to-first-spike as a coding scheme to describe a spike's effect on the emitting and the receiving neuron [30, 98]. Therefore, a reduced version of SRM [19, 50, 98, 99] is of specific interest in this work and is described in equation 3.4.

$$x_j(t) = \sum_{i \in \Gamma_j} w_{ji} y_i(t) \tag{3.4}$$

Where a neuron j has a set of Γ_j of immediate pre-synaptic neurons that receives a set of spikes with firing time t_i, $i \in \Gamma_j$. An SRM neuron fires when its membrane potential $x_j(t)$ reaches a certain threshold θ. Besides, w_{ji} is the synaptic weight to modulate $y_i(t)$, which is the unweighted post-synaptic potential of a spike from a pre-synaptic neuron i to post-synaptic neuron j shown in Figure 3.3.

The unweighted postsynaptic potential $y_i(t)$ denoted in equation 3.5, which describes the strength acting on neuron j at a time t. The input parameters of the neuron j are given by t as the current time, t_i is the firing time if the pre-synaptic neuron i and d_{ji} represents its associated delay.

$$y_i(t) = \varepsilon(t - t_i - d_{ji}) \tag{3.5}$$

The form of a standard post-synaptic potential is described by ε in equation 3.6, τ models the membrane potential decay time constant, that defines the rise and decay time of the post-synaptic potential.

$$\varepsilon(t) = \begin{cases} \frac{t}{\tau} e^{1-t/\tau} & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Figure 3.3 shows an SRM neuron firing spike when the membrane potential $x_j(t)$ exceeds a threshold θ. Point *I* illustrates how each pre-synaptic neuron i sends spikes to a post-synaptic neuron j inside is the membrane potential x_j and evokes a spike. In point *II* the neural dynamics are shown, in which if x_j reaches the threshold θ, an action potential or post-synaptic spike is fired at that precise time. It is possible because the spike trains t_i are formed to evoke the neural behavior [21].

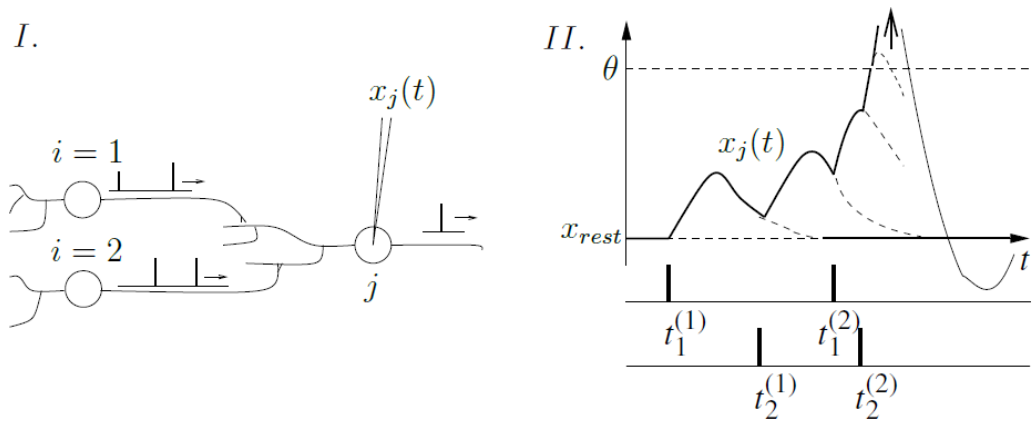

FIGURE 3.3: Representation of an SRM neuron firing spike

3.3 Liquid State Machines

Liquid State Machine (LSM) is based on a rigorous mathematical design that guarantees universal computation power under idealized conditions. The most common approach to design computing in recurrent neural circuits had been used to take control of their high-dimensional dynamics [65]. As a neural microcircuit, the spiking recurrent neural networks are used since they have mighty representational power due to the network's complex time dimension [81]. Neural microcircuits appear to be very good *liquids* for computing perturbations due to the large diversity of their elements, neurons and synapses [72].

An approach to real-computing for LSM considers a series of transient perturbations induced in an excitable environment, such as a *liquid*, by a contentiousness of external disturbances called *inputs*, such as sound, wind, or sequentially dropping some object like rocks into the liquid. Notwithstanding, the perturbed state of the liquid, *at any moment over the time* represents present and past inputs. This perturbed state gives the information to study or analyze many dynamic aspects of a given environment [65].

Different analogies are used to describe an LSM; for example, electroencephalography as a device, measures the brain activity to see the information about the state of the brain [81]. Dropping objects into a liquid container and subsequently reading the ripples created, allows one to know the form of this environment [82]. LSM is compared with Support Vector Machines in [82]. Besides, when a spoon hits the surface of the coffee or sugar cubes that drop into the cup, making perturbances [68].

LSM has a universal computational power for real-time computing with fading memory over analog functions in continuous time. In figure 3.4 the input function $u(\cdot)$ that can be a sequence of disturbances, is injected as input into the liquid filter L^M. A machine M maps the input functions of time $u(\cdot)$ generating, at every time t,

an internal *liquid state* $x^M(t)$ as in equation 3.7.

$$x^M(t) = (L^M u)(t) \tag{3.7}$$

This function corresponds to its current response to preceding perturbations to be transformed by a memory-less readout map f^M which transforms, at every time t, the current liquid state $x^M(t)$ into an output function $y(t)$ (see equation 3.8) of time, which gives a real-time analysis of the $u(\cdot)$ [65].

$$y(t) = f^M(x^M(t)) \tag{3.8}$$

Figure 3.4 shows the architecture of an LSM where an input function of time $u(\cdot)$ is injected into the liquid filter L^M (represented by a *column* [72]), subsequently at a time t the representation of that input as *liquid state* $x^M(t)$ is formed. Afterward, the liquid state is transformed by a memory-less readout f^M to create an output $y(t)$.

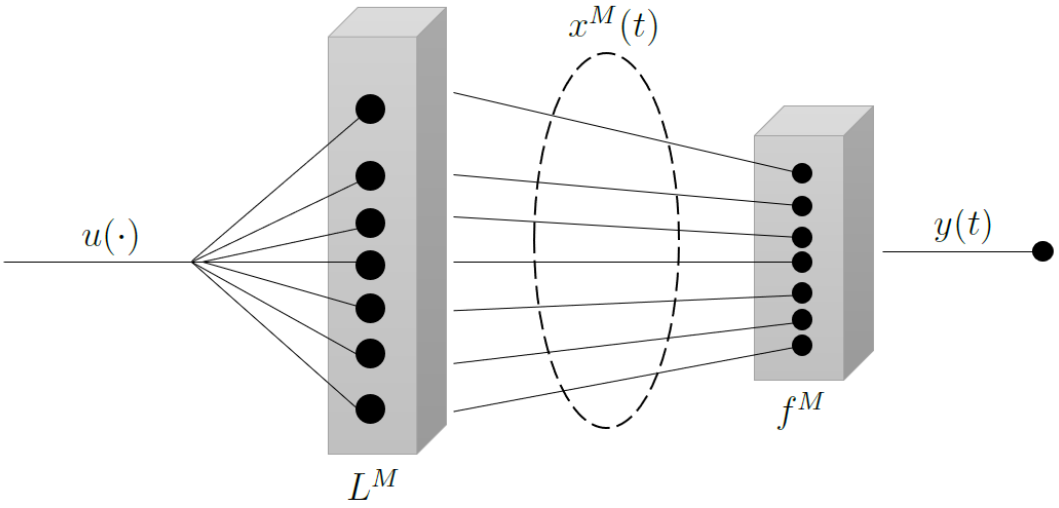

FIGURE 3.4: Architecture of an LSM. Adapted from [65]

Liquid State Machines (LSMs) are a model that can take advantage of recurrent SNNs without training the network. Instead, a randomly reservoir (or liquid) is generated to be used as filter for a simple learning algorithm, for example a perceptron. However many researchers have proposed some works where a liquid is trained besides the output to solve the problem in which many random liquids are generated until a useful liquid is found [83]. Some of these proposals involve unsupervised learning, modifying the structure of the LSM, training the readout layer or evolving some parameters of the LSM [72, 79, 81, 77, 84, 100, 87, 88].

LSMs are composed by two main components:

- Liquid or reservoir, which is formed by a large recurrent SNN.
- Readout function that is distinguished because it can be designed by a simple learning algorithm.

As is shown in Figure 3.5, an input signal is transformed into a function of time or a series of spikes by means of some function. The spike train is introduced into the liquid acting as filter. Thereafter, the state of the liquid (or *state vector*) at time t is employed to feed the readout function, in other words, is used as input to train a learning algorithm [81, 82].

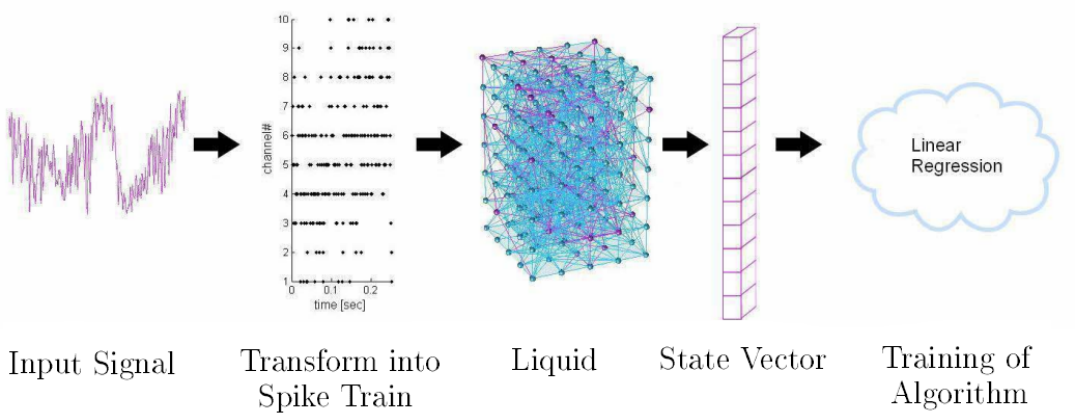

FIGURE 3.5: Flow of a LSM. Taken from [81]

From the LSM study, emerge as necessary and sufficient conditions for powerful real-time computing on perturbations: a *separation property* (SP) and *approximation property* (AP).

- *SP*: addresses the amount of separation between the trajectories of internal states of the liquid caused by two different input streams, i.e., the ability of the liquid to discern different patterns when given different classes of inputs [81].
- *AP*: addresses the resolution and recording capabilities of the readout layer, in other words, the ability to classify the state vectors acquired from the liquid [81].

3.3.1 Lambda Model Synaptic Connection

In [65], Maass proposed to use a randomly connected circuit or liquid, which is formed by 135 I&F neurons, 20% of them are inhibitory. The liquid is structured as a single *column*, similar as in Figure 3.4, locating them on integer points of a $15 \times 3 \times 3$ column in space, and the connectivity structure is defined by means of the probability of a synaptic connection from neuron i to neuron j (as well as a synaptic

connection from neuron j to neuron i) by equation 3.9.

$$P_{ij} = C \cdot e^{-\left(\frac{D(i,j)}{\lambda}\right)^2} \tag{3.9}$$

where the parameter λ controls the average number of connections and the average distance between neurons that are synaptically connected, therefore λ can control the density of the connection , $D(i,j)$ is the Euclidean distance between neuron i and neuron j, influenced independently by the type of neurons i and j, that is, whether they are excitatory (E) or inhibitory (I).

A column with high λ, which means high internal connectivity, achieves higher separation, in contrast a low λ, with lower internal connectivity, tends to chaotic behavior.

3.4 Mono-objective Algorithms

Optimization is the task in which one or more solutions are found with the characteristic that corresponds to minimizing or maximizing one or more specified objectives and can satisfy all constraints if these exist. A mono-objective (or single-objective) optimization problem involves a single objective function that usually gives a single solution, called an optimal solution [101].

To solve the mono-objective optimization problems, the use of Evolutionary Optimization (EO) can be a tool that helps us because it involves a population of solutions that upgrades over generations (at first this seems like overkill but helps to provide implicit parallel search-ability making EO computationally efficient) [102].

EO has some principles [102] that provide a better understanding:

1. EO does not usually involv the use of gradient information during the searching process, therefore EO approaches are direct search procedures, allowing them to be applied to a wide variety of optimization problems.

2. EO procedure uses more than a single solution, forming a *population* approach over an iteration. In addition, the population approach has certain advantages: a) it provides a parallel processing power achieving a general computationally quick search; b) it allows to find multiple optimal solutions; c) it provides the ability to normalize decision variables as well as objective functions and constraint functions within a population during the evolving process using maximum and minimum values from the best population.

3. EO procedure employs stochastic operators, which tend to achieve a desired effect by using biased probability distributions to reach desirable outcomes. This provides to EO algorithm the ability to find multiple optima and other complexities, better achieving a global perspective over the search process.

An Evolutionary Optimization algorithm starts its search with a population of solutions usually randomly initialized within a certain lower and upper bound on

each variable. Subsequently, the algorithm enters into an iterative operation (or generations), which updates the current population to create a new population by means of the use of four principal operators: selection, crossover, mutation, and elite-preservation [102]. This procedure ends when one or more specific termination criteria are met.

3.4.1 Particle Swarm Optimization

Particle Swarm Optimization [103] (PSO) is part of the family of swarm intelligence algorithms, which are inspired from the collective behavior of species such as ants, bees, wasps, fishes, etc. Swarm intelligence is based on the social behaviour of those species that compete for food. The main characteristics of these inspired algorithms are *particles*, which are simple, and non-sophisticated agents. These particles *cooperate* by an indirect communication medium, and perform movements in the decision space [104]. PSO has as characteristic that each solution is called a particle that moves through multidimensional space which represents the search space. The search space depends on the dimension of the variables used to represent the problem to be solved [105, 106]. The position of each particle within the search space is updated through its current location and its velocity vector (this vector tells how fast the particle will move over each dimension).

Algorithm 1 details the template employed for PSO algorithm. In this thesis, PSO algorithm was configured from OMOPSO algorithm (see section 3.5.1) as mono-objective version.

Algorithm 1 PSO Algorithm

Require: Randomly initialization of the whole swarm
while stopping criteria is not met **do**
  Evaluate each particle in the swarm
  for each particle **do**
    Fly
    Mutation
    Evaluate
    Update p_{best}
  end for
  Update swarm
  Store swarm optimized
  crowding swarm, $g = g + 1$
end while

3.5 Multi-objective Algorithms

Multi-objective Optimization Problem (MOP) deals with more than one objective function (usually these are in conflict with each other [107]), in other words, consists in finding an optimal solution set such as to improve a solution does not worsen other solution, then, this result forms a Pareto set. Two important goals of multi-objective problems are defined in [108]:

- 1) Convergence to the optimal Pareto set.
- 2) Maintenance of diversity of optimal Pareto set.

Both goals are independent of each other.

Many real search and optimization problems are naturally present as non-linear programming problems with multiple objectives. Due to lack of adequate solution techniques, such problems were artificially transformed into single-objective problems. A difficulty emerges because these problems give rise to a set of trade-off optimal solutions known as *Pareto-optimal solutions*, instance of a single optimal solution. *Pareto-optimal solutions* arises from the multi-objective optimization problem with conflicting objectives, unlike the usual notion of only one optimal solution associated with a single-objective optimization task. Henceforth, it is important to find as many Pareto-optimal solutions as possible [108].

In MOP, the objective functions form a multi-dimensional space, in addition to the usual decision variable space common to all optimization problems. This additional space is known as the *objective space* Z [108], Ω in [109], S in [110]; Equation 3.10 is derived from [109, 111].

$$\begin{aligned} \text{minimize } & F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x}))^T \\ \text{subject to } & g_i(\vec{x}) \leq 0, \;\; i = 1, ...m \in \Omega \end{aligned} \tag{3.10}$$

where Ω is the solution space, $F : \Omega \rightarrow R^m$ constituted by the k objective functions, and $k \geq 2$. A MOP solution minimizes the components of a vector $F(\vec{x})$, where $\vec{x}$ is an n-dimensional decision variable vector $(\vec{x} = x_1, ..., x_n) \in \Omega$, and there are m constraints. The MOP evaluation function $F : \Omega \rightarrow R^m$ maps decision variables $\vec{x}$ to vectors $(\vec{y} = a_1, ..., a_k)$. This scenario is shown and adopted from [111] in figure 3.6 which illustrates a particular case where $n = 2$, $m = 0$, and $k = 3$.

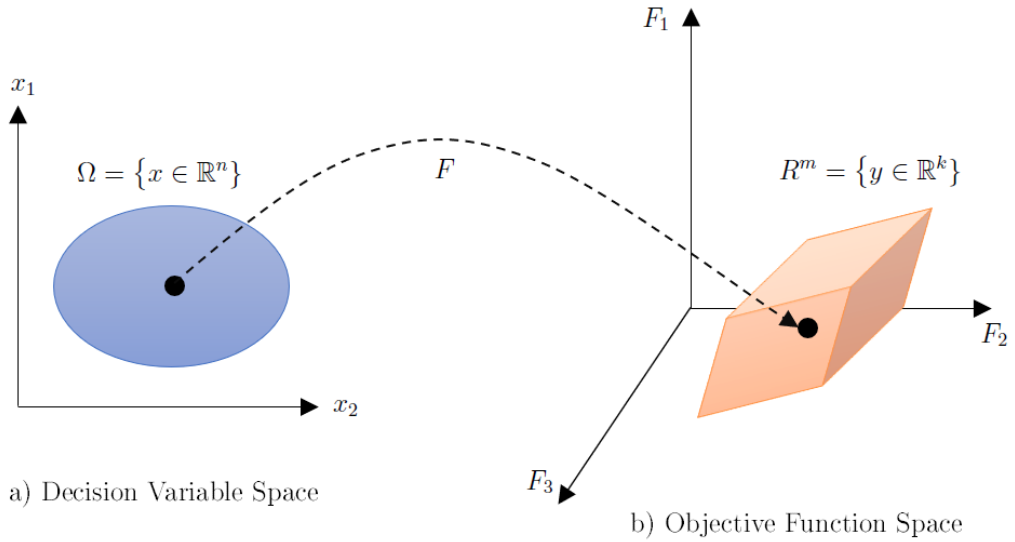

FIGURE 3.6: MOP evaluation mapping

In a minimization problem, a vector x_1 is partially less than another vector x_2, $(x_1 \leq x_2)$, when no value of x_2 is less than x_1 and at least one value of x_2 is precisely greater than x_1. If x_1 is partially less than x_2, then the solution x_1 *dominates* x_2 or the solution x_2 is $inferior$ to x_1. Any value of the solution that is not dominated by any other value, it said to be *non-dominated*. In the same way if the objective is to maximize a function, a dominated point is defined if the corresponding component is not greater than that of a non-dominated point. The optimal solutions to a MOP are non-dominated solutions, and there are know as *Pareto-optimal* solutions [107]. Some important concepts [112] of MOP are *Pareto Concepts*, such as Pareto Dominance, Pareto Optimality, the Pareto Optimal Set and the Pareto Front are detailed as follows:

- *Pareto Dominance*: A vector $\vec{u} = u_1, ..., u_k$ is said to dominate $\vec{v} = (v_1, ..., v_k)$, denoted by $\vec{u} \leq \vec{v}$ iff: u is partially less than v, i.e., $\forall i \in \{1, ..., k\}, u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i \leq v_i$.

- *Pareto Optimality*: A solution $x \in \Omega$ is said to be Pareto optimal with respect to Ω if only if there is no $x' \in \Omega$ for which $\vec{v} = F(x') = (f_1(x'), ..., f_k(x'))$ dominates $\vec{u} = F(x') = (f_1(x'), ..., f_k(x'))$. The term *"Pareto optimal"* is used to refer to the whole decision variable space unless otherwise specified.

- *Pareto Optimal Set*: For a given MOP $F(x)$, the Pareto optimal set (P^*) contains the non-dominated solutions with respect to Ω.

$$P^* := \{x \in \Omega | \neg \exists x' \in \Omega \;\; F(x') \leq F(x)\} \tag{3.11}$$

(P^*) has the non-dominated solutions or *Pareto optimal solutions* also known as *non-solutions, admissible* or *efficient* solutions; their corresponding vectors are called *non-dominated*.

- *Pareto Front*: For a given MOP $F(x)$ and Pareto optimal set (P^*), the Pareto front (PF^*) or PF contains the non-dominated solutions with respect to Ω.

$$PF^* := \{\vec{u} = F(x) = (f_1(x), ..., f_k(x)) | x \in P^*\} \tag{3.12}$$

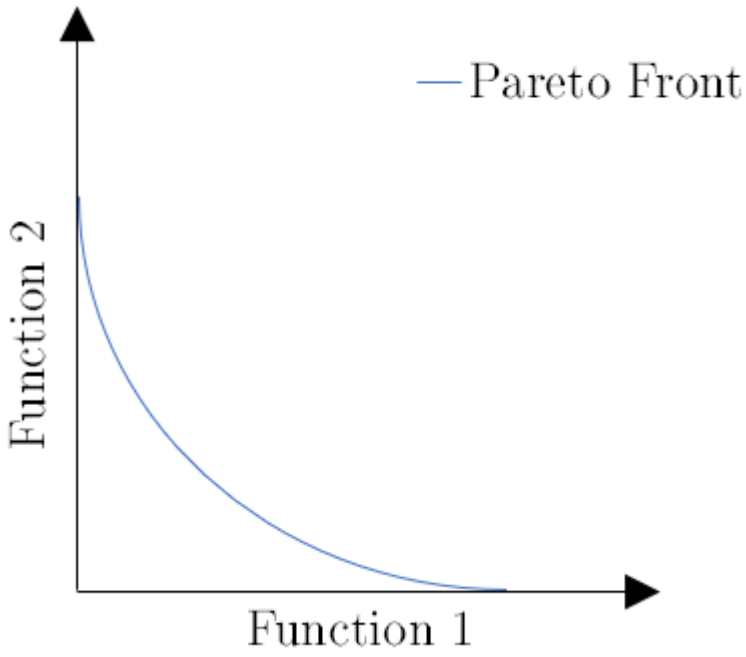

FIGURE 3.7: Pareto Front PF

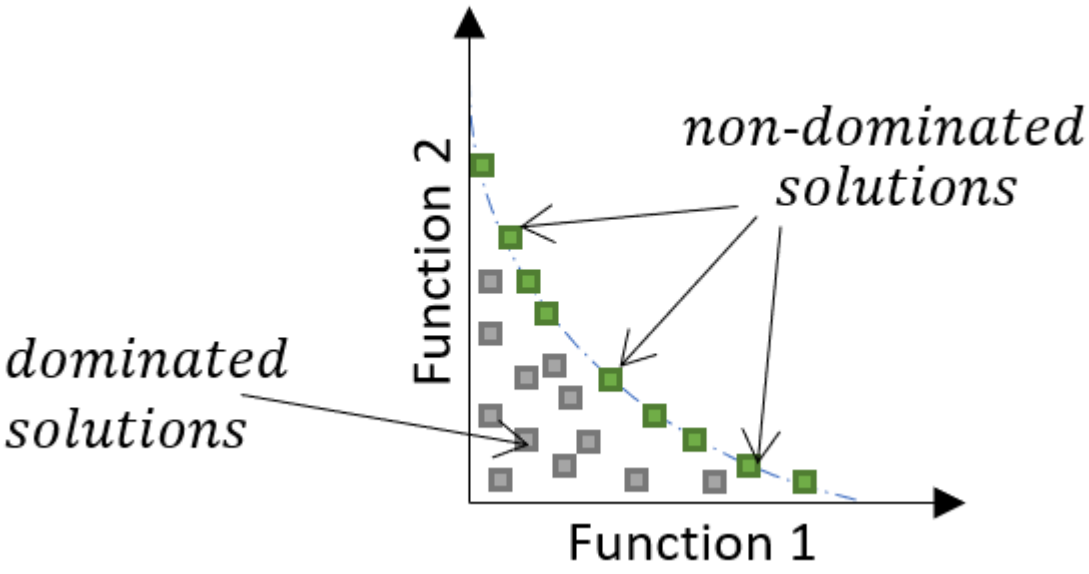

FIGURE 3.8: Pareto Optimal Solutions

Figure 3.7 shows the Pareto front formed by means of successively spread points that represent the non-dominated solutions through the evaluation by two objectives functions, and Figure 3.8 is a view of the representation of the concept for non-dominated solutions (Pareto optimal solutions) and dominated solutions.

3.5.1 Optimized Multi-objective Particle Swarm Optimization

Optimized Multi-objective Particle Swarm Optimization (OMOPSO) was proposed in [113] as an amelioration from [114]. OMOPSO is based on Pareto dominance and elitism selection by means of a crowding factor, and also incorporated two mutation operators: *uniform mutation* and *non-uniform mutation*. The first one, refers to variability range allowed for each decision variable, which is kept constant over generations. The second one, has a variability range allowed for each decision variable, which decreases over time. Finally, the ε-dominance concept that is the final size of the external file that stores the non-dominated solutions. Algorithm 2 shows the OMOPSO algorithm.

Algorithm 2 Optimized Multi-objective Particle Swarm Optimization

Require: Initialize Swarm P_i, Initialize Leaders L_i
1: Send L_i to ε-$file$
2: $crowding(L_i)$, $g = 0$
3: **while** $g \leq g_{max}$ **do**
4: **for** each particle **do**
5: Select Leader
6: Fly
7: Mutation
8: Evaluate
9: Update p_{best}
10: **end for**
11: Update L_i
12: Send L_i to ε-$file$
13: $crowding(L_i)$, $g = g + 1$
14: **end while**
15: Report results in ε-$file$

Algorithm 2 starts with a initial population P_i for $i = 0$ to n where n is the number of particles. Then, it computes the non-dominated particles of P_i and stores them as L_i, which is saved in a ε-$file$. The maximum size of $L_i = P_i$. When it is obtained L_i, the *crowding factor* is applied to each leader in L_i and after, g_{max} is initialized.

Once g_{max} is defined, At each generation, for each particle a leader is selected through a *binary tournament*, which is based on the *crowding value* of L_i. Afterwards, the algorithm compute the fly operator, i.e., the flight of each particle, the changes to the velocity vector are solved by equation 3.13.

$$v_i(t) = Wv_i(t-1) + C_1r_1(x_{pbest_i} - x_i(t)) + C_2r_2(x_{gbest} - x_i(t)) \tag{3.13}$$

where $W = random(0.1, 0.5)$, C_1,$C_2 = random(1.5, 2.0)$ and r_1,$r_2 = random(0.0, 1.0)$. Thereupon, the mutation operator is applied based on the subdivision of P_i. This process divides P_i into three equal subsets, for each one a different mutation is applied. To the first subset the any mutation is applied, the uniform mutation is applied to the second subset, and to the last subset the non-mutation is used. In order to avoid the definition of extra parameters for the mutation operators, the mutation rate is define as $1/number\ of\ variables$.

As next step is to evaluate the particle and update its personal best value (p_best). A new particle replaces its p_best value if such value is dominated by the new particle or if both are non-dominated with respect to each other. Once the particles have been updated, L_i is updated too, so to update the values in L_i, only happens with the particles that outperform their p_best value will try to enter to L_i, and with this, the ε-$file$ is updated.

Finally, update the crowding values of L_i, and eliminate as many leaders as necessary in order to avoid exceeding the size of L_i and follow the ε-dominance concept to fix the maximum size of non-dominated solutions.

A decision vector x_1 is said to ε-dominate a decision vector x_2 when $\varepsilon \leq 0$ if only if: $f_i(x_1)/(1+\varepsilon), \forall i = 1, ..., m$ and $f_i(x_1)/(1+\varepsilon) \leq f_i(x_2)$ for at least one $i = 1, ..., m$. It is worth nothing that, when ε-dominance is used, the size of the ε-file depends on the ε-value, which is usually a parameter defined by a user [115]. In this algorithm, the same value of ε is considered for all the objective functions of a given problem. For each problem, ε is based on the desired amount of points in the final Pareto-front.

3.5.2 Multi-objective Evolutionary Algorithm based on Decomposition with Dynamical Resource Allocation

Multi-objective Evolutionary Algorithm based on Decomposition with Dynamical Resource Allocation (MOEA/D-DRA) developed in [116] is an improvement from the algorithm MOEA/D detailed in [117]. This algorithm arises because last versions of MOEA/D propose to treat all subproblems as equal, in other words, each of them gives similar or roughly the same amount of computational effort, and these subproblems can have different computational difficulties, therefore is rational to assign different amounts of computational effort to different problems. Consequently, the aim with MOEA/D-DRA is to define and compute an *utility* π^i for each subproblem i and thus, the computational efforts are distributed to the subproblems in function of their utilities.

MOEA/D requires a decomposition approach for converting the problem of approximation of the PF into a number of scalar optimization problems. One of them is the Tchebycheff (equation 3.14) [117] approach, which is employed in MOEA/D-DRA as a method of decomposition.

$$\begin{gathered} \text{minimize } g^{te}(x|\lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i |f_i(x) - z_i^*|\} \\ \text{subject to equation 3.10} \end{gathered} \tag{3.14}$$

where $z^* = (z_1^*, ..., z_m^*)^T$ is the reference point, i.e., $z^* = \max\{f_i(x)|x \in \Omega\}$ (in case the aim of MOP is minimization: $z^* = \min\{f_i(x)|x \in \Omega\}$) for each Pareto optimal point x^* there exists a weight vector λ (see Algorithm 3) such that x^* is the optimal solution of equation 3.14 and each solution is a Pareto optimal solution. Next, the Algorithm 3 explains the MOEA/D-DRA algorithm.

Algorithm 3 Multi-objective Evolutionary Algorithm based on Decomposition with Dynamical Resource Allocation

Require: At each generation t, MOEA/D-DRA with *Tchebycheff* approach keeps:

- a population of N points $x^1, ..., x^N \in \Omega$ where x^i is the current solution to the i-th subproblem.
- $FV^1, ..., FV^N$ where FV^i is the F-value of x^i, i.e. $FV^i = F(x^i)$ for each $i = 1, ...N$.
- $z = (z_1, ..., z_m)^T$ where z_i is the best value found so far for objective f_i.
- $\pi^1, ..., \pi^N$ where π^i utility of subproblem i.
- *gen* the current generation number.

Input

- Determinate the multi-objective problem to solve it.
- Set a stopping criteria.
- N is the number of the subproblems considered in MOEA/D-DRA.
- A uniform spread of N weight vectors: $\lambda^1, ..., \lambda^N$.
- T is the number of weight vectors in the neighborhood of each vector.

Output $\{x^1, ..., x^N\}$ and $\{F(x^1), ..., F(x^N)\}$.

Step 1 Initialization

Step 1.1 Compute the Euclidean distances between any two weight vectors and the find the T closest weight vectors to each weight vector.

Step 1.2 Generate an initial population $x^1, ..., x^N$.

Step 1.3 Initialize $z = (z_1, ..., z_m)^T$.

Step 1.4 Set $gen = 0$ and $\pi^i = 1$ for all $i = 1, ..., N$.

Step 2 Selection of Subproblems for Search

Step 3 For each $i \in I$ where I is formed in the previous step, do:

Step 3.1 - Selection of Mating/Update Range

Step 3.2 - Reproduction

Step 3.3 - Repair

Step 3.4 - Update of z

Step 3.5 - Update of Solutions

Step 4 Stopping Criteria

Step 5 *gen=gen+1*

End

Algorithm 3 requires, at each generation during the evaluation, a population of N points, the F-value of x^i, the best value z_i found so far for each objective f_i, the utility π^i of subproblem i and the current generation gen. In the beginning of the algorithm, firstly it has to determinate the MOP, an stopping criteria, the number N of the subproblems in MOEA/D-DRA, a uniform spread of N weight vectors (as in [116]) $\lambda^1, ..., \lambda^N$ and T is the parameter that indicates the number of weighted vectors in the neighborhood of each vector. At the end of the optimization, there are expected $\{x^1, ..., x^N\}$ and $\{F(x^1), ..., F(x^N)\}$.

Into the **Step 1**, four different steps are needed. Firstly, in **Step 1.1** the Euclidean distances are computed between any two weight vectors and the T closest weight

vectors to each weight vector are found. For each $i = 1, ...N$, set $B(i) = \{i_1, ..., i_T\}$ where $\lambda^{i_1}, ..., \lambda^{i_T}$ are the T closest weight vectors to λ^i. Secondly, **Step 1.2** Generates an initial population $x^1, ..., x^N$ by uniformly randomly sampling from the search space. Thirdly, **Step 1.3** indicates to initialize $z = (z_1, ..., z_m)^T$ by setting $z_i = min\{f_i(x^1), f_i(x^2), ..., f_i(x^N)\}$. Finally, **Step 1.4**, set $gen = 0$ and $\pi^i = 1$ for all $i = 1, ..., N$.

Step 2, the selection of subproblems for search is given by the indexes of the subproblems whose objectives are MOP individual objectives f_i are selected to form initial I. By using 10-tournament selections based on π^i, select other $[\frac{N}{5}] - m$ indexes and add them to I. In 10-tournament selection, the index with the highest π^i value from 10 uniformly randomly selected indexes are chosen to enter I. This selection should be done $[\frac{N}{5}] - m$ times.

Step 3 indicates that into loop $i \in I$ some steps are necessary to continue the optimization. **Step 3.1** is about the selection of mating/update range, given by generation of a number $rand\ \mathcal{U} \sim (0, 1)$, then set

$$P = \begin{cases} B(i) & \text{if } rand \leq \delta \\ \{1, ..., N\} & \text{otherwise} \end{cases} \tag{3.15}$$

Reproduction is made in **Step 3.2**, where $r_1 = i$ is set and two indexes r_2 and r_3 are randomly select from P (generated in equation 3.15), and then generate a solution $\bar{y}$ from x^{r_1}, x^{r_2} and x^{r_3} by a DE operator, and then perform a mutation operator on $\bar{y}$ with probability p_m to produce a new solution y. DE operator for each element $\bar{y_k}$ in $\bar{y} = (\bar{y_1}, ..., \bar{y_n})^T$ is given in equation 3.16 and the mutation operator generates $y = (y_1, ..., y_n)^T$ detailed in equations 3.17 and 3.18 from $\bar{y}$.

$$\bar{y_k} = \begin{cases} x_k^{r_1} + F \times (x_k^{r_2} - x_k^{r_3}) & \text{with probability } CR, \\ x_k^{r_1}, & \text{with probability } 1 - CR \end{cases} \tag{3.16}$$

where F and CR are two control parameters.

$$y_k = \begin{cases} \bar{y_k} + \sigma_k \times (b_k - a_k) & \text{with probability } p_m, \\ \bar{y_k} & \text{with probability } 1 - p_m \end{cases} \tag{3.17}$$

with

$$\sigma_k = \begin{cases} (2 \times rand)^{\frac{1}{\eta+1}} - 1 & \text{if } rand \leq 0.5 \end{cases} \tag{3.18}$$

where $rand$ is a uniformly random number from $[0, 1]$. The distribution index η and the mutation rate p_m are two control parameters. a_k is the lower and b_k is the upper bound of the k-th decision variable, respectively.

Chapter 4

Methodology

This section covers two proposal methodologies applied in this thesis. Firstly, it is described a single spiking neuron, which is implemented as a LIF neuron and its synaptic weights are optimized by the OMOPSO algorithm and the PSO algorithm. Finally, an implementation of the concept of LSM with a liquid connected by probabilities is optimized with the MOEA/D-DRA algorithm to find the optimal synaptic weights and delays belonging to the linked neurons into the liquid.

4.1 Single Spiking Neuron with Multi-objective Optimization

Two kinds of experiments were proposed:

1. A neuron trained by maximizing the inter distance of mean firing rates among classes and minimizing the standard deviation of the intra firing rate of each class.

2. In addition to the previous experiment, the dimension reduction of input vector besides of neuron training is proposed.

The LIF neuron model was implemented into jMetal [118, 119] where the OMOPSO algorithm is available, which was used for training the LIF neuron. Furthermore, the OMOPSO algorithm was configured as a mono-objective algorithm (PSO).

The design of this methodology is shown in figure 4.1. Initially, we set up the parameters of the OMOPSO algorithm and the LIF neuron model. Next, it is necessary to initialize the particles and Leaders (L_i) with uniformly random numbers to create a swarm.

Each particle represents a synaptic weight vector $(\bar{w})$ with the same size as the feature input vector $(\bar{x})$. Then, whole particles are evaluated into the LIF neuron model, by means of the objective functions. The non-dominated particles in the swarm will be L_i, which are sent to $\varepsilon\text{-}file$. Besides this, a crowding factor for each L_i is calculated as a second discrimination criterion.

After, an Internal Loop is initialized into an External Loop, and each particle into the Internal Loop is modified, updating the position and applying the mutation operators. Then, each particle is evaluated and its personal best value $(pbest)$ is

updated. A new particle replaces the *pbest* if such value is dominated by the new particle or if both are non-dominated concerning each other.

When all particles have been updated, the L_i are modified in the External Loop. Only the particles that overcome their *pbest* will try to enter into the L_i set. Once the L_i have been updated, they are sent to ε-$file$.

Finally, the crowding values of the L_i set are updated and we eliminate as many leaders as necessary to avoid overflow the size of the L_i set. The process is repeated until finalizing all iterations.

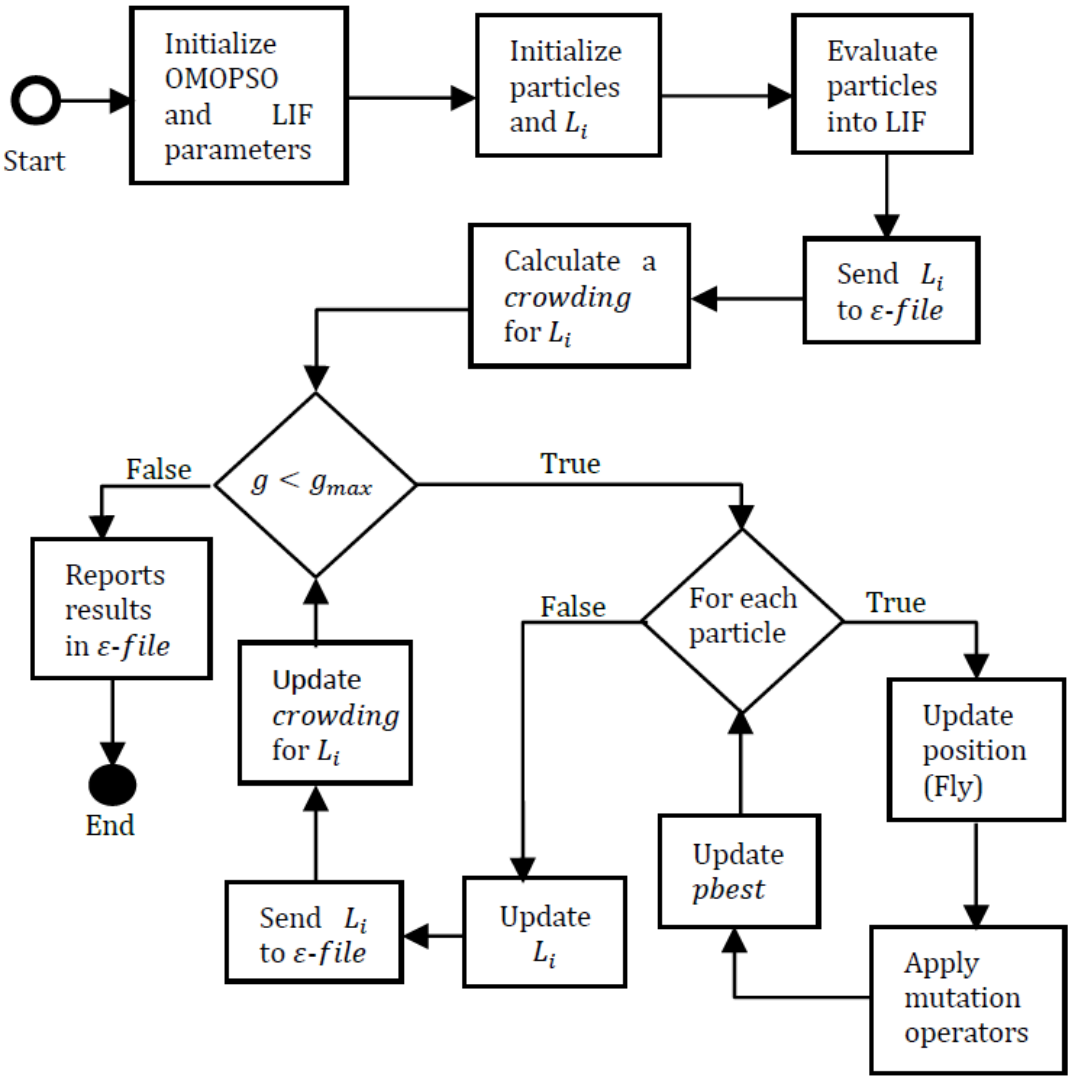

FIGURE 4.1: Methodology scheme for LIF neuron optimization

4.1.1 Search Engine

A vector $(\bar{w})$ is formed by the synaptic weights of the LIF neuron. This vector is used to create a swarm into the PSO and OMOPSO algorithms described in sections 3.4.1 and 3.5.1 respectively. The aim of applying these algorithms is to

optimize the synaptic weights to find an optimal solution to be able to solve classification problems.

4.1.2 Objective Functions

Three different objective functions were considered to guide the search solution to measure the performance of each one (particles).

A. The Euclidean distance between the combination of AFR_i and AFR_j where AFR is the average firing rate of each class and $i \neq j$. For this objective function, we look to maximize the separability between the classes.

$$MAX \ \ dist(AFR_i, AFR_j) \tag{4.1}$$

B. The Standard Deviation of the firing rate for each pattern class $SDFR_k$, where $k = 1, ..., K$ and K is the total of pattern classes. In this objective function, we look to minimize the dispersion of each pattern class.

$$MIN(SDFR_k) \tag{4.2}$$

C. The dimension of the input feature vector ($\bar{x}$). To avoid redundancies in information, we desire to reduce the dimensionality of the feature vectors, by minimizing the total of $1s$ of a binary mask ($\bar{r}$) with the same size of the input feature vector.

In our proposal, the number of objective functions is related to the number of classes of the dataset employed to evaluate the methodology.

4.2 Liquid State Machine with Multi-objective Optimization

The structure layers of LSM are detailed in figure 4.2, where an input signal x is transformed into spike trains through a 1D Coding layer of excitatory neurons, each of them is randomly connected to 20% of neurons (just with excitatory neurons) in the liquid. The liquid's connectivity structure (see section 3.3.1) considers 4:1 ratio between excitatory and inhibitory SRM neurons. Once the liquid has been simulated, a readout layer receives the signal from the neurons belonging to it. Then, the way that the state of the liquid is described is driven by the time-to-first-spike coding scheme allowing that the input spike trains are fed into a spatio-temporal filter which accumulates the temporal information of all input signals into a single high-dimensional intermediate liquid state [120, 75]. Hence, the high-dimensional liquid states are provided to the readout layer aiming to train a linear classifier. In this thesis, as a linear classifier, the Stochastic Gradient Descent (SGD) classifier (in sci-kit learn [121]) was used.

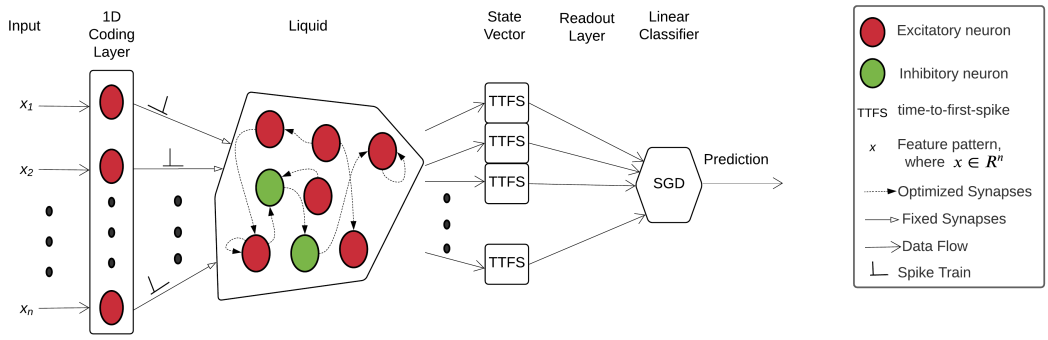

FIGURE 4.2: LSM structure layers.

4.2.1 Data Temporal Encoding

For the sake of preserving the original dimensionality of the input patterns, we seek the liquid to transform the pattern into high-dimensional states that describe it, in this work the encoding scheme *"1-D coding"* [42, 99] is used in order to generate one temporal value or firing time from a real value, allowing the original data dimension to be retained the equation 4.3.

$$y(f) = \left[\frac{b-a}{range} \times f\right] + \left[\frac{(a \times M) - (b \times m)}{range}\right] \tag{4.3}$$

where $y(f)$ is the firing time value, f is the original feature value, a and b represent the temporal coding lower and upper interval limits respectively, $range = M - m$ is the range of the original data, whereas M and m hold the maximum and minimum bounds of the original data that f takes.

4.2.2 Search Engine

Once the structure of the liquid has been created by probability connections with Lambda model (defined in section 3.3.1), a solution is formed into two parts: the synaptic weights and the delays of each pair of neuron connections. In the first part, all synaptic weights are located, then the delays. Figure 4.3 shows the design of a solution since the synaptic weights and delays from the interconnected neurons into the liquid, where w_{ji} and d_{ji} are denoted for the synaptic weight and delay respectively. The pre-synaptic neuron into liquid is denoted by i and the post-synaptic neuron by j.

In this work, the liquid structure is first mapped to a solution (vector) of real values, which consists of the synaptic weights and delays of the whole synapses in the liquid. A set of such solutions will form the population to be evolved by the MOEA/D-DRA algorithm. The liquid was implemented into jMetalPy [122] to be trained.

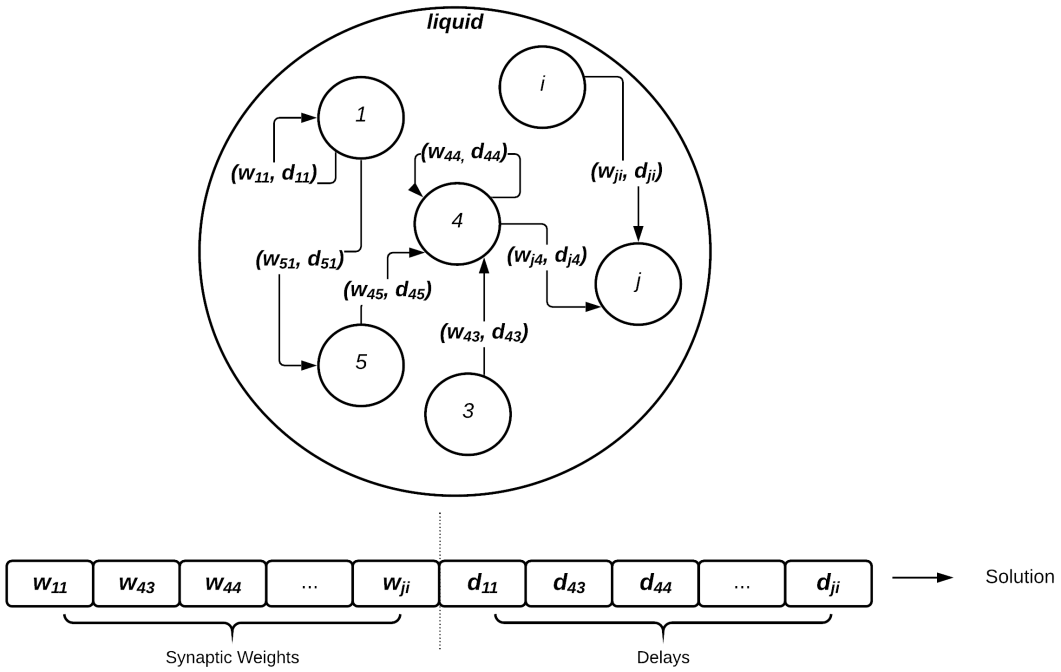

FIGURE 4.3: Representation of a solution formed by two key parts: synaptic weights and delays.

4.2.3 Objective Functions

Following the *separation metric* proposed in [74] and used in [81, 82] to determine the effectiveness of a liquid, i.e., it refers to the ability of the liquid to produce discernibly different patterns when given different classes of inputs [81]. Given that metric, we propose an objective functions approach to address searching in the objective space, which is considered to explore a Pareto set.

It is described as follows from two essential aspects: inter-class distance C_d, and intra-class variance $\rho(O_m)$ defined by equations 4.4 and 4.6, respectively.

Firstly, it requires us to divide the whole of state vectors O generated through the time-to-first-spike of each neuron belonging to the liquid into N subsets, O_m, one for each class, where N is the total number of classes. Individual state vectors are represented by o.

$$MAX \;\; C_d = ||\mu(O_m) - \mu(O_n)||_2, \quad m \neq n \tag{4.4}$$

In equation 4.4, the L_2-norm is computed, $||\cdot||_2$, between the combination of the center mass (equation 4.5) for every pair of classes where $m \neq n$. The objective function aims to maximize the separability between the set of pattern classes.

$$\mu(O_m) = \frac{\sum_{o_n \in O_m} o_n}{|O_m|} \tag{4.5}$$

For clarity in equation 4.5, $|\cdot|$ refers to the cardinality of a set.

In equation 4.6, $\rho(O_m)$ refers to computing the average amount of variance for each state vector within the class m from the center of mass for that class. Concerning to this objective function, it wants to minimize the spread of state vectors belonging to each pattern class.

$$MIN \ \ \rho(O_m) = \frac{\sum_{o_n \in O_m} ||\mu(O_m) - o_n||_2}{|O_m|} \tag{4.6}$$

In this thesis, the number of objective functions is related to the dataset's number of classes.

Chapter 5

Experiments and Results

This chapter covers the results obtained from the methodology described in chapter 4. Once the Theoretical Background and Methodology have been studied, two experimentations are proposed. For the sake of evaluating the performance of each experimentation, the results are carefully analysed and explained.

5.1 Experimental Design

First experimentation details the results from the comparison between the optimization of a single spiking neuron with mono-objective and multi-objective approaches.

5.1.1 Design of Experimentation

Four datasets from the UCI Machine Learning Repository [123] were employed for the experimentation: Glass, Iris Plant, SPECT and Wine. Table 5.1 shows the details of the datasets used.

Each dataset was randomly divided in two subsets with approximately the same size. The first one was employed as training set and the second one as testing set.

TABLE 5.1: Datasets employed for experimentation

Dataset	Instances	Classes	Features
Glass	214	6	9
Iris Plant	150	3	4
SPECT	267	2	22
Wine	178	3	13

With the aim to observer the performance of this proposal, four experiments were configured according to the objective functions seen in section 4.1.2. The characteristics of each experiment are defined below and summarized in table 5.2.

- **Experiment #1** (Exp #1) was defined as multi-objective problem, focusing on the A and B objective functions. The OMOPSO algorithm was used to optimize the synaptic weight vector of the LIF neuron.

- **Experiment #2** (Exp #2) employs the multi-objective approach, considering the A, B and C objective functions. The OMOPSO algorithm was taken to optimize the synaptic weight vector and the dimension of the input vector. Concerning the optimization of the last parameter, a binary mask $(\bar{r})$ was used in equation 3.3 to calculate a modified input current given by equation 5.1.

$$I = \bar{x} \cdot \bar{w} \cdot \bar{r} \cdot \theta \tag{5.1}$$

- **Experiment #3** (Exp #3) was designed as a mono-objective problem. The objective function (equation 5.2) was formed by the weighted sum of two objective functions. The first one is the inverse of the summation of the Euclidean distances among all combinations of AFR_i and AFR_j, and the second objective is the sum of the standard deviation of the firing rate for all classes [47]. PSO algorithm was used to design the synaptic weight vectors.

$$MIN(f) = \frac{1}{dist(\mathbf{AFR})} + \sum_{k=1}^{K} SDFR_k \tag{5.2}$$

- **Experiment #4** (Exp #4) is a mono-objective approach that seeks to optimize the synaptic weight vector and the dimension of the input vector by means of the PSO algorithm. The objective function (equation 5.3) is formed by the weighted sum in equation 5.2 and the rate of T and D, where T is the total of $1s$ in the binary mask $\bar{r}$ and D is the dimension of the input feature vector.

$$MIN(f) = \frac{1}{dist(\mathbf{AFR})} + \sum_{k=1}^{K} SDFR_k + \frac{\mathbf{T}}{\mathbf{D}} \tag{5.3}$$

TABLE 5.2: Configuration for experimentation

	Algorithm	**Optimized Parameters**	**Objective Function**
Exp #1	OMOPSO	synaptic weight vector	A, B
Exp #2	OMOPSO	synaptic weight vector and dimension of input vectors	A, B, C
Exp #3	PSO	synaptic weight vector	A, B
Exp #4	PSO	synaptic weight vector and dimension of input vectors	A, B, C

Table 5.3 shows a compendium of the number of objective functions by experiment for each dataset.

TABLE 5.3: Total of Objective Functions by experiment

Dataset	**Classes**	**Exp #1**	**Exp #2**	**Exp #3**	**Exp #4**
Glass	6	21	22	1	1
Iris Plant	3	6	7	1	1
SPECT	2	3	4	1	1
Wine	3	6	7	1	1

Each experiment consisted of 40 independently executions per dataset, to guarantee statistical significance. The parameters values used in the OMOPSO algorithm and the LIF neuron model [47] are detailed in tables 5.4 and 5.5 respectively.

TABLE 5.4: Configuration OMOPSO Parameters

Max particle size:	100
Max iterations:	1000
ε-*file* size:	100
Uniform Mutation	
Mutation probability:	$\frac{1.0}{Number\ of\ problem\ variables}$
Perturbation index:	0.5
Non-uniform Mutation	
Mutation probability:	$\frac{1.0}{Number\ of\ problem\ variables}$
Perturbation index:	0.5
Max iterations:	1000

TABLE 5.5: Configuration LIF Parameters

a	0.5
b	-0.001
c	-50 mV
v_i	-60 mV
$v_{threshold}$	50 mV
Time	1000 ms
h	1
θ	0.1

The initial synaptic weights were generated randomly $\in [0, 1]$.

5.1.2 Experimental Results

For each execution, at the end of the training phase, the total of particles are evaluated in the LIF neuron model using the training set, and the classification accuracy

is calculated for each particle. Finally, the particle with the best performance is used in the testing phase to obtain the accuracy in the testing set.

Tables 5.6 and 5.7 show the results obtained, where the accuracy values along with the standard deviations grade the performance in the experiments. The accuracy of the training phase corresponds to the average of the performance of the best particles obtained in each experiment, whereas the accuracy of the testing phase is obtained from the average of the performance of these particles applied to the testing set. The highest accuracy values are marked in bold font.

TABLE 5.6: Accuracy of training phase over each experiment

Dataset	**Exp #1**	**Exp #2**	**Exp #3**	**Exp #4**
Glass	**0.5050 ± 0.0441**	0.5031 ± 0.0405	0.39 ± 0.0030	0.3638 ± 0.0726
Iris Plant	**0.9817 ± 0.0131**	0.9793 ± 0.0157	0.9 ± 0.0232	0.8987 ± 0.0219
SPECT	**0.8592 ± 0.0243**	0.8286 ± 0.0227	0.7276 ± 0.0325	0.7273 ± 0.0344
Wine	0.7858 ± 0.0358	**0.8048 ± 0.0336**	0.6849 ± 0.0432	0.6986 ± 0.0301

TABLE 5.7: Accuracy of testing phase over each experiment

Dataset	**Exp #1**	**Exp #2**	**Exp #3**	**Exp #4**
Glass	0.3516 ± 0.1141	**0.3695 ± 0.1163**	0.3472 ± 0.0947	0.3594 ± 0.0960
Iris Plant	0.94 ± 0.0383	**0.9543 ± 0.0220**	0.9003 ± 0.0355	0.8883 ± 0.0303
SPECT	0.7043 ± 0.1051	0.7019 ± 0.1006	**0.7157 ± 0.0545**	0.7073 ± 0.0523
Wine	0.7322 ± 0.0604	**0.7397 ± 0.0610**	0.6706 ± 0.0505	0.6858 ± 0.0451

Table 5.8 shows the average amount of input features employed by the LIF neuron model and its corresponding ratio concerning the total size of the original input feature vector.

TABLE 5.8: Analysis of reduction of features of input vector

	Exp #2		Exp #4	
Dataset	Average number of features employed	Ratio of features used	Average number of features employed	Ratio of features used
Glass	5.550 ± 1.999	0.617	6.00 ± 1.377	0.667
Iris Plant	2.575 ± 0.747	0.640	3.050 ± 0.221	0.760
SPECT	10.325 ± 5.677	0.469	21.675 ± 0.562	0.985
Wine	8.475 ± 3.266	0.652	5.850 ± 1.902	0.450

5.1.3 Statistical Analysis

Several statistic tests were applied to the results shown in the previous section. Firstly, Shapiro-Wilk test was executed to identify the normality of our data, this

will define the kind of parametric or non-parametric tests to be used along with our data, it means to check if the data follows a normal distribution. Shapiro-Wilk is a normality test published in 1965 by Samuel Sanford Shapiro and Martin Wilk [124] where they focus to find if a data set follows a normal distribution. Shapiro-Wilk test has proven to be able to detect a dimension anomaly for an wide variety of statistical distributions, besides those with Gaussian kurtosis values, which has been recommended as a stronger normality general test. This test is based on a correlation of sample "order statistic" with those of a normal distribution [125].

Statistical Analysis was divided into two sections; the results from statistic tests for the Training phase are shown and discussed. Subsequently, the results of statistic tests computed in the Testing phase are analyzed.

Training phase statistical analysis

In Shapiro-Wilk test, the null-hypothesis (H_0) states the samples come from a normal distribution. In Table 5.9, for a significance level $\alpha = 0.05$, the *p-values* obtained show that approximately half of the results do not reject H_0, but the rest of the results reject H_0. Therefore, non-parametric statistic tests were applied.

TABLE 5.9: Shapiro-Wilk test in Training phase

	OMOPSO Experiments		PSO Experiments	
Dataset	#1	#2	#3	#4
Iris Plant	0.0002868 H_0 is rejected	0.000761 H_0 is rejected	0.1257 H_0 is not rejected	0.189 H_0 is not rejected
Wine	0.6275 H_0 is not rejected	0.0407 H_0 is rejected	0.08713 H_0 is not rejected	0.3317 H_0 is not rejected
Glass	0.0002413 H_0 is rejected	0.001126 H_0 is rejected	6.64E-14 H_0 is rejected	1.09E-11 H_0 is rejected
SPECT	0.06032 H_0 is not rejected	0.07665 H_0 is not rejected	0.3015 H_0 is not rejected	0.09427 H_0 is not rejected

Non-parametric statistical tests

Non-parametric tests to be used along with our data. These tests were applied by means of CONTROLTEST [126] package tool for comparison between experiments. Specifically, three non-parametric tests were applied: Friedman, Friedman Aligned Ranks, and Quade. In these tests, the null-hypothesis (H_0) states that the data of

the experiments follow the same distribution [126] (there is no difference in their performance).

Table 5.10 reports the average ranks obtained from these statistical tests on all the experiments. The smaller values in bold font, indicate that Experiment #1 obtained the best performance consistently.

TABLE 5.10: Average rankings of the experiments for the Training phase

Experiment	Friedman	Friedman Aligned Ranks	Quade
#1	**1.25**	**4.0**	**1.2**
#2	1.75	5.0	1.80
#3	3.25	12.25	3.199
#4	3.75	12.75	3.8

Table 5.11 shows the *p-value* for each statistical test and the sentence corresponding to the status of H_0 for a significance level $\alpha = 0.05$. If the *p-value* is greater than α indicates that there does not exist evidence to reject H_0. Therefore, the tests Friedman and Quade rejected H_0. However, these results do not give enough information to select the best experiment, and it was necessary to perform a post-hoc procedure. From Table 10, Experiment #1 was taken as the control experiment.

TABLE 5.11: Contrast the null-hypothesis in Training phase

Statistical Test	*p-value*	Hypothesis Testing
Friedman	0.01694	H_0 is rejected
Friedman Aligned Ranks	0.3806	H_0 is not rejected
Quade	1.04E-04	H_0 is rejected

Table 5.12 shows the results of the post-hoc procedure, where the *p-values* were adjusted by Holm's correction. For $\alpha = 0.05$, the adjusted *p-values* for the comparison between the control experiment and the Experiments #3 and #4 show that the Experiment #1 obtained better performance.

TABLE 5.12: Adjusted *p-values* by Holm's correction for Training phase

Experiment	Friedman Holm	Quade Holm
#1 vs #4	0.01667	0.01667
#1 vs #3	0.025	0.025
#1 vs #2	0.05	0.05

Testing phase statistical analysis

In Shapiro-Wilk test, the null-hypothesis (H_0) states the samples come from a normal distribution. Table 5.13 shows the results where the *p-values* were contrasted with a significance level of $\alpha = 0.05$. The *p-values* obtained show that five results reject H_0, and eleven results do not reject H_0. Then, non-parametric statistic tests were applied.

TABLE 5.13: Shapiro-Wilk test in Testing phase

	OMOPSO Experiments		PSO Experiments	
Dataset	#1	#2	#3	#4
Iris Plant	0.05354 H_0 is not rejected	0.02317 H_0 is rejected	0.4062 H_0 is not rejected	0.2015 H_0 is not rejected
Wine	0.4185 H_0 is not rejected	0.4515 H_0 is not rejected	0.4542 H_0 is not rejected	0.2932 H_0 is not rejected
Glass	0.06616 H_0 is not rejected	0.00249 H_0 is rejected	4.17E-08 H_0 is rejected	4.71E-07 H_0 is rejected
SPECT	0.002891 H_0 is rejected	0.08466 H_0 is not rejected	0.7195 H_0 is not rejected	0.07491 H_0 is not rejected

Non-parametric statistical tests

Non-parametric tests to be used along with our data. These tests were applied by means of CONTROLTEST [126] package tool for comparison between experiments. Specifically, three non-parametric tests were applied: Friedman, Friedman Aligned Ranks, and Quade. In these tests, the null-hypothesis (H_0) states that the data of the experiments follow the same distribution [126].

Table 5.14 shows the average ranks obtained from Friedman, Friedman Aligned Ranks and Quade tests for all the results. In the three tests, the smaller average ranks, in bold font, specify that Experiment #2 had the best performance.

TABLE 5.14: Average rankings of the experiments for the Testing phase

Experiment	**Friedman**	**Friedman Aligned Ranks**	**Quade**
#1	2.5	6.5	2.3
#2	**1.75**	**4.75**	**1.2999**
#3	3.0	11.75	3.4
#4	2.75	11.0	3.0

Table 5.15 shows the *p-value* for each statistical test The significance level was set to $\alpha = 0.05$. Quade test rejects H_0. Nonetheless, this result does not present enough information to choose the best experiment. So, a post-hoc procedure was carried out. From Table 14, Experiment #2 was used as the control experiment.

TABLE 5.15: Contrast the null-hypothesis in Testing phase

Statistical Test	*p-value*	Hypothesis Testing
Friedman	0.5519	H_0 is not rejected
Friedman Aligned Ranks	0.3632	H_0 is not rejected
Quade	0.0381	H_0 is rejected

Table 5.16 shows the results of the post-hoc procedure for Quade test, where *p-values* were adjusted by Holm's correction. The *p-values* were compared against a significance level of $\alpha = 0.05$. The *p-values* for the comparison between the control experiment and the Experiment #3 and #4 show that the Experiment #2 had better performance.

TABLE 5.16: Adjusted *p-values* by Holm's correction for Testing phase

Experiment	Quade Holm
#2 vs #4	0.01667
#2 vs #3	0.025
#2 vs #1	0.05

5.1.4 Discussion

In this first experimentation, a methodology for training full and partially connected LIF spiking neurons is shown, using the OMOPSO algorithm and PSO algorithm for solving pattern recognition problems. The experiments were designed under a multi-objective approach and their results were compared statistically with the results of mono-objective experiments. Each experiment was tested on four well-known benchmark datasets by performing 40 independent executions for each dataset. The results have shown that the Experiments #1 and #2 obtained the best performances in the Training and Testing phases, respectively. Therefore, the multi-objective approach provides an adequate alternative to optimize LIF spiking neurons. One interesting characteristic of this experimentation consists on the reduction of dimensionality of the input feature vectors to avoid redundancies in the input information.

5.2 Second Experimentation

This section details the results obtained from the optimization of LSM's parameters and these are compared with an SNN-based classifier in the state of art.

5.2.1 Design of Experimentation

Twelve supervised classification datasets which can be obtained from the UCI Machine Learning Repository [123] were employed in this experimentation: Balance Scale, Blood Transfusion Service Center (Blood), Breast Cancer Wisconsin (Breast Cancer), Japanese Credit Screening (Card), Pima Indians Diabetes (Diabetes), Fertility, Glass, Ionosphere, Iris Plant, Liver, Parkinson, and Wine. Table 5.17 shows the details of each dataset employed.

TABLE 5.17: Datasets description

Dataset	Instances	Classes	Features
Balance Scale	625	3	4
Blood	748	2	4
Breast Cancer	683	2	9
Card	653	2	15
Diabetes	768	2	8
Fertility	100	2	9
Glass	214	6	9
Ionosphere	351	2	33
Iris Plant	150	3	4
Liver	345	2	6
Parkinson	195	2	22
Wine	178	3	13

Each dataset was randomly divided into two subsets with approximately the same size and similar quantity of samples into each. The first one was used as training set to optimize the liquid through evolutionary optimization and the second one as testing set to prove the performance of the best solution found by the evolutionary optimization.

With the aim to observe the performance of our proposal, an experimental design was conducted to compare our proposal's performance against a previous method developed in [99] for deploying SNN-based classifiers.

Experimental design with Multi-objective approach and SNN-based classifiers: In this experiment we compared our multi-objective approach through the MOEA/D-DRA algorithm, optimizing the synaptic weights and delays generated by the interconnected neurons in the liquids and the configurations' SNN-based classifiers: Gamma 1 (γ_1) and Gamma 2 (γ_2), detailed in [99].

Table 5.18 shows our configurations for this experimental design.

TABLE 5.18: Configurations of the experimental design with Multi-objective approach and SNN-based classifiers.

Configuration	Liquid	Algorithm	Objective Functions
MOEA/D-DRA Lambda	Lambda model	MOEA/D-DRA	MAX C_d MIN $\rho(O_m)$
Gamma 1	-	Differential Evolution	MIN Squared error
Gamma 2	-	Differential Evolution	MIN Accuracy error

In order to achieve statistical significance we executed 33 independent runs in our experimental design.

Table 5.19 reports the number of objective functions implemented to be optimized by the MOEA/D-DRA algorithm in the experiments.

TABLE 5.19: Total of objective functions to optimize with MOEA/D-DRA by each dataset.

Dataset	Classes	Total of Objective Functions
Balance Scale	3	6
Blood	2	4
Breast Cancer	2	4
Card	2	4
Diabetes	2	4
Fertility	2	4
Glass	6	21
Ionosphere	2	4
Iris Plant	3	6
Liver	2	4
Parkinson	2	4
Wine	3	6

Specific parameters used in our experiments are described as follows:

- *Temporal Encoding*: $a = 0.01$, and $b = 10$.
- *SRM model*: threshold $\theta = 1$ mV, $\tau = 9$, synaptic weight range $\in [0.0001, 10]$, and delay range $\in [0.0001, 10]$ms.
- *Liquid with Lambda model*: liquid size=50 neurons, excitatory neurons= 80% and inhibitory neurons= 20%, simulation time= 15 ms, $\lambda = 2$, value of C for each type of synapses are set as $C_{EE} = 0.3$, $C_{EI} = 0.2$, $C_{IE} = 0.4$, $C_{II} = 0.1$, and the probability that neurons from the 1D Coding Layer connect towards the liquid's excitatory neurons is 20% with synaptic weight range $\in [0.00001, 1.0]$, and column size $5 \times 5 \times 2$.

- *MOEA/D-DRA*: population size= 100, call functions= 10000, $F = 0.5$, $C_r = 1.0$, $K = 0.5$, neighbor size= 20, neighbor selection probability= 0.9, variables range$\in [0.0001, 10.0]$, mutation operator= *Polynomial Mutation*, mutation distribution index= 20, and aggregative function= $Tschebycheff$.

SGD classifier parameters are used by defaults configuration in sci-kit learn.

5.2.2 Experimental Results

To measure our configuration's accuracy performance, every solution from the optimal Pareto set is tested using the training set. Once the liquid is simulated with each solution over that dataset, the state vectors are obtained. Then these feed SGD, and finally, the training performance accuracy is calculated for each solution. The testing set is used to find the best performance in the liquid obtained from the previous step. Liquid's state vectors are computed and fed into the SGD classifier to calculate the testing performance accuracy.

A multi-objective approach is taken to compare against a grammatical evolution (GE)-based methodology (see [99] for more details) to automatically design spiking neural networks (SNNs) (referred in this work as SNN-based classifiers). Both methodologies are used to solve pattern classification problems. The comparison can be seen in Table 5.20 and 5.21, both tables show the performance accuracy and standard deviation in the training and testing phase for the same classification problems.

TABLE 5.20: Accuracy (Mean±Std.Dev) of Training and Testing phase for Balance Scale, Blood, Breast Cancer, Card, Diabetes and Fertility datasets.

Dataset	Configuration	Training accuracy	Testing accuracy
Balance	MOEA/D-DRA Lambda	**0.9509±0.0080**	**0.8916±0.0187**
	Gamma 1	0.8528±0.0197	0.8346±0.0261
	Gamma 2	0.8960±0.0062	0.8647±0.0134
Blood	MOEA/D-DRA Lambda	**0.8134±0.0132**	**0.7739±0.0214**
	Gamma 1	0.776±0.0076	0.7618±0.0088
	Gamma 2	0.7957±0.0145	0.7685±0.0155
Breast Cancer	MOEA/D-DRA Lambda	**0.9858±0.0055**	**0.9607±0.0114**
	Gamma 1	0.9574±0.0111	0.9384±0.0140
	Gamma 2	0.9749±0.0062	0.9478±0.0117
Card	MOEA/D-DRA Lambda	0.8861±0.0127	0.8439±0.0166
	Gamma 1	0.8740±0.0134	**0.8596±0.0197**
	Gamma 2	**0.8879±0.0120**	0.8535±0.0166
Diabetes	MOEA/D-DRA Lambda	0.7824±0.0180	0.7232±0.0263
	Gamma 1	0.7810±0.0153	0.7370±0.0152
	Gamma 2	**0.7902±0.0134**	**0.7389±0.0205**
Fertility	MOEA/D-DRA Lambda	**0.9758±0.0240**	0.8236±0.0463
	Gamma 1	0.9455±0.0199	**0.8479±0.0462**
	Gamma 2	0.9370±0.0131	0.8236±0.0484

TABLE 5.21: Accuracy (Mean±Std.Dev) of Training and Testing phase for Glass, Ionosphere, Iris Plant, Liver, Parkinson and Wine datasets.

Dataset	Configuration	Training accuracy	Testing accuracy
Glass	MOEA/D-DRA Lambda	**0.7567±0.0324**	0.5715±0.0483
	Gamma 1	0.4895±0.0574	0.4351±0.0476
	Gamma 2	0.7126±0.0190	**0.6186±0.0413**
Ionosphere	MOEA/D-DRA Lambda	0.9491±0.0147	0.8783±0.0236
	Gamma 1	0.9351±0.0182	0.8907±0.0240
	Gamma 2	**0.9616±0.0113**	**0.9015±0.0201**
Iris Plant	MOEA/D-DRA Lambda	**0.9943±0.0093**	**0.9398±0.0237**
	Gamma 1	0.9794±0.0123	0.9325±0.0164
	Gamma 2	0.9923±0.0074	0.9358±0.0261
Liver	MOEA/D-DRA Lambda	**0.7717±0.0239**	0.6385±0.0420
	Gamma 1	0.7472±0.0183	**0.6723±0.0302**
	Gamma 2	0.7636±0.0196	0.6612±0.0295
Parkinson	MOEA/D-DRA Lambda	0.9032±0.0196	0.8089±0.0402
	Gamma 1	0.9025±0.0266	0.8380±0.0387
	Gamma 2	**0.9200±0.0172**	**0.8494±0.0377**
Wine	MOEA/D-DRA Lambda	**1±0**	**0.9326±0.0326**
	Gamma 1	0.9318±0.0285	0.862±0.0491
	Gamma 2	0.9638±0.0164	0.8684±0.0458

Tables 5.20 and 5.21 suggest our multi-objective approach achieved better performance when learning the data. MOEAD/D-DRA Lambda was better in 8 datasets, and Gamma 2 configuration was better in the rest of the datasets. For the testing phase in 5 times MOEA/D-DRA Lambda was better, it was followed by Gamma 2 in four occasions and Gamma 1 in the rest of datasets. Figures 5.1 and 5.2 show the performance (shown as boxplots) in the testing phase of our multi-objective approach, which shows that it has similar performance to that of the SNN-based classifiers configurations, even the standard deviations are similar too.

The heat-maps in figures 5.3 and 5.4 show the state vectors from the best solution obtained by our MOEA/D-DRA Lambda configuration with the patterns of the training set and testing set, respectively. It can be observed that the liquid has great separability between classes from the Breast Cancer dataset. In the x-axis, the state vector obtained for each pattern is located, and each neuron from the liquid conforms the y-axis. Then, the time-to-first-spike of each neuron by each pattern class is illustrated. An intense red colour represents the value of -1; it means that the neuron did not generate a spike. The variation of colour is composed of the time spike evoked during the simulation as is indicated in the right side of the heat-map, being purple the colour corresponding to the farthest time to generate a spike.

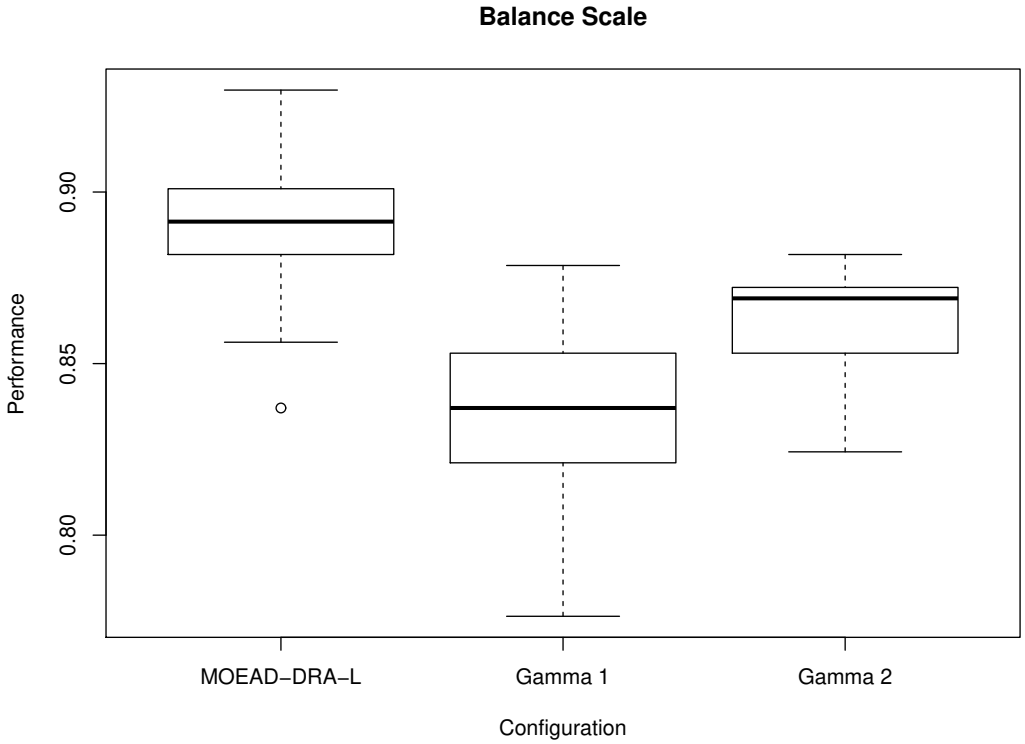

FIGURE 5.1: Boxplots of the performance in Testing phase for the Multi-objective approach and SNN-based classifiers on the Balance Scale dataset, MOEA/D-DRA Lambda (MOEA/D-DRA-L).

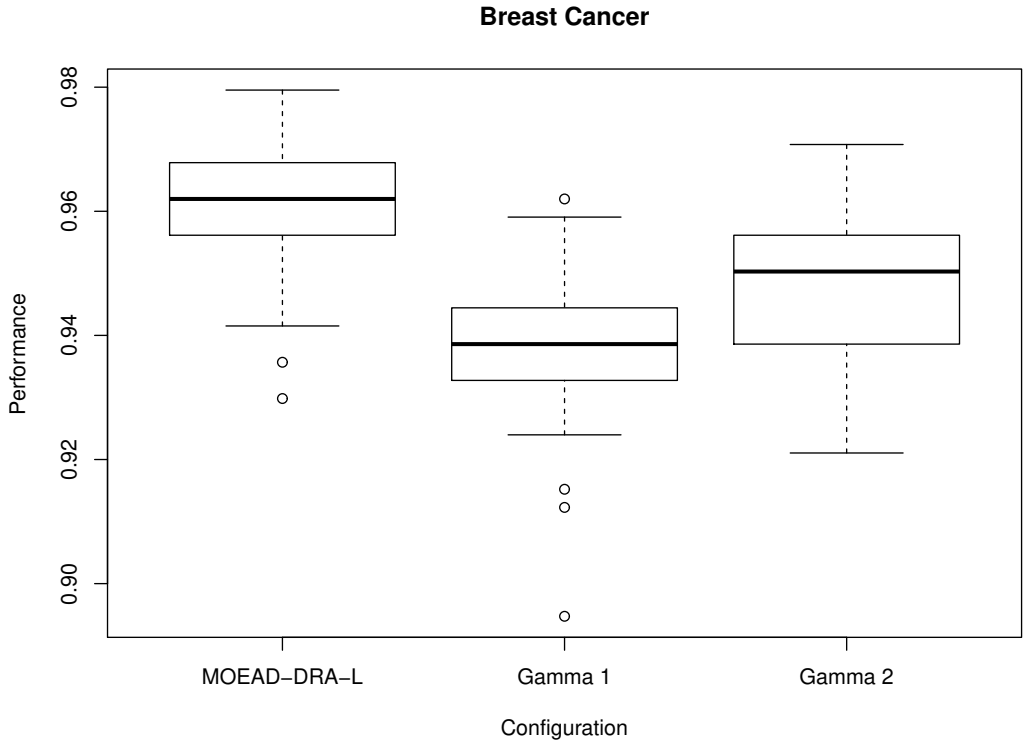

FIGURE 5.2: Boxplots of the performance in Testing phase for the Multi-objective approach and SNN-based classifiers on the Breast Cancer dataset: To refers MOEA/D-DRA Lambda (MOEA/D-DRA-L).

A statistical analysis of this behaviour is presented in the next section.

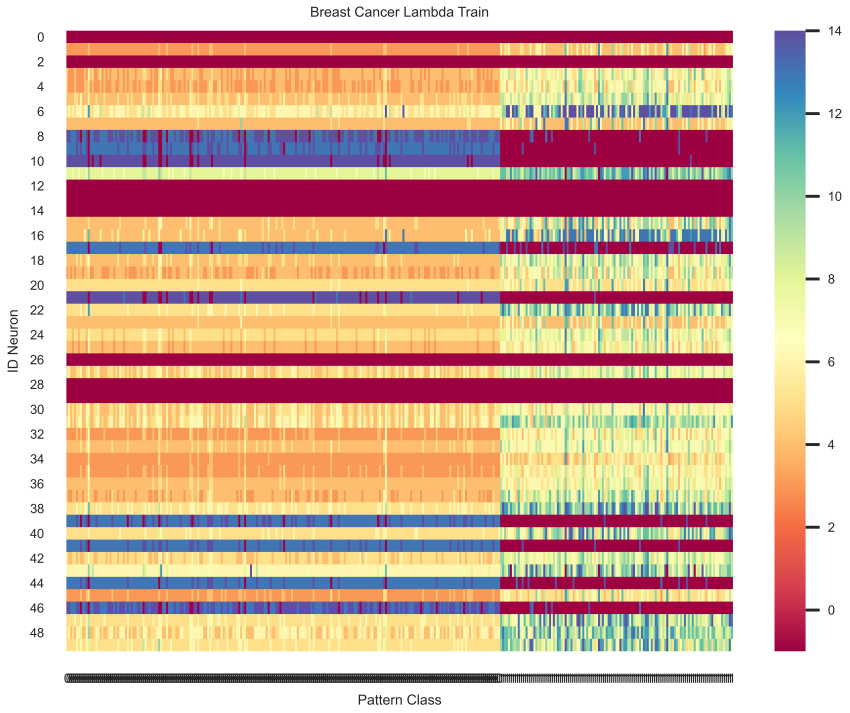

FIGURE 5.3: State Vectors for Breast Cancer dataset with MOEA/D-DRA Lambda configuration in Training phase

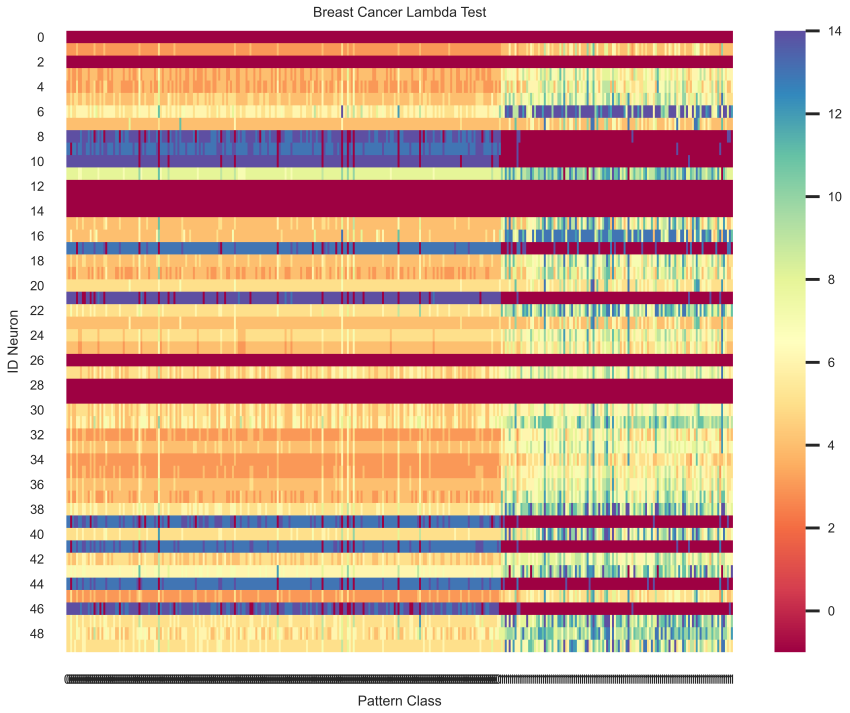

FIGURE 5.4: State Vectors for Breast Cancer dataset with MOEA/D-DRA Lambda configuration in Testing phase

5.2.3 Statistical Analysis

Non-parametric tests

Parametric statistical tests are commonly used to measure and contrast the performance of heuristic methods. However, these tests assume independence, normality and homoscedasticity of the gather data, which are not guaranteed in the case of meta-heuristic algorithms. Shapiro-Wilk test is used to check if the data follows a normal distribution [124, 125]. Non-parametric statistical test deals with this limitation and can be used for comparing heuristic approaches. In this thesis, we used CONTROLTEST [126] which is a tool specially designed for non-parametric comparison among heuristic algorithms to apply several rank tests to our data, these non-parametric tests are analogue to a two-way analysis of variance or ANOVA. The first one; Friedman test is used to identify if at least 2 samples from a bigger set represent populations with different median values. Therefore, it is a multiple-comparisons tests which objective is to detect significant differences between the performance of two or more heuristic algorithms. In addition to the Friedman test and to corroborate our findings, we use the Alignment Friedman test which uses a value of location computed as the average performance achieved by all algorithms in each problem. Finally; a Quade test is applied. This test considers that some problems might be more difficult than others. Accordingly this difficulty, the Quade test assigns a reward value to the winner algorithm of an instance where other algorithms exhibit a wide dispersion in their results. All these tests consider ranks, therefore, the lower the reported value, the better the performance achieved. The null-hypothesis H_0 for each tests states equality of medians between the populations [127, 128].

Multi-objective approach and SNN-based classifier

To contrast our work and a previous relevant paper in the state of the art (SNN-based classifier), we conducted a statistical analysis to determine if there exists an improvement between our proposal and published work. Following the same experimental design (explained in the previous section), we conducted a Shapiro-Wilk test to assess the normality of the data. Table 5.22 shows that for some configurations, the null-hypothesis (H_0) is rejected (with a significance level $\alpha = 0.05$). Therefore we cannot assume that all the results follow a normal distribution; three non-parametric statistical tests were applied to deal with this limitation.

TABLE 5.22: Shapiro-Wilk test in Testing phase for comparison between multi-objective configurations and SNN-based classifier over each dataset.

Dataset	MOEA/D-DRA Lambda	Gamma 1	Gamma 2
Balance Scale	0.3298 H_0 is not rejected	0.09717 H_0 is not rejected	0.008865 H_0 is rejected
Blood	0.4477 H_0 is not rejected	1.71E-06 H_0 is rejected	0.3235 H_0 is not rejected
Breast Cancer	0.08371 H_0 is not rejected	0.101 H_0 is not rejected	0.4931 H_0 is not rejected
Card	0.09307 H_0 is not rejected	0.0006931 H_0 is rejected	0.7892 H_0 is not rejected
Diabetes	0.05583 H_0 is not rejected	0.08288 H_0 is not rejected	0.2633 H_0 is not rejected
Fertility	0.002926 H_0 is rejected	0.0002584 H_0 is rejected	0.1889 H_0 is not rejected
Glass	0.2422 H_0 is not rejected	0.6466 H_0 is not rejected	0.3063 H_0 is not rejected
Ionosphere	0.4081 H_0 is not rejected	0.06188 H_0 is not rejected	0.2823 H_0 is not rejected
Iris Plant	1.99E-05 H_0 is rejected	0.04726 H_0 is rejected	0.1523 H_0 is not rejected
Liver	0.04725 H_0 is rejected	0.7064 H_0 is not rejected	0.4467 H_0 is not rejected
Parkinson	0.32 H_0 is not rejected	0.008275 H_0 is rejected	0.0005978 H_0 is rejected
Wine	0.006471 H_0 is rejected	0.3396 H_0 is not rejected	0.6751 H_0 is not rejected

Table 5.23 details the ranks computed for Friedman, Aligned Friedman and Quade tests. These tests are commonly employed when the assumption of normality in the data cannot be guaranteed. The lower ranks, in bold font, specify that Gamma 2 configuration achieved the best performance. The MOEA/D-DRA Lambda is located in the second place with respect to the performance.

TABLE 5.23: Ranks of experiments according to Friedman, Aligned Friedman and Quade Tests for Testing phase.

Experiment	Friedman	Aligned Friedman	Quade
MOEA/D-DRA Lambda	2.0416	18.3750	2.0576
Gamma 1	2.25	20.6666	2.2884
Gamma 2	**1.7083**	**16.4583**	**1.6538**

The *p-value* for each test is shown in Table 5.24. The significance level α was set

to 0.05. Aligned Friedman test rejects the null-hypothesis (H_0) which dictates that all the data samples follow the same distributions.

TABLE 5.24: *p-values* of non-parametric tests.

Statistical Test	*p-value*	Hypothesis Testing
Friedman	0.4259	H_0 is not rejected
Aligned Friedman	0.0098	H_0 is rejected
Quade	0.3973	H_0 is not rejected

Then, this statistical evidence suggests that Gamma 2 achieved better results in the ranks than the rest of configurations, but the comparison of $p - values$ against the significance level α indicates that Friedman and Quade tests states the all the data samples follow the same distribution. It means that our proposal had less performance than SNN-based classifiers.

5.2.4 Discussion

This experimental design was compared through a rigorous statistical analysis, which indicates that the multi-objective approach does not show statistically significant differences in contrast with SNN-based classifiers to solve pattern classification problems. But it can be underlined that an initial proposal, during the training phase has had better performance than two Gamma configurations. Nevertheless, in testing phase our proposal has failed at generalizing performance. In addition, we propose another method to extract the state of the liquid, employing the time-to-first-spike of each neuron after liquid's simulation. With this way to draw a high-dimensional vector, it can be observed that the liquid produces great separability between classes.

Our results just employ 1% of the call functions required by the SNN-based classifiers, it means less computation power than SNN-based classifiers, which used one million of call functions in the evolutionary procedure. In this work, using just a small quantity of the computation power required by SNN-based classifiers, we have obtained interesting results given a fascinating property of MOEAs, which is the capability to explore the solution space given a Pareto set approximation in a single run without much effort [129]. MOEA/D-DRA explores a wider area of the solution space giving as result an optimal Pareto Front instead of finding a single optimal solution.

Chapter 6

Conclusions & future work

In this work, Spiking Neural Networks parameter's are optimized through a Multi-objective Optimization approach to solve pattern classification problems.

The evolutionary multi-objective approach has been used as an option to optimize a large number of parameters belonging to the third generation of neural networks. Initially, to prove the efficiency of this approach, a single spiking neuron (LIF neuron) was optimized by means of a swarm multi-objective algorithm (OMOPSO), showing better results than a swarm mono-objective algorithm (PSO). In addition, a proposal to reduce the input connections to this neuron was presented. The reduction of the input vector was employed in the two approaches.

By analyzing statistically the obtained results, the multi-objective approach shows that by itself achieves better classification performances than mono-objective approach, improving when we apply reduction of the input vector, which means better performance with less computation power required. Then, with OMOPSO, better solutions are found due to it explore more efficiently the search space than PSO.

Three characteristics are important about our proposed objective functions that are in favour of using the LIF neuronal model: Firstly, neuron's average firing rate by each class and the dispersion of firing rates within each class. By maximizing Euclidean distance between the averages of the firing rate, the separability between each class is sought, the greater the distance between each average firing rate, the greater the separability between classes. Secondly, by minimizing the dispersion of firing rates within each class, the patterns belonging to one class are sought to be close to each other. By means of these two observations, we seek to guide neuron's synaptic weight vector through the search space. Finally, the minimization of the amount of input data received by the neuron through the design of a binary mask, it look to reduce the computational cost from the training process and avoiding redundancy in the data injected into the spiking neuron. In this way, a set of optimal solutions is obtained, formally known as the Pareto-optimal Front.

Since the previous experimentation, favorable results were obtained using the multi-objective approach to optimize a SNN, in this thesis we proposed to optimize all synapses from a liquid, which is a recurrent SNN as a fundamental piece of a LSM.

By employing the MOEA/D-DRA algorithm, we sought to optimize the synapses into the liquid. Based on the objective functions proposed from optimizing a single spiking neuron and an objective function of the state-of-art to optimize a liquid, two objective functions (extending to the number of classes in the dataset) were proposed to evaluate the liquid's separability property using the state vector generated by each input pattern. The first objective function seeks to maximize the L2-norm between each pair of average state vectors of each class. Moreover, the second objective function pursues to minimize the dispersion between the state vectors for each class. These objective functions were used to guide the search for non-dominated solutions uniformly spread in the objective space.

Once it was observed that the liquid, with its optimized parameters, has separability, a series of experiments were carried out, contrasted with the state-of-the-art SNN-based classifiers. The statistical results of the experiments showed us that our proposal has statistically significant differences against the Gamma1 configuration. In contrast, our proposal does not improve against the Gamma2 configuration. This behavior may be because SNN-based classifiers have some advantages. They are optimized by more evaluations (1,000,000) than our proposal (10,000). Our proposal obtained higher performance in the training phase but did not generalize correctly, in contrast with the Gamma1 and Gamma2 configurations. Furthermore, they are designed with a partially connected architecture; therefore, their performance does not show overfitting in the training process and, in turn, show better generalization.

As future work, we propose to review the proposed objective functions to find some modifications to improve the exploration in the search space to optimize LSM parameters. Another objective function that could be considered is to minimize the accuracy error (as proposed in the Gamma2 configuration) adapted with the other objective functions to evaluate if non-dominated solutions found during the evolutionary process have good classification performance. Also, we propose to increase the number of neurons in the liquid, including the optimization of the parameter λ (from Lambda Model Synaptic Connection) to control the density of connections in the liquid to a greater number of neurons, thus aiming to avoid the liquid's chaotic behaviour.

Bibliography

[1] I.A Basheer and M Hajmeer. "Artificial neural networks: fundamentals, computing, design, and application". In: *Journal of Microbiological Methods* 43.1 (2000), pp. 3–31. DOI: 10.1016/s0167-7012(00)00201-3.

[2] Beatriz A. Garro, Humberto Sossa, and Roberto A. Vazquez. "Artificial neural network synthesis by means of artificial bee colony (ABC) algorithm". In: *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011. DOI: 10.1109/cec.2011.5949637.

[3] Khabat Soltanian et al. "Artificial neural networks generation using grammatical evolution". In: *2013 21st Iranian Conference on Electrical Engineering (ICEE)*. IEEE, 2013. DOI: 10.1109/iraniancee.2013.6599788.

[4] Beatriz A. Garro and Roberto A. Vázquez. "Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms". In: *Computational Intelligence and Neuroscience* 2015 (2015), pp. 1–20. DOI: 10.1155/2015/369298.

[5] Wolfgang Maass. "Networks of spiking neurons: The third generation of neural network models". In: *Neural Networks* 10.9 (1997), pp. 1659–1671. DOI: 10.1016/s0893-6080(97)00011-7.

[6] Daniel Gardner, ed. *Neurobiology of Neural Networks*. The MIT Press, 1993. ISBN: 9780262071505. DOI: 10.7551/mitpress/4941.001.0001.

[7] Andrés Espinal et al. "Comparing Metaheuristic Algorithms on the Training Process of Spiking Neural Networks". In: *Studies in Computational Intelligence*. Springer International Publishing, 2014, pp. 391–403. DOI: 10.1007/978-3-319-05170-3_27.

[8] Richard Kempter, Wulfram Gerstner, and J. Leo van Hemmen. "Hebbian learning and spiking neurons". In: *Physical Review E* 59.4 (1999), pp. 4498–4514. DOI: 10.1103/physreve.59.4498.

[9] Wolfgang Maass. "Fast Sigmoidal Networks via Spiking Neurons". In: *Neural Computation* 9.2 (1997), pp. 279–304. DOI: 10.1162/neco.1997.9.2.279.

[10] Ammar Belatreche et al. "An evolutionary strategy for supervised training of biologically plausible neural networks". In: *in Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing*. 2003, pp. 1524–1527.

[11] Geoffrey Hinton. "Artificial Intelligence: Neural Networks". In: *Van Nostrand's Scientific Encyclopedia*. American Cancer Society, 2005. ISBN: 9780471743989. DOI: 10.1002/0471743984.vse0673. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471743984.vse0673. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471743984.vse0673.

[12] Tatt Hee Oong and N. A. M. Isa. "Adaptive Evolutionary Artificial Neural Networks for Pattern Classification". In: *IEEE Transactions on Neural Networks* 22.11 (2011), pp. 1823–1836. DOI: 10.1109/tnn.2011.2169426.

[13] T. Jayalakshmi and A. Santhakumaran. "A Novel Classification Method for Diagnosis of Diabetes Mellitus Using Artificial Neural Networks". In: *2010 International Conference on Data Storage and Data Engineering*. IEEE, 2010. DOI: 10.1109/dsde.2010.58.

[14] Piotr S. Maciag, Marzena Kryszkiewicz, and Robert Bembenik. "Online Evolving Spiking Neural Networks for Incremental Air Pollution Prediction". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020. DOI: 10.1109/ijcnn48605.2020.9206775.

[15] Huaizhi Wang et al. "Probabilistic wind power forecasting based on spiking neural network". In: *Energy* 196 (2020), p. 117072. DOI: 10.1016/j.energy.2020.117072.

[16] Kazuhiro Ohkura, Toshiyuki Yasuda, and Yoshiyuki Matsumura. "Coordinating the adaptive behavior for swarm robotic systems by using topology and weight evolving artificial neural networks". In: *IEEE Congress on Evolutionary Computation*. IEEE, 2010. DOI: 10.1109/cec.2010.5586552.

[17] Erick Israel Guerra-Hernandez et al. "A FPGA-Based Neuromorphic Locomotion System for Multi-Legged Robots". In: *IEEE Access* 5 (2017), pp. 8301–8312. DOI: 10.1109/access.2017.2696985.

[18] Zhenshan Bing et al. "A Survey of Robotics Control Based on Learning-Inspired Spiking Neural Networks". In: *Frontiers in Neurorobotics* 12 (2018). DOI: 10.3389/fnbot.2018.00035.

[19] Sander M. Bohte, Joost N. Kok, and Han La Poutré. "Error-backpropagation in temporally encoded networks of spiking neurons". In: *Neurocomputing* 48.1-4 (2002), pp. 17–37. DOI: 10.1016/s0925-2312(01)00658-0.

[20] Xiaoli Tao and Howard E. Michel. "Data Clustering Via Spiking Neural Networks Through Spike Timing-Dependent Plasticity". In: *Proceeding "IC-AI'04", June 21-24, 2004, Las Vegas*. 2004.

[21] Wulfram Gerstner. *Spiking neuron models : single neurons, populations, plasticity*. Cambridge University Press, 2002. 496 pp. ISBN: 0521890799. DOI: 10.1017/cbo9780511815706.

[22] Wolfgang Maass and Michael Schmitt. "On the complexity of learning for a spiking neuron (extended abstract)". In: *Proceedings of the tenth annual conference on Computational learning theory - COLT '97*. ACM Press, 1997. DOI: 10.1145/267460.267477.

[23] Filip Ponulak and Andrzej Kasinski. "Introduction to spiking neural networks: Information processing, learning and applications". In: *Acta neurobiologiae experimentalis* 71.4 (2011), 409—433. ISSN: 0065-1400. URL: http://europepmc.org/abstract/MED/22237491.

[24] Wolfgang Maass and Michael Schmitt. "On the Complexity of Learning for Spiking Neurons with Temporal Coding". In: *Information and Computation* 153.1 (1999), pp. 26–46. DOI: 10.1006/inco.1999.2806.

[25] Samanwoy Ghosh-Dastidar and Hojjat Adeli. "Third Generation Neural Networks: Spiking Neural Networks". In: *Advances in Intelligent and Soft Computing*. Springer Berlin Heidelberg, 2009, pp. 167–178. DOI: 10.1007/978-3-642-03156-4_17.

[26] Iulia M. Comsa et al. "Temporal Coding in Spiking Neural Networks with Alpha Synaptic Function". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020. DOI: 10.1109/icassp40776.2020.9053856.

[27] J. Stephen. Jubb. *Neural network design and the complexity of learning*. MIT Press, 1990.

[28] TM McGinnity et al. "Supervised Training of Spiking Neural Networks with Weight Limitation Constraints". In: (2004).

[29] A. L. Hodgkin and A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". In: *The Journal of Physiology* 117.4 (1952), pp. 500–544. DOI: 10.1113/jphysiol.1952.sp004764.

[30] Wulfram Gerstner. "Time structure of the activity in neural network models". In: *Physical Review E* 51.1 (1995), pp. 738–758. DOI: 10.1103/physreve.51.738.

[31] Wulfram Gerstner. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014. ISBN: 978-1107635197.

[32] E.M. Izhikevich. "Which Model to Use for Cortical Spiking Neurons?" In: *IEEE Transactions on Neural Networks* 15.5 (2004), pp. 1063–1070. DOI: 10.1109/tnn.2004.832719.

[33] Gregory D. Smith et al. "Fourier Analysis of Sinusoidally Driven Thalamocortical Relay Neurons and a Minimal Integrate-and-Fire-or-Burst Model". In: *Journal of Neurophysiology* 83.1 (2000), pp. 588–610. DOI: 10.1152/jn.2000.83.1.588.

[34] Wulfram Gerstner. "Spiking Neurons". In: *Pulsed Neural Networks*. Cambridge, MA, USA: MIT Press, 1999, 1–53. ISBN: 0626133504.

[35] E.M. Izhikevich. "Simple model of spiking neurons". In: *IEEE Transactions on Neural Networks* 14.6 (2003), pp. 1569–1572. DOI: 10.1109/tnn.2003.820440.

[36] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. "Spike-based strategies for rapid processing". In: *Neural Networks* 14.6-7 (2001), pp. 715–725. DOI: 10.1016/s0893-6080(01)00083-1.

[37] Hélène Paugam-Moisy and Sander Bohte. "Computing with Spiking Neuron Networks". In: *Handbook of Natural Computing*. Vol. 1. Springer Berlin Heidelberg, 2012, pp. 335–376. DOI: 10.1007/978-3-540-92910-9_10.

[38] Jesus L. Lobo et al. "Spiking Neural Networks and online learning: An overview and perspectives". In: *Neural Networks* 121 (2020), pp. 88–100. DOI: 10.1016/j.neunet.2019.09.004.

[39] Aboozar Taherkhani et al. "A review of learning in biologically plausible spiking neural networks". In: *Neural Networks* 122 (2020), pp. 253–272. DOI: 10.1016/j.neunet.2019.09.036.

[40] S. Bohté, J. Kok, and H. L. Poutré. "SpikeProp: backpropagation for networks of spiking neurons". In: *ESANN*. 2000.

[41] Ammar Belatreche et al. "Evolutionary Design of Spiking Neural Networks". In: *New Mathematics and Natural Computation* 02.03 (2006), pp. 237–253. DOI: 10.1142/s179300570600049x.

[42] Ammar Belatreche, Liam P. Maguire, and Martin McGinnity. "Advances in Design and Application of Spiking Neural Networks". In: *Soft Computing* 11.3 (2006), pp. 239–248. DOI: 10.1007/s00500-006-0065-7.

[43] Jinling Wang et al. "An online supervised learning method for spiking neural networks with adaptive structure". In: *Neurocomputing* 144 (2014), pp. 526–536. DOI: 10.1016/j.neucom.2014.04.017.

[44] Jinling Wang et al. "SpikeComp: An Evolving Spiking Neural Network with Adaptive Compact Structure for Pattern Classification". In: *Neural Information Processing*. Springer International Publishing, 2015, pp. 259–267. DOI: 10.1007/978-3-319-26535-3_30.

[45] Joseph M. Brader, Walter Senn, and Stefano Fusi. "Learning Real-World Stimuli in a Neural Network with Spike-Driven Synaptic Dynamics". In: *Neural Computation* 19.11 (2007), pp. 2881–2912. DOI: 10.1162/neco.2007.19.11.2881.

[46] Alexander Vandesompele, Florian Walter, and Florian Rohrbein. "Neuro-evolution of spiking neural networks on SpiNNaker neuromorphic hardware". In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2016. DOI: 10.1109/ssci.2016.7850250.

[47] Roberto A. Vazquez and Aleister Cachon. "Integrate and Fire neurons and their application in pattern recognition". In: *2010 7th International Conference on Electrical Engineering Computing Science and Automatic Control*. IEEE, 2010. DOI: 10.1109/iceee.2010.5608622.

[48] Roberto Vazquez. "Izhikevich neuron model and its application in pattern recognition". In: *Australian Journal of Intelligent Information Processing Systems* 11.1 (2010), pp. 35–40.

[49] N.G. Pavlidis et al. "Spiking neural network training using evolutionary algorithms". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* IEEE, 2005. DOI: 10.1109/ijcnn.2005.1556240.

[50] Shen Hong et al. "A cooperative method for supervised learning in Spiking neural networks". In: *The 2010 14th International Conference on Computer Supported Cooperative Work in Design*. IEEE, 2010. DOI: 10.1109/cscwd.2010.5472007.

[51] Roberto A. Vazquez. "Training spiking neural models using cuckoo search algorithm". In: *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011. DOI: 10.1109/cec.2011.5949684.

[52] Ahmed A. Abusnaina, Rosni Abdullah, and Ali Kattan. "Supervised Training of Spiking Neural Network by Adapting the E-MWO Algorithm for Pattern Classification". In: *Neural Processing Letters* 49.2 (2018), pp. 661–682. DOI: 10.1007/s11063-018-9846-0.

[53] Ammar Mohemmed et al. "Optimization of Spiking Neural Networks with dynamic synapses for spike sequence generation using PSO". In: *The 2011 International Joint Conference on Neural Networks*. IEEE, 2011. DOI: 10.1109/ijcnn.2011.6033611.

[54] Yaochu Jin, Ruojing Wen, and Bernhard Sendhoff. "Evolutionary Multi-objective Optimization of Spiking Neural Networks". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 370–379. DOI: 10.1007/978-3-540-74690-4_38.

[55] Abdulrazak Yahya Saleh, Siti Mariyam Shamsuddin, and Haza Nuzly Abdull Hamed. "Multi-Objective Differential Evolution of Evolving Spiking Neural Networks for Classification Problems". In: *IFIP Advances in Information and Communication Technology*. Springer International Publishing, 2015, pp. 351–368. DOI: 10.1007/978-3-319-23868-5_25.

[56] R. Batllori et al. "Evolving spiking neural networks for robot control". In: *Procedia Computer Science* 6 (2011), pp. 329–334. DOI: 10.1016/j.procs.2011.08.060.

[57] Llewyn Salt et al. "Parameter Optimization and Learning in a Spiking Neural Network for UAV Obstacle Avoidance Targeting Neuromorphic Processors". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2020), pp. 3305–3318. DOI: 10.1109/tnnls.2019.2941506.

[58] Javier Pérez et al. "Bio-inspired spiking neural network for nonlinear systems control". In: *Neural Networks* 104 (2018), pp. 15–25. DOI: 10.1016/j.neunet.2018.04.002.

[59] Rubén Crespo-Cano et al. "On the Automatic Tuning of a Retina Model by Using a Multi-objective Optimization Genetic Algorithm". In: *Artificial Computation in Biology and Medicine*. Springer International Publishing, 2015, pp. 108–118. DOI: 10.1007/978-3-319-18914-7_12.

[60] Antonio Martínez-Álvarez et al. "Automatic Tuning of a Retina Model for a Cortical Visual Neuroprosthesis Using a Multi-Objective Optimization Genetic Algorithm". In: *International Journal of Neural Systems* 26.07 (2016), p. 1650021. DOI: 10.1142/s0129065716500210.

[61] Mihaela Dimovska et al. "Multi-Objective Optimization for Size and Resilience of Spiking Neural Networks". In: *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2019.

[62] Junxiu Liu et al. "Multi-objective Spiking Neural Network Hardware Mapping Based on Immune Genetic Algorithm". In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Theoretical Neural Computation*. Springer International Publishing, 2019, pp. 745–757. DOI: 10.1007/978-3-030-30487-4_58.

[63] Herbert Jaeger, Wolfgang Maass, and Jose Principe. "Special issue on echo state networks and liquid state machines". In: *Neural Networks* 20.3 (2007), pp. 287–289. DOI: 10.1016/j.neunet.2007.04.001.

[64] Mantas Lukoševičius and Herbert Jaeger. "Reservoir computing approaches to recurrent neural network training". In: *Computer Science Review* 3.3 (2009), pp. 127–149. DOI: 10.1016/j.cosrev.2009.03.005.

[65] Wolfgang Maass, Thomas Natschläger, and Henry Markram. "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations". In: *Neural Computation* 14.11 (2002), pp. 2531–2560. DOI: 10.1162/089976602760407955.

[66] E. Goodman and D. Ventura. "Effectively using recurrently-connected spiking neural networks". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* IEEE, 2005. DOI: 10.1109/ijcnn.2005.1556107.

[67] Jacques Kaiser et al. "Scaling up liquid state machines to predict over address events from dynamic vision sensors". In: *Bioinspiration & Biomimetics* 12.5 (2017), p. 055001. DOI: 10.1088/1748-3190/aa7663.

[68] Thomas Natschläger, Wolfgang Maass, and Henry Markram. "The" liquid computer": A novel strategy for real-time computing on time series". In: *Special issue on Foundations of Information Processing of TELEMATIK* 8.ARTICLE (2002), pp. 39–43.

[69] Wolfgang Maass and Henry Markram. "On the computational power of circuits of spiking neurons". In: *Journal of Computer and System Sciences* 69.4 (2004), pp. 593–616. DOI: 10.1016/j.jcss.2004.04.001.

[70] D. Verstraeten et al. "Isolated word recognition with the Liquid State Machine: a case study". In: *Information Processing Letters* 95.6 (2005), pp. 521–528. DOI: 10.1016/j.ipl.2005.05.019.

[71] Chrisantha Fernando and Sampsa Sojakka. "Pattern Recognition in a Bucket". In: *Advances in Artificial Life*. Springer Berlin Heidelberg, 2003, pp. 588–597. DOI: 10.1007/978-3-540-39432-7_63.

[72] Grzegorz M. Wojcik and Wieslaw A. Kaminski. "Liquid state machine built of Hodgkin–Huxley neurons and pattern recognition". In: *Neurocomputing* 58-60 (2004), pp. 245–251. DOI: 10.1016/j.neucom.2004.01.051.

[73] Harald Burgsteiner et al. "Movement prediction from real-world images using a liquid state machine". In: *Applied Intelligence* 26.2 (2006), pp. 99–109. DOI: 10.1007/s10489-006-0007-1.

[74] E. Goodman and D. Ventura. "Spatiotemporal Pattern Recognition via Liquid State Machines". In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006. DOI: 10.1109/ijcnn.2006.246880.

[75] Stefan Schliebs, Haza Nuzly Abdull Hamed, and Nikola Kasabov. "Reservoir-Based Evolving Spiking Neural Network for Spatio-temporal Pattern Recognition". In: *Neural Information Processing*. Springer Berlin Heidelberg, 2011, pp. 160–168. DOI: 10.1007/978-3-642-24958-7_19.

[76] François Rhéaume, Dominic Grenier, and Éloi Bossé. "Multistate combination approaches for liquid state machine in supervised spatiotemporal pattern classification". In: *Neurocomputing* 74.17 (2011), pp. 2842–2851. DOI: 10.1016/j.neucom.2011.03.033.

[77] Beata J. Grzyb et al. "Which model to use for the Liquid State Machine?" In: *2009 International Joint Conference on Neural Networks*. IEEE, 2009. DOI: 10.1109/ijcnn.2009.5178822.

[78] Grzegorz M. Wojcik and Wieslaw A. Kaminski. "Liquid state machine and its separation ability as function of electrical parameters of cell". In: *Neurocomputing* 70.13-15 (2007), pp. 2593–2597. DOI: 10.1016/j.neucom.2006.12.015.

[79] Hélène Paugam-Moisy, Régis Martinez, and Samy Bengio. "Delay learning and polychronization for reservoir computing". In: *Neurocomputing* 71.7-9 (2008), pp. 1143–1158. DOI: 10.1016/j.neucom.2007.12.027.

[80] Eugene M. Izhikevich. "Polychronization: Computation with Spikes". In: *Neural Computation* 18.2 (2006), pp. 245–282. DOI: 10.1162/089976606775093882.

[81] D. Norton and D. Ventura. "Preparing More Effective Liquid State Machines Using Hebbian Learning". In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. IEEE, 2006. DOI: 10.1109/ijcnn.2006.246996.

[82] David Norton and Dan Ventura. "Improving the separability of a reservoir facilitates learning transfer". In: *2009 International Joint Conference on Neural Networks*. IEEE, 2009. DOI: 10.1109/ijcnn.2009.5178656.

[83] David Norton and Dan Ventura. "Improving liquid state machines through iterative refinement of the reservoir". In: *Neurocomputing* 73.16-18 (2010), pp. 2893–2904. DOI: 10.1016/j.neucom.2010.08.005.

[84] Stefan Schliebs, Maurizio Fiasché, and Nikola Kasabov. "Constructing Robust Liquid State Machines to Process Highly Variable Data Streams". In: *Artificial Neural Networks and Machine Learning – ICANN 2012*. Springer Berlin Heidelberg, 2012, pp. 604–611. DOI: 10.1007/978-3-642-33269-2_76.

[85] Han Ju et al. "Effects of synaptic connectivity on liquid state machine performance". In: *Neural Networks* 38 (2013), pp. 39–51. DOI: 10.1016/j.neunet.2012.11.003.

[86] Emmanouil Hourdakis and Panos Trahanias. "Use of the separation property to derive Liquid State Machines with enhanced classification performance". In: *Neurocomputing* 107 (2013), pp. 40–48. DOI: 10.1016/j.neucom.2012.07.032.

[87] Yan Zhou, Yaochu Jin, and Jinliang Ding. "Evolutionary Optimization of Liquid State Machines for Robust Learning". In: *Advances in Neural Networks – ISNN 2019*. Springer International Publishing, 2019, pp. 389–398. DOI: 10.1007/978-3-030-22796-8_41.

[88] Yan Zhou, Yaochu Jin, and Jinliang Ding. "Surrogate-Assisted Evolutionary Search of Spiking Neural Architectures in Liquid State Machines". In: *Neurocomputing* 406 (2020), pp. 12–23. DOI: 10.1016/j.neucom.2020.04.079.

[89] Menahem Friedman and Abraham Kandel. *Introduction to Pattern Recognition*. World Scientific, 1999. DOI: 10.1142/3641.

[90] T.L. Fine. "Fundamentals of Artificial Neural Networks [Book Reviews]". In: *IEEE Transactions on Information Theory* 42.4 (1996), p. 1322. DOI: 10.1109/tit.1996.508868.

[91] Kishan Mehrotra. *Elements of artificial neural networks*. Cambridge, Mass: MIT Press, 1997. ISBN: 9780262133289.

[92] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/bf02478259.

[93] W. Gerstner. "Chapter 12 A framework for spiking neuron models: The spike response model". In: *Neuro-Informatics and Neural Modelling*. Elsevier, 2001, pp. 469–516. DOI: 10.1016/s1383-8121(01)80015-4.

[94] Carlo R. Laing and Carson C. Chow. "A Spiking Neuron Model for Binocular Rivalry". In: *Journal of Computational Neuroscience* 12.1 (2002), pp. 39–53. DOI: 10.1023/a:1014942129705.

[95] Dazhi Wei and J.G. Harris. "Signal reconstruction from spiking neuron models". In: *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*. IEEE, 2004. DOI: 10.1109/iscas.2004.1329535.

[96] Louis Lapicque. "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation". In: *J. Physiol. Pathol. Gen.* 9 (1907), pp. 620–635.

[97] Aleister Cachón and Roberto A. Vázquez. "Tuning the parameters of an integrate and fire neuron via a genetic algorithm for solving pattern recognition problems". In: *Neurocomputing* 148 (2015), pp. 187–197. DOI: 10.1016/j.neucom.2012.11.059.

[98] Andrés Espinal et al. "Developing Architectures of Spiking Neural Networks by Using Grammatical Evolution Based on Evolutionary Strategy". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 71–80. DOI: 10.1007/978-3-319-07491-7_8.

[99] G. López-Vázquez et al. "Evolutionary Spiking Neural Networks for Solving Supervised Classification Problems". In: *Computational Intelligence and Neuroscience* 2019 (2019), pp. 1–13. DOI: 10.1155/2019/4182639.

[100] Joseph Chrol-Cannon and Yaochu Jin. "Learning structure of sensory inputs with synaptic plasticity leads to interference". In: *Frontiers in Computational Neuroscience* 9 (2015). DOI: 10.3389/fncom.2015.00103.

[101] Jürgen Branke et al., eds. *Multiobjective Optimization*. Springer Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-88908-3.

[102] Kalyanmoy Deb. "Introduction to Evolutionary Multiobjective Optimization". In: *Multiobjective Optimization*. Springer Berlin Heidelberg, 2008, pp. 59–96. DOI: 10.1007/978-3-540-88908-3_3.

[103] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. IEEE, 1995. DOI: 10.1109/icnn.1995.488968.

[104] El-Ghazali Talbi. *Metaheuristics : from design to implementation*. Hoboken, N.J: John Wiley & Sons, 2009. ISBN: 978-0470278581.

[105] Andres Espinal et al. "Comparison of PSO and DE for Training Neural Networks". In: *2011 10th Mexican International Conference on Artificial Intelligence*. IEEE, 2011. DOI: 10.1109/micai.2011.16.

[106] Marco Aurelio Sotelo-Figueroa et al. "Improving the Bin Packing Heuristic through Grammatical Evolution Based on Swarm Intelligence". In: *Mathematical Problems in Engineering* 2014 (2014), pp. 1–12. DOI: 10.1155/2014/545191.

[107] N. Srinivas and Kalyanmoy Deb. "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms". In: *Evolutionary Computation* 2.3 (1994), pp. 221–248. DOI: 10.1162/evco.1994.2.3.221.

[108] Kalyanmoy Deb. "Multi-Objective Optimization". In: *Search Methodologies*. Springer US, 2005, pp. 273–316. DOI: 10.1007/0-387-28356-0_10.

[109] Peng Hu et al. "Multiple Swarms Multi-Objective Particle Swarm Optimization Based on Decomposition". In: *Procedia Engineering* 15 (2011), pp. 3371–3375. DOI: 10.1016/j.proeng.2011.08.632.

[110] Andy Pryke, Sanaz Mostaghim, and Alireza Nazemi. "Heatmap Visualization of Population Based Multi Objective Algorithms". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 361–375. DOI: 10.1007/978-3-540-70928-2_29.

[111] David A. Van Veldhuizen and Gary B. Lamont. "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art". In: *Evolutionary Computation* 8.2 (2000), pp. 125–147. DOI: 10.1162/106365600568158.

[112] G. Lamont and D. V. Veldhuizen. "Multiobjective evolutionary algorithms: classifications, analyses, and new innovations". In: 1999.

[113] Margarita Reyes Sierra and Carlos A. Coello Coello. "Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and $\in$-Dominance". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 505–519. DOI: 10.1007/978-3-540-31880-4_35.

[114] C.A. Coello Coello and M.S. Lechuga. "MOPSO: a proposal for multiple objective particle swarm optimization". In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*. IEEE, 2002. DOI: 10.1109/cec.2002.1004388.

[115] Marco Laumanns et al. "Combining Convergence and Diversity in Evolutionary Multiobjective Optimization". In: *Evolutionary Computation* 10.3 (2002), pp. 263–282. DOI: 10.1162/106365602760234108.

[116] Qingfu Zhang, Wudong Liu, and Hui Li. "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances". In: *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009. DOI: 10.1109/cec.2009.4982949.

[117] Qingfu Zhang and Hui Li. "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition". In: *IEEE Transactions on Evolutionary Computation* 11.6 (2007), pp. 712–731. DOI: 10.1109/tevc.2007.892759.

[118] Juan J. Durillo, Antonio J. Nebro, and Enrique Alba. "The jMetal framework for multi-objective optimization: Design and architecture". In: *IEEE Congress on Evolutionary Computation*. IEEE, 2010. DOI: 10.1109/cec.2010.5586354.

[119] Juan J. Durillo and Antonio J. Nebro. "jMetal: A Java framework for multi-objective optimization". In: *Advances in Engineering Software* 42.10 (2011), pp. 760–771. DOI: 10.1016/j.advengsoft.2011.05.014.

[120] Stefan Schliebs, Nuttapod Nuntalid, and Nikola Kasabov. "Towards Spatio-Temporal Pattern Recognition Using Evolving Spiking Neural Networks". In: *Neural Information Processing. Theory and Algorithms*. Springer Berlin Heidelberg, 2010, pp. 163–170. DOI: 10.1007/978-3-642-17537-4_21.

[121] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[122] Antonio Benítez-Hidalgo et al. "jMetalPy: A Python framework for multi-objective optimization with metaheuristics". In: *Swarm and Evolutionary Computation* 51 (2019), p. 100598. ISSN: 2210-6502. DOI: 10.1016/j.swevo.2019.100598. URL: http://www.sciencedirect.com/science/article/pii/S2210650219301397.

[123] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences. 2017. URL: http://archive.ics.uci.edu/ml.

[124] Jorge A. Soria-Alcaraz, Marco A. Sotelo-Figueroa, and Andrés Espinal. "Statistical Comparative Between Selection Rules for Adaptive Operator Selection in Vehicle Routing and Multi-knapsack Problems". In: *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*. Springer International Publishing, 2018, pp. 389–400. DOI: 10.1007/978-3-319-71008-2_28.

[125] Kexin Jia and Zishu He. "DOA Identification of Communication Emitters Based on Shapiro-Wilk Test and Divisive Hierarchical Cluster Analysis". In: *2010 International Conference on Computational Intelligence and Software Engineering*. IEEE, 2010. DOI: 10.1109/wicom.2010.5601432.

[126] Joaquín Derrac et al. "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms". In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 3–18. DOI: 10.1016/j.swevo.2011.02.002.

[127] Milton Friedman. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance". In: *Journal of the American Statistical Association* 32.200 (1937), pp. 675–701. DOI: 10.1080/01621459.1937.10503522.

[128] Milton Friedman. "A Comparison of Alternative Tests of Significance for the Problem of *m* Rankings". In: *The Annals of Mathematical Statistics* 11.1 (1940), pp. 86–92. DOI: 10.1214/aoms/1177731944.

[129] Lucas Prestes et al. "Boosting the Performance of MOEA/D-DRA with a Multi-Objective Hyper-Heuristic Based on Irace and UCB Method for Heuristic Selection". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018. DOI: 10.1109/cec.2018.8477661.

Appendix A

Research Article for the *Workshop on Computational Intelligence 2019* (WCI 2019)

Single Spiking Neuron Multi-Objective Optimization for Pattern Classification

Submitted: 20th December 2019; accepted: 30th March 2020

Carlos Juarez-Santini, Manuel Ornelas-Rodriguez, Jorge Alberto Soria-Alcaraz, Alfonso Rojas-Domínguez, Hector J. Puga-Soberanes, Andrés Espinal, Horacio Rostro-Gonzalez

DOI: 10.14313/JAMRIS/1-2020/9

Abstract: *As neuron models become more plausible, fewer computing units may be required to solve some problems; such as static pattern classification. Herein, this problem is solved by using a single spiking neuron with rate coding scheme. The spiking neuron is trained by a variant of Multi-objective Particle Swarm Optimization algorithm known as OMOPSO. There were carried out two kind of experiments: the first one deals with neuron trained by maximizing the inter distance of mean firing rates among classes and minimizing standard deviation of the intra firing rate of each class; the second one deals with dimension reduction of input vector besides of neuron training. The results of two kind of experiments are statistically analyzed and compared again a Mono-objective optimization version which uses a fitness function as a weighted sum of objectives.*

Keywords: *Multi-objective Optimization, Spiking Neuron, Pattern Classification*

1. Introduction

Artificial Neural Networks (ANNs) try to simulate the behavior of the brain when they generate, process or transform information. An ANN is a system formed of simple processing units, which offers the property, and capability of input-output mapping. ANNs learn to solve complex problems in a reasonable amount of time [1]. The ability of learning of ANNs become a powerful tool for wide applications, for instance: pattern recognition works, classifying, clustering, vision tasks and forecasting [2].

ANNs can be distinguished in three generations according to their computational units [3]. The first one is based on McCulloch-Pitts neuron as computational units that can handle digital data [3]. The second one is characterized by a multilayer architecture, connectivity separating input, intermediate, and output units and applying activation functions with a continuous set of possible output values to a weighted sum of the inputs [4]. The third generation has been developed with the purpose of design neural models more plausible to the biological neurons. These are known as Spiking Neural Networks (SNNs) [5], [6].

ANNs are conformed by neurons organized in input, hidden and output layers, which are inter-connected by synaptic weights. These simulate the neuron synapsis of the human brain. During the training process of an ANN, a set of synaptic weights constantly is changing until the knowledge acquired is enough. Once the knowledge process has finished, it is necessary to evaluate the performance of the ANN. It is expected that the ANN can classify with acceptable accuracy the patterns from a particular problem during the testing phase [7]. The training process is an optimization task since it is desired to find the optimal weight set of the ANN. Methods based on gradient-descent have been applied to the training phase [8], but these techniques can be trapped at local minima. Then to overcome this situation, the researchers have proposed different global optimization methods [9] to optimize the ANNs by Evolutionary Algorithms (EAs). These EAs can be used to calibrate the connection weights, optimize the architecture and selecting the input features of ANNs [10].

The present research proposes a method for training full and partially connected SNNs based on the Leaky Integrate and Fire (LIF) model, by using a variant of Multi-objective Particle Swarm Optimization known as OMOPSO. This methodology is designed to solve pattern recognition problems. The results are statistically analyzed and compared with a version of mono-objective optimization using the Particle Swarm Optimization algorithm (PSO).

This paper is organized as follows: Section 2 presents the theoretical fundamentals used in this work. Section 3 explains the implemented methodology. Section 4 shows the results and statistical analysis. Finally, in section 5 are presented the conclusions and future work.

2. Background

This section describes the LIF model and the Optimized Multi-objective Particle Swarm Optimization (OMOPSO), which were used in this work.

2.1. Leaky Integrate and Fire Model

The LIF neuron model is one of the most used in the field of computational neuroscience given this model has an easier implementation and a lower computational cost in comparison with other spiking neuron models [11].

The mathematical representation for this model is shown in [11], [12] and it is given by the potential dynamic of the membrane:

$$\tau \frac{dv_i}{dt} = -g_{leak}\left(v_i - E_{leak}\right) + I(t) \tag{1}$$

where g_{leak} and E_{leak} are the conductance and the reversal potential of the leak current, τ is the membrane time constant and $I(t)$ is a current injected into the neuron.

In this work, it was used the representation proposed in [11],[13] defined as:

$$v' = I + a - bv,$$
$$\text{if } v \geq v_{threshold}\text{, then } v \leftarrow c \tag{2}$$

where I is the input current of the neuron, v denotes the membrane potential, a and b are parameters to configure the behavior of the neuron, c is the rest state voltage and $v_{threshold}$ is the threshold for the spike (firing) of the neuron. Besides, an initial condition v_0 is necessary to solve the differential equation by numerical methods.

Since the input patterns cannot be directly processed by the LIF neuron, they must be transformed to input currents by means of the equation:

$$I = \bar{x} \cdot \bar{w} \cdot \theta \tag{3}$$

where $\bar{x} \in \mathbb{R}^n$ is the input pattern vector, $\bar{w} \in \mathbb{R}^n$ is the set of synaptic weights and θ is a gain factor.

Fig. 1 shows the representation of a LIF neuron. When I is computed, it continues to solve the equation (2) to obtain the output spike train belonging to the input pattern.

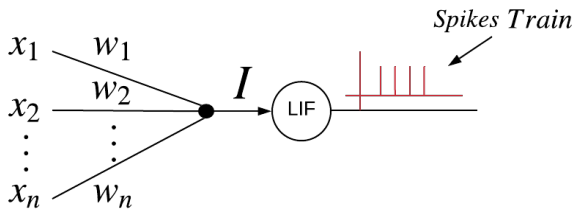

Fig. 1. Representation of a LIF neuron

2.2. Optimized Multi-Objective Particle Swarm Optimization (OMOPSO)

Regarding multi-objective optimization, a considerable number of algorithms can be found in the literature. For instance, the Multi-objective Particle Swarm (MOPSO) was proposed by Coello in [14]. In this work, we used the OMOPSO algorithm described in [15], which is based on Pareto dominance and an elitist selection through crowding factor. Beside this, the authors incorporated two mutation operators (uniform mutation and non-uniform mutation). The uniform mutation refers to variability range allowed for each decision variable, which is kept constant over generations and the non-uniform mutation has a characteristic variability range allowed for each decision variable, which decreases over time. Finally, it was added the ε-dominance concept which is the final size of the external file where stores the non-dominated solutions. Algorithm 1 shows the OMOPSO.

Algorithm 1. OMOPSO

Require: Initialize Swarm P_i, Initialize Leaders L_i

1: Send L_i to *ε-file*
2: *crowding*(L_i), g = 0
3: **while** $g < g_{max}$ **do**
4: **for** each particle
5: Select leader
6: Fly
7: Mutation
8: Evaluate
9: Update p_{best}
10: **end for**
11: Update L_i
12: Send L_i to *ε-file*
13: *crowding*(L_i), $g = g + 1$
14: **end while**
15: Report results in *ε-file*

3. Methodology

This section shows the methodology used in our work. There were proposed two kinds of experiments: the first one treats with neuron trained by maximizing the inter distance of mean firing rates among classes and minimizing the standard deviation of the intra firing rate of each class; the second one deals with dimension reduction of input vector besides of neuron training.

The LIF neuron model was implemented into jMetal [16], [17] where is available the OMOPSO algorithm, which was used for training the LIF neuron. Furthermore, the OMOPSO algorithm was configured as a mono-objective algorithm (PSO).

The design of the methodology is shown in Fig. 2. Initially, we set up the parameters of the OMOPSO algorithm and the LIF neuron model. Next, it is necessary to initialize the particles and Leaders (L_i) with uniformly random numbers to make a swarm. Each particle represents a synaptic weight vector ($\bar{w}$) with the same size as the feature input vector ($\bar{x}$). Then, whole particles

are evaluated into the LIF neuron model, by means of the objective functions. The non-dominated particles in the swarm will be L_i, which are sent to *ε-file*. Besides this, it is calculated a crowding factor for each L_i as a second discrimination criterion.

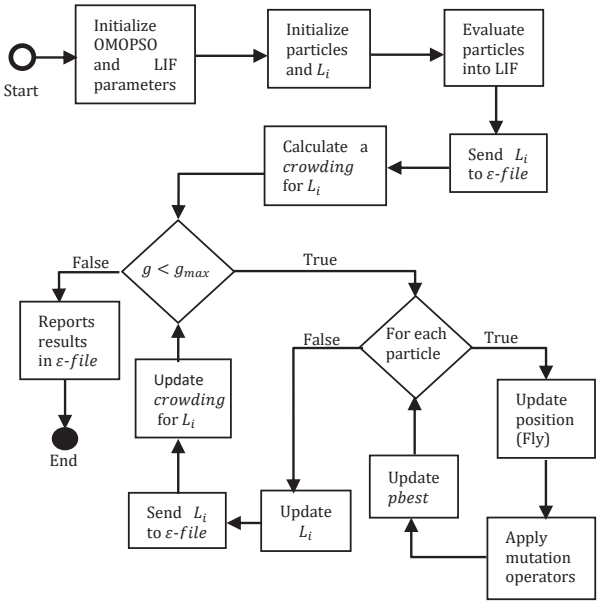

Fig. 2. Methodology schema

After it is initialized an Internal Loop into an External Loop, and each particle is modified into the Internal Loop, updating the position and applying the mutation operators. Then, each particle is evaluated and updated its personal best value (p_{best}). A new particle replaces the p_{best} if such value is dominated by the new particle or if both are non-dominated concerning each other.

When all particles have been updated, the L_i are modified in the External Loop. Only the particles that overcome their p_{best} will try to enter to L_i set. Once the L_i have been updated, they are sent to *ε-file*. Finally, the crowding values of the set of L_i is updated and we eliminate as many leaders as necessary to avoid overflow of the size of the L_i set. The process is repeated until finalizing all iterations.

3.1. Objective Functions

Three different objective functions were considered to measure the performance of the solutions (particles):

A. The Euclidean distance between the combination of AFR_i and AFR_j, where *AFR* is the average firing rate of each class and $i \neq j$. For this objective function, we looking for maximize the separability between the classes:

$$MAXdist\left(AFR_i, AFR_j\right) \tag{4}$$

B. The Standard Deviation of the firing rate for each pattern class $SDFR_k$, where $k = 1, ..., K$ and K is the total of pattern classes. In this objective function, we looking for minimize the dispersion of each pattern class:

$$MIN\left(SDFR_k\right) \tag{5}$$

C. The dimension of the input feature vector ($\bar{x}$). To avoid redundancies in information, we desire to reduce the dimensionality of the feature vectors, by minimizing the total of 1's of a binary mask a binary mask ($\bar{r}$) with the same size of the input feature vector.

In our proposal, the number of objective functions is related to the number of classes of the dataset.

3.2. Experiments

Four supervised classification datasets from the UCI Machine Learning Repository [18] were employed for experimentation: Iris Plant, Wine, Glass, and SPECT. Table 1 shows the details of the datasets used.

Each dataset was randomly divided in two subsets with approximately the same size. The first one was employed as training set and the second one as testing set.

Tab. 1. Datasets employed for experimentation

Dataset	Instances	Classes	Features
Iris Plant	150	3	4
Wine	178	3	13
Glass	214	6	9
SPECT	267	2	22

With the aim to observe the performance of our proposal, four experiments were configurated according to the objective functions seen in section 3.1. The characteristics of each experiment are defined below and summarized in Table 2.

i. Experiment #1 was defined as a multi-objective problem, focusing on the A and B objective functions. The OMOPSO algorithm was used to optimize the synaptic weight vector of the LIF neuron.
ii. Experiment #2 employs the multi-objective approach, considering the A, B and C objective functions. The OMOPSO algorithm was taken to optimize the synaptic weight vector and the dimension of the input vector. Concerning the optimization of the last parameter, a binary mask ($\bar{r}$) was used in equation (3) to calculate a modified input current given by equation (6).

$$I = \bar{x} \cdot \bar{w} \cdot \bar{r} \cdot \theta \tag{6}$$

iii. Experiment #3 was designed as a mono-objective problem. The objective function (eq. 7) was formed by the weighted sum of two objective functions. The first one is the inverse of the summation of the Euclidean distances among all combinations of AFR_i and AFR_j and the second objective is the sum of the standard deviation of the firing rate for all classes as shown in equation 7 [11]. PSO algorithm was used to design the synaptic weight vectors.

$$MIN(f) = \frac{1}{dist(\boldsymbol{AFR})} + \sum_{k=1}^{K} SDFR_k \quad (7)$$

iv. Experiment #4 is a mono-objective approach that seeks to optimize the synaptic weight vector and the dimension of the input vector with the PSO algorithm. The objective function (eq. 8) is formed by the weighted sum of the equation (7) and the rate of *T* and *D*, where *T* is total of 1's in the binary mask $(\bar{r})$ and *D* is the dimension of the input feature vector.

$$MIN(f) = \frac{1}{dist(AFR)} + \sum_{k=1}^{K} SDFR_k + \frac{\boldsymbol{T}}{\boldsymbol{D}} \quad (8)$$

Tab. 2. Configuration for experimentation

	Algorithm	Optimized Parameters	Objective Functions
Exp #1	OMOPSO	synaptic weight vector	A, B
Exp #2	OMOPSO	synaptic weight vector and dimension of input vectors	A, B, C
Exp #3	PSO	synaptic weight vector	A, B
Exp #4	PSO	synaptic weight vector and dimension of input vectors	A, B, C

Table 3 shows a compendium of the number of objective functions by experiment for each dataset.

Tab. 3. Total of Objective Functions by experiment

		Objective Functions in			
Dataset	Classes	Exp #1	Exp #2	Exp #3	Exp #4
Iris Plant	3	6	7	1	1
Wine	3	6	7	1	1
Glass	6	21	22	1	1
SPECT	2	3	4	1	1

Each experiment consisted of 40 independently executions per each dataset to guarantee statistical significance. The parameter values used in the OMOPSO algorithm and the LIF neuron model [11] are detailed in Table 4 and 5 respectively.

The initial synaptic weights were generated randomly $\theta \in [0,1]$.

4. Results and Statistical Analysis

This section describes the results obtained from the experimentation proposed in section 3. The results are statistically analyzed and discussed below.

Tab. 4. Configuration OMOPSO Parameters

Max particle size:	100
Max iterations:	1000
ε-file size:	100
Uniform Mutation	
Mutation probability:	$\frac{1.0}{Number\ of\ problem\ variables}$
Perturbation index:	0.5
Non-uniform Mutation	
Mutation probability:	$\frac{1.0}{Number\ of\ problem\ variables}$
Perturbation index:	0.5
Max iterations:	1000

Tab. 5. Configuration LIF Parameters

a	0.5
b	-0.001
c	-50 mV
v_i	-60 mV
$v_{threshold}$	50 mV
Time	1000 ms
h	1
θ	0.1

For each execution, at the end of the training phase, the total of particles is evaluated in the LIF neuron model using the training set, and the classification accuracy is calculated for each particle. Finally, the particle with the best performance is used in the testing phase for obtaining the accuracy in the testing set.

Tab. 6. Accuracy of training phase over each experiment

	OMOPSO		PSO	
	Experiments		Experiments	
Dataset	#1	#2	#3	#4
Iris Plant	**0.9817 ± 0.0131**	0.9793 ± 0.0157	0.9 ± 0.0232	0.8987 ± 0.0219
Wine	0.7858 ± 0.0358	**0.8048 ± 0.0336**	0.6849 ± 0.0432	0.6986 ± 0.0301
Glass	**0.5050 ± 0.0441**	0.5031 ± 0.0405	0.39 ± 0.0030	0.3638 ± 0.0726
SPECT	**0.8592 ± 0.0243**	0.8286 ± 0.0227	0.7276 ± 0.0325	0.7273 ± 0.0344

Tables 6 and 7 show the results obtained from the methodology proposed. The accuracy values along with the standard deviations grade the performance of the experiments. The accuracy of the training phase corresponds to the average of the performance of the

best particles obtained in each experiment, whereas that the accuracy of the testing phase is obtained from the average of the performance of these particles applied to the testing set. The highest accuracy values are remarked in bold font.

Tab. 7. Accuracy of testing phase over each experiment

	OMOPSO		**PSO**	
	Experiments		**Experiments**	
Dataset	#1	#2	#3	#4
Iris Plant	0.94 ± 0.0383	**0.9543 ± 0.0220**	0.9003 ± 0.0355	0.8883 ± 0.0303
Wine	0.7322 ± 0.0604	**0.7397 ± 0.0610**	0.6706 ± 0.0505	0.6858 ± 0.0451
Glass	0.3516 ± 0.1141	**0.3695 ± 0.1163**	0.3472 ± 0.0947	0.3594 ± 0.0960
SPECT	0.7043 ± 0.1051	0.7019 ± 0.1006	**0.7157 ± 0.0545**	0.7073 ± 0.0523

Tab. 8. Analysis of reduction of features of input vector

	Experiments			
	#2		**#4**	
Dataset	Average number of features employed	Rate of features used	Average number of features employed	Rate of features used
Iris Plant	2.575 ± 0.747	0.640	3.050 ± 0.221	0.760
Wine	8.475 ± 3.266	0.652	5.850 ± 1.902	0.450
Glass	5.550 ± 1.999	0.617	6.00 ± 1.377	0.667
SPECT	10.325 ± 5.677	0.469	21.675 ± 0.526	0.985

Table 8 shows the average amount of input features employed by the LIF neuron model and its corresponding rate concerning the total size of the original input feature vector.

Several statistic tests were applied to the obtained results. Firstly, Shapiro-Wilk test was executed to identify the kind of parametric or non-parametric tests to be used along with our data. Our tests were implemented using R programming language, and the CONTROLTEST package tool (available at http://sci2s.ugr.es/sicidm/) was used for non-parametric comparison between experiments. Specifically, three non-parametric tests were applied: Friedman, Friedman Aligned Ranks, and Quade.

Firstly, the results from statistic tests for the Training phase are shown and discussed. Subsequently, the results of statistic tests computed in the Testing phase are analyzed.

In the Shapiro-Wilk test, the null-hypothesis (H_0) states the samples come from a normal distribution. In Table 9, for a significance level of α = 0.05, the *P-values* obtained show that approximately half of the results do not reject H_0, but the rest of the results reject H_0. Therefore, non-parametric statistics were applied since such tests include both cases.

Tab. 9. Shapiro-Wilk test in Training phase

	OMOPSO		**PSO**	
	Experiments		**Experiments**	
Dataset	#1	#2	#3	#4
Iris Plant	0.0002868	0.000761	0.1257	0.189
	H_0 is rejected	H_0 is rejected	H_0 is not rejected	H_0 is not rejected
Wine	0.6275	0.0407	0.08713	0.3317
	H_0 is not rejected	H_0 is rejected	H_0 is not rejected	H_0 is not rejected
Glass	0.0002413	0.001126	6.64E-14	1.09E-11
	H_0 is rejected	H_0 is rejected	H_0 is rejected	H_0 is rejected
SPECT	0.06032	0.07665	0.3015	0.09427
	H_0 is not rejected	H_0 is not rejected	H_0 is not rejected	H_0 is not rejected

Friedman, Friedman Aligned Ranks, and Quade tests were applied to the obtained results. In these tests, the null-hypothesis (H_0) states that the data of the experiments follow the same distribution [19] (there is no difference in their performance).

Table 10 reports the average ranks obtained from these tests on the whole experiments. The smaller values, in bold font, indicate that Experiment #1 had consistently the best performance.

Tab. 10. Average rankings of the experiments for the Training phase

Experiment	**Friedman**	**Friedman Aligned Ranks**	**Quade**
#1	**1.25**	**4.0**	**1.2**
#2	1.75	5.0	1.80
#3	3.25	12.25	3.199
#4	3.75	12.75	3.8

Table 11 shows the *P-value* for each statistical test and the sentence corresponding to the status of H_0 for a significance level α = 0.05. If the *P-value* is greater than α then indicates that not exist evidence to reject H_0. Therefore, the tests Friedman and Quade rejected H_0. However, these results do not give enough information

to select the best experiment, so that, a post-hoc procedure was necessary to do. From Table 10, Experiment #1 was taken as the control experiment.

Tab. 11. Contrast the null-hypothesis in Training phase

	Friedman	**Friedman Aligned Ranks**	**Quade**
P-values	0.01694	0.3806	1.04E-04
	H_0 is rejected	H_0 is not rejected	H_0 is rejected

Table 12 shows the results of the post-hoc procedure, where the *P-values* were adjusted by Holm's correction. For α = 0.05, the adjusted *P-values* for the comparison between the control experiment and the Experiments #3 and #4 show that the Experiment #1 had better performance.

Then, it is presented the results for the Testing phase.

Table 13 shows the results of the Shapiro-Wilk test where the *P-values* were contrasted with a significance level of α = 0.05. The *P-values* obtained show that five results reject H_0, and eleven results do not reject H_0. Next, non-parametric statistic tests were applied.

Tab. 12. Adjusted P-values for Training phase

	Friedman	**Quade**
Experiment	Holm	Holm
#1 vs #4	0.01667	0.01667
#1 vs #3	0.025	0.025
#1 vs #2	0.05	0.05

Tab. 13. Shapiro-Wilk test in Testing phase

	OMOPSO		**PSO**	
	Experiments		**Experiments**	
Dataset	#1	#2	#3	#4
Iris Plant	0.05354	0.02317	0.4062	0.2015
	H_0 is not rejected	H_0 is rejected	H_0 is not rejected	H_0 is not rejected
Wine	0.4185	0.4515	0.4542	0.2932
	H_0 is not rejected	H_0 is not rejected	H_0 is not rejected	H_0 is not rejected
Glass	0.06616	0.00249	4.17E-08	4.71E-07
	H_0 is not rejected	H_0 is rejected	H_0 is rejected	H_0 is rejected
SPECT	0.002891	0.08466	0.7195	0.07491
	H_0 is rejected	H_0 is not rejected	H_0 is not rejected	H_0 is not rejected

Table 14 shows the average ranks obtained from Friedman, Friedman Aligned Ranks and Quade tests for whole results. In the three tests, the smaller average ranks, in bold font, specify that Experiment #2 had the best performance.

Tab. 14. Average rankings of the experiments for the Testing phase

Experiment	**Friedman**	**Friedman Aligned Ranks**	**Quade**
#1	2.5	6.5	2.3
#2	**1.75**	**4.75**	**1.2999**
#3	3.0	11.75	3.4
#4	2.75	11.0	3.0

Tab. 15. Contrast the null-hypothesis in Testing phase

	Friedman	**Friedman Aligned Ranks**	**Quade**
P-values	0.5519	0.3632	0.0381
	H_0 is not rejected	H_0 is not rejected	H_0 is rejected

Table 15 shows the *P-value* for each statistical test. The significance level was set up α = 0.05. Quade test rejects H_0. Nonetheless, this result does not present enough information to choose of the best experiment. So, a post-hoc procedure was made. From Table 14, Experiment #2 was used as the control experiment.

Tab. 16. Adjusted P-values for Testing phase

	Quade
Experiment	Holm
#2 vs #4	0.01667
#2 vs #3	0.025
#2 vs #1	0.05

Table 16 shows the results of the post-hoc procedure for Quade test, where *P-values* were adjusted by Holm's correction. The *P-values* were compared against a significance level of α = 0.05. The *P-values* for the comparison between the control experiment and the Experiment #3 and #4 show that the Experiment #2 had better performance.

5. Conclusion

This paper presents a methodology for training full and partially connected LIF spiking neurons using the OMOPSO algorithm for solving pattern recognition problems. The experiments were designed with a multi-objective approach and their results were compared statistically with the results of mono-objective experiments. Each experiment was tested on four well-known benchmark datasets by performing 40 independently executions for each dataset.

The results have shown that the Experiments #1 and #2 had the best performances in the Training and Testing phases respectively. Therefore, the multi-ob-

jective approach provides an adequate alternative to optimize LIF spiking neurons.

One interesting characteristic of our methodology consists on the reduction of dimensionality of the input feature vectors to avoid redundancies in the input information.

As future work, we propose to include the LIF parameters into the OMOPSO algorithm to explore better non-dominated solutions and to implement more multi-objective algorithms from state of the art for training LIF spiking neurons.

ACKNOWLEDGEMENTS

The authors express their gratitude to the National Technology of Mexico and the University of Guanajuato. C. Juarez-Santini and A. Rojas-Dominguez thank the National Council of Science and Technology of Mexico (CONACYT) for the support provided by means of the Scholarship for Postgraduate Studies and research grant CATEDRAS-2598, respectively.

AUTHORS

Carlos Juarez-Santini – Postgraduate Studies and Research Division, Leon Institute of Technology – National Technology of Mexico, Leon, Guanajuato, Mexico
e-mail: jusca_94@hotmail.com.

Manuel Ornelas-Rodriguez* – Postgraduate Studies and Research Division, Leon Institute of Technology – National Technology of Mexico, Leon, Guanajuato, Mexico
e-mail: mornelas67@yahoo.com.mx.

Jorge Alberto Soria-Alcaraz – Department of Organizational Studies, DCEA-University of Guanajuato, Guanajuato, Mexico, e-mail: jorge.soria@ugto.mx.

Alfonso Rojas-Domínguez – Postgraduate Studies and Research Division, Leon Institute of Technology – National Technology of Mexico, Leon, Guanajuato, Mexico
e-mail: alfonso.rojas@gmail.com.

Hector J. Puga-Soberanes – Postgraduate Studies and Research Division, Leon Institute of Technology – National Technology of Mexico, Leon, Guanajuato, Mexico
e-mail: pugahector@yahoo.com.mx.

Andrés Espinal – Department of Organizational Studies, DCEA-University of Guanajuato, Guanajuato, Mexico, e-mail: aespinal@ugto.mx.

Horacio Rostro-Gonzalez – Department of Electronics, DICIS-University of Guanajuato, Salamanca, Guanajuato, Mexico, e-mail: hrostrog@ugto.mx.

* Corresponding author

REFERENCES

[1] M. van Gerven and S. Bohte, "Editorial: Artificial Neural Networks as Models of Neural Information Processing", *Frontiers in Computational Neuroscience*, vol. 11, 2017, 1–2, DOI: 10.3389/fncom.2017.00114.

[2] K. Soltanian, F. A. Tab, F. A. Zar and I. Tsoulos, "Artificial neural networks generation using grammatical evolution". In: *2013 21st Iranian Conference on Electrical Engineering (ICEE)*, 2013, 1–5, DOI: 10.1109/IranianCEE.2013.6599788.

[3] W. Maass, "Networks of spiking neurons: The third generation of neural network models", *Neural Networks*, vol. 10, no. 9, 1997, 1659–1671, DOI: 10.1016/S0893-6080(97)00011-7.

[4] D. Gardner, *The Neurobiology of neural networks*, MIT Press, 1993.

[5] N. G. Pavlidis, O. K. Tasoulis, V. P. Plagianakos, G. Nikiforidis and M. N. Vrahatis, "Spiking neural network training using evolutionary algorithms". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks,* vol. 4, 2005, 2190–2194, DOI: 10.1109/IJCNN.2005.1556240.

[6] A. A. Hopgood, *Intelligent Systems for Engineers and Scientists*, CRC Press/Taylor & Francis Group, 2012.

[7] B. A. Garro and R. A. Vázquez, "Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms", *Computational Intelligence and Neuroscience*, 2015, 1–20, DOI: 10.1155/2015/369298.

[8] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation". In: *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*, MIT Press, 1986, 318–362.

[9] D. Karaboga, B. Akay and C. Ozturk, "Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks". In: V. Torra, Y. Narukawa and Y. Yoshida (eds.), *Modeling Decisions for Artificial Intelligence*, vol. 4617, 2007, 318–329, DOI: 10.1007/978-3-540-73729-2_30.

[10] S. Ding, H. Li, C. Su, J. Yu and F. Jin, "Evolutionary artificial neural networks: a review", *Artificial Intelligence Review*, vol. 39, no. 3, 2013, 251–260, DOI: 10.1007/s10462-011-9270-6.

[11] R. A. Vazquez and A. Cachon, "Integrate and Fire neurons and their application in pattern recognition". In: *2010 7th International Conference on Electrical Engineering Computing Science and Automatic Control*, 2010, 424–428, DOI: 10.1109/ICEEE.2010.5608622.

[12] A. Cachón and R. A. Vázquez, "Tuning the parameters of an integrate and fire neuron via a genetic algorithm for solving pattern recognition problems", *Neurocomputing*, vol. 148, 2015, 187–197, DOI: 10.1016/j.neucom.2012.11.059.

[13] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons?", *IEEE Transactions on Neural Networks*, vol. 15, no. 5, 2004, 1063–1070, DOI: 10.1109/TNN.2004.832719.

[14] C. A. Coello Coello and M. S. Lechuga, "MOPSO: a proposal for multiple objective particle swarm optimization". In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, vol. 2, 2002, 1051–1056, DOI: 10.1109/CEC.2002.1004388.

[15] M. R. Sierra and C. A. Coello Coello, "Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ∈-Dominance". In: C. A. Coello Coello, A. Hernández Aguirre and E. Zitzler (eds.), *Evolutionary Multi-Criterion Optimization*, vol. 3410, 2005, 505–519, DOI: 10.1007/978-3-540-31880-4_35.

[16] J. J. Durillo, A. J. Nebro and E. Alba, "The jMetal framework for multi-objective optimization: Design and architecture". In: *IEEE Congress on Evolutionary Computation*, 2010, 1–8, DOI: 10.1109/CEC.2010.5586354.

[17] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization", *Advances in Engineering Software*, vol. 42, no. 10, 2011, 760–771, DOI: 10.1016/j.advengsoft.2011.05.014.

[18] "UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science". D. Dua and C. Graff, http://archive.ics.uci.edu/ml. Accessed on: 2020-05-28.

[19] J. Derrac, S. García, D. Molina and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", *Swarm and Evolutionary Computation*, vol. 1, no. 1, 2011, 3–18, DOI: 10.1016/j.swevo.2011.02.002.