جامعة سيدي محمد بن عبد الله بفاس
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵓⵃⵎⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ ⴱ ⴼⴰⵙ
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES

كلية العلوم والتقنيات فاس
ⵜⴰⵖⵡⵍⵉⵜ ⵏ ⵜⵣⵔⴼⵜⵉⵏ ⴷ ⵜⵙⵙⵏⵜⵉⵏ ⴼⴰⵙ
FACULTÉ DES SCIENCES ET TECHNIQUES DE FÈS

Centre d'Etudes Doctorales : Sciences et Techniques de l'Ingénieur

N° d'ordre 07 /2021

THESE DE DOCTORAT

Présentée par

Mme : Kaoutar SENHAJI

Spécialité : Recherche Opérationnelle et Informatique

Sujet de la thèse : Contributions à la résolution des problèmes d'optimisation multiobjectif : Application à l'apprentissage automatique et aux problèmes socio-économique

Thèse présentée et soutenue le 12 janvier 2021 devant le jury composé de :

Nom Prénom	Titre	Etablissement	
Abdelmajid HILALI	PES	Faculté des Sciences et Techniques de Fès	Président
Abdellatif EL AFIA	PES	Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes de Rabat	Rapporteur
Mohammed SABBANE	PES	Faculté des Sciences de Meknès	Rapporteur
Khalid HADDOUCH	PH	Ecole Nationale des Sciences Appliquées de Fès	Rapporteur
Youssef AKDIM	PES	Faculté des Sciences Dhar El Mehraz de Fès	Examinateur
Rachid EL KHAOULANI EL IDRISSI	PH	Faculté des Sciences et Techniques de Fès	Examinateur
Mohamed ETTAOUIL	PES	Faculté des Sciences et Techniques de Fès	Directeur de thèse

Noureddine JOUDAR	ENSAM Rabat	Invité

Laboratoire d'accueil : Modélisation et Structure Mathématique

Etablissement : Faculté des Sciences et Techniques de Fès

Sidi Mohamed Ben Abdellah University - USMBA

Contribution for solving the multiobjective optimization problems: Applications to the machine learning and socio-economic problems

by
Kaoutar SENHAJI

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Sciences and Techniques FST
Modeling and Mathematical Structures Laboratory (MSM)

January 2021

Declaration of Authorship

I, Kaoutar SENHAJI , declare that this thesis titled , 'Contribution for solving the multiobjective optimization problems : Applications to the machine learning and socio-economic problems' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"It's one small step for man, one giant leap for mankind."

Neil Armstrong

Contents

Abstract **5**
Acknowledgement **6**
Introduction **10**
Chapter 1: Résumé des travaux de recherche **14**
1.1 Introduction 14
1.2 Optimisation multiobjectif 17
1.2.1 Généralités 17
1.2.2 Approches de résolution multiobjectif 19
1.2.3 La méthode "Non-dominated Sorting Genetic Algorithm II" (Non-dominated Sorting Genetic Algorithm (NSGA II)) . . . 21
1.3 Réseaux de neurone artificiel: architectures et apprentissage 24
1.3.1 Neurone formel 25
1.3.2 Fonction d'activation 26
1.3.3 Réseau à propagation avant : Perceptron multicouche 26
1.3.4 Réseaux récurent "feed-back": Réseaux de Hopfield continu . . 28
1.3.5 Réseaux de neurones artificiels et apprentissage 31
1.4 Approches neuro-metaheuristiques pour le problème de selection du portfeuille 36
1.4.1 Probléme de gestion du portfeuille 36
1.4.2 Modèles rationnels proposés 41
1.4.3 Résolution des problèmes à variables mixtes 41
1.4.4 Réseau Hopfield continu pour les modèles rationnels 43
1.4.5 Algorithme génétique hybride pour le modèle de variance moyenne rationnelle 45
1.4.6 Adaptation de l'algorithme génétique 46
1.4.7 Hybride multiobjectif algorithme génétique pour le modèle de semi-variance rationnelle moyenne 47
1.4.8 Résultats expérimentaux 48
1.5 Multiobjectif approche pour l'apprentissage du percetron multicouche 50
1.5.1 Adaptation de l'algorithme NSGA II 57
1.5.2 Résultats expérimentaux 59
1.6 Conclusion 61
Chapter 2: Multiobjective optimization **62**
2.1 Introduction 62
2.2 Generality and denfinitions 62

2.2.1 Definitions . . . 62
2.2.2 Dominance . . . 63
2.2.3 Pareto front . . . 64
2.3 Resolution methods of multiobjective optimization problem . . . 67
2.3.1 Scalarization methods . . . 67
2.3.2 Metaheuristics approachs: Evolutionary algorithms . . . 72
2.4 Conclusion . . . 78
Chapter 3: Artificial Neural Network: Learning and Optimization 80
3.1 Introduction . . . 80
3.2 Artificial Neural Networks History . . . 80
3.3 Multilayer Perceptron . . . 81
3.3.1 MLPNN Model . . . 82
3.3.2 MLPNN Training . . . 85
3.4 Continous Hopfield Network . . . 91
3.4.1 Continuous Hopfield Network Architecture . . . 91
3.4.2 Hopfield Energy and the Equilibrium State . . . 93
3.5 Conclusion . . . 94
Chapter 4: Neural-Metaheuristics approach for portfolio selection problem 96
4.1 Introduction . . . 96
4.2 Portfolio selection problem Models . . . 96
4.2.1 Introduction . . . 97
4.2.2 Mean-variance Model . . . 100
4.2.3 Efficient Frontier . . . 101
4.3 Proposed Rational Models . . . 102
4.4 Continuous Hopfield Network for Rational Models . . . 102
4.4.1 Energy function . . . 102
4.4.2 Activation function . . . 103
4.4.3 Parameter setting . . . 105
4.5 Hybrid Genetic Algorithm for Rational Mean-variance Model . . . 105
4.5.1 Algorithm . . . 106
4.5.2 Genetic Algorithm Adaptation . . . 107
4.6 Hybrid Multiobjective Genetic Algorithm (GA) for Rational Mean-semivariance Model . . . 110
4.7 Experimental Results . . . 111
4.7.1 Rational Mean-Variance Model Results . . . 111
4.7.2 Rational Mean-Semivariance Model Results . . . 116
4.8 Conclusion . . . 121
Chapter 5: Multiobjective learning approaches for multilayer perceptron 124
5.1 Introduction . . . 124
5.2 Multiobjective Training Model for MLPNN . . . 125
5.2.1 Notation . . . 125
5.2.2 Training Objective . . . 126
5.2.3 Modified Laplace regularizer . . . 126
5.2.4 Non-Smooth $L_{1/2}$ regularizer . . . 128

5.3 Non-dominated Sorting Genetic Algorithm for Multiobjective Training Models . . . 130
5.3.1 NSGA II General Used Algorithm . . . 131
5.3.2 General NSGA II Adaptation . . . 131
5.3.3 Modified Laplace regularizer model adaptation . . . 133
5.3.4 Non-Smooth $L_{1/2}$ regularizer model Adaptation . . . 134
5.3.5 Models identification . . . 135
5.4 Experiment Results . . . 136
5.4.1 Description of used Data set . . . 136
5.4.2 Network parameter . . . 137
5.4.3 NSGA II parameter . . . 138
5.4.4 Performance measurement . . . 138
5.4.5 Modified Laplace Regularizer Numerical Results . . . 139
5.4.6 Non-Smooth $L_{1/2}$ Regularizer Numerical Results . . . 140
5.5 Conclusion . . . 143
General conclussion and perspectives **145**

Abstract

In an optimization context, there are different real-world problems that face many difficulties. Indeed, decision-makers are sometimes targeting several objectives at the same time, increasing the research space as well as the complexity of the problem. Multi-objective optimization provides an answer to similar problems in order to balance the different objectives. Considering that objectives are often contradictory, the resolution via classical methods is very complicated or even impossible, knowing the difficulties that this type of modelling generates. This work aims to model and resolve several multi-objective models, in particular, the portfolio management problem. This problem presents an interesting example of objective conflict, as we are trying to obtain maximum profit with minimum risk. In a first step, we solved the proposed model by a neural approach using the continuous Hopfield network. The approach has clearly produced interesting results in terms of risk minimization, as it is classified as a scalarization approach. Subsequently, we suggest to improve the Continuous Hopfield Network (CHN) responses by a multiobjective approach using the genetic algorithm. The second part of this thesis is directed towards the field of artificial intelligence, we have treated, precisely, the generalization of the multilayer perceptron. In this context, we proposed to balance two main objectives: the accuracy of the perceptron and its complexity, since the latter objective has a great influence on the generalization of the neural network. Thus, we proposed two different multi-objective models including different constraints. We have also solved the models using a meta-heuristic approach called Non-dominate Genetic Algorithm II (NSGA II). The obtained results showed high efficiency in terms of accuracy and network complexity.

Key-words: Artificiel neural network, multi-layer perceptron; multiobjective learning, optimization, portofolio selection problem, multiojbective genetic algorithm.

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisors Pr. **Mohamed ETTAOUIL** and Pr. **Mohamed MASRAR** for the continuous support of my PhD study and related research, for Their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study.

Besides my advisor, I would like to thank the rest of my thesis committee: Pr. **Abdellatif EL AFIA**, Pr. **Mohammed SABBANE**, and Pr. **Khalid HADDOUCH**, for their reviews and hard question which incented me to widen my research from various perspectives, as well as Pr. **Abdelmajid HILALI** for presiding the doctoral committee. Further, a special thanks to the professors: **Youssef AKDIM**, **Rachid EL KHAOULANI IDRISSI** and **Noureddine JOUDAR** for their insightful comments and encouragement.

I thank my fellow labmates for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last years. In particular, Dr. **Hassan RAMCHOUN**.

Getting through my dissertation, I learned that PhD studies required more than academic support, and I have many, many people to thank for listening to and, at times, having to tolerate me over the past years. I cannot begin to express my gratitude and appreciation for their friendship. **Selma Makboul** and **Yassir EL KARKRI** have been unwavering in their personal and professional support during my PhD journey, especially during the time confinement of Covid-19.

Most importantly, none of this could have happened without my family. I would like to dedicate a special thanks to my parents and sister for supporting me spiritually throughout writing this thesis and my life in general. it would be an understatement to say that, as a family, we have experienced some ups and downs in the past three years. Every time I was ready to quit, you did not let me and I am forever grateful. This dissertation stands as a testament to your unconditional love and encouragement.

List of Figures

1.1 Exemple de front de Pareto 19
1.2 L'optimalité locale au sens de Pareto [21] 19
1.3 Fonctionnement de l'algorithme NSGA II [26] 22
1.4 Mesure d'entassement (crowding) [26] 23
1.5 Modèle d'un neurone formel [11] 25
1.6 Réseau de Hopfield continu à n neurones [11] 29

2.1 Pareto front representation [21] 65
2.2 The most common Pareto front forms for two objectives [21]. 66
2.3 The weighted Objective Functions method Presentation [21] 69
2.4 The Corney method Presentation [21] 71
2.5 Schematic of VEG Algorithm [19] 73
2.6 Efficacity computing [21] 75
2.7 The Crowding Distance Computing [26] 77

3.1 Neural Network History Timeline 81
3.2 Multiple layer perceptron neural network model 82
3.3 Activation function rule [1] 83
3.4 Step Functions 84
3.5 Linear Activation Function 84
3.6 Diferent Non-Linear Activation Functions 86
3.7 Electronic representation of Hopfield neural network [11] 92

4.1 Portfolio Efficient Frontier [89] 101
4.2 The slope activation function for slope values (0.5, 0.4, 0.3, 0.24, 0.1 and 0.08). 104
4.3 Hybrid GA flowchart 106
4.4 Individual Coding 107
4.5 Crossover Operator 108
4.6 Mutation Operator 108
4.7 Rational Mean-variance Model Efficient Frontier 112
4.8 Hybrid GA Obtained Results 114
4.9 hybrid GA Efficient Frontier 116
4.10 CHN and hybrid GA performance comparison 116
4.11 (Estrada Mean-Semivariance, CHN) Efficient frontier 117

4.12 (Estrada mean-Semivariance CHN) and (mean-variance, CHN) Comparing . . . 118
4.13 Obtained Efficient Front of Hybrid Multiobjctive Genetic Algorithm . 119
4.14 Graphic comparing of the proposed approach and multiobjctive genetic algorithm . . . 121

5.1 Multilayer Perceptron Neural Network (MLPNN) structure coding using the connections matrix (C_{ij}^k), weight matrix (W_{ij}^k) and neurons decision vector (U_i). . . . 132
5.2 The Coresponding Individual Coding . . . 133
5.3 Corresponding MLPNN network to an individual coding. . . . 134
5.4 The binary crossover process. . . . 135
5.5 The mutation process for binary variables. . . . 136
5.6 The flowchart of the methodology. . . . 136
5.7 MLPNN architecture before and after training. . . . 138
5.8 The obtained Pareto front. . . . 141
5.9 The waves performance measure. . . . 141
5.10 Obtained neural network for Iris Data. . . . 142

List of Tables

4.1 CHN Optimal Portfolios 112
4.2 Comparison mean return result between CHN and GA for a fixed mean return value 113
4.3 Comparison mean return result between CHN and GA for a fixed risk level 113
4.4 Hybrid GA parameter setting 114
4.5 Exemple of Obtained Hybrid GA Results 115
4.6 Example of Obtained Efficient Portfolio 117
4.7 Multiobjective Genetic Algorithm Parameters adjustment 119
4.8 Example of an Optimized Solution 120
4.9 Table Comparing the proposed Approach and CHN in term of Mean-return criteria 122
4.10 Table Comparing the proposed Approach and CHN in term of risk-level 122

5.1 Characteristics of used database. 137
5.2 NSGA II Parameter setting for each tested data. 138
5.3 Modified Laplace Model Obtained Results 140
5.4 Table of compared results. 143

Introduction

A wide variety of problems in engineering, industry, and many other fields, involve the simultaneous optimization of several objectives. In many cases, the researchers are faceted to a multiplicity of goals, and they present some contradictory (i.e., one objective cannot be improved without infecting, at least, another objective). These problems are called Multiobjective Optimization Problems (Multiobjective Problems (MOPs)). In single-objective optimization, it is relatively simple to compare possible solutions. As a result, a single optimal solution is obtained. However, in multiobjective optimization there does not exist a straightforward method to determine if a solution is better than others. It more an evaluation process called Pareto dominance relation [27]. Which, alternatively to a single optimal solution, leads to a set of alternatives with different trade-offs among the objectives. These solutions are called Pareto optimal solutions or non-dominated solutions. The resolution of a multiobjective optimization problem is also a difficult task considering the contradictory objectives and the multiplicity of the solutions. In this context, two main approaches to solve multiobjective optimization problems are distinguished. The first is called the Multi-Criteria Decision Making (Multicriteria Decision Making (MCDM)) approach which can be characterized by the use of mathematical programming techniques and a decision-making method in an intertwined manner. In most of the MCDM's methods, the decision-maker plays a major role in providing information to build a preference model. The second approach is the evolutionary Multiobjective Optimization (Evolutif Multiobjectif Optimization (EMO)). It is a useful approach to solve multiobjective optimization problems. Since evolutionary algorithms use a population-based approach in addition to the Pareto-based concept, contrary to Multi-Criteria Decision Making, they usually find an approximation of the whole Pareto front in one run. Although in the EMO community the decision-making task has not received too much attention in the past, in recent years a considerable number of works have addressed the incorporation of preference in Multi-Objective Evolutionary Algorithms (MultiObjective Evolutionary Algorithms (MOEAs)).

The second axis of this thesis is artificial neuron networks. Artificiel Neural Network (ANNs) are a computational model based on the structure and functions of biological neural networks. It is like an artificial human nervous system for receiving, processing, and transmitting information in terms of Computer Science. The data explosion in modern drug discovery research requires sophisticated analysis methods to uncover the hidden causal relationships between single or multiple responses and a

large set of properties. Compared to a traditional regression approach, the ANNs is capable of modelling complex nonlinear relationships. The ANNs also has excellent fault tolerance and is fast and highly scalable with parallel processing. Several ANNs exists in the literature such as multilayer perceptron [14]. It is a feedforward network intended to perform non-linear classification. A multilayer perceptron (MLPNN) is a deep artificial neural network. It is composed of more than one perceptron. MLPNN with one hidden layer is capable of approximating any continuous function. Multilayer perceptrons are often applied to supervised learning problems they train on a set of input-output pairs and learn to model the correlation between those inputs and outputs. However, the performance of the MLPNN is closely depending on MLPNN learning, a parameter (the weights and biases) setting of MLPNN to fit the treated data without falling into overfitting or underfitting area. Indeed, learning is a multiobjective model since we are looking for a tradeoff between accuracy and complexity. The MLPNN training looks to minimize the training error in the first order, these why several training approaches were concentrated in it such as Back-propagation. However, more objectives are taken into consideration. the second type of ANNs is the recurrent neural network, it is a feed-forward neural network having one or more hidden layers with at least one feedback loop. the feedback may be a self-feedback, where the output of each neuron is fed back to its input. We are interested in the recurrent neural network: Hopfield neural networks [45], it was invented by Dr John Hopfield in 1982, it consists of a single layer which contains one or more fully connected recurrent neurons. Hopfield networks are commonly used for auto-association and optimization tasks [58].

As part of the theses research, the portfolio selection problem was treated, since this later is a considerably important problem in the economic fields. Besides, it presents contradictory objectives searching for an answer to the investor for his must difficult question, how to allocate his capital between many actives, with the best profitability and minimum risk level. The first model was the Markowitz model or Mean-variance model in 1952 [75] [51]. Markowitz was the pioneer of the first rigorous treatment of the investor's dilemma. Many models were proposed after, however, each approach was focusing on one of the cited objectives. Considering the multi-objectivity of the portfolio selection problem, we propose new models, by adding a new constraint to the Markowitz's models. Regarding its effectiveness, continuous Hopfield network is used to solve the proposed models. In this regard, we construct an original energy function that makes a compromise between the risk, profit and cardinality constraints. To ensure the equilibrium state of the network, the parameters of the energy function are chosen based on a consistency mathematical results; besides, the slop of the activation functions is chosen such that the behaviour of each neuron is almost leaner. The proposed method produces good solutions. However, the proposed energy function seems to be linear at a point due to the behaviour of the risk aversion parameter. To avoid the aforementioned dilemma, we propose hybrid approaches basing on continuous Hopfield network (CHN) and Genetic Algorithm (GA). This latter product better solution compared to literature methods, basing on real financial data. The courant chapter was based on three

publications (two articles [103], [107] and one proceeding puplication [104]).

Our research interests also focused on multi-layer perceptron learning, indeed, the learning task is a multi-criteria task. In this context, we propose a new approach for the MLPNN learning to face the generalization problem. As known, most supervised learning algorithms aim to minimize the training error. However, the mentioned method, based only on error minimizing can generate insufficient generalization performance. The proposed approaches consist of a multiobjective modelling implying two mean objectives: training and complexity. Indeed, the learning tasks is assured by minimizing the two objectives simultaneously, according to the Pareto domination concept using the Non-dominated Sorting Genetic Algorithm ($NSGA$) as a solver. This method leads to a set of solutions called the Pareto front, being the optimal set of solutions, the adequate $MLPNN$ must be extracted. The results show that the proposed method is capable of reducing the neural network, besides, the selected network provides a good classification rate compared to different methods. The present chapter is an instant for three publication (one article [106], one chapter [105] and one proceeding publication [56])

The remains Theses is organized in the following layout: The first chapter named "Résumé des travaux de recherche", presents a summary of the following chapters. Well, the second chapter: Multiobjective optimization explore the multiobjective optimization problem framework. At first, the essential generality and definitions as the problem formulation, optimality and Pareto front. Then, the different approaches proposed for the resolution of multiobjective problems are presented. Principally, two approaches are disdained; the first one mains to reduce the multiobjective problem to a single mono-objective problem using several technics such as the weighted objective functions. Besides to the ϵ-constraints Method. The second approach will be to use a developed evolutionary algorithm to keep the spirit of multiobjective and produce a potential Pareto front in a single run.The second chapter titled: Artificial neural network: learning and optimization; The chapter presents a brief description of the artificial neural network history and fundamentals. The theory of learning is explained as well as ANNs architectures. Therefore, the multilayer perceptron is detailed especially the different technics and approaches proposed to improve the MLPNN training. In the end, the continuous Hopfield network is carefully explained through his architecture, dynamic and his mode of convergence. The next chapter: Neural-Metaheuristics approach for the portfolio selection problem is dedicated to the portfolio selection problem, the first part of the chapter explored a part of the problem literature. The problem was defined and the different models were presented. The second part presents the proposed multiobjective models, besides the adaptation of the continuous Hopfield network and the hybrid approach are detailed. the last part of the chapter exhibits the different obtained results. The last chapter; Multiobjective learning approaches for multilayer perceptron; In this chapter, a multiobjective approach is proposed for the multilayer learning, in this context, a description of the used methodologies is presented. At first, the proposed models are well explained. The second part of the chapter deals with the non-dominated Sorting Genetic Algorithm II (NSGA II) adaptation to the problem. Eventually, the

obtained results are presented and examined.

Chapter 1

Résumé des travaux de recherche

1.1 Introduction

Une grande variété de problèmes dans le domaine d'ingénierie, d'industrie et de nombreux autres domaines impliquent l'optimisation simultanée de plusieurs objectifs. Dans de nombreux cas, les chercheurs sont confrontés à une diversité d'objectifs en présentant certains objectifs contradictoires (c'est-à-dire qu'un objectif ne peut être amélioré sans affecter négativement, au moins, un autre objectif). Ces problèmes sont appelés problèmes d'optimisation multiobjectifs (MOPs). En optimisation mono-objectif, il est relativement simple de comparer les solutions possibles. Par conséquent, une solution optimale unique est obtenue. Cependant, dans l'optimisation multiobjectif, il n'existe pas une méthode simple pour déterminer la meilleure solution. En fait, il s'agit plus d'un processus d'évaluation appelé relation de dominance de Pareto qui conduit à un ensemble d'alternatives avec différents compromis entre les objectifs [27]. Ces derniers sont appelées solutions optimales de Pareto ou solutions non dominées. La résolution d'un problème d'optimisation multiobjectif est également une tâche difficile compte tenu des objectifs contradictoires et de la multiplicité des solutions. Dans ce contexte, deux approches principales pour résoudre les problèmes d'optimisation multiobjectifs sont distinguées : la première est appelée l'approche de prise de décision multicritère (MCDM) caractérisée par l'utilisation de programmation mathématiques d'une part et la prise de décision de manière entrelacée d'autre part [96]. Dans la plupart des méthodes du MCDM, le décideur joue un rôle majeur en fournissant des informations pour construire un modèle de préférence. La deuxième approche est l'optimisation évolutive multiobjectif (EMO). Il s'agit d'une approche utile pour résoudre les problèmes d'optimisation multiobjectifs. Étant donné que ces derniers utilisent une approche basée sur la population en plus du concept de Pareto, contrairement à la prise de décision multicritère, ils trouvent généralement une approximation de l'ensemble du front de Pareto en une seule fois. Bien que, dans le passé, l'intégration de la prise de décision dans le processus d'optimisation reste faiblement considérée par la communauté EMO. Ces dernières années, un nombre considérable de travaux ont porté sur l'incorporation de la préférence dans les algorithmes évolutifs multiobjec-

tifs (MOEAs) [17].

Le deuxième axe de la présente thèse est les réseaux de neurones artificiels (Réseaux de Neurone Artificiel (RNA)) Ces derniers se considèrent comme un modèle de calcul basé sur la structure et les fonctions des réseaux neuronaux biologiques similaire à un système nerveux humain artificiel afin de recevoir, traiter et transmettre des informations en termes d'informatique. En fait le débordement des données dans la recherche moderne nécessite des méthodes d'analyse sophistiquées afin de découvrir les relations causales cachées entre des réponses uniques ou multiples à un large ensemble de propriétés. En comparant un réseau de neurones artificiels à une approche de régression traditionnelle, un RNA est capable de modéliser des relations non linéaires complexes; rapidité ainsi que hautement évolutif concernant le traitement parallèle. Plusieurs typologies de RNA existent dans la littérature comme le réseau de Hopfield [45], la carte auto-organisatrice (Self Organization Map (SOM)) [63] et notamment le perceptron multicouche [14]. Ce dernier est un réseau à action directe destinée à effectuer une classification non linéaire. Un perceptron multicouche (MLPNN) est un réseau neuronal profond composé de plusieurs couches. Les MLPNN avec une couche cachée sont capables d'approcher n'importe quelle fonction continue. Les perceptrons multicouches sont souvent appliqués aux problèmes d'apprentissage supervisés qu'ils entraînent sur un ensemble de paires entrées-sorties et apprennent à modéliser la corrélation entre ces entrées et sorties. Cependant, les performances des MLPNN dépendent rigoureusement de l'apprentissage. Dans le but de s'adapter aux données traitées sans se retrouver dans une zone de surajustement ou de sous-ajustement. Effectivement, l'apprentissage est un modèle multiobjectif puisque nous recherchons un compromis entre la précision et la complexité. L'apprentissage du MLPNN cherche, en effet, à minimiser l'erreur d'apprentissage dans le premier ordre, ce qui explique pourquoi plusieurs approches d'apprentissage y étaient concentrées telles que la rétropropagation [74]. Cependant, nouveaux objectifs sont pris en considération. Le deuxième type de RNA est les réseaux récurrents, c'est un réseau neuronal à action directe ayant une ou plusieurs couches cachées avec au moins une boucle de rétroaction. Elle peut être une auto-rétroaction, où la sortie de chaque neurone est renvoyée à son entrée. Nous nous intéressons aux réseaux de neurones de Hopfield [45], ces derniers ont été inventé par le Dr John Hopfield en 1982, en effet, ils se composent d'une seule couche qui contient un ou plusieurs neurones récurrents entièrement connectés. Les réseaux Hopfield sont couramment utilisés pour les tâches d'auto-association et d'optimisation.

Dans le cadre de nos recherches, le problème de sélection de portefeuille a été traité, vu son importance dans les domaines économiques. En outre, il présente des objectifs contradictoires à la recherche d'une réponse à l'investisseur pour sa question difficile par exemple, comment répartir son capital entre de nombreux actifs, et ceci avec une meilleure rentabilité et un niveau de risque réduit. Le premier modèle était le modèle de Markowitz ou le modèle à variance moyenne de 1952 [75] [51]. Markowitz a été le pionnier du premier traitement rigoureux du dilemme de l'investisseur. De nombreux modèles ont été proposés après, cependant, chaque approche se concentrait sur l'un des objectifs cités. Compte tenu de la multiobjectivité

du problème de sélection de portefeuille, nous proposons de nouveaux modèles, en ajoutant une nouvelle contrainte aux modèles de Markowitz. Concernant son efficacité, le réseau Hopfield continu est utilisé pour résoudre les modèles proposés. À cet égard, nous construisons une fonction énergétique originale qui réalise un compromis entre les contraintes de risque, de profit et de cardinalité. Pour assurer l'état d'équilibre du réseau, les paramètres de la fonction énergétique sont choisis en fonction d'une cohérence des résultats mathématiques ; de plus, la pente des fonctions d'activation est choisie de telle sorte que le comportement de chaque neurone est presque plus léger. La méthode proposée produit de bonnes solutions. Cependant, la fonction d'énergie proposée semble être linéaire en un point en raison du comportement du paramètre d'aversion au risque. Pour éviter le dilemme susmentionné, nous proposons des approches hybrides basées sur le réseau de Hopfield continu (CHN) et l'algorithme génétique (GA). En utilisant une base de données financières réelle, le modèle proposé génère une meilleure solution par rapport aux méthodes classiques.

Nos intérêts de recherche ont porté aussi sur l'apprentissage du perceptron multicouche, en effet, l'apprentissage est une tâche multicritère. Dans ce contexte, nous proposons une nouvelle approche pour l'apprentissage du réseau ($MLPNN$), pour faire face au problème de généralisation. Comme mentionné, la plupart des algorithmes d'apprentissage supervisé visent à minimiser l'erreur d'apprentissage. Cependant, les méthodes mentionnées, basées uniquement sur la minimisation des erreurs, peuvent générer une solution avec des performances de généralisation insuffisantes. Ce travail propose des modèles d'optimisation impliquant deux objectifs : la précision et la complexité car le problème d'apprentissage est par nature multiobjectif. La tâche d'apprentissage se poursuit en minimisant les deux objectifs simultanément, selon le concept de domination de Pareto, en utilisant $NSGAII$ (Non-dominated Sorting Genetic Algorithm II) comme solveur. Cette méthode nous conduit à un ensemble de solutions nommé front de Pareto, étant l'ensemble de solutions optimales, le $MLPNN$ adéquat doit être extrait. Nous montrons empiriquement que la méthode proposée est capable de réduire la topologie du réseau neuronal et d'améliorer les performances de généralisation accompagnée par un bon taux de classification.

Le présent chapitre est organisé comme suit : la premiére section nommé "Optimisation multiobjectif" concerne la partie d'exploitation de la littérature principalement le problème d'optimisations multiobjectifs. Dans un premier temps, nous présenterons la généralité essentielle, les définitions, la formulation de la problématique ainsi que les différentes approches proposées pour la résolution de problèmes multiobjectifs. Principalement, deux approches sont distinguées ; la première vise à réduire le problème multiobjectif à un seul problème mono-objectif en utilisant plusieurs techniques tels que les fonctions d'objectifs pondérés, entre autres, la méthode ϵ -contraints. La deuxième approche consiste à utiliser un algorithme évolutif développé pour conserver l'esprit du multiobjectif et produire un front de Pareto potentiel en une seule fois. La seconde section noté "Réseaux de neurones artificiels : architectures et apprentissage" est une brève description des principes fondamentaux des réseaux de neurones artificiels. Nous adresserons, en premier, la théorie

de l'apprentissage ainsi que les différentes architectures RNA. Ensuite, nous détaillerons les différentes techniques dédiées au perceptron multicouche, en particulier, approches proposées pour améliorer l'apprentissage. Finalement, le réseau continu de Hopfield est soigneusement expliqué à travers son architecture, sa dynamique et son mode de convergence. La suivante section abordera l'une des contributions "Approche neuro-métaheuristique pour le problème de sélection du portefeuille", nous présenterons le problème de sélection de portefeuille, la première partie de la section représente une partie de la littérature de la problématique. Celle-ci sera définie ainsi que les différents modèles proposés. La deuxième partie présentera les approches multiobjectifs, en outre l'adaptation du réseau continu de Hopfield et l'approche hybride sont détaillées. La dernière partie représente les différents résultats obtenus. Ainsi, cette contribution a fait l'objet de trois différentes publications (deux articles [103], [107] et une publication proceeding [104]). La dernière section "Multiobjectif approche pour l'apprentissage du perceptron multicouche" proposera une approche multiobjectif pour l'apprentissage multicouche, dans ce contexte, une description des méthodologies utilisées sera présentée. Dans un premier temps, les modèles proposés sont bien expliqués. Ensuite, l'adaptation de l'algorithme (NSGA II) au problème est présentée. Finalement, les résultats obtenus sont présentés et examinés. Cette dernière section présente trois publications (un article[106], un chapitre [105] et une publication procceding [56]).

1.2 Optimisation multiobjectif

L'optimisation mathématique cherche à analyser, modéliser et résoudre les problèmes de différents domaines par plusieurs méthodes, techniques et outils scientifiques, c'est-à-dire chercher le minimum ou le maximum (l'optimum) d'une donnée sous un ensemble de contraintes. Ils sont généralement des problèmes réalistes représentant un ou plusieurs critères. En fait, nous cherchons à satisfaire l'ensemble des critères représentant un problème. Parfois, l'optimisation mono-objectif est incapable de fournir une réponse convenable. Pour faire face, les chercheurs ont exploité d'autres méthodes et techniques comme l'optimisation multicritère.

1.2.1 Généralités

L'optimisation multiobjectif peut être définie comme un problème dont on cherche l'action qui satisfait un ensemble de contraintes et optimise un vecteur de fonctions objectif. Le principe d'une optimisation multiobjectif est différent du principe d'une approche mono-objectif. En effet, il y a plus qu'une fonction objective, chacune pouvant avoir une solution optimale différente. Le but est de trouver un « bon compromis » entre les critères plutôt qu'une seule solution, ce qui génère un ensemble de solutions et non pas une solution optimale.

La difficulté d'un problème multicritère réside dans le fait qu'il n'existe pas une seule solution optimale mais un ensemble de solutions optimales. Alors, le décideur peut choisir une solution à d'autres, pourtant, il n'existe pas une solution meilleure

que toutes les autres. Or résoudre un problème multiobjectif ne consiste pas à rechercher la solution optimale mais l'ensemble des solutions satisfaisantes pour lesquelles on ne pourra pas effectuer une opération de classement.
Dans ce qui suit nous allons présenter les différentes définitions et théorèmes englobant cette thématique.

Definition 1.2.1 *Un problème d'optimisation multiobjectif peut être formulé de la manière suivante :*
Trouver le vecteur $x^* = (x_1^*, x_2^*, ..., x_n^*)$ *qui satisfait les m contraintes d'inégalités et les p contraintes d'égalités suivantes :*

$$\begin{array}{l} g_i(x) \geq 0 \quad i = 1, ..., m \\ h_j(x) = 0 \quad j = 1, .., p \end{array} \tag{1.2.1}$$

En optimisant (minimiser ou maximiser) le vecteur de fonctions suivant :

$$f(x) = (f(x_1), f(x_2), ..., f(x_k))^t \tag{1.2.2}$$

Au $19^{éme}$ Siècle, Vilfredo Pareto formule le concept suivant : dans un problème multiobjectif, il existe un équilibre tel que l'on ne peut pas améliorer un objectif sans détériorer au moins un des autres objectifs, qui est basé sur la notion de dominance. pour cela, la notion d'optimalité change d'un cadre mono-objectif à un autre multiobjectif. Ce concept génère un ensemble de solutions dites Pareto optimal. Le nombre de solutions optimales croît en fonction de la taille des problèmes et le nombre de critères utilisés.

Definition 1.2.2 (Dominance) *Une solution A domine une solution B si*

$$\forall i \in 1, ..., k : f_i(A) \leq f_i(B) \quad et \quad \exists j \in 1, ..., k : f_j(A) < f_j(B) \tag{1.2.3}$$

la définition (1.2.2) est présenté dans un cas d'un problème de minimisation. En général, une solution A domine une solution B, si et seulement si, A est équivalente à B pour tout critères et A est meilleur que B au moins en un critère.
En utilisant le concept de dominance, une solution x^* est dite Pareto optimal s'il n'existe pas de solution dans l'espace de décision qui domine x^*.

Definition 1.2.3 *l'ensemble Pareto optimal d'un problème d'optimisation multi objectif est l'ensemble de toutes les solutions Pareto optimales du problème.*

L'image de l'ensemble Pareto optimal dans l'espace des objectifs est appelée **"le front de Pareto"**, ou surface de compromis. L'allure de cette frontière prend des formes différentes selon les objectifs à atteindre (minimiser, maximiser ou les deux).

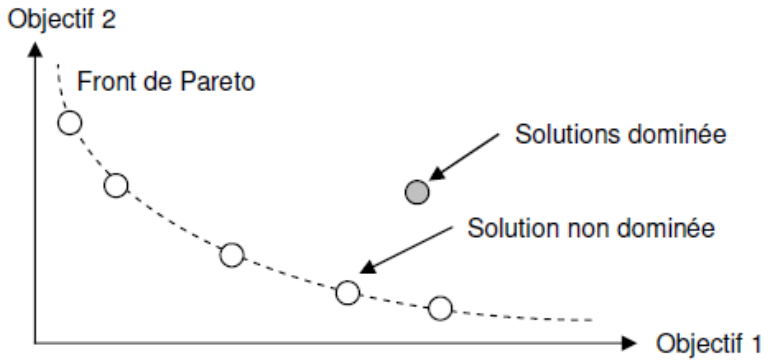


Figure 1.1: Exemple de front de Pareto

La figure (1.1) constitue l'exemple d'un front de Pareto pour un problème de minimisation de deux objectifs.

Definition 1.2.4 (optimalité locale au sens de Pareto) *Un vecteur $x \in \mathbb{R}^n$ est localement optimal au sens de Pareto s'il existe un réel $\Delta > 0$ tel qu'il n'y ait pas de vecteur x^* qui domine le vecteur x avec $x^* \in \mathbb{R} \cap B(x, \Delta)$ où $B(x, \Delta)$ représente la boule de centre x et de rayon Δ.*

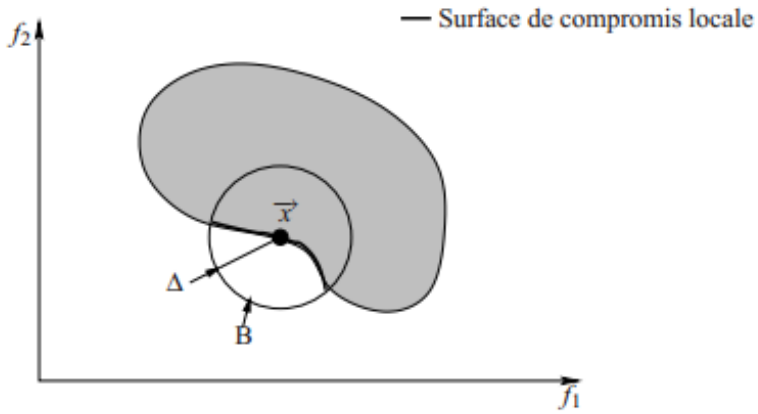


Figure 1.2: L'optimalité locale au sens de Pareto [21]

Definition 1.2.5 (optimalité globale au sens de Pareto) *Un vecteur x est optimal globalement au sens de Pareto (ou optimal au sens de Pareto) s'il n'existe pas de vecteur x^* qui domine le vecteur x.*

La différence entre cette définition et celle de l'optimalité locale tient dans le fait que l'on ne considère plus une restriction de l'ensemble $\mathbb{R}^n$.

1.2.2 Approches de résolution multiobjectif

Dans cette partie, nous considérons les approches recherchent des solutions de meilleurs compromis. Dans la littérature, nous distinguons deux différentes approches de résolution de problèmes multiobjectifs. La première adopte un point de vue décideur,

les approches sont classées en fonction de l'usage que le décideur désire en faire. La deuxième adopte un point de vue concepteur, les approches sont triées de leur façon de traiter les fonctions objectives. Ainsi avant d'entamer l'étape de résolution d'un problème multiobjectif, il faut se questionner sur le type d'approche de résolution à utiliser.

Dans un cas plus général, On peut citer trois schémas possibles. Soit le décideur intervient dès le début de la définition du problème, en exprimant ses préférences, afin de transformer un problème multiobjectif en un problème mono-objectif. Ou bien, le décideur affine son choix de compromis au fur et à mesure du déroulement de l'optimisation. Soit le décideur effectue son choix dans l'ensemble des solutions proposées par le solveur multiobjectif :

- **Les approches à priori**: le décideur intervient en aval du processus d'optimisation, pour définir la fonction d'agrégation modélisant le compromis que l'on désire faire entre les différents objectifs. Dans ce cas le décideur est supposé connaître a priori le poids de chaque objectif afin de les mélanger dans une fonction unique. Cela revient à résoudre un problème mono-objectif. Cependant dans la plupart des cas, le décideur ne peut pas exprimer clairement sa fonction d'utilité, vu que les différents objectifs sont non commensurables (exprimés dans des unités différentes);

- **Les approches interactives**: combinent de manière cyclique et incrémentale les processus de décision et d'optimisation. Le décideur intervient de manière à modifier certaines variables ou contraintes afin de diriger le processus d'optimisation. Le décideur modifie ainsi interactivement le compromis entre ses préférences et les résultats. Cette approche permet donc de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche;

- **Les approches a posteriori**: cherche à fournir au décideur un large choix de bonnes solutions bien reparties. Le décideur, par la suite, peut sélectionner celle qui lui semble le plus approprié. En effet, le décideur n'intervient pas au cours d'optimisation. Ainsi, les préférences du décideur ne sont pas requises.

Quant à la classification "point de vue concepteur", cette classe adopte un point de vue plus théorique. Les approches utilisées pour la résolution de problèmes multiobjectifs peuvent être classées en deux familles [7]:

- **Les méthodes non-Pareto**: ces approches, comme indique leur nom, proposent de ramener le modèle multiobjectif initial en un ou plusieurs autre modèle mono-objectif. Les méthodes non-Pareto sont classées en deux catégories : les approches scalaires, qui transforment le problème multiobjectif en problème mono-objectif, nous pouvons y distinguer la méthode de pondération des fonctions objectives [20], le principe consiste à transformer le modèle en utilisant une somme pondéré des objectifs étant la fonction objective du nouveau modèle mono-objectif. Les pondérations peuvent présenter les préférences du

décideur. Une autre technique appelée la méthode du compromis ou bien la méthode de la contrainte ϵ consiste à choisir un critère prioritaire qui sera optimisé et limité les critères restant en contraintes [81]. Alors que les approches non scalaires, qui gardent l'approche multiobjectif, mais en traitant séparément chacun des objectifs tels que Vector Evaluated Genetic Algorithm (VEGA) (Vecteur Evaluated Genetic Algorithm) [102]. Ce dernier est la première version de l'algorithme génétique développé pour un problème multiobjectif. Cette version propose de subdivisionner la population en k sous populations, étant k le nombre de critères à optimiser, ainsi n/k meilleur individu est sélectionné pour chaque objectif. Par la suite, il faut mélanger afin d'obtenir une nouvelle population de taille n. Le processus se termine par l'application des opérateurs génétiques (croisement et mutation).

- **Les méthodes Pareto**: les approches Pareto utilisent directement la notion de dominance dans la sélection et l'évaluation des solutions. Ces méthodes ont l'avantage de traiter les problèmes multiobjectifs sans transformation, sans promouvoir un critère par au dépit des autres et ils peuvent générer des solutions supportées et non supportées en un passage unique de l'algorithme. Les méthodes les plus connues sont les métaheuristiques. Les méthodes à base de population reposant sur un ensemble de solutions potentielles, comme les algorithmes évolutionnistes, les essaims particulières ou les colonies de fourmis se voient de bonnes candidates à la résolution des problèmes multicritères. Ce qui nous intéresse dans cette approche sont les algorithmes évolutionnistes, telle que Multi-Objective Genetic Algorithm (MOGA) [82], Non-dominated Sorting Genetic Algorithm (NSGA) [113], NSGA II [26].

Dans ce qui suit nous nous intéresserons à la méthode NSGA II qui sera détaillée dans la suivante section.

1.2.3 La méthode "Non-dominated Sorting Genetic Algorithm II" (NSGA II)

L'algorithme NSGA II est une version améliorée de l'algorithme NSGA, il a était proposé à fin de tenter de résoudre toutes les critiques faites sur NSGA : complexité, non élitiste et utilisation de "sharing". Avant de détailler plus l'algorithme NSGA II, une description est présentée dans les sous-sections suivantes.

1.2.3.1 "Non-dominated Sorting genetic algorithm" (NSGA)

l'algorithme génétique de tri non dominé (NSGA) est suggéré par Srinivas et Deb [113]. Il est propose une classification à plusieurs couches des individus comme suggérés par Goldberg [130]. Avant d'effectuer une sélection, la population est classée sur la base de la non-domination ou bien le rang de dominance: tous les individus non dominés sont classés en une seule catégorie (avec une valeur fictive de fitness, proportionnelle à la taille de la population) voir l'algorithme(3). Pour maintenir

la diversité de la population, ces individus classés sont partagés avec leurs valeurs fictives de fitness. Ensuite, ce groupe d'individus classés est ignoré et une autre couche d'individus non dominés est considérée. Le processus se poursuit jusqu'à ce que tous les individus de la population soient classés. Étant donné que les individus du premier front ont la valeur de fitness maximale, ils obtiennent toujours plus de copies que le reste de la population. L'algorithme du NSGA n'est pas très efficace, car le classement de Pareto doit être répété plusieurs fois.

1.2.3.2 "Non-dominated Sorting Genetic Algorithm II" (NSGA II)

L'algorithme a était développé par Deb et al. [26], il propose d'intègre un opérateur de sélection, basé sur la distance de "Crowding", très différent de celui de NSGA. NSGA II est un algorithme élitiste, différemment à d'autre algorithmes, il n'utilisant pas d'archives externes pour stocker l'élite. NSGA II conserve, à chaque nouvelle génération, les meilleurs individus rencontrés à fin d'assurer l'élitisme. Il consiste à créer une population intermédiaire des meilleurs individus produits par les opérateurs (sélection, croisement, mutation). Ainsi, les deux populations sont triées par le rang de dominance (ranking) ne gardant que les N (taille de la population) meilleur individu. La figure (1.3) présente le fonctionnement de l'algorithme NSGA II.

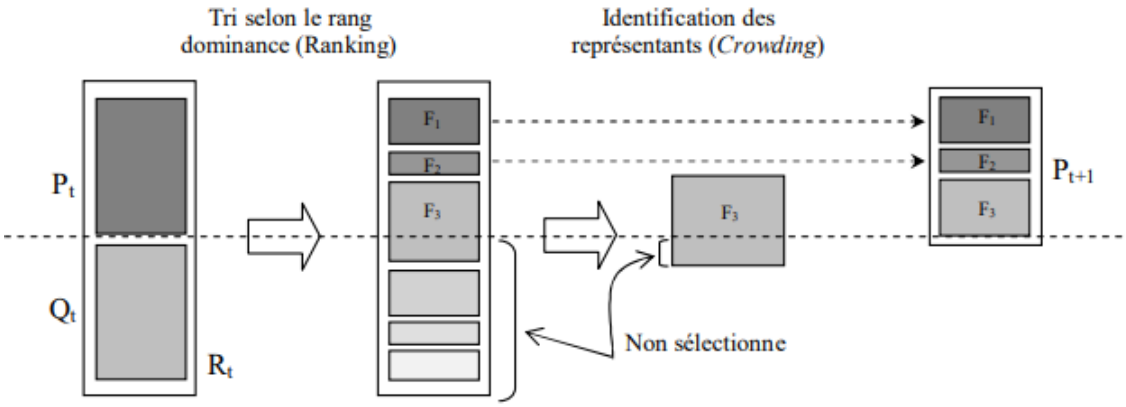


Figure 1.3: Fonctionnement de l'algorithme NSGA II [26]

Distance d'entassement (crowding) La mesure (ou distance) d'entassement est une estimation de la densité des solutions entourant une solution donnée. Commençons par calculer la distance moyenne entre le point représentant cette solution sur le graphe des objectifs et les deux points les plus proches des deux côtés de ce point (voir Fig. 1.4) et ce par rapport à chaque point. Par la moyenne des distances calculées nous obtenons la distance d'entassement de la solution considérée. Le calcul de la distance d'entassement nécessite un tri de la population dans un ordre croissant de chaque objectif. Les solutions en positions limites (avec la valeur la plus petite et la plus grande) prennent une distance infinie. Toutes autres solutions intermédiaires prennent une valeur de distance égale à la valeur absolue de la différence

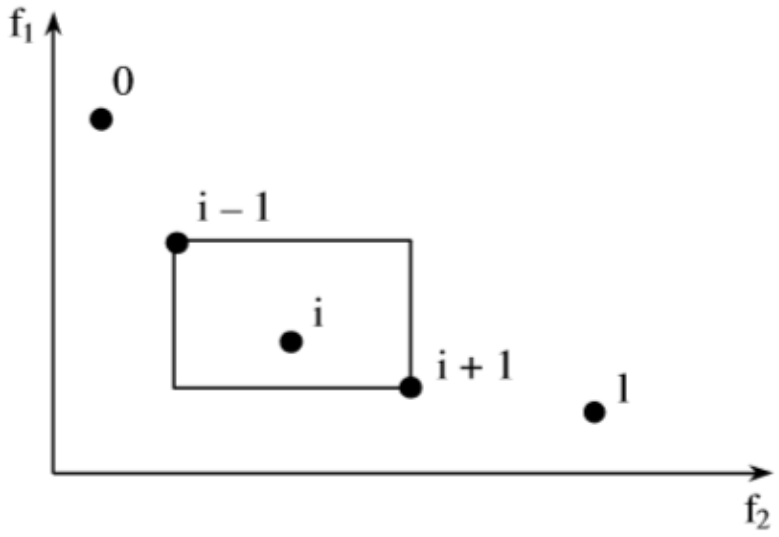


Figure 1.4: Mesure d'entassement (crowding) [26]

normalisée des valeurs de la fonction objective des deux solutions adjacentes dans la liste triée. Cette valeur est calculée pour chaque objectif. La somme des distances d'entassement par rapport à chaque objectif représente la distance d'entassement général d'une solution donnée. Il faut noter que la distance d'entassement est calculée par rapport aux solutions du même front.
L'algorithme (1) décrive la procédure de calcul des distances d'entassement pour toutes les solutions d'un ensemble non dominée I. La procédure trier(I,j) permet de

Algorithm 1: Procédure de calcul de la distance d'entassement (Crowding)

Require: I un front donnée
1: $l = |I|$
2: **for** i=1 to l **do**
3: $I[i]_{dist} = 0$
4: **end for**
5: **for** j=1 to m **do**
6: I=trier(I,j) ; // m: la taille de l'ensemble des objectives
7: $I[1]_{dist} = I[l]_{dist} = \infty$
8: **for** i=2 to l-1 **do**
9: $I[i]_{dist} = I[i]_{dist} + \frac{I[i+1]^j - I[i-1]^j}{f_j^{max} - f_j^{min}}$
10: **end for**
11: **end for**

trier les solutions d'un front donné I par ordre croissant suivant chaque objectif j.
$I[i]^j$ désigne la valeur de l'objectif j relatif à $i^{éme}$ solutions d'un front I.
f_j^{max} et f_j^{min} présentent les valeurs maximales et minimales de la fonction objectif j.

À fin d'obtenir une bonne répartition des solutions au sein de la population, une approche de comparaison des solutions du même front, basé sur la distance d'entassement est alors utilisée. Cette procédure est appelée comparaison d'entassement ("Crowded-comparaison approach"). L'approche consiste à comparer les proximités des solutions. Une solution avec une valeur de distance d'entassement plus petite est une solution plus entassée par rapport aux autres solutions. Cette approche engendre un opérateur de comparaison qui permet de déterminer un ordre partiel des solutions ($\prec_n$) en considérant deux attributs:

- Le rang de la solution : i_{rank},
- La distance d'entassement : i_{dist}.

L'opérateur de comparaison est alors défini comme suit:

$$(i \prec_n j) \leftrightarrow \begin{cases} i_{rank} < j_{rank} \\ \quad ou \\ (i_{rank} = j_{rank}) \quad et \quad (i_{dist} > j_{dist}) \end{cases} \tag{1.2.4}$$

Si deux solutions ont du rang différent, la solution portant un rang inférieur est considérée de meilleure qualité. Alors que pour deux solutions du même front, la solution portant une distance d'entassement plus large est préférée.
L'algorithme (2) présent le déroulement de la méthode NSGA II. Nous considérons que P_t est la population initiale et Q_t est la population des enfants créés par les opérateurs de productions.

Algorithm 2: L'algorithme de NSGA II

1: $R_t = P_t \cup Q_t$
2: F =tri par non-dominance de R_t
3: $P_{t+1} = \emptyset$ et $i = 1$
4: **while** $|P_{t+1}| + |F_i| \leq N$ **do**
5: calculer la distance d'entassement de F_i
6: $P_{t+1} = P_t \cup F_i$
7: $i = i + 1$
8: **end while**
9: trier(F_i,$\prec_n$)
10: FS_i =les premiers N-$|P_{t+1}|$ solutions de F_i
11: $P_{t+1} = P_t \cup FS_i$
12: Q_{t+1} = enfants de P_{t+1}

1.3 Réseaux de neurone artificiel: architectures et apprentissage

Les réseaux de neurones artificiels (RNA) sont des systèmes de traitement d'informations, inspirer du réseau neuronal du système nerveux. Les RNA sont destinées à ef-

fectuer des tâches compliquées semblables au raisonnement humain tel que les tâches de prédiction et classification. De nos jours, les réseaux de neurones sont utilisés dans de nombreux domaines vus leurs propriétés, en particulier leur capacité d'apprentissage. Différents types de RNA peuvent être distingué, suivant leur architecture. Le premier réseau de neurones artificiel a été proposé en 1958 grâce aux travaux de Rosemblatt, appelé "Perceptron" [95]. Il est inspiré du système visuel (en termes d'architecture neurobiologique). L'architecture de ce dernier est composé d'une couche de neurones d'entrée et une autre de sortie ("décisionnelle"). Ce réseau parvient à apprendre à identifier des formes simples et à calculer certaines fonctions logiques [3]. Cependant le perceptron est incapable de traiter des problèmes non linéaires, cela à était bien discuté par M. Minsky et S. Papert dans leur ouvrage publier en 1969 [78]. Vu ces limitations, les chercheurs abandonnent les recherches dans le domaine (surtout en U.S.A), jusqu'au début des années 80. Hopfield restaure l'intérêt pour ce domaine en 1982 [43] et présente l'utilisation des réseaux récurrents (dits "feed-back") pour la modélisation des processus. En parallèle, une version améliorer du perceptron connu sous le nom: "perceptron multicouche (MLPNN)" est développé par Schmidhuber et Hochreiter en 1997 [41], cette fois-ci le réseau et du type "feed-forword".

1.3.1 Neurone formel

Le neurone formel est une modélisation mathématique qui reprend les principes du fonctionnement du neurone biologique, en particulier la sommation des entrées (fig. 1.5). Sachant qu'au niveau biologique, les synapses n'ont pas toutes la même « valeur » (les connexions entre neurones étant plus au moins fortes), les chercheurs ont donc créé un algorithme qui pondère la somme de ses entrées par des poids synaptiques (coefficients de pondération). En général, un neurone formel est un élément de traitement possédant n entrées x_1, x_2,..., x_i,..., x_n (qui sont les entrées externes ou les sorties des autres neurones) et une ou plusieurs sorties. Son traitement consiste à effectuer à sa sortie y_i le résultat d'une fonction de seuillage f (dite aussi la fonction d'activation) de la somme pondérée.

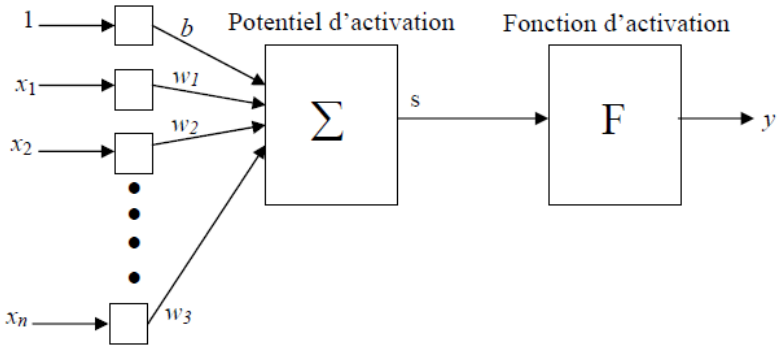


Figure 1.5: Modèle d'un neurone formel [11]

Avec :

Les x_i sont les entrées du réseau.
S est le potentiel d'activation.
Les w_i représentent les poids synaptiques.
y_i la sortie du réseau tels que : $y = f(S); \quad y = \sum_{i=0}^{n} w_i.x_i + b.$

1.3.2 Fonction d'activation

La fonction d'activation [86] (ou fonction de seuillage, ou encore fonction de transfert) sert à introduire une non linéarité dans le fonctionnement du neurone. Les fonctions de seuillage présentent généralement trois intervalles :

- En dessous du seuil, le neurone est non actif (souvent dans ce cas, sa sortie vaut 0 ou -1),
- Aux alentours du seuil, une phase de transition,
- Au dessus du seuil, le neurone est actif (souvent dans ce cas, sa sortie vaut 1).

Dans sa première version, le neurone formel était implémenté avec une fonction à seuil, mais de nombreuses versions existent. Ainsi, le neurone de McCulloch et Pitts a été généralisé de différentes manières, en choisissant d'autres fonctions d'activations. Les trois fonctions les plus utilisées sont les fonctions « seuil », « linéaire », « sigmoïdes ».

1.3.3 Réseau à propagation avant : Perceptron multicouche

Le plus classique des réseaux de neurones est le perceptron multicouche. C'est un Réseaux de neurones à propagation avant (acyclique) "feedforward", c'est-à-dire que l'information est traitée de l'entrée vers la sortie. Comme indique son nom, le perceptron multicouche est structuré de plusieurs couches. Il est ainsi composé d'une couche d'entrée, d'une ou plusieurs couches intermédiaires dites cachées et d'une couche de sortie. L'ensemble de ses nœuds de chacune des couches est relié à l'ensemble des nœuds de la couche précédente ainsi que la suivante. La figure (3.2) présente un perceptron multicouche avec $p-1$ couches cachées.

L'origine du perceptron multicouches remonte à F. Rosenblatt qui en 1957 s'inspire des travaux sur les neurones formels [38] et sur la règle de Hebb introduite dans [80] pour développer le modèle du perceptron mais également la méthode qui permet à ce modèle d'apprendre [94]. Ce modèle n'a qu'une seule couche mais permet déjà de résoudre des tâches simples. Il est cependant impossible avec la méthode formulée par F. Rosenblatt d'entraîner un système ayant plusieurs couches ce qui se révèle plus tard très restrictif. En effet, le perceptron est incapable d'apprendre des fonctions si celles-ci ne sont pas séparables linéairement (comme par exemple, la fonction booléenne XOR). Ainsi, il est nécessaire d'avoir au moins une couche de neurones supplémentaire (soit un perceptron multicouche) pour pouvoir régler le problème. Mais il n'existe alors aucune méthode permettant d'entraîner un perceptron multicouche. Il faut attendre plus de dix ans pour que des chercheurs mettent au point la

technique d'apprentissage qui permet de régler les paramètres d'un perceptron multicouches. En effet, c'est P. Werbos qui le premier propose l'idée d'appliquer aux ANNs la technique de rétropropagation du gradient des erreurs développée dans les années 1960. En 1986, D. Rumelhart, G. Hinton et R. Williams montrent dans [97] qu'un perceptron multicouche entrainé par l'algorithme de rétropropagation du gradient des erreurs peut enfin dépasser les limites du perceptron qui avait été soulevé par M. Minsky et S. Papert en 1969. En outre, le perceptron multicouche est capable de traiter des problèmes non linéaires complexes et peut approximer, avec une seule couche cachée et un nombre de neurones suffisant, n'importe quelle fonction non linéaire et continue sur un espace compact avec une précision arbitraire [47]. Le perceptron multicouche est ainsi appelé un approximateur universel de fonctions.

1.3.3.1 Description mathématique du modèle

Un MLPNN est défini par le nombre de couches n qui le composent et qui se succèdent. La couche $j \in [1, N]$ d'un perceptron multicouche est définie par : $L_j = (n_j, a_j, f_j)$ tel que:

- $n_j \in \mathbb{N}$ est le nombre de neurones dans la couche j,
- $a_j : \mathbb{R}^{n_{j-1}} \to \mathbb{R}^{n_j}$ est la transformation affine définie par la matrice $W_j \in \mathbb{R}^{n_{j-1} * n_j}$ est le vecteur $b_j \in \mathbb{R}^{n_j}$,
- $f_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_j}$ est la fonction d'activation de la couche j.

Il est à noter que :

- La matrice W_j est appelée matrice des poids entre la couche $j-1$ et j,
- Le vecteur b_j est appelé le vecteur de biais de la couche j,
- La fonction f_j est une fonction de $\mathbb{R}^{n_j} \to \mathbb{R}^{n_j}$, non-lineaires, bornée et dérivables. Choisie parmi les fonctions d'activation présentée dans la section 3.3.1.1. La fonction d'activation est appliquée à chacun des éléments des vecteurs de dimensions n_j.

Un perceptron multicouche prend en entrée un vecteur de dimension n_0 et dont le vecteur de sortie est de dimension n_N, alors pour un vecteur d'entrée donnée $x \in \mathbb{R}^{n_0}$, on obtient le vecteur de sortie $y \in \mathbb{R}^{n_N}$ en initialisant le vecteur $h_0 = x$ puis appliquant itérativement chacune des couches L_j, $j \in [1, N]$ au vecteur de sortie de la couche précédente:

- appliquez la transformation affine a_j au vecteur de sortie de la couche précédente:
$$g_j = a_j(h_{j-1}) = W_j.h_{j-1} + b_j \tag{1.3.1}$$
- puis appliquez la fonction d'activation au résultat :
$$h_j = f_j(g_j) \tag{1.3.2}$$

Avec $h_j, g_j \in \mathbb{R}^{n_j}$ et $y = h_N$.
Un perceptron multicouche (MLPNN) est donc une fonction $g_{mlp} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_N}$ définie par la composition successive d'une transformation affine et d'une fonction d'activation non-linéaire et ce pour chacune de ses couches:

$$y = g_{mlp}(x) = f_N \circ a_N \circ f_{N-1} \circ a_{N-1} \circ \cdots \circ f_1 \circ a_1(x) \tag{1.3.3}$$

1.3.4 Réseaux récurent "feed-back": Réseaux de Hopfield continu

Contrairement à un réseau de neurones acyclique (Feed-Forward Neural Network), un réseau de neurones récurrent (Recurrent Neural Network - Recurrent Neural Networks (RNNs)) est un réseau de neurones dont le graphe des connexions contient au moins un cycle, notamment le réseau de Hopfield continu. Ce type des RNA est largement exploité dans plusieurs domaines comme l'analyse des données, la reconnaissance de formes, le traitement de la parole, les mémoires associatives et l'optimisation combinatoire[12], [45].

Le réseau de Hopfield a été inventé par le physicien J. Hopfield en 1982 [43]. Sa découverte a permis de raviver l'intérêt des recherches dans les réseaux de neurones. Par ailleurs, le réseau de Hopfield s'inscrit dans la catégorie des réseaux récurrents. Il est caractérisé par sa capacité de laisser l'information circuler récursivement d'une manière partielle ou bien totale. Il s'agit d'un système à minimisation d'énergie. Ce modèle a été utilisé pour résoudre une grande variété de problèmes d'optimisation combinatoire [109], [112].

La présente section concerne donc la conception générale du réseau de Hopfield continu (CHN) et ses différents attributs essentiels: son architecture, l'équation d'évolution de ses états, ses points d'équilibre et sa fonction d'énergie. Dans ce contexte, nous présentons les principes d'association des variables du problème de programmation quadratique avec les paramètres de la fonction d'énergie à minimiser.

1.3.4.1 Architecture du réseau de Hopfield continu

Le réseau de Hopfield possède une topologie cyclique. Dans ce modèle, le réseau est constitué d'une seule couche de n neurones entièrement interconnectés. Ils évoluent à partir d'un état initial.

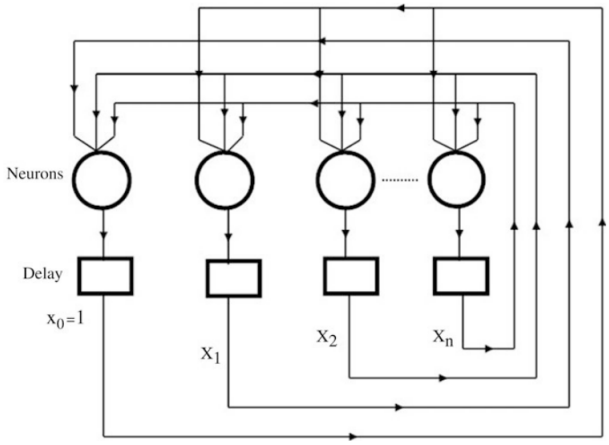


Figure 1.6: Réseau de Hopfield continu à n neurones [11]

La figure (1.6) représente un réseau de Hopfield composé de n neurones. La sortie de chaque neurone est bouclée sur les entrées des autres neurones additionnées à un vecteur du bruit provenant de l'extérieur I_i^b. Un neurone i est relié à un neurone j par des poids symétriques $T_{ij} = T_{ji}$, mais il n'est pas connecté à lui-même $T_{ii} = 0$. De plus, le neurone i est décrit par deux autres variables:

- Le potentiel intérieur u_i calculé comme une somme des informations pondérées par les poids des connexions des autres neurones.

- La sortie v_i définie par $v_i = h(u_i)$. Avec h désigne la fonction d'activation du neurone i.

Souvent, on utilise la fonction d'activation du type tangente hyperbolique qui permet d'assurer la convergence des sorties v_i vers les sommets de l'hypercube $[0,1]^n$.

$$v_i = h(u_i) = \frac{1}{2}(1 + \tanh(\frac{u_i}{u_0})) \quad \forall i \in \{1, ..., n\}, \quad u_0 > 0 \tag{1.3.4}$$

En se référant au cas d'une réalisation physique d'un réseau électronique, basée sur des amplificateurs opérationnels et résistances, l'équation différentielle qui gouverne l'évolution du réseau de Hopfield continu est donnée par[116], [117]:

$$\frac{du}{dt} = -\frac{u}{\tau} + Tv + I^b \tag{1.3.5}$$

Ce système est appelé l'équation d'état du réseau de Hopfield.
Les vecteurs u, v et I^b représentent, respectivement, le vecteur des états, le vecteur des sorties et le vecteur du bruit provenant de l'extérieur. T représente la matrice des poids des connexions entre les neurones.

1.3.4.2 Énergie et stabilité des réseaux de Hopfield continus

Nous présentons dans la suite, la fonction d'énergie associée à l'équation d'état du réseau de Hopfield. Puis après, nous étudions également l'existence d'une solution (point d'équilibre) de cette équation, ainsi que les méthodes d'approximation d'un point d'équilibre lorsque celui-ci existé [115].
L'évolution du réseau de Hopfield est caractérisée par un système d'équations différentielles, alors l'étude de l'existence d'une solution (point de stabilité) de ce système se base sur l'existence d'une fonction d'énergie dite fonction de Lyapunov [43].

Definition 1.3.1 *Soit $v(0)$ le vecteur qui représente les états initiaux des neurones du réseau de Hopfield. Un vecteur v^e est appelé un point de stabilité du réseau de Hopfield s'il existe un instant t^e tel que :*

$$\forall t \geq t^e, \quad v(t) = v^e \tag{1.3.6}$$

Dans [44], Hopfield a démontré l'existence d'un point d'équilibre pour le système (1.3.5). En effet, par un argument de la fonction de Lyapunov, Hopfield a démontré que les systèmes différentiels de la dynamique d'état de son réseau sont globalement asymptotiquement stables, i.e. de n'importe quel état initial, le système converge vers un état stable.
La fonction d'énergie de Hopfield définie sur $H = [0,1]^n$ est donnée par :

$$E(v) = -\frac{1}{2} v^t T v - \left(I^b\right)^t v + \frac{1}{\tau} \sum_{i=1}^{n} \int_0^{v_i} g^{-1}(x)\, dx \tag{1.3.7}$$

Il faut noter que cette fonction E est bornée et sa dérivée est négative. Par conséquent, l'évolution de l'état global du système est un déplacement dans l'espace des états à la recherche d'un minimum, qui peut être local de E.

Theorem 1.3.1 (Lyaponuv) *Si pour un système différentiel, une fonction d'énergie E décroissante par rapport au temps existe, tel que :$\frac{dE(u(t))}{dt} = 0$ pour $\frac{du(t)}{dt} = 0_{\mathbb{R}^n}$ alors ce système est stable ; c'est-à-dire $u(t)$ tend vers un point stable.*

Le théorème suivant donne la forme de la fonction d'énergie associée au système (1.3.5).

Theorem 1.3.2 *Si T est une matrice zéro-diagonale et symétrique, alors la fonction définie par (1.3.7) est une fonction d'énergie associée au système (1.3.5) qui vérifie les conditions du théorème de Lyapunov.*

Hopfield a défini une fonction d'énergie E pour le réseau proposé, qui mesure son état à un instant donné et qui est très proche de l'énergie d'un système thermodynamique. De cette façon, l'énergie minimale du réseau de Hopfield correspond à un état d'équilibre de ce réseau. Plus précisément, Hopfield a montré que l'énergie E ne peut que baisser ou rester stable en un minimum local. Aussi, il a prouvé que la symétrie de la matrice des poids est une condition suffisante pour l'existence de la fonction de Lyapunov [43], [44].

Un point $v \in H$, peut être un point d'équilibre du réseau de Hopfield si et seulement si, les trois conditions suivantes sont satisfaites [115]:

$$E_i(v) = \frac{\partial E(v)}{\partial v_i} \geq 0 \quad \forall i \in \{1, ..., n\} \quad tel\ que \quad v_i = 0 \tag{1.3.8}$$

$$E_i(v) = \frac{\partial E(v)}{\partial v_i} \leq 0 \quad \forall i \in \{1, ..., n\} \quad tel\ que \quad v_i = 1 \tag{1.3.9}$$

$$E_i(v) = \frac{\partial E(v)}{\partial v_i} = 0 \; \forall i \in \{1, ..., n\} \quad tel\ que \quad v_i \in]0, 1[\tag{1.3.10}$$

1.3.5 Réseaux de neurones artificiels et apprentissage

L'apprentissage d'un RNA est une phase importante permettant d'améliorer la performance de celui-ci. L'apprentissage contrôle le comportement du réseau, cette phase modifie les paramétrer du réseau afin d'obtenir un comportement souhaitable. Nous pouvons distinguer trois classiques catégories d'apprentissage:

Apprentissage supervisé : Ce type d'apprentissage nécessite des informations a priori sur les réponses de la problématique traité sous forme d'exemples étiquetés appelés ensemble d'apprentissage tel que $D_t = \{(x_1, d_1), (x_2, d_2), (x_3, d_3), ..., (x_N, d_N)\}$ pour x_i désigne un stimulant (vecteur entrée du réseau) et d_i l'objet de relance appelé : *sortie désirée*. Le couple (x_i, d_i) de l'ensemble d'apprentissage correspond au comportement attendu du réseau pour un stimulant donné. L'apprentissage supervisé ajuste les paramétrés du réseau par une comparaison directe entre la sortie réelle (y_i) et la sortie désirée (d_i). La mesure d'erreur utilisée pour guider le processus d'apprentissage est généralement définie par l'erreur quadratique moyenne (Mean Squared Error (MSE)):

$$E = \frac{1}{N} \sum_{i=1}^{N} \|d_i - y_i\|^2 \tag{1.3.11}$$

Tel que N est la taille de l'ensemble d'apprentissage et y_i est la sortie réseau du modèle i. Le processus d'apprentissage est réalisé lorsque l'équation 1.3.11 est suffisamment petite ou qu'un critère d'arrêt est satisfait. L'apprentissage supervisé [69] est largement utilisé pour la classification, l'approximation, le traitement du signal et l'optimisation.

Apprentissage non-supervisé : Cette catégorie ne nécessite aucune valeur cible. Il essaie de décrire les associations et les modèles parmi un ensemble de mesures d'entrée. Ainsi, seules les caractéristiques sont observées, de plus, les résultats ne sont pas mesurables. Principalement, la tâche d'apprentissage consiste davantage à décrire les données, nous parlons d'organisation ou de classification. Un apprentissage non supervisé ou "apprentissage sans enseignant" [8] est entièrement basé sur les corrélations entre les données. Il est utilisé pour trouver les modèles ou caractéristiques significatives dans les données d'entrée sans l'aide d'un décideur ou de

guide. Ce type d'apprentissage est principalement utilisé pour "clustering", la quantification vectorielle, l'extraction d'entités, le codage de signaux, en plus de l'analyse des données.

Apprentissage par renforcement : Ce paradigme peut être considéré comme un cas particulier de l'apprentissage supervisé, où la sortie souhaitée exacte est inconnue. Le formateur ne fournit que des commentaires sur le succès ou l'échec d'une réponse. L'apprentissage par renforcement [114] essaie de maximiser un signal de récompense au lieu d'essayer de trouver une structure cachée. De plus, la procédure d'apprentissage récompense le réseau neuronal pour son bon résultat de sortie et le punit pour un mauvais résultat. L'apprentissage par renforcement est généralement utilisé dans le contrôle et l'intelligence artificielle.
L'apprentissage du perceptron multicouche est soumis à un paradigme d'apprentissage supervisé. Cependant, l'apprentissage d'un MLPNN peut être présenté comme un problème d'optimisation. Les sections suivantes couvriront les différentes méthodes et techniques proposées pour l'apprentissage du perceptron multicouche.

1.3.5.1 Apprentissage et optimisation

Cependant, l'atteinte d'une fonction objective adéquate est la seule question, qui représente au mieux les sélections de poids. Une fois le modèle d'optimisation établi, il existe de nombreux algorithmes prêts à l'emploi, que ce soit les méthodes classiques telles que la rétropropagation [97] ou une approche d'optimisation sophistiquée telle que l'algorithme génétique [32] le modèle MLPNN peut le faire fonctionner. Avant d'aller plus loin, rappelons l'algorithme d'apprentissage le plus utilisé, l'algorithme d'apprentissage à rétropropagation.

Algorithme d'apprentissage à propagation arrière : L'algorithme a été proposé pour la première fois par Werbos [123] en 1974, pour être ensuite exploité par Rumelhart et al. dans le processus cognitif de simulation. L'algorithme d'apprentissage de rétropropagation réalise une recherche de descente de gradient dans l'espace des poids à la recherche de la fonction d'erreur (MSE) minimum. L'algorithme 4 présente les principales étapes de l'algorithme de rétropropagation.

L'algorithme d'apprentissage de rétropropagation s'est avéré être un algorithme robuste; cependant, l'algorithme converge lentement. En outre, l'algorithme mentionné, en tant qu'algorithme d'optimisation, permet d'obtenir des informations sur les valeurs optimales des poids du réseau, mais l'algorithme ne fournit pas d'informations sur la complexité du réseau.

De plus, dans la littérature, de nombreuses approches ont été proposées pour l'apprentissage du MLPNN, en utilisant l'erreur quadratique moyenne (MSE) comme fonction objective pour le modèle d'optimisation MLPNN. Néanmoins, minimiser uniquement l'erreur d'apprentissage peut donner lieu à des modèles de sur-ajustement. Ainsi, d'autres travaux introduisent des contraintes au modèle d'optimisation, visant à contrôler le nombre de connexions et de neurones du réseau [15], [93]. Ce processus

produit une architecture de réseau neuronal optimisée. Cependant, la régularité du réseau reste à améliorer. Pour cette raison, les chercheurs proposent de régulariser le réseau en incluant des informations supplémentaires pour éviter un sur-ajustement, le processus consiste à ajouter un terme de pénalité avec un paramètre de régularisation (λ) à l'erreur de formation MLPNN, représentant la complexité du réseau. Au cours du processus, les modèles à petits coefficients sont privilégiés [16], [92]. Avant d'explorer davantage les problématiques d'apprentissage du MLPNN, détaillons d'abord la théorie de la régularisation.

1.3.5.2 Régularisation d'un réseau de neurone artificiel

La régularisation est une technique utilisée pour trouver une solution stable lorsqu'un problème d'identification de paramètre est exposé à des erreurs considérables. Alors, si une fonction objectif est complexe, même un petit changement des paramètres peut conduire à un changement significatif de la valeur de la fonction objective. La régularisation contribue à la stabilisation d'une telle fonction. Une régularisation de réseau cherche à contrôler la variation et la complexité du réseau en contrôlant les valeurs de pondération du réseau. Bartlett [9] montre que les valeurs des poids étaient plus importantes que leur nombre dans le réseau afin d'obtenir un modèle de réseau de neurones ajusté. Plusieurs méthodes ont été proposées pour la régularisation du réseau, nous pouvons présenter:

Early Stopping : Étant donné que l'apprentissage supervisé consiste à minimiser la fonction de coût d'apprentissage (MSE), la technique "Early Stopping" [90] propose d'arrêter les étapes de l'algorithme d'apprentissage juste avant que la convergence de l'algorithme tel que le modèle ne soit pas finement défini sur l'ensemble d'apprentissage et alors le sur-ajustement est limité.

Pénalité de poids (Weight Decay) : la technique de réduction de poids consiste à introduire une fonction de coût à l'erreur de formation afin de surmonter du sur-ajustement du réseau.

$$E = E + \lambda\Omega \tag{1.3.12}$$

Tels que λ est un hyperparamètre qui contrôle la force de régularisation et Ω est le terme de régularisation. Ci-dessous, nous présentons les termes de régularisation les plus courants.

La norme L_1 ou Last Absolue Shrinkage and Selection Operator (LASSO) régularisation : Ce terme tente de minimiser la somme de la différence absolue entre les valeurs réelles et les valeurs prédites. La régularisation LASSO offre la possibilité de sélectionner facilement les caractéristiques en forçant une représentation clairsemée. la régularisation se présente comme:

$$\Omega_{L_1} = \sum_k |w_i| \tag{1.3.13}$$

Tel que $k = 1, ..., p$ avec p est le nombre de poids dans le réseau.

La norme L_2 ou Tikhonov régularisation : Le terme Tikhonov [118] essaye de minimiser la somme de la différence au carré entre les valeurs réelles et les valeurs prédites. Ce terme est plus rapide que le régularisation LASSO, cependant, il n'est pas capable de ramener à zéro de petits poids non pertinents. Le régularisation Tikhonov est présenté par:

$$\Omega_{L_1} = \sum_k w_i^2 \tag{1.3.14}$$

Le terme de la régularisation dans l'équation (1.3.13) et l'équation (1.3.14) ont également été des études concernant l'apprentissage bayésien, connu sous le nom de la régularisation de LASSO et la régularisation gaussien.

Considérant le paramètre λ, également appelé paramètre de régularisation, il contrôle le compromis entre la fluidité du réseau et l'erreur d'apprentissage. Cependant, la valeur du paramètre n'est pas significative et généralement fixée en fonction du choix de l'utilisateur. Ainsi, la spécification de λ devient un problème fondamental dans l'identification des paramètres régularisés. En d'autres termes, l'approche scalaire [85], [87] est appliquée pour optimiser l'erreur d'apprentissage et la complexité du réseau. Cette approche tente de réduire les problèmes d'optimisation multiobjectifs à un seul objectif, où il est plus facile de trouver un compromis entre plusieurs objectifs car cette approche permet d'utiliser les méthodes classiques d'optimisation. Portant cette technique a plusieurs faiblesses; la détermination de l'hyperparamètre approprié (λ) n'est pas un problème insignifiant. Le paramètre mentionné ci-dessus doit refléter correctement le but de l'utilisateur. Une telle localisation est peu fréquente, en particulier lorsque des choix arbitraires sont appliqués. Ainsi, en utilisant l'approche de scalarisation, nous limitons la variété des solutions car une seule solution peut être obtenue. En conséquence, la perspicacité du problème n'est pas entièrement explorée. Pour les principales raisons citées et d'autres, de nouvelles approches ont été envisagées. La section suivante couvre une nouvelle approche dite apprentissage multiobjectif.

1.3.5.3 Multiobjective Learning

L'apprentissage automatique est principalement multicritères. Puisque, dans l'apprentissage supervisé, plusieurs objectifs doivent être pris en considération. Le modèle d'apprentissage doit obtenir, et pas seulement, une performance d'approximation bénéfique concernant l'apprentissage des données, bien que sur des données invisibles du même problème. Compte tenu de la faiblesse mentionnée des méthodes classiques pour faire face à la multiobjectivité de l'apprentissage automatique, une nouvelle méthode a été exploitée, telle que l'optimisation multiobjectif [53].

Formulation du problème : Pour éviter l'utilisation d'un facteur de pondération, dans ce cas, l'hyperparamètre de régularisation λ, il est plus judicieux de trouver des solutions qui ne dépendent pas de la valeur de l'hyperparamètre en le supprimant entièrement de la formulation du problème. Il en résulte une formulation

multiobjectif et l'objectif du problème s'exprime ainsi:

$$f(w)^T = [E(w), \Omega(w)] \tag{1.3.15}$$

Le problème de régularisation formulé comme un problème d'optimisation multiobjectif se caractérise de manière concluante par:

- Le problème est multiobjectif,
- les fonctions objectives $f(x)$ sont continues mais peuvent être complexes comme non convexes,
- le problème est soumis à des contraintes d'égalité et des contraintes d'inégalité,
- l'espace de recherche est continu.

De même, la solution du problème est un ensemble de solutions appelé front de Pareto [84], qui représente les résultats de différents compromis. Les solutions sont obtenues en utilisant l'optimalité de Pareto, ce concept est utilisé pour calculer entre les solutions réalisables. Une fois l'apprentissage terminé, l'utilisateur doit choisir la solution la plus adéquate en fonction des performances de l'utilisateur, ce qui permet à ce dernier, d'atteindre une meilleure préférence lors de la sélection de la solution finale. En outre, l'approche basée sur Pareto peut épargner l'algorithme d'apprentissage des optimaux locaux.

Actuellement, plusieurs articles appellent l'approche multiobjectif pour traiter l'apprentissage MLPNN, le premier modèle proposé dans ce contexte a été par G. Liu et V. Kadirkamanathan [70], a proposé de considérer trois indices de performance différents comme fonctions objectives du modèle multiobjectif. De plus, Ricardo H. C. Takahashi et al [5] ont équilibré l'erreur de formation (MSE) et les poids de connexion pendant l'apprentissage pour améliorer la généralisation MLPNN Bien que dans [13], les auteurs ont proposé l'algorithme d'apprentissage Multi-OBJective (Multi-OBJective training algorithm (MOBJ)), composé des deux étapes, la première vise à trouver la solution Pareto-optimale et la seconde est une étape de décision pour choisir le modèle MLPNN adéquat, une des solutions de l'ensemble Pareto-optimal, à l'aide d'un ensemble de données de validation. En 2006, Costa, M. A., & Braga, A. P. [24] a proposé d'optimiser simultanément les performances de généralisation MLPNN et la sélection des poids par un algorithme prenant en compte "linear last absolute shrinkage and selection operator" (LASSO) régularisation. En outre, Jin, Y., & Sendhoff, B. [54] ont donné une revue des recherches actuelles sur l'apprentissage automatique multiobjectif, décrivant l'apprentissage supervisé. Dans le même contexte, nous devons considérer nos travaux précédents [57], [108], où nous avons introduit de nouvelles contraintes et une nouvelle variable de décision au modèle d'apprentissage MLPNN multiobjectif.

1.4 Approches neuro-metaheuristiques pour le problème de selection du portfeuille

La gestion de portefeuilles est un problème très important dans le domaine économétrique. Il donne la réponse à l'investisseur pour sa question difficile, comment répartir son capital entre de nombreux actifs? Le premier modèle était le modèle de Markowitz ou modèle à variance moyenne de 1952 [75] [51]. Markowitz a été le pionnier du premier traitement rigoureux du dilemme de l'investisseur. Dans ce chapitre, nous proposons de nouveaux modèles, en ajoutant une nouvelle contrainte aux modèles de Markowitz, pour éviter d'investir sur les actifs d'un rendement négatif. Compte tenu de son efficacité, un réseau continu de Hopfield est utilisé pour résoudre les modèles proposés. Dans ce regard, nous construisons une fonction énergétique originale qui fait un compromis entre les contraintes de risque, de profit et de cardinalité. Pour assurer la faisabilité du point d'équilibre, les paramètres de la fonction énergétique sont choisis en fonction d'une cohérence des résultats mathématiques; de plus, la pente des fonctions d'activation est choisie de telle sorte que le comportement de chaque neurone est presque linéaire. La méthode proposée produit de bonnes solutions. Cependant, la fonction d'énergie proposée semble être linéaire en un point en raison du comportement du paramètre d'aversion au risque. Pour éviter le dilemme mentionné, nous proposons une approche hybride basée sur le réseau de Hopfield continu (CHN) et l'algorithme génétique (GA). Ce dernier produit une meilleure solution comparée aux méthodes de la littérature, sur la base de données financières réelles.

1.4.1 Probléme de gestion du portfeuille

Le concept de gestion du portefeuille était utilisé bien avant le milieu du siècle dernier cependant la théorie du portefeuille est connu un éclat au début des années cinquante à la suite des travaux de Markowitz qui peut être considéré comme le père de la théorie moderne du portefeuille [22]. Avant ces travaux, les investisseurs avaient comme objectif de maximiser le rendement de leur portefeuille sans considération l'existence d'un risque. L'apport principal de Markowitz (1952) [75] a été d'introduire le risque dans le modèle de choix des titres d'un portefeuille. Ce choix est lié au concept de la diversification, l'investisseur pourra répartir ses investissements sur plusieurs titres au lieu de se contenter d'un seul. Ce qui lui permet d'atteindre un meilleur rapport rendement/risque.
Avant d'entamer la partie modélisation nous présentons quelques définitions reliées à la problématique traitée.

Un actif financier est un titre produisant à son propriétaire des revenus ou un gain en capital sur le marché financier. Cet actif produit une certaine prise de risque et peut-être transmis ou négocié sur le marché. Il est considéré comme un placement et il est comptabilisé dans le patrimoine de l'individu.

Un portefeuille est un ensemble d'actifs financiers détenus par un investisseur. Ces actifs peuvent provenir de différentes classes : actions, obligations, produits dérivés, matières premières, fonds, cash, etc. L'investisseur, pour diminuer son risque, procède souvent à une diversification de ses actifs. Ces derniers possèdent chacun une volatilité qui leur est propre et sont plus ou moins corrélés entre eux. La détention de plusieurs actifs différents tend donc, généralement, à diminuer la volatilité globale du portefeuille. Le degré de risque (nature des titres détenus, volatilité historique, diversification), le dynamisme de la stratégie (fréquence de réajustement de l'allocation) et le rendement obtenu sont des caractéristiques importantes, servant à comparer les portefeuilles boursiers entre eux.

1.4.1.1 Le rendement

La rentabilité est un concept fondamental en finance et apparaît dans l'expression de la plupart des modèles de gestion de portefeuilles.

Definition 1.4.1 *La rentabilité mesure l'appréciation (ou la dépréciation) relative de la valeur d'un actif financier ou d'un portefeuille d'actifs financiers entre deux instants successifs [4].*

Soit t et $t+1$ deux moments successifs. On note P_t et P_{t+1} les valeurs (prix) de l'actif aux temps respectivement t et $t+1$. La rentabilité exécutée dans l'intervalle $[t, t+1]$ peut être représentée par:

Rentabilité arithmétique Une valeur appelée rentabilité arithmétique toute rentabilité observée en $[t, t+1]$ la valeur définie par:

$$r_t = \frac{P_{t+1} - P_t}{P_t} \tag{1.4.1}$$

Rentabilité logarithmique Une valeur appelée rentabilité logarithmique toute rentabilité observée en $[t, t+1]$ la valeur définie par:

$$r_t = \ln(\frac{P_{t+1} - P_t}{P_t}) \tag{1.4.2}$$

De nombreux modèles financiers utilisent des séries de données historiques pour les cours des actions pour estimer les propriétés stochastiques de la rentabilité correspondante, comme la rentabilité moyenne. De plus, plusieurs méthodes sont disponibles. Celui-ci présenté est l'estimation des rendements de la sous période, puis la formule suivante est utilisée.

$$\overline{r} = \frac{1}{T}\sum_{1}^{T} r_t \tag{1.4.3}$$

L'hypothèse selon laquelle les rendements des actifs financiers sont normalement distribués est souvent avancée dans la littérature financière. Mais cela ne correspond pas à la réalité. En général, la rentabilité des marchés ne suit pas la distribution gaussienne. Dans ce cas, deux autres propriétés stochastiques entrent en jeu, ce sont l'asymétrie et le kurtosis. Dans cette section, nous considérons que les rendements boursiers possèdent des densités de probabilité normales et identiques.

1.4.1.2 Analyse du risque

Le risque peut être défini d'une manière quantitative en ayant recours à des mesures statistiques telles que l'écart-type ou les quantiles (notion de valeur à risque VaR). Ces mesures sont utilisées d'une manière systématique. Il existe bien un nombre d'outils constituant la base de la gestion du risque ayant démontré leur efficacité. Nous abordons ici les plus célèbres et utilisés d'entre eux :

Variance la variance, est une mesure classique du risque. Il est bien connu que Markowitz est le premier chercheur à utiliser la variance comme mesure du risque [75], [76], pour la première fois, le problème de la sélection de portefeuille a été clairement développé et résolu. Le modèle proposé était basé sur deux notions de probabilité, les rendements moyens et la variance des rendements. Ainsi, le modèle de Markowitz tente de partager la richesse entre les actifs tout en réduisant la variance du portefeuille.

Definition 1.4.2 *Soit f la fonction de densité d'une variable aléatoire r, on définit la variance de r par*

$$Var(r) = \sigma^2(r) := \int_{-\infty}^{\infty} (r - E(r))^2 f(r) dr \tag{1.4.4}$$

où E est l'espérance mathématique du rendement et r est une variable continue. Dans le cas où r est une variable discrète, la variance de r est définie par

$$Var(r) = \sigma^2(r) := \sum_{r} (r - E(r))^2 f(r) \tag{1.4.5}$$

Ou la somme est effectuée sur toutes les valeurs possibles de r.

Si r est le rendement du portefeuille, la variance sera la différence entre le rendement moyen et le rendement en cours. Supposons qu'une liste des retours historiques soit disponible. La liste contient les enregistrements historiques des retours de n actifs pendants m périodes. Compte tenu de la liste et de l'équité mentionnées, Equ. 4.2.3, nous définissons le rendement moyen de chaque actif en utilisant la formule suivante:

$$r_i := \frac{1}{m} \sum_{j=1}^{m} r_{ij} \tag{1.4.6}$$

Où r_{ij} est le rendement de l'actif i dans la période j tel que $i = 1, ..., n$ et $j = 1, ..., m$. Soit $x_1, x_2, ..., x_n$ les actifs n à risque qui composent les portefeuilles. De là, nous supposons que $x_i \geq 0$ et $\sum_{i=1}^{n} x_i = 1$ et que les variables x_i correspondent aux rendement aléatoire $r = (r_1, r_2, ..., r_n)^t$. Le rendement du portefeuille est alors $R = x^t r$ et le risque du portefeuille est défini par $\sigma^2 = x^t Q x$. telle que Q est la matrice de variance-covariance, où l'élément (i, j) est estimé par la formule suivante:

$$\sigma_{ij} = \frac{1}{m} \sum_{k=1}^{m} (r_{ik} - r_i)(r_{jk} - r_j) \tag{1.4.7}$$

L'utilisation de la variance comme mesure du risque a ses avantages. La simplicité peut être le premier avantage de cette mesure, d'ailleurs, en utilisant la matrice de variance-covariance, on peut construire un modèle de programmation quadratique convexe [65] pour calculer des portefeuilles efficaces. Cependant, l'approche du risque par la variance présente certains inconvénients. Étant donné que la variance pénalise de la même manière les gains et les pertes qui sont trop éloignés de la moyenne [65]. Dans ce contexte, des modèles alternatifs ont été développés pour remédier aux inconvénients du modèle Mean Variance model (MV).

Semi-variance En 1959, Markowitz réalise les limites de variance comme mesure de risquez, il propose alors la semi-variance. cette mesure ne prend en considération que les écarts inférieurs à un objectif. La mesure mentionnée est classée comme une mesure de risque à la baisse, cette classe de mesure des risques est également utile lorsque les rendements suivent une distribution non gaussienne. Mathématiquement, la mesure de semi-variance est définie comme suit [34]:

$$\Sigma_T := \sum_r ([r - T]^-)^2 f(r) \tag{1.4.8}$$

tel que T est la cible et f est la fonction de densité du rendement aléatoire. Markowitz suppose que T est le rendement moyen, $E(r)$, d'où le nom de semi-variance, dans ce contexte la formule 1.4.8 devient:

$$\Sigma_T := \sum_r ([r - E(r)]^-)^2 f(r) \tag{1.4.9}$$

La semi-variance ne prend en compte que les écarts associés aux valeurs de r qui sont inférieures à l'espérance de rendement. Cependant, la matrice de semi-covariance est plus difficile à définir, Hogan et warren [42] définissent la semi-covariance entre les actifs i et j comme:

$$\Sigma_{ij}^{HW} = E[(r_i - r_f).min(r_j - r_f, 0)] \tag{1.4.10}$$

Tel que HW indique qu'il s'agit de la définition proposée par Hogan et Warren et que r_f est le rendement attendu. Néanmoins, cette définition présente deux inconvénients:

- Le rendement de l'indice de référence est limité au taux sans risque et ne peut être adapté à aucun autre indice de référence souhaité;
- la matrice de semi-covariance est généralement asymétrique ($\Sigma_{ij}^{HW} \neq \Sigma_{ji}^{HW}$), cette caractéristique est particulièrement limitative. Par ailleurs, l'interprétation de la contribution des actifs i et j au risque de portefeuille n'est pas claire.

pour dépasser les caractéristiques précitées, Estrada [28] a proposé de définir la semi-variance entre les actifs i et j par rapport au benchmark T comme:

$$\Sigma_{ijB} = E[Min(r_i - B, 0).min(r_j - B, 0)] = \frac{1}{T}\sum_{t=1}^{T}[min(r_{it} - B, 0).min(r_{jt} - B, 0)] \tag{1.4.11}$$

Selon Estrada, cette définition peut être adaptée à tout B souhaité et génère une matrice de semi-variance symétrique et exogène, ou en utilisant la semi-variance comme mesure du risque, le risque du portefeuille peut être présenté comme:

$$\Sigma_{pB}^2 = \frac{1}{T}\sum_{t=1}^{T}[Min(R_{pt} - B, 0)]^2 \tag{1.4.12}$$

Où R_{pt} désigne les rendements du portefeuille et Σ_{pB}^2 leur semi-variance. En considérant la formule 1.4.12, la matrice de semi-covariance, selon Estrada, est endogène; c'est-à-dire qu'un changement de pondération affecte les périodes au cours desquelles le portefeuille est sous-performant par rapport à l'indice de référence, ce qui affecte à son tour les éléments de la matrice de semi-covariance. En 2008, Estrada a proposé une alternative heuristique [29] pour utiliser la mesure du risque de semi-variance supportant les caractéristiques de la variance. L'heuristique proposée permet de résoudre tous les problèmes de semi-variance moyenne avec les mêmes solutions bien connues de forme fermée largement utilisées pour résoudre les problèmes de variance moyenne. Plus précisément, la semi-variance d'un portefeuille par rapport à un indice de référence B peut être approximée par l'expression suivante:

$$\Sigma_{pB}^2 \approx \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \Sigma_{ijB} \tag{1.4.13}$$

Cette expression 4.2.13 renvoie une matrice de semi-variance symétrique et exogène, selon Estrada, la formule peut être utilisée de la même manière que la matrice de covariance utilisée dans les problèmes de variance moyenne.

1.4.1.3 Frontière efficiente

Chaque portefeuille qui permet le rendement attendu le plus élevé pour un niveau de risque donné, ou inversement, chaque portefeuille qui présente le risque le plus faible pour un rendement attendu donné.
L'ensemble des portefeuilles efficaces forme une frontière que Markowitz a appelée «frontière efficace». La figure (4.1) illustre la forme efficace de la frontière. Cependant, la sélection de portefeuille est un exemple d'une famille de problèmes d'optimisation multiobjectifs, visant à équilibrer le niveau de risque et le rendement du portefeuille. Cet équilibre est plus connu sous le nom de front de Pareto. De plus, Jarow [51] a proposé d'utiliser un paramètre λ appelé risque d'aversion pour atteindre l'équilibre tel que les deux objectifs (risque min et rendement max) sont combinés sur un modèle mono-objectif, cette technique est connue sous le nom de la méthode de somme pondérée [77]. Le modèle résultant est présenté comme suit:

$$\begin{cases} \min\limits_{x} & \lambda \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \sigma_{ij} & - & (1-\lambda)\sum_{i=1}^{n} x_i r_i & \\ \text{s.t.} & & & & \\ & \sum\limits_{i=1}^{n} x_i & = & 1 & \\ & x_i & \geq & 0 & \forall i \in 1,..,n \end{cases} \tag{1.4.14}$$

Où λ est un paramètre d'aversion au risque tel que $\lambda \in [0,1]$. En considérons $\lambda = 1$, la situation correspond à l'investisseur préoccupé du niveau de risque. Cependant, pour $\lambda = 0$, la situation convient à un investisseur n'ayant aucune considération au niveau risque. Ainsi, en varions les valeurs de $\lambda \in [0,1]$, nous pouvons construire une frontière efficiente.

1.4.2 Modèles rationnels proposés

Dans cette partie, nous nous intéressons au modèle de frontière efficace, car ce dernier est un modèle quadratique multiobjectif. Cependant, nous proposons d'ajouter une nouvelle contrainte à celui-ci visant à réduire le rendement négatif des actifs et à éviter l'influence négative dans le portefeuille, en plus d'éviter la perte de capital d'investissement. Le retour peut alors atteindre son maximum et le risque le plus bas. La contrainte s'exprime par:

$$\sum_{j \in I} x_j = 0 \tag{1.4.15}$$

Tels que I est l'ensemble des rendements négatifs des actifs formulés par:

$$I = \{j \mid r_j \leq 0\} \tag{1.4.16}$$

Le problème de sélection de portefeuille peut s'exprimer comme suit:

$$\begin{cases} \min\limits_{x} & \lambda\theta_r & - & (1-\lambda)\sum_{i=1}^{n} x_i r_i & & \\ \text{s.t.} & & & & & \\ & \sum\limits_{i=1}^{n} x_i & = & 1 & & \\ & \sum_{j \in I} x_j & = & 0 & I = \{j \mid r_j \leq 0\} & \\ & x_i & \geq & 0 & \forall i \in 1,..,n & \end{cases} \tag{1.4.17}$$

Tels que θ_r présentent la mesure du risque. Dans un premier temps, nous utilisons la variance comme mesure du risque pour résoudre le problème, puis nous avons adopté la semi-variance Estrada [29]. les modèles émergents sont appelés modèles rationnels. Néanmoins, le problème a été résolu avec différentes méthodes.

1.4.3 Résolution des problèmes à variables mixtes

Les problèmes d'optimisation à variables mixtes [48] sont ceux utilisant des produits de variables combinatoires et continues. Ceux-ci peuvent apparaître, par exemple, dans les problèmes de planification de production comme en problème de transport. Ces problèmes se caractérisent par leur difficulté de résolution puisqu'ils ont NP-difficile.

Plusieurs stratégies existent afin de résoudre tels problèmes. Notamment la relaxation, c'est une manipulation classique en optimisation combinatoire. Elle permet d'obtenir les bornes intéressantes (borne inférieure en minimisation et borne supérieure en maximisation). Nous pouvons distinguée deux types de relaxation :

- **Relaxation linéaire** ou la relaxation des contraintes, permet d'élargir l'ensemble des solutions sur un espace plus large et facile à optimiser, en relâchant les contraintes d'intégrité.

- **Relaxation Lagrangienne** consiste à modifier la fonction objectif de telle façon à intégrer les contraintes difficiles dans la fonction objectif sous forme de combinaisons linéaires, où les coefficients sont des multiplicateurs de Lagrange. Ces derniers pénalisent la fonction objectif si l'une des contraintes intégrées est violée.

Une fois le programme relâché résolu, nous pourrions arrondir aux valeurs entières les plus proches. Dans certains cas, cela peut fonctionner, mais l'exemple du sac à dos montre que ce n'est pas toujours vérifié. Cette méthode par arrondissement est même parfois désastreuse.
Une autre intéressante technique consiste à diviser le problème initial en un certain nombre de sous-problèmes, nommé **Branch & Bound** (méthodes de séparation et évaluation) [68]. La stratégie suit la technique diviser-pour-régner: à chaque itération, décomposer le problème en sous-problèmes plus simples pour déterminer l'optimum de l'ensemble des solutions réalisables associées au sous-problème en question, ou pour prouver que cet ensemble ne contient pas de solution optimale. Ensuite, combiner la résolution de ces sous-problèmes pour obtenir la solution du problème original, chaque sous-problème correspond à un sommet dans l'arbre des solutions. La méthode la plus utilisé pour l'étape de séparation est la relaxation du sous-problème courant. Pour ensuite à déterminer un minorant (majorant pour une maximisation) des solutions contenues dans l'ensemble. Si le minorant est supérieur à la valeur de la meilleure solution trouvée, cela affirme que le sous-ensemble ne contient pas l'optimum. Par contre, si les deux bornes supérieure et inférieure sont égales, alors une solution optimale est trouvée.
Cependant ces techniques ne sont applicables que pour les problèmes de forme linéaire, dans ce contexte, elles restent utilisables tant que le nombre de variables ne dépassent pas les vingtaines. Au-delà de cela, elles atteignent leurs limites. Dans un cas contraire, il est préférable de s'orienté vers des méthodes approché tel que les méta-heuristiques [127] et les méthodes hybrides [35], [55]. Celle-ci est capable de résoudre tout type de problème d'optimisation. Elles sont caractérisées par leur caractère stochastique, ainsi que par leur origine discrète. Les méta-heuristiques sont inspirées des analogies de la physique (recuit simulé, recuit microcanonique), de la biologie (algorithmes évolutionnaires) ou encore l'éthologie (colonies de fourmis, essaims particulaires). Cependant, elles peuvent demander plusieurs paramètres à régler, ceci peut être considéré comme le principal inconvénient de ces méthodes. Il est, aussi, à souligner que les métaheuristiques se prêtent à toutes sortes d'extensions, notamment en optimisation multiobjectif.
Dans les sections suivantes, nous proposons différentes approches pour le modèle rationnel proposé, en se basant sur une approche neuronal et des méta-heuristiques.

1.4.4 Réseau Hopfield continu pour les modèles rationnels

Dans cette partie, nous proposons un réseau adaptatif continu de Hopfield, détaillé dans la section (1.3.5), pour résoudre les modèles proposés représentant le problème de sélection de portefeuille. L'approche doit faire un compromis entre la fonction objectif et les contraintes. Pour cela, différentes étapes sont requis, l'adaptation complète proposée est détaillée ci-dessous:

1.4.4.1 Fonction d'énergie du réseau CHN

Pour résoudre le (PM) via le CHN, nous proposons la fonction d'énergie suivante:

$$E(x) = \alpha\frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \sigma_{ij} - \frac{\alpha}{2}(1-\lambda)\sum_{i=1}^{n} r_i x_i + \beta_1 \tau^t x + \beta_2 (\tau^t x)^2 + \varphi_1 \tau_I^t x + \varphi_2 (\tau_I^t x)^2 \tag{1.4.18}$$

Tel que

$$\tau^t = (1, ..., 1)^t \in \mathbb{R}^n \tag{1.4.19}$$

$$\tau_I^t = (\delta_1, ..., \delta_n)^t \in \mathbb{R}^n \tag{1.4.20}$$

En considérant:

$$\delta_i = \begin{cases} 1 & if \quad i \in I \\ 0 & else \end{cases} \tag{1.4.21}$$

$$I = \{ i \mid r_i \leq 0 \ \} \tag{1.4.22}$$

Les variables α, β_1, β_2, φ_1 et φ_2 sont les paramètres de pénalité de la fonction d'énergie CHN. Ainsi, le gradient de la fonction d'énergie CHN peut être exprimée par:

$$\nabla E(x) = \alpha\lambda Q x - (1-\lambda)\alpha r + \beta_1 \tau + \beta_2 \tau (\tau^t x) + \varphi_1 \tau_I + \varphi_2 \tau_I (\tau_I^t x) \tag{1.4.23}$$

à partir de la formule obtenue (Equ. 1.4.23), le biais est donné par:

$$-\nabla E(0) = \alpha(1-\lambda) r - \beta_1 \tau - \varphi_1 \tau_I \tag{1.4.24}$$

Mais nous avons aussi:

$$\nabla^2 E(x) = \alpha\lambda Q + \beta_2 \tau\tau^t + \varphi_2 \tau_I \tau_I^t \tag{1.4.25}$$

En se basant sur Equ. 1.4.25, la matrice de poids est donnée par:

$$T = -\alpha\lambda Q - \beta_2 \tau\tau^t - \varphi_2 \tau_I \tau_I^t \tag{1.4.26}$$

D'après les équations obtenu, la matrice de poids et le vecteur de biais dépendent des paramètres de pénalité; pour assurer la faisabilité du point d'équilibre, nous utilisons une méthode mathématique (voir dans la section 1.4.4.3) pour sélectionner les poids et biais adéquats.

1.4.4.2 Fonction d'activation

Comme le problème de portefeuille recherche des valeurs continues, la fonction d'activation doit être choisie de telle sorte que chaque valeur dans $[0,1]^n$ ait la même probabilité d'être prise. La figure (4.2) est une présentation de la fonction $f(x) = \arctan(Tx)$ pour plusieurs valeurs de la pente T sous $[a,b]$. La meilleure pente est celle qui dépense

- presque linéaire
- Presque surjectif sous $[a,b]$

Premièrement, nous devons définir l'ensemble $[a,b]^n$ qui contient toutes les valeurs possibles des états neuronaux notés U_i. Comme on le sait, l'état U_i est donné par $U_i = T_i x + I_i$ où T_i est la ligne vectorielle de la matrice T. Dans ce sens, vérifiez l'égalité suivante:

$$\sum_{j=1}^{n} min(0, T_{ij}) + I_i \leq U_i \leq \sum_{j=1}^{n} max(0, T_{ij}) + I_i \tag{1.4.27}$$

Soient $\underline{U}$ et $\overline{U}$ deux instants défendus par:

$$\underline{U} = min \left\{ \sum_{j=1}^{n} min(0, T_{ij}) + I_i \right\} \tag{1.4.28}$$

$$\overline{U} = max \left\{ \sum_{j=1}^{n} max(0, T_{ij}) + I_i \right\} \tag{1.4.29}$$

En ce sens, nous avons

$$\forall i \in \{1, ..., n\} \ \underline{U} \leq U_i \leq \overline{U} \tag{1.4.30}$$

Puisque la fonction $f(x)$ est une fonction croissante, nous avons:

$$\forall i \in \{1, ..., n\} \ f(\underline{U}) \leq f(U_i) \leq f(\overline{U}) \tag{1.4.31}$$

En conséquence, les paramètres a et b sont donnés par les formules:

$$a = f(\underline{U}) \ et \ b = f(\overline{U}) \tag{1.4.32}$$

En considérons les paramètres a et b, nous avons choisi la pente de la fonction d'activation, de telle sorte que celle-ci est presque plus linéaire et subjective dans l'intervalle $[a,b]$. Ce faisant, la fonction d'activation ne fait la promotion d'aucun élément.

1.4.4.3 Réglages de paramètres

L'étape de paramétrage vise à trouver le paramètre adéquat pour garantir la stabilité du réseau, car nous visons à minimiser la quantité:

$$\frac{1}{2}(1-\lambda)x^tQx - \lambda r^t x \tag{1.4.33}$$

Le paramètre α doit être strictement positif $\alpha > 0$. Si pour x_i il existe un neurone dans $i \in I$ tel que $x_i > 0$. Ensuite, $E_i = \frac{\delta E(x)}{\delta x_i}$ doit retirer: $E_i(x) < \epsilon$ ($\epsilon \in \mathbb{R}^{*+}$). Ainsi, la condition suffisante est:

$$(1-\lambda)\sum_{j=1}^{n} q_{ij}\alpha\lambda r_i + \beta_1 + \varphi_1\delta_i\varphi_2\delta_i\,|I| \leq \epsilon \tag{1.4.34}$$

Tel que

$$\varphi_2 > 0\, et\, \beta_1 < 0 \tag{1.4.35}$$

Ou la condition de liberté suffisante est:

$$\alpha(1-\lambda)\overline{\Sigma Q} - \alpha\lambda\underline{r} + \beta_1 + \varphi_1 + \varphi_2\,|I| \leq \epsilon \tag{1.4.36}$$

De telle manière que

$$\varphi_1 > 0\,; \overline{\Sigma Q} = min\sum_{j=1}^{n} q_{ij}\,; \underline{r} = minr_i \tag{1.4.37}$$

Finalement, toute solution dans le système (S) (Eq. 1.4.38) converge à un portefeuille réalisable.

$$(S)\begin{cases} \epsilon > 0 \\ \alpha > 0 \\ \beta_1 < 0 \\ \varphi_1 > 0 \\ \varphi_2 > 0 \\ \alpha(1-\lambda)\overline{\Sigma Q} - \alpha\lambda\underline{r} + \beta_1 + \varphi_1 + \varphi_2\,|I| \leq \epsilon \end{cases} \tag{1.4.38}$$

En considérant le deuxième modèle, le modèle rationnel du semi-variance et moyenne, nous construisons un réseau de Hopfield continu similaire en remplaçant la formule de variance par celle de semi-variance Estrada. Par conséquent, seule la matrice de poids est modifiée.

1.4.5 Algorithme génétique hybride pour le modèle de variance moyenne rationnelle

L'utilisation des algorithme génétique (GA) vise à améliorer les réponses CHN. Nous sommes convaincus qu'une telle combinaison donne de bons résultats. GA aidera le CHN à surmonter le problème de linéarité de la fonction d'énergie puisque quand le paramètre d'équilibre converge vers zéro, la fonction d'énergie, admise comme fonction quadratique, s'accorde avec une fonction linéaire. D'un autre coté, CHN accélérera l'AG en obtenant une solution adaptée à la population initiale.

1.4.5.1 Algorithme du AG

Les étapes de l'algorithme GA adaptées sont présentées comme suit:
Etape 1 : Initialisez la population initial par les solutions obtenu grâce au réseau de Hopfield continu
Etape 2: Evaluez la fitnesse les individus de la population
Etape 3: Sélectionnez les meilleurs solutions pour la production
Etape 4: Appliquez le croisement a deux point a 60% des individus sélectionné
Etape 5: Appliquez la mutation a 40% des individu choisi
Etape 6: Les enfants obtenus du la production passent par un processus de vérification vis à des contraintes. Si les solutions (enfants) obtenues ne vérifie pas les contraintes du problème, alors le processus nécessite une étape de correction. Sinon, passez directement à l'étape 7.
Etape 7: Remplacez la génération actuelle par la nouvelle génération.
Etape 8: Vérifiez les critères d'arrêt, si celle-ci sont atteintes alors arrêtez le processus, sinon passez à l'étape 2.
L'organigramme correspondant à l'AG hybride est présenté par la figure (4.3).

1.4.6 Adaptation de l'algorithme génétique

Comme l'algorithme génétique nécessite une adaptation au problème du portefeuille, nous présentons dans cette partie les différentes modifications utilisées pour adapter l'algorithme.
Codage de l'individu: Chaque individu généré est une solution potentielle au problème de portefeuille codé correspondant à la variable du problème: x_{ij} un vecteur continu se référant à la proportion de capital investi. La figure (4.4) représente un individu possible formé par plusieurs gènes, tel que chaque gène est produite dans l'intervalle $[0, 1]$. La population d'AG est souvent générée de manière aléatoire. Cependant, les solutions générées doivent satisfaire les contraintes du problème.
Fonction d'évaluation: La fonction d'évaluation d'un individu définit sa capacité à rivaliser avec d'autres individus de la population. La probabilité qu'un individu soit sélectionné pour la reproduction est basée sur son score. Étant donné que chaque problème a une fonction de fitness spécifique, nous proposons d'utiliser la fonction objective proposée du problème de portefeuille comme fonction d'évaluation, étant donné que ce dernier a un score continu qui ne posera aucun problème de mise en œuvre. Tels que l'individu avec une fonction d'évaluation minimal est plus approprié pour être la solution du problème.
Sélection: Un mécanisme de sélection est utilisé pour sélectionner les individus qui serviront de parents pendant la production de la prochaine génération. Dans ce contexte, plusieurs méthodes ont été proposées [52]. Cependant, nous proposons d'utiliser la sélection de tournois [40] pour sélectionner les parents. Cette méthode est une variante des méthodes de sélection basées sur le classement. Il consiste à sélectionner aléatoirement un ensemble de k-individus, puis les solutions de k-choix sont triées en fonction de leur aptitude relative et l'individu le plus pertinent est

sélectionné pour la reproduction. Ce processus est répété jusqu'à ce que la proportion de sélection soit atteinte. Par conséquent, la probabilité de sélection de chaque individu peut être présentée par:

$$p(i) = \begin{cases} \frac{C_{n-1}^{k-1}}{C_n^k} & sii \in [1, n-k-1] \\ 0 & sii \in [n-k, n] \end{cases} \tag{1.4.39}$$

Croisement: Le rôle de l'opérateur de croisement est de fournir de différents individus qui convergent vers un sous-espace. l'opérateur recombine les informations de deux bons «parents» en deux nouveaux individus appelés enfants. Comme la population sélectionnée est divisée en deux parties, selon la probabilité de croisement p_c, une partie est choisie pour passer par l'opérateur de croisement. Dans la littérature, plusieurs représentations possibles [88] ont été proposées, nous avons choisi d'utiliser la méthode "position listing représentation pour accomplir le croisement. Ce dernier, choisit au hasard deux points de croisement et échange tous les gènes entre les points de croisement. La figure (4.5) représente l'opérateur de croisement utilisé.
Mutation: l'objectif principal de l'utilisation d'un opérateur de mutation consiste à maintenir la diversité de la population ainsi qu'à empêcher la convergence prématurée et les optima locaux. L'opérateur génère une population de descendants en changeant au hasard un ou plusieurs gènes chez un individu sélectionné. La figure (4.6) décrit le mécanisme de mutation utilisé.
Correction: Étant donné que les opérateurs de production créent la population de la nouvelle génération, il est important de vérifier la qualité des solutions produites. Chaque nouveau individu devrais passé par une vérification des contraintes, si l'individu produit ne respecte pas la contrainte du problème, une étape de correction est nécessaire. Par conséquent, nous proposons une heuristique adaptative pour l'étape de correction GA, répondant aux contraintes du problème de portefeuille. L'algorithme (5) présente l'heuristique de correction proposée.

1.4.7 Hybride multiobjectif algorithme génétique pour le modèle de semi-variance rationnelle moyenne

Comme nous recherchons une solution de problème qui équilibre entre le niveau de risque et la rentabilité du modèle proposé, le modèle vise à répondre à plusieurs objectifs, nous pouvons considérer une modélisation multiobjective [25]. Pour résoudre un tel problème, de nombreuses méthodes ont été proposées et classées, telles que les méthodes d'agrégation pondérée, les méthodes non agrégées non Pareto et l'approche basée sur Pareto (voir la section 1.2 pour plus de détails). La première section examine une approche d'agrégation pondérée utilisant le paramètre d'aversion au risque (λ). Dans cette partie, nous proposons d'utiliser une méthode basée sur "Pareto" pour améliorer les solutions obtenues et éviter l'influence du paramètre d'aversion au risque. Ainsi, nous appliquons un GA multiobjectif développé pour résoudre des problèmes d'optimisation multiobjectifs. Diverses variantes MGA ont été développées comme VEGA [101] (méthodes non agrégées non Pareto), MOGA [30], NSGA

[113], Niched Pareto Genetic Algorithm (NPGA) [46] basé sur Pareto); Strength Pareto Evolutionary Algorithm (SPEA) [131], Pareto Archived Evolution Strategy (PAES) [62], Pareto envelope-based Selection Algorithm (PESA) [23] NSGA II [26] (élitiste de Pareto). Comme les contraintes du problème sont linéaires, nous pouvons utiliser l'algorithme génétique multiobjectif du solveur "Matlab", l'option «gamultiobj» est alors utilisée. Cette fonction tente de créer un ensemble de Pareto optimal pour un problème multiobjectif. Vous pouvez éventuellement définir des limites et des contraintes linéaires sur les variables. «Gamultiobj» utilise l'algorithme génétique pour trouver des optimaux de Pareto locaux. Comme dans la fonction «ga», vous pouvez spécifier une population initiale ou demander au solveur d'en générer une automatiquement. Cette fonction prend en compte les options suivantes:

- N'admet pas les contraintes non linéaires, donc sa syntaxe a moins d'entrées,
- Prend en option «DistanceMeasureFcn», une fonction qui attribue une mesure de distance à chaque individu pour ses voisins,
- Prend en option "ParetoFraction", un nombre compris entre 0 et 1 qui spécifie la fraction de la population sur la meilleure frontière de Pareto à conserver lors de l'optimisation. S'il n'y a qu'une seule frontière de Pareto, cette option est ignorée,
- Utilise uniquement la fonction de sélection de tournoi,
- Utilise les individus d'élite, différemment de «ga», il ne trie aucun individu au-dessus des celle inférieurs, il utilise donc automatiquement les individus d'élite,
- N'a qu'une seule fonction hybride: "fgoalattain",
- Ne prend pas en charge une limite de temps de décrochage,
- Possède différentes fonctions de présentations graphiques disponibles,
- Ne prend pas en charge un choix de fonction de mise à l'échelle.

Durant la résolution du modèle nous utilisons les différentes solutions obtenues du réseau de Hopfield continu (en variant les valeurs de $\lambda \in [0, 1]$) comme population initiale de l'algorithme. Cette hybridation permettra d'accélérer l'algorithme d'un côté et d'un autre d'atteindre des zones non atteintes de l'espace de recherche vu que la population initiale est composée de solution considérée optimale.

1.4.8 Résultats expérimentaux

Pour souligner l'efficacité d'une mesure de risque choisie, le problème de sélection de portefeuille est résolu à l'aide de différentes mesures. Premièrement, en utilisant la variance; ensuite, la semi-variance de Estrada est requise comme mesure du

risque du portefeuille rationnel. Comme cette mesure a une forme symétrique, la matrice de poids de Hopfield est également symétrique; par conséquent, la stabilité du réseau neuronal proposé est garantie [45]. Les résultats du réseau de Hopfield continu sont ensuite éjectés en tant que population initiale de l'algorithme génétique adapté, pour le modèle rationnel de variance moyenne; de plus, les résultats de semi-variance modèle sont améliorés par un algorithme génétique multiobjectif. Les données utilisées sont de vraies données financières de la base de données yahoo finance [126]. Les données comprennent trois portefeuilles, chacun comprenant 20 actifs sur 20 périodes; l'étude couvrira la période du 29 septembre 2014 au 17 février 2015 par période hebdomadaire.

1.4.8.1 Résultats du modèle rationnelle (moyenne, variance)

D'après les résultats que nous avons obtenus, le réseau de Hopfield continu a pu converger vers un état de stabilité. Le réseau a été lancer plusieurs fois pour différentes $\lambda \in [0, 1]$. Enfin les meilleurs résultats ont été gardés (le tableau 4.1 présente les résultats du réseau CHN). Malgré la performance des résultats, nous avons considéré un processus de comparaison avec l'algorithme génétique dans le but de confirmer nos observations. Dans un premier temps, nous avons observé les niveaux de risque obtenu par chacun des optimisateurs utilisés (CHN et GA) pour un rendement fixé (pour plus de détail, veuillez consulter la section (4.7.1.1). Le résultat de nos observations confirme la supériorité du réseau de Hopfield continu, car ce dernier obtient de niveau de risque minimal pour un rendement fixé, comparé aux résultats de l'algorithme génétique. La seconde partie du processus consiste à observer les variations du rendement obtenu par chaque optimisateur pour un niveau de risque fixé (voir section 4.7.1.1 tableau 4.3). Cette fois-ci, nos observations ne sont pas très prononcées. Étant donnée que pour un niveau de risque fixé, l'algorithme génétique obtient de meilleurs rendements. Après réflexion, nous pouvons expliquer un tel résultat par le comportement du paramétré λ. Celui-ci, converge vers 0, si nous souhaitons favorisé le critère du rendement, ainsi, exclure le niveau de risque de l'équation. De telle action, produit un modèle linéaire (vu que le rendement est une fonction linéaire). Dans ce cas la fonction d'énergie du réseau est perturbée. Ce qui a réduit la performance du réseau. Sous ses conditions, et à fin d'y remédier à cette problématique, nous proposons d'améliorer les résultats du CHN en les injectons comme population initiale de l'algorithme génétique.

L'algorithme génétique hybridé produit meilleurs résultats (voir tableau 4.5). Nous pouvons remarquer que pour des petites valeurs de λ le niveau de risque est élevé cependant le rendement est plus bénéfique. contrairement quand celle-ci prend de plus grandes valeurs. Ce processus, varier les valeurs de λ, permet de construire la frontière efficiente (figure 4.8). Ainsi, nous proposons d'opter pour une comparaison graphique de la frontière du CHN vise celle-ci obtenu par GA hybridé. Nous remarquons de la figure (4.10) que la frontière du CHN est positionnée au-dessous de celle de l'algorithme génétique hybridé (voir section 1.2). Nous pouvons conclure que l'approche proposée produit de meilleurs résultats. La stratégie d'éjecter les solutions du réseau de Hopfield continu comme population initiale de l'algorithme

génétique a permis d'accélérer l'algorithme GA. Ainsi d'élargir l'espace de recherche. D'un autre côté, le réseau de Hopfield continu est de meilleure performance que GA tant que la fonction objective est quadratique, l'utilisation de l'algorithme génétique a permis d'éviter la linéarité de celle-ci et, ainsi, améliorer les résultats du CHN.

1.4.8.2 Résultats du modèle rationnelle (moyenne, semi-variance)

Concernant le modèle rationnel (moyen, Semi-variance), nous utilisons la même procédure pour different valeur de λ dans l'intervalle $[0, 1]$. Le réseau de Hopfield continu a réalisé de satisfaisants résultats (Fig. 4.11) . À notre connaissance, l'utilisation de la mesure semi-variance d'Estrada est très limitée. Pour cette raison, nous tenons de comparer les résultats obtenus par le réseau de Hopfield continu à celle-ci obtenu pour le modèle (moyen, variance). La figure (4.12) illustre la performance de chacun des systèmes pour un rendement fixé. Il est clair que la mesure semi-variance est plus efficace. Cependant, de la figure (4.11), nous constatons que la linéarité de la fonction d'énergie du CHN semble persister. Cette fois-ci, nous proposons d'éviter l'utilisation du paramètre λ. Ainsi, nous proposons d'utiliser une approche de résolution basée sur le principe de "Pareto".

La nouvelle approche consiste à commencer la résolution aux points d'arrêt du réseau de Hopfield continu. Les solutions sont éjectés à un processus basé Pareto, qui permet d'obtenir toutes solutions jugées (en utilisant le principe de dominance) équilibrant le rendement du portefeuille a son niveau de risque. Lafigure (4.13) présente la frontière efficiente (front de Pareto) obtenu. Il faut rappeler qu'il est difficile de choisir parmi les solutions de la frontière, c'est au décideur de choisir la solution qui le convient. Dans ce cas s'est à l'investisseur d'exprimer ses préférences via la solution choisie. Ainsi, nous nous contenterons de comparer l'approche multiobjectif au réseau de Hopfield continu via leurs frontières efficient.

Une lecture, plus profond, de la figure (4.14), permettent de conclure. L'approche multiobjectif, non seulement, permet d'obtenir une frontière plus définie mais aussi de meilleures solutions. Vu que pour le même niveau de risque l'approche multiobjectif obtient un meilleur rendement.

1.5 Multiobjectif approche pour l'apprentissage du percetron multicouche

La présente section propose de nouvelles approches pour optimiser le réseau neuronal: perceptron multicouche ($MLPNN$), pour faire face au problème de généralisation. Comme on le sait, la plupart des algorithmes d'apprentissage supervisé visent à minimiser l'erreur d'apprentissage. Cependant, les méthodes mentionnées, basées uniquement sur la minimisation des erreurs, peuvent générer une solution avec des performances de généralisation insuffisantes. Notre travail propose deux modelés d'optimisation multiobjectif impliquant deux objectifs: précision et complexité car le problème d'apprentissage est par nature multiobjectif. La tâche d'apprentissage se

poursuit en minimisant les deux objectifs simultanément, selon le concept de domination de Pareto, en utilisant $NSGAII$ (Non-dominated Sorting Genetic Algorithm II) comme solveur. Cette méthode nous conduit à un ensemble de solutions, front de Pareto, étant l'ensemble de solutions optimal, le $MLPNN$ adéquat doit être extrait. Nous montrons empiriquement que la méthode proposée est capable de réduire la topologie du réseau neuronal et d'améliorer les performances de généralisation, en plus d'un meilleur taux de classification par rapport aux différentes méthodes.

1.5.0.1 Apprentissage du percetron multicouche

l'apprentissage du perceptron multicouche (MLPNN) a été largement discuté ces dernières années. Respectivement, plusieurs approches ont été proposées telles que l'apprentissage classique [99] ou les méthodes d'optimisation [93] [2] ainsi que leur efficacité a été largement discutée. Alors pourquoi converger vers une approche multiobjective basée sur le principe de Pareto? Une fois que différents objectifs doivent être optimisés, ces algorithmes s'avèrent impuissant, car un seul objectif peut être minimisé directement et peut produire un MLPNN sur-ajusté. L'approche classique semble être faible pour gérer les objectifs compétitifs dans l'apprentissage [79]. Ce dernier ne produit pas nécessairement de bons modèles de généralisation [72], ses méthodes présentent plusieurs inconvénients, comme une lente convergence, le problème des minima locaux et l'incertitude de sélection du modèle. pour traiter ce qui précède, les méthodes classiques ont ajouté, pour équilibrer les deux objectifs, un paramètre de pénalité (λ) à la fonction objectif du modèle. cette technique est connue sous le nom de régularisation du réseau [16].

Malheureusement, le paramètre mentionné devient problématique. Au lieu de cela, nous proposons d'utiliser une approche multiobjectifs basée sur le concept de Pareto [26]. L'un des avantages importants de l'approche peut être présenté par le nombre de solutions optimales de Pareto, ce précédent peut atteindre un grand nombre de solutions appelées front de Pareto, à partir desquelles l'utilisateur est capable de dériver les informations sur le problème et de prendre une décision améliorée unique. De là, le compromis (apprentissage-complexité), en utilisant la méthodologie basée sur Pareto, a été examiné. À cet égard, les recherches antérieures sur l'apprentissage supervisé par concept basé sur Pareto peuvent généralement être divisées en trois catégories en fonction de leurs motivations, telles que la génération de modèles interprétables ou la sélection de modèles pour la généralisation et la génération d'ensemble.

Dans cette section, nous proposons une nouvelle approche pour sélectionner le bon modèle pour une meilleure généralisation du MLPNN L'objectif proposé est de régulariser la complexité du réseau pendant l'apprentissage du MLPNN, ainsi, trouver le compromis entre précision et complexité du réseau. De plus, nous proposons une méthodologie multiobjectifs comprenant deux objectifs: minimiser l'erreur générale du perceptron et les termes de régularisation choisis supportés par une variable de décision, sous des contraintes représentant la communication entre les couches du réseau. Ainsi, nous résolvons les modèles émergents en utilisant NSGA II (Non-dominated Sorting Genetic Algorithm II) [128] l'un des algorithmes

génétiques multiobjectifs les plus efficaces.

1.5.0.2 Modèle d'apprentissage multiobjectif du perceptron multicouhce

L'utilisation d'un modèle multiobjectif en apprentissage automatique est, en effet, une idée essentielle, car nous recherchons un compromis entre précision et complexité. Comme raisons pour lesquelles une topologie inadaptée augmente le temps d'apprentissage et peut entraîner une non-convergence. Principalement, les raisons mentionnées sont généralement responsables de l'insuffisance des performances du MLPNN [110]. En général, une fois la complexité réduite, le modèle est plus contrôlé et le sur-ajustement est évité. Par conséquent, nous proposons d'optimiser, en plus de l'erreur d'apprentissage du MLPNN, deux termes de régularisation différents: un LASSO modifié et la norme $L_{1/2}$ modifiée pour régulariser la complexité du réseau. Les régularisations mentionné permettrons un meilleur contrôle sur les valeurs de poids, en outre, cette méthode produira une topologie MLPNN appropriée. Puisqu'il est un critère important influençant l'efficacité du MLPNN. Les modèles soutiennent l'introduction d'une nouvelle variable de décision dans la formule des objectifs, qui joue le rôle d'une technique d'élagage (pruning technique), visant à réduire les neurones non fonctionnels dans l'architecture du MLPNN Cette technique permettra une grande élimination des poids, ce qui permettra d'atteindre d'excellents résultats. De plus, les contraintes du modèle imposent une communication importante entre les couches.

En se basant sur les travaux de O. Ludwig et U. Nunes [71], il a été constaté qu'une seul couche cachée est généralement suffisante pour la plupart des problèmes. Pour cette raison spécifique, le réseau étudié est composé d'une couche cachée individuelle. Les modèles proposés seront expliqués dans les sections suivantes.

Notation :
N : La taille de la couche caché.
n_0 : La taille de l'entrée du réseau.
n : La taille de la couche de sortie.
X : Données d'entrée du réseau neuronal.
Y : Sortie calculée du réseau.
d : La sortie désiré.
f : Fonction d'activation.
h : La sortie de la couche caché.
w : La matrice poid du réseau.

C_{ij}^k : Variable binaire tel que:

$$C_{ij}^k = \begin{cases} 1 \text{ si } \textit{la connexion entre le neurone } i \textit{ et } j \textit{ est gardée} \\ 0 \text{ si non} \end{cases} \tag{1.5.1}$$

k : est l'indice de connexion entre deux couches successives.
U_i : Variable binaire tel que:

$$U_i = \begin{cases} 1 \text{ si } \textit{le neurone i est gardé dans la couche caché} \\ 0 \text{ si non} \end{cases} \tag{1.5.2}$$

Fonction objectif :
De toute évidence, la première fonction objectif est la fonction d'apprentissage du perceptron multiobjectif, présenté par l'erreur quadratique moyenne (MSE), telle que :

$$E\left(w, C, U\right) = \frac{1}{2J} \sum_{j=1}^{J} ||F_j\left(U, C, X, w\right) - d_j||^2 \tag{1.5.3}$$

J est la taille de la base d'apprentissage. Concernant le deuxième objectif; la complexité du réseau; nous proposons deux fonctions différentes. La première est la régularisation LASSO modifié et la deuxième est représenté par la régularisation $L_{1/2}$.

Régularisation LASSO modifié :
Le modèle proposé vise à augmenter la capacité d'apprentissage du réseau avec une complexité de réseau réduite. Cependant, plusieurs travaux ont proposé une réalisation de complexité différente, dans cette partie, nous proposons de présenter cette dernière par un régularisation Lasso modifié. Ainsi, le modèle est développé.

Calcul de la sortie de la couche caché: La couche cachée du perceptron multicouche est directement connectée à la couche d'entrée, de sorte que la suppression des neurones de la couche cachée et les poids reliant la couche d'entrée à la couche cachée sont représentés dans la sortie de chaque neurone par:

$$h = \begin{pmatrix} h_1 \\ \vdots \\ h_i \\ \vdots \\ h_N \end{pmatrix} = \begin{pmatrix} U_1 \left(1 - \prod_{j=1}^{n_0} \left(1 - C_{j1}^0\right)\right) f\left(\sum_{j=1}^{n_0} C_{j1}^0 w_{j1}^0 x_j\right) \\ \vdots \\ U_i \left(1 - \prod_{j=1}^{n_0} \left(1 - C_{ji}^0\right)\right) f\left(\sum_{j=1}^{n_0} C_{ji}^0 w_{ji}^0 x_j\right) \\ \vdots \\ U_N \left(1 - \prod_{j=1}^{n_0} \left(1 - C_{jN}^0\right)\right) f\left(\sum_{j=1}^{n_0} C_{jN}^0 w_{jN}^0 x_j\right) \end{pmatrix} \tag{1.5.4}$$

Calcul de la sortie du perceptron multicouche: La sortie du réseau est calculée en utilisant les poids reliant la couche cachée à la couche de sortie car, à ce stade, seules

les connexions sont élaguées, de sorte que la sortie du réseau est calculée par:

$$F(U,C,X,w) = \begin{pmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_n \end{pmatrix} = \begin{pmatrix} \left(1 - \prod_{j=1}^{N}\left(1 - C_{j1}^{1}\right)\right) f\left(\sum_{j=1}^{N} C_{j1}^{1} w_{j1}^{1} h_j\right) \\ \vdots \\ \left(1 - \prod_{j=1}^{N}\left(1 - C_{ji}^{1}\right)\right) f\left(\sum_{j=1}^{N} C_{ji}^{1} w_{ji}^{1} h_j\right) \\ \vdots \\ \left(1 - \prod_{j=1}^{N}\left(1 - C_{jn}^{1}\right)\right) f\left(\sum_{j=1}^{N} C_{jn}^{1} w_{jn}^{1} h_j\right) \end{pmatrix} \tag{1.5.5}$$

Fonction objectif de complexité du réseau: La complexité du réseau est présentée par la régularisation LASSO modifié [33], ainsi, nous introduisons une variable binaire dans la formule connue. Cette présentation vise à réduire le nombre de poids du réseau afin de contrôler les variations de poids lors de l'apprentissage pour une meilleure généralisation. La formule résultante est présentée comme suit:

$$\Phi_1(w,C,U) = \sum_{i=0}^{n_0}\sum_{j=0}^{N}\left|C_{ij}^{0} w_{ij}^{0}\right| + \sum_{i=0}^{N}\sum_{j=0}^{n}\left|C_{ij}^{1} w_{ij}^{1}\right| \tag{1.5.6}$$

Contraintes du modèle: La première contrainte garantit l'existence d'une couche cachée par l'existence d'au moins, un neurone en elle, la contrainte est exprimé par:

$$\sum_{i=1}^{N} U_i \geq 1 \tag{1.5.7}$$

Puisque nous supprimons les connexions et les neurones du réseau, la communication entre eux doit être garantie. Un neurone, qui n'a pas de connexions, doit être supprimé de l'architecture du réseau. De même, lorsque nous supprimons un neurone, nous devons également supprimer ses connexions. La contrainte s'exprime donc par:

- La communication entre la couche d'entrée et la couche cachée:

$$U_j \prod_{i=1}^{n_0}\left(1 - C_{ij}^{0}\right) = 0 \qquad \forall i = 1, ..., N \tag{1.5.8}$$

- La communication entre la couche cachée et la couche de sortie

$$U_i \prod_{j=1}^{n}\left(1 - C_{ij}^{1}\right) = 0 \qquad \forall i = 1, ..., N \tag{1.5.9}$$

Modèle obtenu: Le problème d'optimisation du perceptron multicouche est formulé

comme:

$$\begin{cases} \min\ E(w,C,U) \\ \min\ \Phi_1(w,C,U) \\ S.c \\ \sum_{i=1}^{N} U_i \geq 1 \\ U_j . \prod_{i=1}^{n_0} \left(1 - C_{ij}^0\right) = 0 \qquad \forall j = 1 \dots N \\ U_i . \prod_{j=1}^{n} \left(1 - C_{ij}^1\right) = 0 \qquad \forall i = 1 \dots N \\ w = \left(w_{ij}^k\right)_{\substack{k=\{1,2\} \\ 1 \leq i \leq N \\ 1 \leq j \leq n}} \quad telque \quad w_{ij}^k \in \mathbb{R} \\ C = \left(C_{ij}^k\right)_{\substack{k=\{1,2\} \\ 1 \leq i \leq N \\ 1 \leq j \leq n}} \quad telque \quad C_{ij}^k \in \{0,1\} \\ U = (U_i)_{1 \leq i \leq N} \quad telque \quad U_i \in \{0,1\} \end{cases} \tag{1.5.10}$$

régularisation "Non-Smooth $L_{1/2}$" :
Dans cette section, nous présenterons un nouveau modèle multiobjectif basé sur une régularisation "Non-Smooth $L_{1/2}$" tel que celui-ci présent plus tard la complexité du réseau. Ainsi, de nouvelles contraintes sont introduites. plus de détails sont donnés dans la section.

La sortie de la couche cachée: Compte tenu de la structure du réseau, l'entrée réseau est directement connectée à la seule couche cachée par le perceptron. Concernant le processus de suppression proposé (poids et neurones), l'équation de sortie 1.5.5 est conservée.

La sortie du perceptron multicouche: A ce stade, seule l'élimination des poids est prise en compte car il faut éviter de supprimer les sorties du réseau. Pour cette raison, la sortie est calculée par:

$$F(U,C,X,w) = \begin{pmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_n \end{pmatrix} = \begin{pmatrix} f\left(\sum_{j=1}^{N} C_{j1}^1 w_{j1}^1 h_j\right) \\ \vdots \\ f\left(\sum_{j=1}^{N} C_{ji}^1 w_{ji}^1 h_j\right) \\ \vdots \\ f\left(\sum_{j=1}^{N} C_{jn}^1 w_{jn}^1 h_j\right) \end{pmatrix} \tag{1.5.11}$$

Présentation de la complexité du réseau: Dans cette partie, la complexité du réseau est présentée par la régularisation $L_{1/2}$ [124], [125] modifié, qui est une pénalité non convexe et possède de nombreuses propriétés prometteuses. En se basant sur les

expériences [121], $L_{1/2}$ permettent un meilleur élagage que la décroissance du poids [67] et l'élimination du poids [122]. La formule correspondante est:

$$\Phi_2(w,C,U) = \sum_{j=1}^{N} U_j \left(\sum_{i=1}^{n_0} C_{ij}^0 \left| w_{ij}^0 \right|^{\frac{1}{2}} + \sum_{k=1}^{n} C_{jk}^1 \left| w_{jk}^1 \right|^{\frac{1}{2}} \right) \tag{1.5.12}$$

Contraintes du modèle: Comme toujours, nous cherchons à éviter l'élimination totale de la couche cachée, il est nécessaire de garder au moins un neurone dans la couche cachée. La contrainte 5.2.7 est conservée dans le modèle. Cependant, nous proposons de nouvelles contraintes pour maintenir la communication entre les couches. Étant donné que le réseau passe par un processus d'élagage aléatoire. De plus, les neurones sans relation avec les couches doivent être supprimés de l'architecture. Les contraintes s'expriment donc par:

- La famille de contraints suivante exprime la suppression complète de toute connexion entre le neurone supprimé et les couches d'entrée-sortie:

$$(1-U_j)\left(\sum_{i=1}^{n_0} C_{ij}^0 + \sum_{k=1}^{n} C_{jk}^1 \right) = 0 \qquad \forall j = 1 \ldots N \tag{1.5.13}$$

- Cependant la communication entre les connexions supprimées et le neurone associé est assurée par:

$$U_j . \left(\prod_{i=1}^{n_0} \left(1 - C_{ij}^0\right) . \prod_{k=1}^{n} \left(1 - C_{jk}^1\right) \right) = 0 \qquad \forall j = 1 \ldots N \tag{1.5.14}$$

modèle d'apprentissage multiobjectif: Le modèle d'apprentissage multiobjectif du perceptron multicouche (MLPNN) peut être exprimé par:

$$\begin{cases} \min E(w,C,U) \\ \min \Phi_2(w,C,U) \\ S.c \\ \sum\limits_{i=1}^{N} U_i \geq 1 \\ (1-U_j)\left(\sum\limits_{i=1}^{n_0} C_{ij}^0 + \sum\limits_{k=1}^{n} C_{jk}^1 \right) = 0 \qquad \forall j = 1 \ldots N \\ U_j . \left(\prod\limits_{i=1}^{n_0} \left(1 - C_{ij}^0\right) . \prod\limits_{k=1}^{n} \left(1 - C_{jk}^1\right) \right) = 0 \qquad \forall j = 1 \ldots N \\ w = \left(w_{ij}^k \right)_{\substack{k=\{0,1\} \\ 1 \leq i \leq \{n_0, N\} \\ 1 \leq j \leq \{N, n\}}} \quad telque \quad w_{ij}^k \in \mathbb{R} \\ C = \left(C_{ij}^k \right)_{\substack{k=\{0,1\} \\ 1 \leq i \leq \{n_0, N\} \\ 1 \leq j \leq \{N, n\}}} \quad telque \quad C_{ij}^k \in \{0,1\} \\ U = (U_i)_{1 \leq i \leq N} \quad telque \quad U_i \in \{0,1\} \end{cases} \tag{1.5.15}$$

1.5.1 Adaptation de l'algorithme NSGA II

NSGA II est un algorithme évolutif multiobjectif proposé par Deb et al [26], il a été classé comme l'un des algorithmes évolutifs multiobjectifs les plus efficaces, en outre, NSGA II possède les caractéristiques requises pour une bonne AG multiobjectif pour sa méthode de sélection. L'algorithme utilise une heuristique pour évaluer les éléments nuisibles du même front, appelé distance d'entassement (Crowding distance) [64], comme celui situé dans une zone moins encombrée. La sélection de tournois bondée garantit la sélection du meilleur classement possible.

Étant donné que nous optimisons deux modèles multiobjectifs différents en utilisant NSGA II comme solveur, la section est distribuée en trois parties, la première est dédié au partie similaire des deux modèles, ainsi les parties restantes discuteront des détails spécifiques de chaque adaptation.

1.5.1.1 Générale Adaptation du NSGA II

Puisque la méthode utilisée est basée sur le processus de l'algorithme génétique, cette partie considère l'adaptation NSGA II commune des modèles proposés. Ainsi, étant donné que les modèles traitent le même problème, l'adaptation prend plusieurs parties communes. les opérateurs suivants sont utilisés pour les modèles considérés.

Codage d'individu. L'étape de codage est très importante; chaque individu dans la population simule une structure MLPNN possible au problème, la figure (5.1) présente un exemple d'un individu et la structure MLPNN correspondante. Le codage est proposé en fonction des variables problématiques qui peuvent être un chromosome entier, continu, binaire ou mixte. Compte tenu des modèles proposés, un seul individu contient trois chromosomes présentant les trois variables problématiques: W_{ij}^k matrice de poids contenant des variables continues, C_{ij}^k matrice de connexions contenant des variables binaires et U_i décision des neurones vecteur; ainsi l'individu NSGA II est structuré comme figure (5.2). Ainsi, chaque individu simule une structure du perceptron multicouche, la figure (5.3) présente un exemple d'un individu et de la structure MLPNN correspondante.

Population initial. La population initiale du AG est le bain des solutions initiale, elle est souvent réalisée aléatoirement, nous adoptons la même procédure. Cependant, l'individu généré doit satisfaire les contraintes du problème. Le vecteur U_i et la matrice C_{ij}^k sont binaires; De plus, W_{ij}^k la matrice de poids contient des variables continues, elles sont générées dans l'intervalle [-0.5,0.5].

Fonction efficacité. La population d'individus est évaluée selon la théorie de la domination de Pareto. Ainsi, chaque individu est caractérisé par les valeurs du rang et de la distance d'entassement. Tels que, les meilleurs individus ont un rang plus petit et une plus grande valeur de distance.

Sélection des parents. La phase de sélection assure la création de la population de descendants pour la prochaine génération. Les meilleurs individus (solutions) sont choisis pour créer des individus de nouvelle génération meilleurs et plus en forme. Dans la littérature, de nombreuses méthodes de sélection sont proposées [31], [52]. Cependant, puisque nous utilisons l'algorithme NSGA II, l'opérateur de sélection

(crowded tournament) est employé pour choisir la prochaine génération de parents. La méthode consiste à sélectionner aléatoirement un ensemble d'individus k, puis, classés en fonction de leur forme physique relative (rang et distance) et l'individu le plus apte est sélectionné pour la reproduction. L'ensemble du processus est répété n fois jusqu'à ce que la proportion de sélection soit atteinte.

Correction. A ce stade, chaque enfant produit est une solution potentielle au modèle proposé, en conséquence, une étape de vérification-correction est nécessaire. Chaque individu résultant des opérateurs de production passe par une vérification des contraintes du modèle, une fois un individu est detecté violant les contraintes du modéle, celui-ci passe à l'étape de correction pour s'adapter aux contraintes du modèle proposé.

Replacement. Étant donné que le NSGA II est un algorithme élitiste, la stratégie de remplacement de la population en régime permanent est utilisée. La population de descendants est combinée avec la population actuelle. De plus, les N (taille de la population) les meilleurs individus sont sélectionnés, en utilisant la stratégie de classement non dominé, pour créer la prochaine génération de population.

Condition d'arrêt. Nous utilisons un nombre d'itération maximum comme critère d'arrêt.

1.5.1.2 Adaptation au modèle LASSO modifié

Cependant, pour chaque modèle, nous proposons une adaptation spécifique présentée par différents opérateurs. Dans cette partie, nous détaillons les opérateurs utilisés pour le modèle LASSO modifié.

Croisement. Cette étape est un opérateur de convergence, qui vise à tirer la population vers l'optimalité locale [88]. Nous avons appliqué l'opération en deux points choisis au hasard.

Mutation. Est un opérateur de divergence. Il est destiné à séparer occasionnellement un ou plusieurs membres d'une population d'un espace local minimum / maximum et potentiellement découvrir un meilleur espace optimal. Le point de mutation est choisi au hasard.

1.5.1.3 Adaptation au modèle Non-Smooth $L_{1/2}$

Pour cette partie de section, les opérateurs particuliers du modèle de régularisation Non-Smooth $L_{1/2}$ sont développés.

Opérateur de production. La population issue de la sélection des parents, population progéniture, est utilisée pour créer de nouveaux individus appelés enfants. Ainsi, la population de descendants est divisée en deux parties; chaque parent subira des opérateurs de croisement ou de mutation. De nombreux types différents de croisement et de mutation existent dans la littérature, comme suit; nous décrivons les opérateurs adaptés à notre problème spécifique:

- Croisement: Le schéma de croisement utilisé dans l'algorithme actuel simule le croisement binaire assisté dans la nature. Nous appliquons la technique de croisement pour chaque variable individuellement.

Croisement des variables binaire(C_{ij}^k matrice de connexion et U_i vecteur décision associé aux neurones de la couche caché): Le croisement utilisé est donné ci-dessous:

$$Y_{1,k} = \alpha_k X_{1,k} + (1 - \alpha_k) X_{2,k} \qquad \forall k = 1, \ldots, m \tag{1.5.16}$$

$$Y_{2,k} = \alpha_k X_{2,k} + (1 - \alpha_k) X_{1,k} \qquad \forall k = 1, \ldots, m \tag{1.5.17}$$

Tels que $X_{(i,k)}$ est le parent sélectionné, $Y_{(1,k)}$ est l'enfant i^{th} avec le composant k^{th}. Où α_k est une variable binaire choisie au hasard. La formule peut être considérée comme un croisement multi-point choisi au hasard, comme expliqué par la figure (5.4);

Croisement des variables continu (W_{ij}^k matrice des poids): nous adoptons les mêmes équations utilisées pour le croisement binaire, avec $alpha_k$ est un paramètre continu, choisi par une distribution uniforme dans [0,1].

- Mutation: La mutation employée est décrite comme suit:

  Mutation des variables continu les gènes à muter, du parent sélectionné, sont choisis en fonction d'un taux de mutation. Une fois les gènes sélectionnés, nous appliquons la formule suivante:

  $$Y(i) = X(i) + (\alpha - 0.5) * 14 \tag{1.5.18}$$

  Tels que i est le gène sélectionné, X est le parent sélectionné et Y est l'enfant obtenu. De plus, α est un paramètre continu choisi par une distribution uniforme dans [0,1].

  Mutation des variables binaire Pour les variables binaire, nous choisissons d'appliquer une mutation simple, c'est-à-dire de pivoter plusieurs valeurs de gènes choisies aléatoirement. Les gènes à permuter sont sélectionnés en fonction d'un taux de mutation. La figure (5.5) représente le processus décrit.

1.5.2 Résultats expérimentaux

Dans cette partie, nous effectuons une série de tests de l'approche proposée en utilisant des ensembles de données de référence standard du référentiel d'apprentissage automatique de l'Université de Californie à Irvine (University of California Irvine (UCI)) [119]. Nous avons choisi quatre problèmes de classification: l'ensemble de données: l'iris de Fisher, les données Win, le Breast cancer Wisconsin et l'ensemble de données Seed. Le tableau (5.1) présente un aperçu des caractéristiques de l'ensemble de données utilisé: le nombre d'exemples, d'attributs et de classe est mentionné. Pour Chacune des données mentionnée, une architecture de départ a était adapté. L'architecture initiale consiste en une seule couche cachée contenant 10 neurones cachés pour Iris, 15 pour Wine, Cancer et Seed. De plus, les poids MLPNN sont générés aléatoirement dans [-0.5,0.5].

1.5.2.1 Résultats du modèle LASSO modifié

Dans un premier temps, nous obtenons le front de Pareto, en un seul fonctionnement de l'algorithme. Cependant, il faut noter que pour le modèle du perceptron multicouche, nous sommes intéressés que par une seule solution considéré la plus adéquate, réalisant un équilibre entre la complexité et la performance du réseau. Dans ce cas, nous nous positionnons en décideur, car pour de telles problématiques, le choix de la solution final est plus important. Dans ce contexte, plusieurs approches ont été proposée [49]. Ainsi, pour le modèle de LASSO modifié, nous avons adapté une approche basée sur la distance d'entassement pour choisir la solution finale. le tableau (5.3) présente le résultat obtenu. Toutefois, nous proposons d'évaluer la performance du réseau obtenu via une comparaison à un réseau régularisé et entraîné par la méthode de rétropropagation accompagnée du terme de régularisation LASSO (BP+LASSO) [voir la section 5.4.5]. En considérant les résultats du tableau (5.3), nous remarquons que la performance de l'approche proposée en matière de régularisation du réseau est considérablement meilleure. Notre approche a pu éliminer 23,8% de connexion pour Iris. Ainsi, 66,6% pour data Wine et 95,5% pour data cancer. Nous pouvons constater que le nombre de connexions ainsi que de neurone a été considérablement réduit. Notre approche est aussi meilleure en matière de classification, puisque celle-ci enregistre un meilleur taux de mal classé. De ce fait, le modèle proposé est capable de fournir une perception généralisée.

1.5.2.2 Resultats du modéle "Non-Smooth $L_{1/2}$" modifié

Comme déjà mentionné, le front de Pareto n'est qu'une première partie de l'optimisation du perceptron multicouche. La figure (5.8) présente le front de Pareto obtenu pour chaque ensemble de données. Il reste à choisir une solution appropriée compromettant la précision et la complexité du réseau. Par conséquent, la solution finale du problème est le résultat à la fois d'une optimisation et d'un processus de décision. Cette fois-ci, nous proposons de choisir aux dépens du taux de mal classifié; tel que la solution portant le meilleur taux sera considérée comme meilleur. Le tableau (5.4) pressente la solution sélectionnée pour chaque ensemble de données. En général, nous pouvons considérer que la taille du réseau a été remarquablement réduite. Pour l'ensemble iris nous avons constaté une réduction de 85,54% des poids du réseau, tend dit 67,82% pour data Wine. Pour l'ensemble data cancer, 77,91% a été éliminer. Une large élimination des poids du réseau conduit à une élimination de neurones de la couche cachée, ce qui explique la réduction de la complexité du réseau. Ainsi, nous proposons de comparer les résultats de cette approche à nos travaux intérieurs autant qu' approches considérées basées sur le principe Pareto et d'autres plus classique (section 5.4.6).

Une lecture détaillée du tableau (5.4) permet de rapporter l'efficacité de l'approche proposée. On considérant le taux de mal classifié, notre approche enregistre les meilleurs résultats. Cependant, on considérant la réduction de la taille du réseau, nous nous réservons à prendre une décision. Nous remarquons la supériorité de différentes méthodes en différant les ensembles de données. Quoique notre approche

enregistre un bon résultat. (Pour plus de détail, voire section 5.4.6).

1.6 Conclusion

L'optimisation multiobjectif offre à l'utilisateur plus de liberté pour modéliser un problème donné. Il met également en évidence certaines difficultés de modélisation qui sont souvent évitées dans l'optimisation mono-objectif. En effet, il fait souvent un compromis entre certains objectifs, à l'insu du concepteur du modèle. L'optimisation multiobjectif met en évidence ce compromis et c'est aussi la majeure difficulté de cette approche. Il faut ainsi garder à l'esprit que l'optimisation multiobjectif malgré son efficacité reste à multi-choix, il appartiendra toujours au décideur de décider et de choisir une solution plutôt qu'une autre. Cependant, l'optimisation multiobjectif offre des outils pour aider le décideur à décider. Durant cette thèse, nous avons eu l'occasion d'exploité deux différentes approches pour des problèmes classés comme étant multiobjectif. Le problème de gestion de portefeuilles, étant traité dans un premier temps par une méthode de scalarisation, révèle les défauts de cette approche. Cependant, les résultats obtenus peuvent être jugés satisfaisants, néanmoins, les résultats obtenus ont tendance à privilégier le niveau de risque du portefeuille au bénéfice de celui-ci. Cette remarque peut être expliquée par le comportement du paramètre d'aversion, qui a une certaine valeur rend la fonction d'énergie du réseau de Hopfield continue presque linéaire. Converger vers une approche Pareto parait le mieux adéquate, vu les perturbations causées a l'issue de l'utilisation d'un paramètre d'aversion. Le second problème traité, l'apprentissage du perceptron multicouche, pause un autre défi issu de notre préférence à l'approche Pareto. Était donnée que celle-ci produite un front de solutions dites Pareto optimal, le choix d'une solution adéquat augmente la difficulté de ce problème, puisque nous cherchons à balancer la précision du réseau et sa complexité, le facteur de choix reste fortement dépendant de la nature du traitement via le réseau. Notre application porter sur des données de classification, pour cette raison, notre critère de choix portée sur le taux de mal classifier; La solution de Pareto portant le meilleur taux de mal classifier est considérée comme la meilleure solution possible.

Chapter 2

Multiobjective optimization

2.1 Introduction

Engineers face frequently complex technical problems that arise in a wide variety of areas, such as processing images, mechanical systems design, operations research and electronics. The problem, to be solved, can often be expressed in the general form of an optimization problem, in which we define an objective function, or cost function (or even more), which we seek to minimize in relation to all the parameters concerned. There are many "classic" optimization methods for solving such problems, applicable when certain mathematical conditions are satisfied. Therefore, linear programming effectively deals with the case where the objective functions, as well as the constraints, are expressed linearly as a function decision variables. Unfortunately, the situations encountered in practice often involve one or more complications, which put these methods at fault. For example, the objective function can be non-linear, or even not be expressed analytically as a function of the parameters; or the problem may require the simultaneous consideration of several conflicting objectives. In this context, the multiobjective optimization is introduced. This area, to which this chapter is devoted, deals with the simultaneous presence of several objectives. The different generalities and methodologies to deal with such problems are presented.

2.2 Generality and denfinitions

The multi-objective optimization (by Vilfredo Pareto) is an area of multiple-criteria decision making, optimizing several objective functions simultaneously. In this section, the general definitions and concepts will be explored.

2.2.1 Definitions

Multiobjective problem The known formulation of an optimization problem is related to a single objective function. However, sometimes during a problem modelization, we are faced with a multiplicity of objectives. For example, the decision-maker wants a performance system as well as low consumption of the system. In this case,

we talk about a multiobjective problem or multicriteria optimization. Moreover, a multiobjective problem is a problem in the form:

$$\begin{cases} \min[f_1(X), f_2(X), ..., f_k(X)] \\ \text{s.t.} \\ \quad g_i(X) \leq 0 \quad i = 1, ..., m \\ \quad h_j(X) = 0 \quad i = 1, ..., p \end{cases} \tag{2.2.1}$$

Where k is the number of objective functions $f_i : \mathbb{R}^n \to \mathbb{R}$,
m is the inequality constraints number,
p is the equality constraints number,
The vector $X = [x_1, ..., x_n]^T$ is the vector of decision variables.
The principle purpose of a multiobjective problem resolution is to minimize, as possible, the different objectives. Considering the nature of the problem objectives, it is often that objectives are contradictory. Two objectives are contradictory when decreasing one objective increases other objectives. In the next section, this idea will be more detailed.

Solutions multiplicity Considering an optimization problem, a single solution is taken into evidence. This scenario is rarely met, in multiobjective optimization, most time there is a multitude of solutions, regarding the contradictory objectives. So when a multiobjective problem is optimized, we will get a set of solutions. These solutions will not be optimal since this later doesn't minimize all the problem objectives. A new Concept is then introduced: compromise solutions. Indeed, the obtained solutions are a compromise such as some objectives are optimized the performance of reminding objectives is degraded.

2.2.2 Dominance

As in optimization, a potential solution is detected by performance measurement, it is often by computing the objective function. However, in a multiobjective optimization problem, it is more difficult to compute solutions basing on the objective functions, since this latter is the most time contradictory. So, to be interested in a solution, there must be a dominance relation [27] between the considered solution and other ones, such as:

the dominance relationship A vector X_1 is said dominates X_2 if:

- X_1 is at least as good as X_2 in all objectives,
- X_1 is strictly better than X_2 in at least one objective

To illustrate this definition, we consider the model 2.2.1 a minimization problem, thus the dominance relationship can be defined as:

Definition 2.2.1 *A vector X_1 is said dominates X_2 if and only if:*

$$\forall i \in \{1, 2, \ldots, n\} : f_i(X_1) \leq f_i(X_2) \tag{2.2.2}$$

Such as:

$$\exists j \in \{1, 2, \ldots, n\} : f_j(X_1) < f_j(X_2) \tag{2.2.3}$$

Solutions That dominate other solutions but aren't dominated are called Pareto optimal or non-dominated solutions. Pareto local and global optimality is defined as follow:

Definition 2.2.2 (Pareto local optimality) *A vector $X \in \mathbb{R}^n$ is localy Pareto optimal if there is a real $\delta > 0$ such as there is no vector X' dominates the vector X with $X' \in \mathbb{R}^n \cap B(X, \delta)$.*
Where $B(X, \delta)$ represent a ball of center X and radius δ.

A vector X is then Pareto locally optimal if it is Pareto optimal in a restriction of $\mathbb{R}$.

Definition 2.2.3 (Pareto global optimality) *A vector X is globally Pareto optimal if there isn't any vector X' such as X' dominates X.*

The diffrence between the 2.2.2 and 2.2.3 that no restriction in $\mathbb{R}$ is considered. Moreover, the Pareto optimum set is the set of all Pareto optimum solutions of the problem:

$$POS = \{x \in S, \exists y \in S, f(y) < f(x)\} \tag{2.2.4}$$

POS: Pareto Optimum set

2.2.3 Pareto front

Dominance rank The dominance concept is relevant for the multiobjective optimization problem since the solutions performance measurement isn't that easy. However, dominance alone isn't enough. it can be used to establish solutions ranking such as the $rank = 1$ is for the solutions that dominate all the solutions set but aren't dominated between them. These solutions, thereby, are Pareto optimal. The idea is repeated on the remaining solutions in the solutions set, the process is stopped when no solutions remained to compare and all the set solutions are assigned to a rank.

The pseudo-code of the ranking proccess is presented by the algorithm 3. In the algorithm, the variable N denotes the solutions set cardinality.

Pareto front The rank=1 of selected solutions using the dominance concept form a front called the compromised surface or Pareto front.

Algorithm 3: Ranking Procedure

Require: $X \in S$

```
CourantRank=1
m=N
while N ≠ 0 do
    for i=1 to m do
        if X_i isn't dominated then
            Rank(X_i,t)=CourantRank
        end if
    end for
    for i=1 to m do
        if Rank(X_i,t)=CourantRank then
            put X_i in a temporary population
            N=N-1
        end if
    end for
    CourantRank=CourantRank+1
end while
```

Example Late take the case of two objectives minimization $(\min(f_1(X), f_2(X))$ under constraints $g(X) \leq 0$ and $h(X) = 0$.
S will be the set of $(f_1(X), f_2(X))$ values when X respect the constraints $g(X)$ and $h(X)$. P the Pareto front. By representing the two notation (S, P), Fig. 2.1 is obtained.

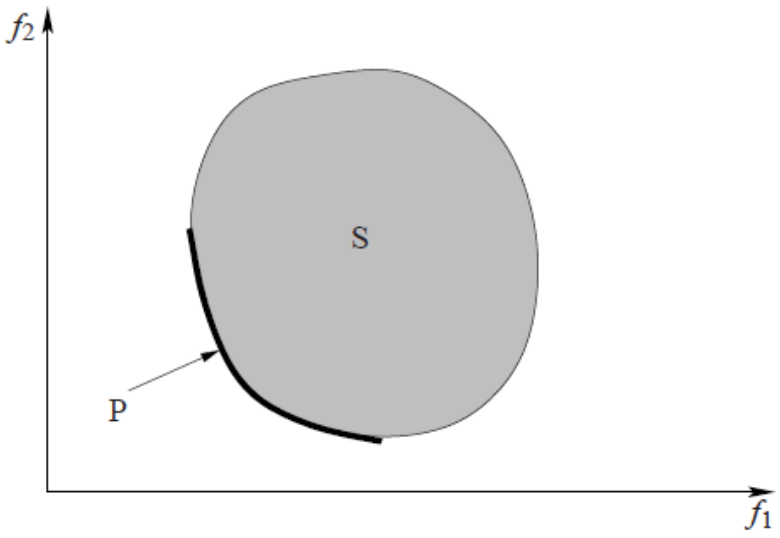


Figure 2.1: Pareto front representation [21]

Fig. 2.1 is not always obtained as presented, it depends on the problem classification. The current forms of the Pareto front are presented in Fig. 2.2. These

forms are typical of optimization problem multiobjective in convex solutions set. It is the must problem current problem faced in the multiobjective optimization. Two characteristics are then associated with the Pareto front:

Ideal point The coordinates of the ideal point are obtained by optimizing each objective separately.

Nadir point The coordinates of this point correspond to the worst values obtained by each objective function when the solutions space is restricted to the Pareto front.

The ideal point is used in many optimization methods as a refereeing point. On the other hand, the "Nadir" point is used to restrain the research space, also, used in some interactive optimization methods.

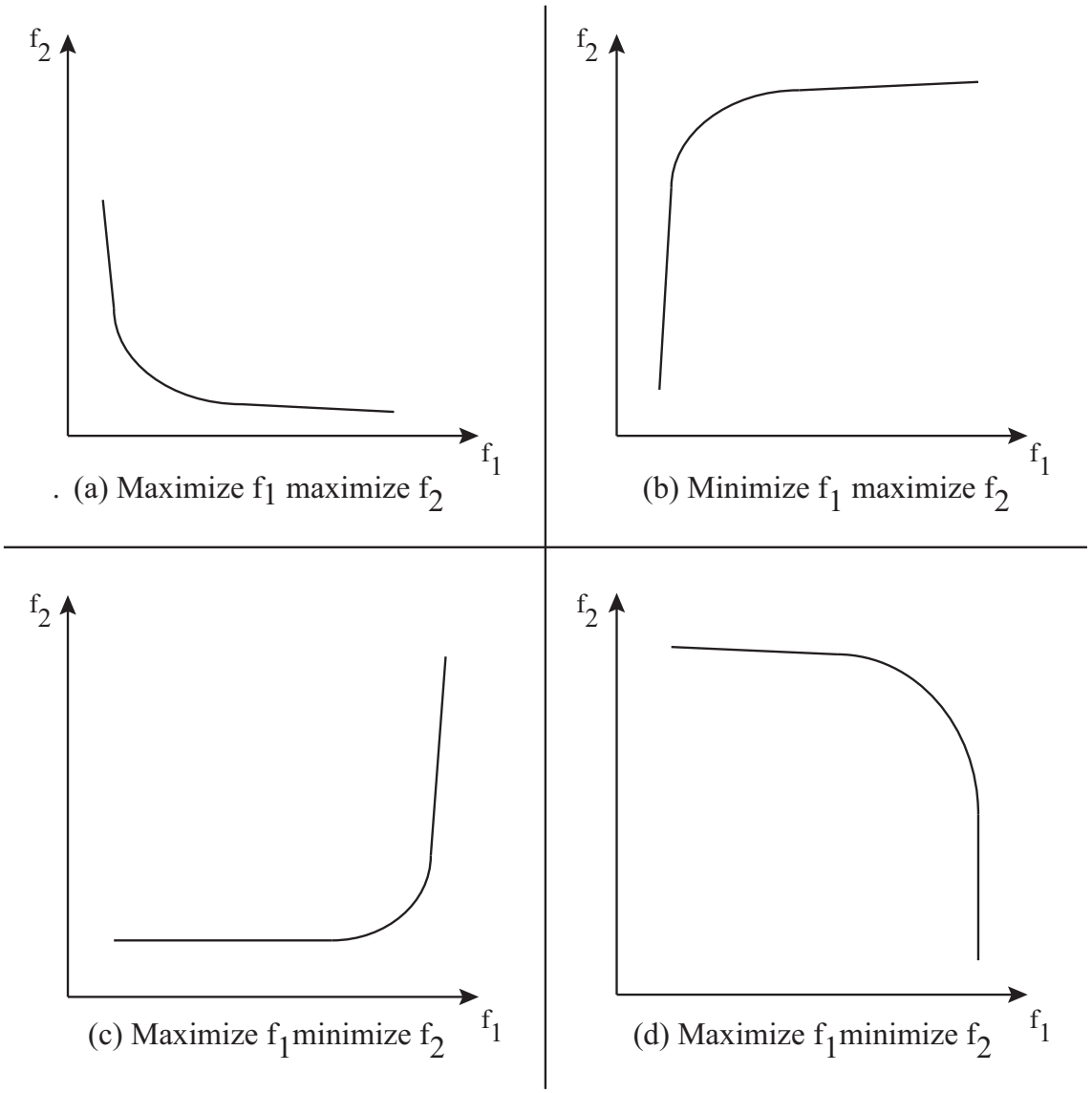


Figure 2.2: The most common Pareto front forms for two objectives [21].

2.3 Resolution methods of multiobjective optimization problem

There are many methods for the multiobjective optimization [21], generally, they can be classified into three different families:

A priori preference methods: the user defines the compromise he wishes to make before launching the optimization method. This category groups most of the aggregation methods where the objective functions are consolidated into one objective function.

Progressive preference methods: In this category, the user can refine his choice of compromise progressively in the optimization process, such as for the interactive method.

A posteriori preference methods: the categorized methods allow the user to choose a compromise solution by examining all the solutions extracted by the optimization method. At the end, the methods of this family provide the complete Pareto front.

There are multi-objective optimization methods that do not fit exclusively into a family. For example, we can use a priori preference method by providing it with randomly chosen preferences. The result will then be a large number of solutions that will be presented to the user to decide on the compromise solution. Then this combination forms a posteriori preference method. Therefore, the presented classification only allows getting an idea of the used approach to obtain an adequate result. A more "fine" classification is developed in the following sections.

2.3.1 Scalarization methods

The methods aiming to transform the multiobjective problem into a mono-objective one are called scalarization methods, these methods can belong to the different families cited previously. In follow, the most known methods of this class are presented.

2.3.1.1 Weighted objective functions

Principe: This approach is most evident, it is also called the naive approach [20]. It is an aggregation method, trying to obtain a mono-objective problem optimization, which their several resolution methods available. Furthermore, The method consists of applying a weighted sum of objective functions such as each weight answer to an objective function. Finally, a new single objective function is obtained.

Method Presentation: Considering the model 2.2.1, using the weighted objective functions, the model is transformed in the following way:

$$\begin{cases} \min f_{eq}(X) = \sum_{i=1}^{k} w_i.f_i(X) \\ \text{s.t.} \\ \quad g_i(X) \leq 0 \quad i = 1, ..., m \\ \quad h_j(X) = 0 \quad j = 1, ..., p \end{cases} \tag{2.3.1}$$

Such as $X \in \mathbb{R}$. Frequently, the weighting coefficients respect the following relation:

$$\sum_{i=1}^{k} w_i = 1 \tag{2.3.2}$$

Such as $w_i \geq 0 \quad i = 1, ..., k$.
The method can be graphically illustrated considering two objectives as follows:

$$\begin{cases} \min f_1(X) \\ \min f_2(X) \\ \text{s.t.} \\ \quad g(X) \leq 0 \\ \quad h(X) = 0 \end{cases} \tag{2.3.3}$$

Such as $g(X) = (g_1(X), ...g_i(X), ..., g_m(X))$ and $h(X) = (h_1(X), ...h_j(X), ..., h_p(X))$
The transformation will consider the following objective function:

$$f_{eq}(X) = w_1 f_1(X) + w_2 f_2(X) \tag{2.3.4}$$

The Equ. 2.3.4 is the expression a straight line in the plane f_1, f_2. Indeed, if we try to minimize f_{eq} we are, actually, looking for a constant C of the following straight line equation as small as possible:

$$f_2(X) = -\frac{w_1}{w_2} f_1(X) + C \tag{2.3.5}$$

This straight line equation corresponds to the isovalue curve of the equivalent objective function.

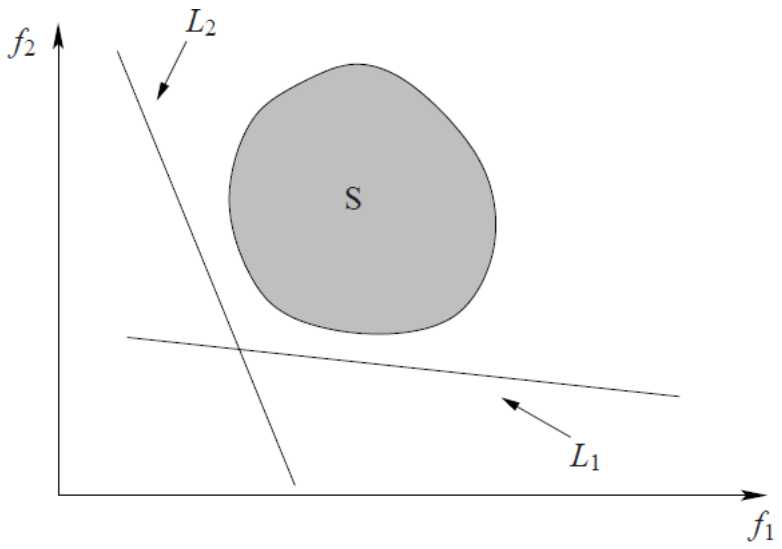


Figure 2.3: The weighted Objective Functions method Presentation [21]

In Fig. 2.3 the set S corespond to the couple $(f_1(X), f_2(X))$ values set respecting the constraints defined by $h(X)$ and $g(X)$. The L_1 and L_2 correspond to two couples of different weights coefficients (w_1, w_2).

This method consists in "making tangent" the line L_1 and the line L_2 with the set S. The point of tangency is then the solution looking for. If the process is repeated for several weights coefficient, the different obtained solutions shape the Pareto front.

This method is only applicable to convex sets S. Otherwise, it does not find the entire Pareto front.

Discussion: This method was one of the first used. From an algorithmic point of view, it is very effective. Moreover, it is possible, by varying the weight coefficient values, to find the Pareto front, if the realizable domain is convex.

However, this method does not make it possible to find the solutions enclosed in a concavity. In certain optimization problems of mechanical structures, it can behave catastrophically [66].

2.3.1.2 Keeney-Raiffa method

Principe: The method uses the product of objective functions to reduce multiobjective functions to a problem of single-function optimization. It is also an aggregation method, similar to the method of weighted objective functions. The objective function, thus obtained is called the utility function of Keeney-Raiffa. This function is found in the multi-attribute utility theory [59], it is originated from the sciences decision support, discusses properties and how to create a "utility function".

Method presentation: considering the model 2.2.1, We transform the "objectives" part using the following function:

$$f_{eq}(X) = K \prod_{i=1}^{m} (k_i u_i(f_i(x)) + 1) \tag{2.3.6}$$

Such as:
k_i is a normalisation coefficient with $k_i \in \mathbb{R}$,
$u_i(\vec{y})$ is a function strictly non-decreasing. For example, $u_i(y) = y^2$ is a strictly decreasing function if $y \in \mathbb{R}^+$. To illustrate the method application, let take $u_i(y))y^2$, $K = 1$ and $k_i = 1$. The problem 2.2.1 is transformed as follow:

$$\begin{cases} \min f_{eq}(X) = \prod_{i=1}^{m} (f_i(x)^2 + 1) \\ \text{s.t.} \\ \quad g(X) \leq 0 \\ \quad h(X) = 0 \end{cases} \tag{2.3.7}$$

2.3.1.3 ϵ-constraints Method

Principe: The ϵ-constraints method [81] provides a mono-objective form starting by a multiobjective problem, using some additional constraints. It is classified as a progressive preference method. The method process is as follows:

- Choose an objective to optimize as a priority
- Choose a vector of initial constraints
- Transform the problem by holding the priority objective and transforming the others objectives into inequality constraints.

Method Presentation: It always starting with the model 2.2.1, we assume that the priority objective function is the rank 1 function. A constraints vector ϵ_i is chosen such as $i \in 2, ..., k, \quad \epsilon_i \geq 0$. The problem is transformed as follows:

$$\begin{cases} \min f_1(X) \\ \text{s.t.} \\ \quad f_2(X) \leq \epsilon_2 \\ \quad . \\ \quad . \\ \quad . \\ \quad f_k(X) \leq \epsilon_k \\ \quad g(X) \leq 0 \\ \quad h(X) = 0 \end{cases} \tag{2.3.8}$$

Discussion: This method can be reduced to the weighted objective functions through a slight transformation. The main disadvantages of this method are as follows: she is devouring regarding the computation time and algorithm programming can be extremely difficult if there are too many constrained functions. However, the relative simplicity of the method made it popular.

2.3.1.4 Corley methods

Principe: The Corley method is a hybrid method, it uses several knows methods to form a new one. in the case of the Corey method, it uses the weighted objective functions and the ϵ-constraints such as starting by the model 2.2.1 the problem is formulated as:

$$\begin{cases} \min \sum_{i=1}^{k} w_i f(X) \\ \text{s.t.} \\ \quad f_j(X) \leq \epsilon_j \quad j = 1, .., k \\ \quad g(X) \leq 0 \\ \quad h(X) = 0 \end{cases} \tag{2.3.9}$$

Method presentation: The process of the method has been shown for a problem with two objectives in Fig. 2.4.

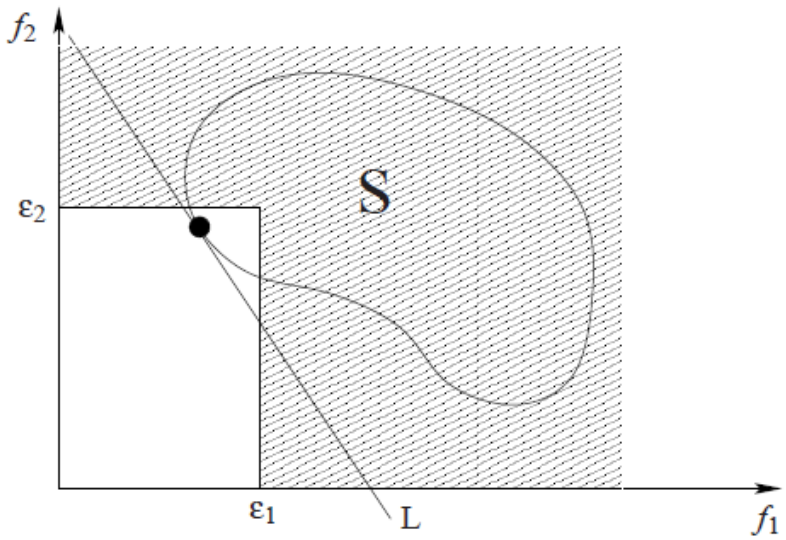


Figure 2.4: The Corney method Presentation [21]

From Fig. 2.4, we can see that the addition of the two constraints $f_1(X) \leq \epsilon_1$ and $f_2(X) \leq \epsilon_2$ allowed to restrict the search space. Then the weighted objective functions method is applied to this limited research space. For a vector $w = (w_1, w_2)$ to which the straight-line L correspond, searching for an optimal value amounts to doing tangent line L with the limited search space.

Discusion: The Corley method allows combining the advantages of the two methods mentioned above (the weighted objective functions method and the ϵ-constraints method). hence, it is effective on different types of problems, whether convex or non-convex. However, a difficulty arises: the number of parameters to be determined has been multiplied. It will be much more difficult for the user to express his preference by playing all the parameters.

2.3.2 Metaheuristics approachs: Evolutionary algorithms

Metaheuristics are general research methods dedicated to "difficult optimization" problems [100]. These methods, which include genetic algorithms, the taboo research method, ant colony algorithms, etc. appeared, since the 1980s, with a common ambition: to best solve difficult optimization problems. Besides, they have the following characteristics:

- They are partly stochastic: this makes possible to face the combinatorial explosion of possibilities;
- they are originally combinatorial: they have the decisive advantage in the continuous case of being direct, that means that they do not resort to the calculation, often problematic, of the gradients of the objective function;
- they are inspired from analogies: with physics (simulated annealing, simulated diffusion, etc.), from biology (genetic algorithms, taboo research, etc.) or aetiology (Ant Colonies, Particles Swarms, etc.));
- they are capable of guiding, in a particular task, another specialized search method (for example, another heuristic, or a local exploration method);
- they also share the same disadvantages: the difficulties of adjusting the very parameters of the method, and the high computation time.

Metaheuristics are, in general, presented as a concept. In this section, we are more interested in the evolutionary one. The evolutionary algorithms are a stochastic class of optimization methods that simulates the process of natural evolution such as genetic algorithms. These approaches operate on a set of candidate solutions called population and by adopting strong simplifications, the population is then subsequently modified through two basic evolution principles: selection and variation. The selection presents the competition for resources among living beings, in evolutionary algorithms, natural selection is simulated by a stochastic selection process. The variation principle reflects the natural capacity of creating "new" living beings employing recombination and mutation. The evolutionary algorithms are classified as the most efficient algorithms for optimization. Besides, evolutionary algorithms suites more multiobjective optimization since this later can achieve solutions called Pareto-optimal in a single simulation and may exploit solutions similarities by recombination. In this context, the multiobjective evolutionary algorithms can mainly be classified as algorithms Incorporating Pareto dominance concept to their selection

mechanism such as NSGA [113] and there that doesn't such as VEGA [102]. Historically, the MOEAs has evaluated in two generations, the first one used the Pareto ranking and niching or fitness sharing, the principle is relatively simple, besides to population-based approaches. The second one introduced the elitism concept by using $(\mu + \lambda)$ selection and a secondary population. In this section, each the MOEAs generation will be explored, especially the most popular and efficient.

2.3.2.1 Vector Evaluated Genetic Algorithm (VEGA)

Principe: This method is various of genetic algorithm [111], it is the first implementation of a real multiobjective evolutionary algorithm, proposed in 1984 by David Shaffer [102]. The VEG Algorithm is a based-population, the method modified the simple three-operators of the genetic algorithm: selection, crossover, and mutation to perform independent selection cycles according to each objective. This method is classified as a non-aggression method, but also non-Pareto based.

Method presentation: VEGA uses subpopulations that optimize each objective separately. It considers a population of N individuals. These N individuals are distributed into k groups (k being the number of the objective functions) of $\frac{N}{k}$ individuals (with N multiple of k). Each group is associated with an objective function. This objective function makes it possible to determine the efficiency of an individual within the group. Then the individuals are mixed and the production operators are applied taking into account the efficiency of each individual. Fig. 2.5 represents the different operating sequences of the method.

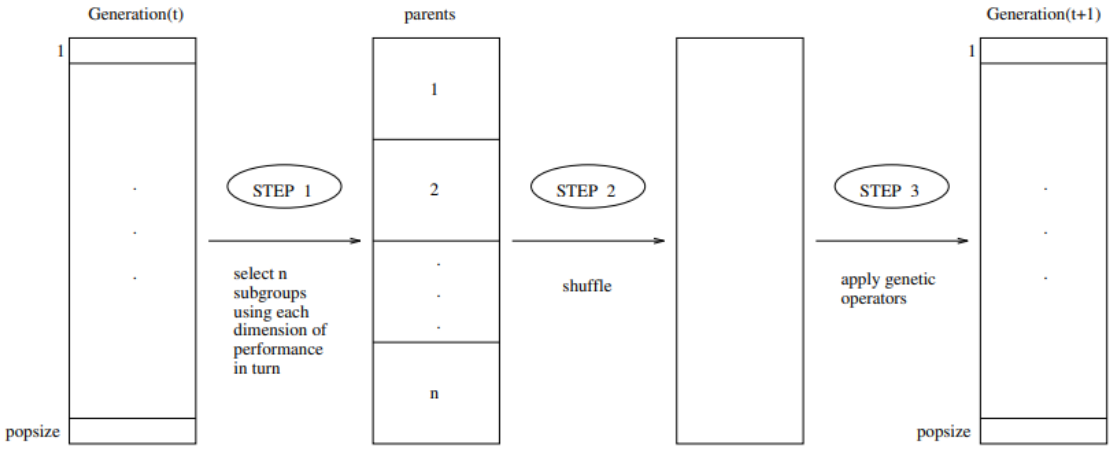


Figure 2.5: Schematic of VEG Algorithm [19]

Here is the description of the different stages:
Step 1: initilize a population of size N;
Step 2: Creation of k groups (sub-populations);
Step 3: compute the individuals efficiencies and mix the individuals;

Step 4: apply the classical genetic algorithm (crossing-mutation-selection);
Step 5: Go back to step 2 and repeat the operation until the stopping criterion is reached.

Discusion: The mean VEGA disadvantage is the possibility to obtain, in the end, a population made up of average individuals in all objectives. Such a population does not allow a well-designed compromise surface to be achieved. Indeed, the population will be focused around a Medium "point". several tricks have proposed to prevent this effect such as the use of species within a group, etc. Besides, it has been shown that this method is similar to the weighted objective function method. Therefore, it does not allow finding solutions in a concave space.

2.3.2.2 Non-dominated Sorting Genetic Algorithm (NSGA)

Principe: NSGA is also a variant of genetic algorithm uses the dominance concept to determine the individual's efficiency, it is suggested by Srinivas and Deb [113], it is based on several layers of classifications of the individuals as suggested by Goldberg [130], Before the selection operator, the individuals are ranked according to the dominance ranking algorithm, then for each individual ranked a function is applied called sharing function. The NSGA principe is similar to the Multiple Objective Genetic Algorithm (MOGA [30], the main difference is in the individual's efficiency calculating.

Method presentation: As mentioned before, All non-dominated individuals of the same rank are classified in a category. Then, a dummy efficiency is assigned to each category, which is inversely proportional to the Pareto rank of the category considered. To maintain the diversity of the population, the classified individuals are affected by a new value of efficiency. the following formula is used:

$$m_i = \sum_{j=1}^{k} Sh(d(i,j)) \tag{2.3.10}$$

Such as:

$$Sh(d(i,j)) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_s hare}\right)^2 & if \quad d(i,j) < \sigma_s hare \\ 0 & if \quad not \end{cases} \tag{2.3.11}$$

k denotes the number of individuals in the category considered, $d(i,j)$ is the distance between individual i and individual j and $\sigma_s hare$ is the influence distance.

From the expression 2.3.10, all individuals who are close enough (whose distance $d(i,j)$ is less than $\sigma_s hare$) will be taken into account in the calculation of m_i. The rest will be ignored. The efficiency value of individual i within the considered category will then be:

$$f_i = \frac{F}{m_i} \tag{2.3.12}$$

Such as F presents the efficiency value assigned to the category to which the individual belongs.
Figure 2.6 shows the different parameters involved in the individual efficiency computing.

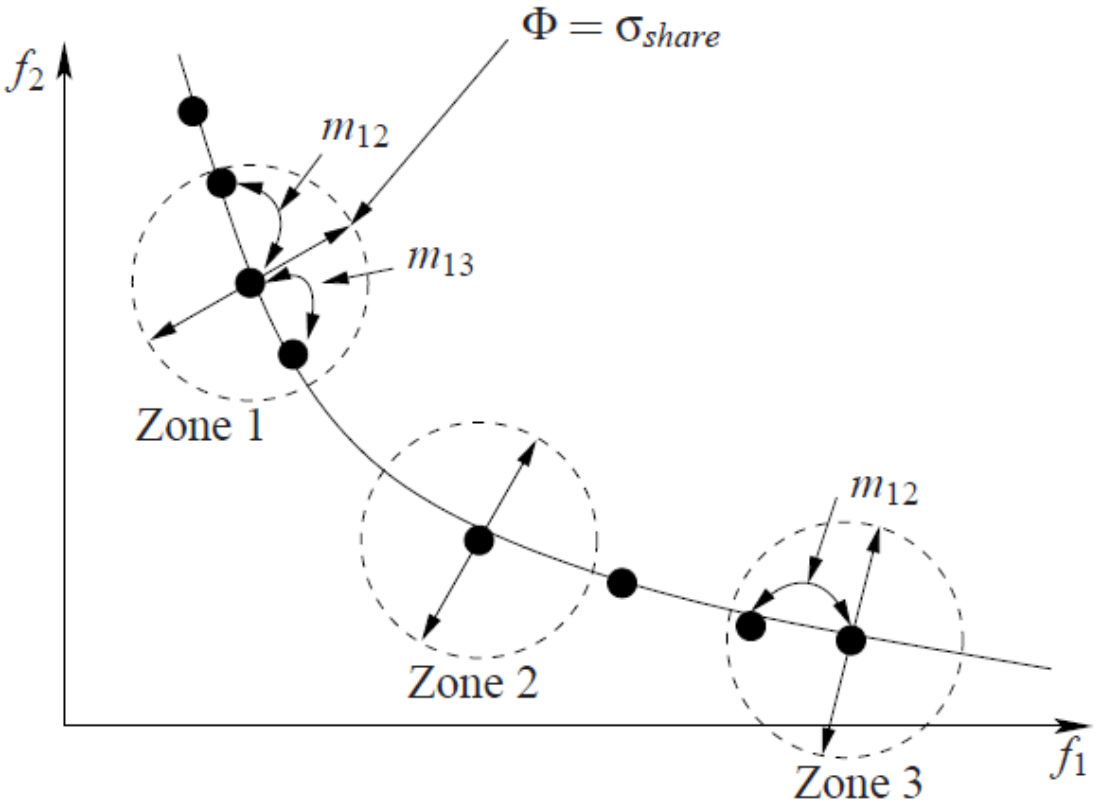


Figure 2.6: Efficacity computing [21]

The individuals of current rank are ignored and the process, then repeated with the remaining individuals, where a new classification is operated, a new categorization and a new sharing operation. This process is repeated until all individuals have been treated.

As individuals with a Pareto rank of 1 have better efficiency, they reproduce in greater numbers than others. This property allows faster convergence to the compromised surface. Sharing, on the other hand, maintains a "uniform" distribution over the Pareto front.

Different niches exist, depending on the used distance. The genotypic niche use the Hamming distance [37], the niches are counted is the gene space. The second niche type is phenotypic niches, it uses the Euclidean distance, niches are counted in the phenotype space. Another type of niches can be considered, the combined

niches, that use both distances such as:

$$Sh(d(i,j)) = \begin{cases} 1 - \left(\frac{d(i,j)_{ham}}{\sigma_{gen}} . \frac{d(i,j)_{euclid}}{\sigma_{phen}}\right)^2 & if \quad d(i,j)_{ham} < \sigma_{gen} \quad and \quad d(i,j)_{euclid} < \sigma_{phen} \\ 1 - \left(\frac{d(i,j)_{euclid}}{\sigma_{phen}}\right)^2 & if \quad d(i,j)_{ham} \geq \sigma_{gen} \quad and \quad d(i,j)_{euclid} < \sigma_{phen} \\ 1 - \left(\frac{d(i,j)_{ham}}{\sigma_{gen}}\right)^2 & if \quad d(i,j)_{ham} < \sigma_{gen} \quad and \quad d(i,j)_{euclid} \geq \sigma_{phen} \\ 0 & if \quad not \end{cases} \tag{2.3.13}$$

Niches are counted in the phenotype space and the gene space. A phenotypic niche allows us to "settle" the diversity of solutions in the space of objective functions while the genotypic niche allows regulating the diversity of solutions in the decision space. The combined niche provides diverse solutions in both the decision space and the objective function space.
Here is the description of the different stages:
Step 1: initilize a population of size N;
Step 2: evaluate the objective functions;
Step 3: Assign a rank based on the rank of dominance on each compromise surface;
Step 4: compute the neighbor account;
Step 5: assign a shared effectiveness for each individual;
Step 6: Apply the selection, mutation, crossover operator;
Step 7: go back to step 2 and repeat the proccess until the stopping criterion is reached.

Discusion: The effectiveness of this method lies in the fact that the objectives are reduced to a dummy efficiency value obtained by using the classification according to the rank of Pareto. it has also the disadvantage of being sensitive to the σ_{share} value choice. Besides that, the Pareto ranking has to be repeated over and over again.

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

Principe: The Non-dominated Sorting Genetic Algorithm II is an improved version of the NSGA, belonging to the second generation, which is more efficient (computationally speaking), it uses elitism and a crowded comparison operator that keeps diversity without specifying any additional parameters. The method was introduced by Deb and Agarwal [26]. It was classified as one of the most efficient multiobjective evolutionary algorithms, besides, NSGA II has the features required for a good multi-objective GA for his selection method.

Method presentation: The algorithm uses a heuristic to evaluate invidious of the same front, called crowding distance [64], such as the one situated in a less crowded area is picked. The crowded tournament selection guarantees the selection of the best rank possible.

The crowding distance (Crowding Distance (CD)) approach aims to obtain uniform solutions spread through the Pareto front without using fitness parameter, it considers the perimeter of the cuboid value (Fig. 2.7) created by the surrounding solutions.

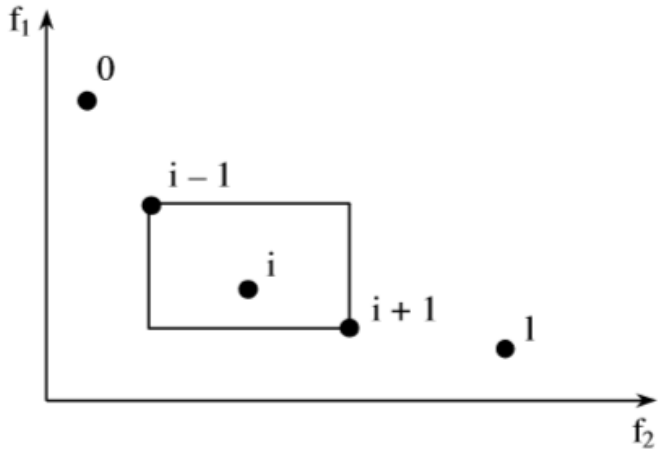


Figure 2.7: The Crowding Distance Computing [26]

The measure of the population density around a solution, computed by the CD approach, is calculated without requiring an additional user-defined parameter. this can be a real advantage of using this approach compared to different multi-objective methods [64]. NSGA II use the crowding distance (CD) measure as a tiebreaker to a selection technique called crowded tournament selection operator.

Crowding Distance Algorithm:

To determine the crowding distance of each point in the front F_r, the following algorithm is used:

Step 1: For each solution in the set, F_r accredit 0 to the corresponding crowding distance;
Step 2: For each objective function $f_m, m = 1, 2, \ldots, M$, arrange the set in worse adjustment of f_m;
Step 3: accredit a wide distance to the border solutions $d^r(1) = d^r(l) = \infty$ (1) is the first solution and (l) is the last one in the front F_r(after sorting). On the other hand, for all other solutions $i = 2, \ldots, l-1$ assign:

$$d^r(i) = d^r(i) + \frac{\left(f_m^r(i+1) - f_m^r(i-1)\right)}{f_m^{max} - f_m^{min}} \tag{2.3.14}$$

Such as:

F_r The r^{th} Pareto set.
$f_m^r(i+1)$ is the m^{th} objective function value of the $(i+1)$ solution in the set F_r;
$f_m^r(i-1)$ is the m^{th} objective function value of the $(i-1)$ solution in the set F_r;
f_m^{max} is the maximum value of the m^{th} objective;
f_m^{min} is the minimum value of the m^{th} objective;

The total crowding distance $d^r(i)$ is the sum of the solution (i) crowding distances with respect to each objective.

Crowded Tournament Selection Operator:

To select the candidates of the next generation, the NSGA II uses the tournament selection through crowding distance criteria. Each population individual is judged according to two characteristics: non-domination rank i_{rank} and crowding distance $d^r(i)$. Furthermore, the crowded tournament selection operator can be defined by:

$$f\ (i_{rank} < j_{rank})\ or\ ((i_{rank} = j_{rank})\ and\ (i_{Cr} > j_{Cr})) \tag{2.3.15}$$

Considering two solutions, **i and j selected randomly**, have equal non-dominated ranks ($r_i = r_j$). The solution with the highest crowding distance value is selected. Otherwise, the selection is based on the rank value. Moreover, the solution with the minimized rank is preferred. This procedure maintains the solution diversity in the population and prevents premature convergence.

Discusion: The principal advantage of NSGA II is the elitist mechanism of the NSGA II. It consists of combining the best parents with the best offspring obtained. Besides the algorithm doesn't require any extra parameter to evaluate the individuals.

2.4 Conclusion

As seen throughout this chapter, multi-objective optimization is not an easy task. The first difficulty encountered is the definition of a multi-objective optimization. A general and well-established definition is the Pareto optimality. Since many problems look to optimize contradictory objectives. The next step presents the resolution mode. Depending on the knowledge of the problem, you can:

- go in search of a single solution if the decision-maker preferences are well determined and if the latter does not wish to choose from a range of possibilities;
- specify the preferences of the decision-maker during the optimization process;

- approach the Pareto front with a "cloud" of solutions. Basing on your choose, different optimization methods are available, some are presented in this chapter.

Multi-objective optimization offers the user more freedom to model their problem. It also highlights certain modelling difficulties that are often avoided in mono-objective optimization. Indeed, it often makes a compromise between certain objectives, without the knowledge of the designer of the model. Multi-objective optimization brings this compromise to light and this is also the difficulty of multi-objective optimization. It must be kept in mind that multi-objective optimization will not treat this compromise with a magic wand. It will always be in the decision-maker to decide and choose one solution over another. still, multi-objective optimization offers tools to help the decision-maker to decide.

Chapter 3

Artificial Neural Network: Learning and Optimization

3.1 Introduction

The artificial neural networks (ANNs) are artificial intelligence tools used for a variety of fields such as Natural Language Processing, medicine, financial analysis and more. They have been widely applied given their learning ability. Indeed, describing ANNs with a simple definition is difficult. However, ANNs are an information processing paradigm that is inspired by the way the biological nervous system such as brain process information. They can be considered as a black box having multiple inputs and multiple outputs. As humans, the ANNs can learn by examples and create behaviors for given situations. The artificial neural network can be classified into two categories: Feed-forward neural networks (Feed-Forward Neural Networks (FFNNs)), where the information spreads from the input to output. The network that takes the output as input is called a recurrent neural network (RNNs). Each category, consider several ANNs such as Multilayer perceptron as FFNNs or continuous Hopfield network as RNNs. Furthermore, the choice of employing the ANNs is often related to the nature of the treated problem: does the problem require a supervised or an unsupervised approach.

The present chapter is dedicated to details the abovementioned ANNs: The Multilayer perceptron as an efficient classifier under supervised learning. In addition to the recurrent neural network: Continuous Hopfield network as an efficient nonlinear optimizer.

3.2 Artificial Neural Networks History

The first artificial neuron network was proposed by Warren McCulloch and Walter Pitts in 1943 [80]. The neurophysiologist and the young mathematician proposed a simple neural network model with electrical circuits. In 1949, Donald Hebb reinforced the concept of neurons [39] by confirming that neural pathways are strengthened each time they are used. Moreover, Nathanial Rochester, IBM research labo-

ratories member, managed the first simulated neural network in the 1950s. In 1958, the first perceptron was introduced, Frank Rosenblatt proposed a single-layer perceptron [95] by observing the eye of a fly since the flee treatment of a fly is done in its eye. The perceptron is useful in classification and still in use today. However, Marvin Minsky and Seymour Papert showed that Perceptron is limited [78] in 1969 by showing the perceptron inability to compute some logical functions like XoR. As a consequence, many Artificial Intelligence researchers concluded that the study of neural networks is not promising. Yet, In 1982, John Hopfield proposed an approach to create useful devices [43]; introducing a recurrent neural network. Further, a recurrent neural network framework was proposed by Schmidhuber and Hochreiter in 1997 [41]. Besides, Remulhart and al proposed an approach to overcome the perceptron limit by introducing a training algorithm called back-propagation algorithm in 1987 [98].

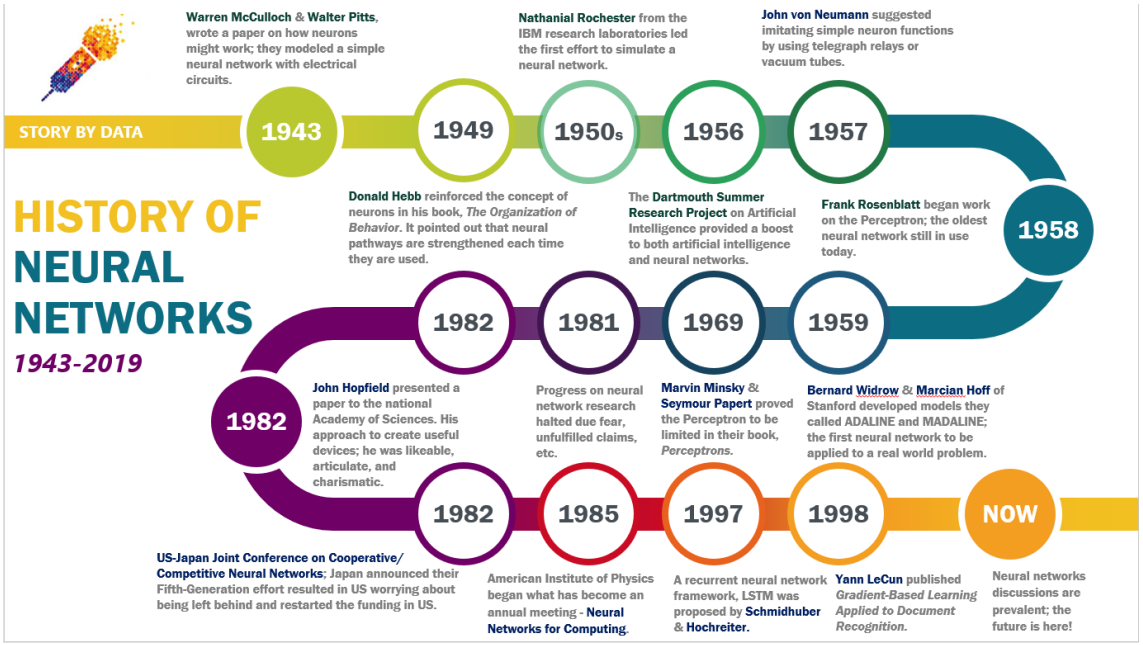


Figure 3.1: Neural Network History Timeline

3.3 Multilayer Perceptron

A Multilayer Perceptron is a variant of the original Perceptron model proposed by Rosenblatt in the 1950s [14]. It is the most known and frequently used neural network today. This latter is classified as a feed-forward neural network, indeed, the information is transmitted within the networks in the direct direction: from the input to output. The network represents a genetic function for approximation and classification. It's can also be seen as a logistic regression classifier. Furthermore, the MLPNN has the power to detect data that are not linearly separable, indeed, the network, first, transforms the input using a learnt non-linear transformation ϕ,

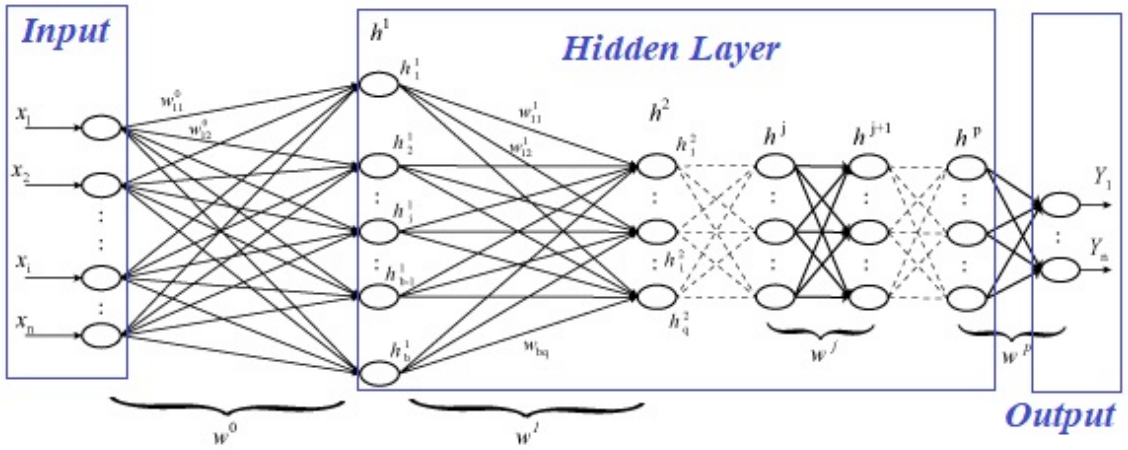


Figure 3.2: Multiple layer perceptron neural network model

projecting the input data into a linearly separable space. For these special characteristics, the MLPNN has been widely used. In the coming section, the multilayer perceptron neural network will generally be discussed.

3.3.1 MLPNN Model

In the coming section, the following notation is used:
x_i: the i^{th} network input
y_i: the i^{th} network output
h_i^k: the i^{th} output of the hidden layer k
$w_i^k j$: the weight between the neuron i of layer $k-1$ and j of the layer k
An MLPNN has three types of layers: An input layer; supporting the problem data; each neuron in the input layer sends weighted information to all neurons in the first hidden layer, where the information is preceded by applying an adequate activation function to feed it as input to the next hidden layer; until the information reaches the last layer, the output layer representing the network answers (see Fig. 3.2). Moreover, the output of each hidden neuron is performed by the sum of the product of the neuron input and their corresponding weights, and then the activation function is applied to it. Each neuron output of k hidden layer is given by :

$$h_j^k = f(\sum_{i=1}^{n} h_i^{k-1} w_i^{k-1} + b_j^k) \tag{3.3.1}$$

Such as k is the current hidden layer and b_j^k is the bias of the neuron j of the hidden layer k.

3.3.1.1 Activation Function

Neural Network activation function is a crucial component of neural networks, is responsible for the model output, its accuracy and the computational efficiency of

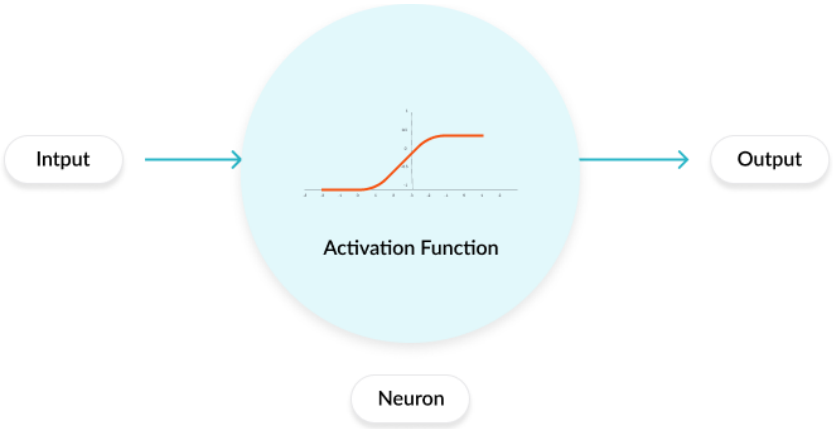


Figure 3.3: Activation function rule [1]

the training model. It has a major effect on the neural network convergence speed. Essentially, the activation function determines if the neuron should be activated or not based on whether each input is relevant to the model response. Moreover, the activation function is a mathematical answer treating the input feeding the current neuron to get its output going to the next layer (see Fig. 3.3). There are different proposed functions to sure neuron activation. However, three principal categories can be defined.

Step Function; the neuron is activated corresponding to the signal sign such as Heaviside step function defined by Equ. 3.3.2, Fig. 3.4 is the function presentation, such as v is an input signal, Or bipolar step function defined by Equ. 3.3.3.

$$f(v) = \begin{cases} 1 \; if \; v \geq 0 \\ 0 \; if \; v < 0 \end{cases} \tag{3.3.2}$$

$$f(v) = \begin{cases} 1 \; if \; v > 0 \\ -1 \; if \; v \leq 0 \end{cases} \tag{3.3.3}$$

Linear Activation Function;As indicate the function name, it is a straight line function where activation is proportional to the neuron input. Fig. 3.5 present the seed function.

Non-Linear Activation Function; it is a non-linear transformation to the input making it capable to learn and perform more complex tasks, which are essential for learning and modeling complex data. There are deferent non-linear activation functions, we present the most used one as follow:

Logistic Activation Function The logistic or sigmoid function performs an output between 0 to 1 with an 'S' shape (see Fig. 3.6.a). The logistic function can be described as the probability of a neuron 'firing' given its inputs. Moreover, it is a smooth function and continuously differentiable. However, the logistic activation function has some disadvantages. First, the function isn't zero-centered. Moreover, the function produces a vanishing gradient problem, That means that for very high

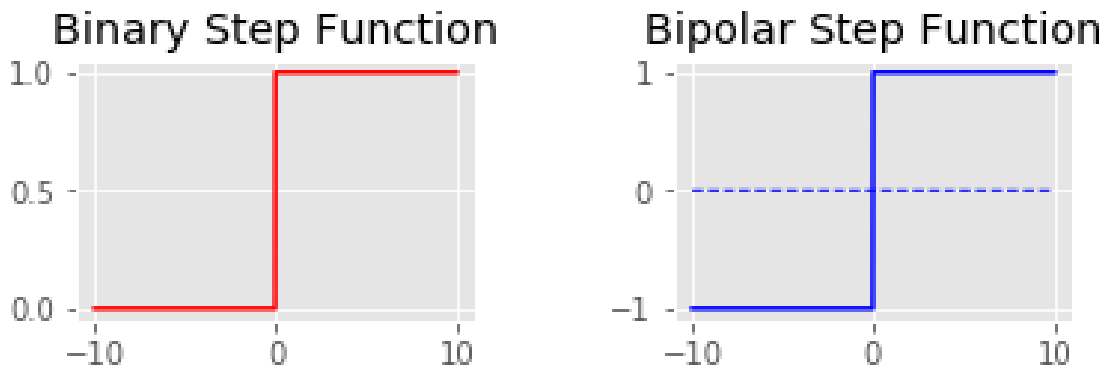


Figure 3.4: Step Functions

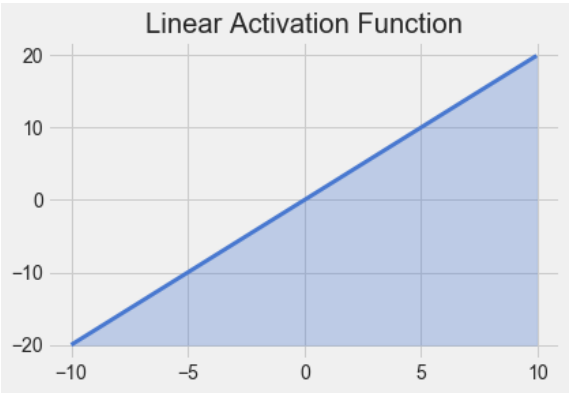


Figure 3.5: Linear Activation Function

or very low input values, there is almost no change to the model. Besides, the function is highly computed-intensive causing a slower convergence.

TanH Activation Function It is similar to the logistic function (by a mathematical transformation), the function produces outputs on the range of $[-1, 1]$. Unlike the logistic activation function, the TanH is a zero-centred function. So it overcomes some of the logistic function disadvantages, still, it causes the gradient to vanish, as the logistic function. Fig. 3.6.b present the function graphic presentation.

ReLU Activation Function The rectified Linear Unit is the most used function because of its simplicity. This function presents several advantages, it is computationally efficient, allowing the network to quickly converge. Besides, the function acts as a linear function (see Fig. 3.6.c). But still, really is not a zero-centered function.

Softmax Activation Function This function is a multivariate function computing the probabilities distribution of the event over 'n' different events, calculating each target class probability over all possible target classes. Equ. 3.3.4 perform the softmax formulation. The main advantage of using this specific function is the output probabilities range such as the output range is in $[0, 1]$ and the sum of all probabilities will be equal to one, giving the softmax function the ability to handle multiple classes.

$$f(x_i)_{i=1,...,n} = \frac{e^{x_i}}{\sum_{j=1}^{n}} \tag{3.3.4}$$

Swish Activation Function Swish is a new, self-gated activation function discovered by researchers at Google [91] it is a smooth, non-monotonic function that performs better than ReLU with a similar level of computational efficiency. The graphic presentation of the Swish activation function is performed in Fig. 3.6.d.

The choice of using one of the presented activation functions is based on the ANNs architecture and the target application. Fig. 3.6 present different mentioned non-linear activation function.

3.3.2 MLPNN Training

The learning phase is very important to the development of the network, this latter controls the network behavior such as it is modified, in order to obtain desirable behavior. Moreover, the learning methods are conventionally divided into three categories:

Supervised Learning; requires a priori information of the problem answers as labeled examples called training set such as $D_t = \{(x_1, d_1), (x_2, d_2), (x_3, d_3), ..., (x_N, d_N)\}$ where x_i designed an stimulus (input) and d_i the stimulus target called *desired output*, each couple (x_i,d_i) of the training set corresponds to the expected network

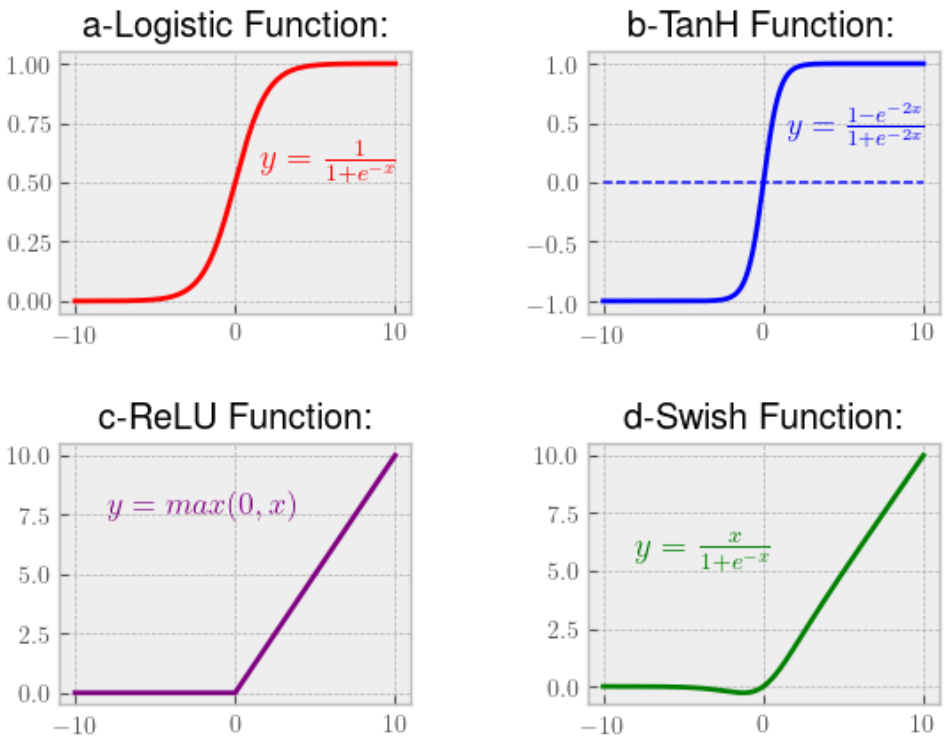


Figure 3.6: Diferent Non-Linear Activation Functions

behavior for a given stimulus. The supervised learning adjusts the network parameters by a direct comparison within the actual network output (y_i) and the expected output (d_i). The error measure used to guide the learning process is usually defined by the mean squared error (MSE):

$$E = \frac{1}{N} \sum_{i=1}^{N} \|d_i - y_i\|^2 \tag{3.3.5}$$

Where N is the training set size and y_i is the network output of the pattern i. The learning process is achieved when the Equ. 3.3.5 is sufficiently small or a failure criterion is met.
Supervised learning [69] is widely used in classification, approximation, control, modeling and identification, signal processing and optimization.
Unsupervised Learning; The said category requires no target values. It tries to describe the associations and patterns among a set of input measures. Moreover, only the features are observed, in addition, the outcomes aren't measurable. Mainly, the learning task is more about describing the data are organization or clusterization. Unsupervised learning or "learning without a teacher" [8] is completely based on the correlations among the input data and is used to find the significant patterns or features in the input data without the help of a decider or guider. This type of learning is mainly used for clustering, vector quantization, feature extraction, signal coding, in addition to data analysis.

Reinforcement learning; The paradigm can be considered as a special case of supervised learning, where the exact desired output is unknown. The trainer supplies only feedback about the success or failure of an answer. Reinforcement learning [114] is trying to maximize a reward signal instead of trying to find a hidden structure. Moreover, the learning procedure rewards the neural network for its good output result and punishes it for the bad output result. Reinforcement learning is usually used in control and artificial intelligence.

The multilayer perceptron learning is under the supervised learning paradigm. However, MLPNN training can be presented as an optimization problem. The coming sections will cover the different proposed methods and technics for the multilayer perceptron learning.

3.3.2.1 Learning by optimization

However, attaining an adequate objective function is the only matter, which represents at best the weights selections. Once the optimization model is drawn up, there are many off-the-shelf algorithms ready to employ, whether its classical methods such as back-propagation [97] or either sophisticated optimization approach such as genetic algorithm [32] the MLPNN model can operate it. Before going further, let get a look at the most used learning algorithm, the back-propagation learning algorithm.

Back-propagation learning algorithm; The algorithm was first proposed by Werbos [123] in 1974, to be next exploited by Rumelhart et al. in the cognitive process

simulating. The back-propagation learning algorithm achieves a gradient descent search in weights space looking for the error function (MSE) minimum. The algorithm 4 presents the main steps oh the back-propagation algorithm.

Algorithm 4: Back-propagation learning algorithm

Require: x^i: Input of the example i
d_i: Desired output of the example i
y_i: Network output for the x^i input
D_t: Training set
N: Training set size
$f(x)$: Activation function
S_i^L: The output of the i^{th} neuron of the L^{th} layer
it: Iteration number
it_{max}: Maximum of iteration number
E_t: pre-specified error threshold
E: Error function

1: Rondomly initialize the network weights by small values
2: Set it=0
3: **repeat**
4: **for** each $x^i \in D_t$ such as $i \in \{1, ..., N\}$ **do**
5: Propagate the signal forwardly through the network
6: Compute: $\delta_i^L = f'(S_i^L)(d_i - y_i)$; // δ_i^L in the output layer
7: Compute: $\delta_i^k = f'(S_i^k) \sum_j w_{ij}^{k+1} \delta_j^{k+1} \; \forall k = L-1, \ldots, 1$; // δ_i^k preceding layers
8: Update weuights as: $\Delta w_{ij}^k = \eta \delta_i^k y_i^{k-1}$
9: **end for**
10: $it = it + 1$
11: **until** $E == E_t \,||\, it \geq it_{max}$

The back-propagation learning algorithm has proven to be a robust algorithm; however, the algorithm converges slowly. Besides, the aforementioned algorithm, as an optimization algorithm, allows obtaining information about the optimal values of the network weights, still, the algorithm doesn't provide information on the network complexity.

Moreover, in literature, many approaches were proposed for the MLPNN training, using the mean square error (MSE) as an objective function for the MLPNN optimization model. Nevertheless, minimizing only the training error can yield to overfitting models. Plus, other works introduce constraints to the optimization model, aiming to control the number of connections and neurons of the network [93]-[15]. This process produces an optimized neural network architecture. However, the network smoothness still to ameliorate. For this reason, the researchers propose to regularize the network by including additional information to avoid overfitting, the process consists of adding a penalty term with a regularization parameter (λ) to the MLPNN training error, representing the network complexity. During the

process, models with small coefficients are privileged [16]-[92]. Before exploring more the MLPNN learning problematics, first detailing the regularization theory.

3.3.2.2 Regularization

Regularization is a technique used in finding a stable solution when a parameter identification problem is exposed to considerable errors. Moreover, If an objective function is complex, even a small change of the parameters may lead to a significant change to the value of the objective function. The regularization contributes to the stabilization of such a function. A network regularization looks for controlling the network variation and complexity by controlling the network weights values. Bartlett [9] shows that the values of the weights were more important than their number in the network in order to obtain an adjusted neuron network model. Several methods were proposed for the network regularization, we may present:

Early Stopping ; Since the supervised learning consists of minimizing the training cost function (MSE), the early stopping technique [90] proposes to stop the training algorithm steps just before the algorithm convergence such as the model isn't finely set to the training set and then the over-fitting is limited.

Weight Decay ; The weight penalty technique consists of introducing a cost function to training error in order to overcome the network overfitting.

$$E = E + \lambda\Omega \tag{3.3.6}$$

Such as λ is a hyperparameter that controls the regularization strength and Ω is the regularization term. In the following, we present the most popular regularization terms.

L_1 **norm or LASSO regularization**; The LASSO attempt to minimize the sum of the absolute difference between real values and predicted values. The LASSO regulazation offers the possibility to easily select the characteristics by forcing a sparse representation. The regularization is presented as:

$$\Omega_{L_1} = \sum_k |w_i| \tag{3.3.7}$$

where k is an index summing up all weights.
L_2 **norm or Tikhonov regularization**; The Tikhonov [118] Attempt to minimize the sum of the squared difference between real values and predicted values. This term is faster than the LASSO regularization, however, it is not able to drive small irrelevant weights to zero. The Tikhonov regularization is presented by:

$$\Omega_{L_1} = \sum_k w_i^2 \tag{3.3.8}$$

Both regularization terms in equation (Equ. 3.3.7) and equation (Equ. 3.3.8) have also been studies regarding Bayesian learning, known as the LASSO regularization and the Gaussian regularization.

Regarding the hyper-parameter λ, known also as the regularization parameter, it's controlling the trade-off between the network smoothness and the training error. Although, the parameter value isn't significant and is usually fixed according to the user choice. Moreover, the specification of λ becomes a fundamental issue in the regularized parameter identification. In other words, the scalarized approach [85], [87] is applied to optimize the training error and network complexity. This approach attempts to reduce multi-objective optimization problems to single-objective, where is more easy to find a compromise between several objectives as this approach allows to use of classical optimizer. Scalarizing objectives have his weakness; the determination of the appropriate hyper-parameter (λ) isn't an insignificant problem. The above-mentioned parameter needs to reflect properly the purpose of the user. Such localization is infrequent, especially when arbitrary choices are applied. Moreover, using a scalarized approach, limit the variety of solutions since only a single solution can be obtained. Accordingly, the problem insight isn't fully explored. For the principal cited reasons and others, new approaches have been considered.The following section covers a new technique called multi-objective learning.

3.3.2.3 Multiobjective Learning

Machine learning is mostly multi-criteria. Since, in supervised learning, several objectives should be considered. The learning model must obtain, not merely, a beneficial approximation performance regarding the data learning, although on unseen data from the same problem. Considering the above-mentioned weakness of the classical methods to deal with the multi-objectivity of machine learning a new method was explored such as Pareto-based approach [53].

Problem formulation; To avoid the use of a weighting factor, in this case, the regularization hyperparameter λ, it is more wisdom to find solutions that do not depend on the hyperparameter value by entirely removing him from the problem formulation. This results into a multi-objective formulation and the objective of the problem is thus expressed as:

$$f(w)^T = [E(w), \Omega(w)] \tag{3.3.9}$$

The regularization problem formulated as a multi-objective optimization problem is conclusively characterized by:

- The problem is multi-objective,
- the objective functions $f(x)$ are continuous but can be complex such as non-convex,
- the problem is subject to equality constraints and inequality constraints,
- the search space is continuous.

Equally, the problem solution is a set of solutions called the Pareto front [84], which represents the results of different tradeoffs. The solutions are achieved using the

Pareto optimality, this concept is used to compute between the feasible solutions. Once the learning is achieved, the user's needs to pick the most adequate solution our more according to the user performance, by doing, the user can reach a better preference whenever selecting the final solution. Besides, the Pareto-based approach may spare the learning algorithm of local optima. More papers are calling multiobjective approach to deal with MLPNN learning, the first proposed model in this context was by G. Liu and V. Kadirkamanathan [70], proposed to consider three different performance indices as objective function of the multiobjective model. Furthermore, Ricardo H. C. Takahashi et al [5] have balanced the training error (MSE) and connection weights during training to improve MLPNN generalization. Although in [13], the authors proposed the Multi-OBJective training (MOBJ) algorithm, composed of the two steps, the first one aims to find the Pareto-optimal solution and the second one is a decision step to choose the adequate solution from the Pareto-optimal set, using a set of validation data. In 2006, Costa, M. A., & Braga, A. P. [24] proposed to optimize simultaneously the MLPNN generalization performance and weight selection by an algorithm considering the linear last absolute shrinkage and selection operator (LASSO). Furthermore, Jin, Y., & Sendhoff, B. [54] has given a review of the current research on multiobjective machine learning, outlining the supervised learning. In the same context, we must consider our previous works [57]-[108], where we introduced new constraints and a new variable decision to the multiobjective MLPNN training model.

3.4 Continous Hopfield Network

Hopfield neural network is a recurrent neural network proposed by Dr. John J. Hopfield in 1982 to solve several optimization problems such as traveling salesman problem [45], also, used for auto-association problem. Since this latter is an energy minimization network; there is no "input" or "output", it is a fully interconnected neuron; the HNN use Lyapunov function for the activity dynamics such as the state of the system evolves from an initial state to a final state, the (local) minimum of the Lyapunov function. The network can be characterised by the network number neurons n, an activation function f and symmetric transfer Matrix W. The continuous or continuous-time HNN use sigmoid or hyperbolic tangent functions as common output functions. In this section, we present the so-called continuous Hopfield network.

3.4.1 Continuous Hopfield Network Architecture

As mentioned before, the Hopfield neural network is a recurrent network. Fig. 3.7 represent a continuous Hopfield network with 3 neurons. The output of each neuron is looped back to the inputs of the other neurons. In addition to a noise vector coming from outside noted I^b. Besides, each neuron i is connected to neuron j by symmetric weight $w_{ij} = w_{ji}$, however, the neuron isn't self-connected $w_{ii} = 0$.

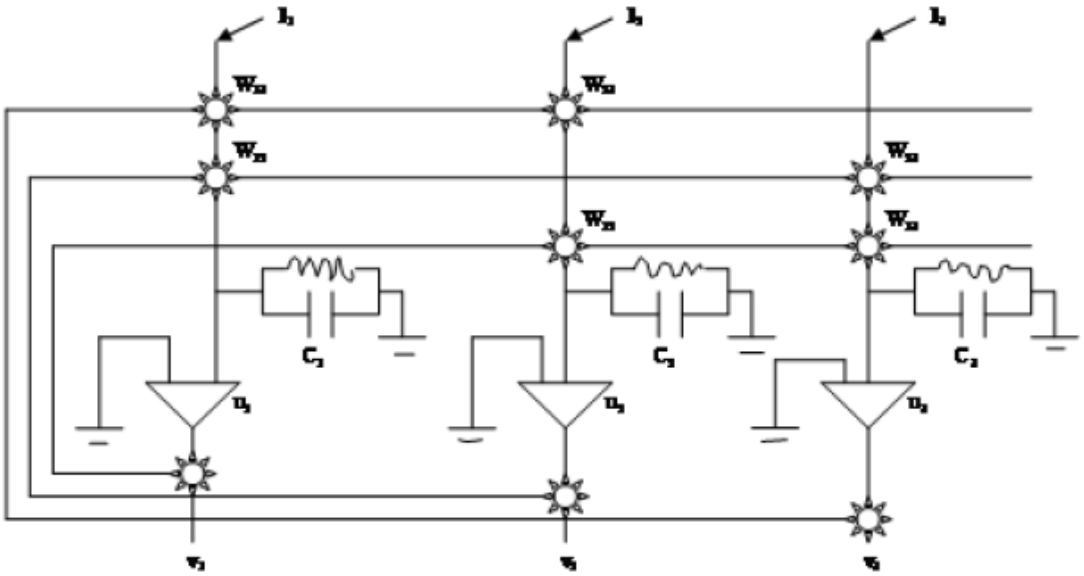


Figure 3.7: Electronic representation of Hopfield neural network [11]

The network units are neurons with a continuous activation function, such as u_i is the internal potential calculated as the sum of the weighted information of other neurons and the output x_i. The continuous activation function is presented by:

$$\forall i \in \{1, ..., n\}, \qquad x_i = g(u_i) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i}{u_0}\right)\right) \qquad with\ u_0 > 0 \qquad (3.4.1)$$

The state of each neuron i at time t is given by a real value $u_i(t)$. This value is calculated as the sum of the information x_j from of the other neurons weighted by w_{ij}, this sum is disturbed by noise I_i coming from the outside. Fig. 3.7 presents the based electronic Hopfield model composed of resistors, non-linear amplifier, and an outside current. Starting from an initial state $u_0 \in \mathbb{R}$, the network global dynamic is defined by a differential equation system:

$$C_i \frac{du_i}{dt} + t_i u_i = \sum_{j=1}^{n} w_{ij} x_j + I_i \qquad (3.4.2)$$

such as :
w_{ij}: Conductance between amplifier i and j,
t_i: Conductance in the output of each amplifier i,
C_i: Capacity at the amplifier input i,
u_i: Voltage at the amplifier input i,
x_i: Voltage at the amplifier output i.
The voltage x_i and u_i are related by the Equ. 3.4.1. Furthermore, we suppose that:

$$\forall (i,j) \in 1, ..., n^2 \quad C_i = C, \quad t_i = t, \quad T_{ij} = \frac{w_{ij}}{C}, \quad I_i^b = \frac{I_i}{C}, \quad and \quad \tau = \frac{C}{t} \quad (3.4.3)$$

Therefore the differential equation of the continuous Hopfield network became:

$$\frac{du}{dt} = -\frac{u}{\tau} + Tx + I^b \tag{3.4.4}$$

The system 3.4.4 is called the state equation of the Hopfield network. The vector x, u and I^b are, respectively, the output vector, the state vector, and the outside noise vector. Moreover, T represents the weights matrix.

3.4.2 Hopfield Energy and the Equilibrium State

As seen in the previous section, the network state change. In this section, the network energy function, associated with the network state equation particular, called Lyapunov function [18] is studied, in particular, the existence of an equilibrium state [115]. The Lyapunov stability is a mathematic form for an elementary finding: if the net energy increase in time then the system converges to an equilibrium state. The direct method seeks to generate a scalar function of an energetic type which admits a negative temporal drift.

Definition 3.4.1 *The vector $u(0)$ presents the initial state of the Hopfield neural network. The vector u^e is called stability point of the Hopfield neural network if it exists a moment t^e such as*

$$\forall t \geq t^e \quad u(t) = u^e \tag{3.4.5}$$

So the existence of an equilibrium point is closely linked to the existence of an energy function. Hopfield showed in [44], by a Lyapunov function argument, that the state dynamics differential systems of its network are globally asymptotically stable, i.e. from any initial state, the system converges to a stable state. Indeed, to prove the network stability point, Hopfield proposed the following function defined on the hypercube $H = [0, 1]^n$.

$$E(x) = -\frac{1}{2}x^t Tx - \left(I^b\right)^t x + \frac{1}{\tau}\sum_{i=0}^{n} g^{-1}(x)dx \tag{3.4.6}$$

Note that the function $E^(x)$ is , therefore the global state evolution of the system is a displacement in the states space looking for a minimum, which can be a local minimum of E.

Theorem 3.4.1 (Lyapunov theorem) *If for a differential system, the associated energy function, decreasing in time, exists such asLyapunov $\frac{E(u(t))}{dt} = 0$ for $\frac{du(t)}{dt} = 0_{\mathbb{R}^n}$, the mentioned system is stable. That means that $u(t)$ converge to a stable point.*

The Hopfield network may be used to resolve difficult optimization problems. However, the resolution requires the construction of an adequate energy function. the following theorem gives the correct energy function form associated with the Hopfield system 3.4.4.

Theorem 3.4.2 *If T is a symmetric matrix, the function 3.4.6 is then the energy associated with the system 3.4.4, supporting the Lyapunov theorem condition 3.4.1.*

Hopfield has determined the network energy E, which measures the network of the thermodynamic system. In this way, the minimum energy of the Hopfield network corresponds to an equilibrium of this network. More specifically, Hopfield has shown that energy E can only decrease or remain stable at a local minimum. Also, Hopfield proved that the symmetry of the weight matrix is a sufficient condition for the existence of the Lyapunov function [44].
A point $x^e \in H$ can be the Hopfield network equilibrium point, if and only if, the three following conditions are satisfied [117]:

$$E_i(x^e) = \frac{\partial E_i(x^e)}{\partial x_i} \geq 0 \quad \forall i \in \{1, ..., n\} \quad such \quad as \quad x_i^e = 0 \tag{3.4.7}$$

$$E_i(x^e) = \frac{\partial E_i(x^e)}{\partial x_i} \leq 0 \quad \forall i \in \{1, ..., n\} \quad such \quad as \quad x_i^e = 1 \tag{3.4.8}$$

$$E_i(x^e) = \frac{\delta E(x^e)}{\delta x_i} = 0 \quad \forall i \in 1, ..., n \quad such \quad as \quad x_i^e \in]0, 1[\tag{3.4.9}$$

The Hopfield networks are energy minimizer networks. They can solve certain optimization problems. Such a problem is specified by the set of inputs supplied by the network, these inputs correspond to an initial state from which the network evolves and the constraints of the problem represent the network connections. While the problem solution represents the stable state reached by the network, it corresponds to a minimum energy configuration of the network. The stable state is the one that best satisfies the problem constraints. When the parameter τ take large values in the energy function 3.4.6, the network CHN can solve the combinatorial optimization problems, which we can extract a quadratic energy function from:

$$E(x) = -\frac{1}{2}x^t T x - (I^b)^t x \tag{3.4.10}$$

Furthermore, the network may be trapped in a local minimum which is not necessarily the global minimum. The energy function extremums are among the vertices of the hypercube $H = [0, 1]^n$. The continuous Hopfield network is faster and more realistic compared to the binary version of the network.
The energy function quadratic form 3.4.10 will be used to resolve, thereafter, the multiobjective problem addressed in chapter 4.

3.5 Conclusion

The presented chapter discussed the different artificial neural network types and architectures, the feedforward and recurrent neural network being neural network variance in several application fields. The chapter mainly focused on the multilayer perceptron, since this letter is one of the most efficient neural networks and the

most used one in the classification and approximation. In this context, the chapter developed the essential step in the MLPNN efficiency: the network training. Indeed, different methods have been proposed to increase the MLPNN efficiency by improving the MLPNN training journey. The most popular algorithm is the Back-propagation learning algorithm. However, many approaches were developed seeing MLPNN training as an optimization problem to then be resolved by on the shelf optimization method.

The second network discussed, in the chapter, is the recurrent neural network: continuous Hopfield network. As well, the architecture and different attributes of CHN were widely detailed, especially, the quadratic form of the network energy function. Since the network is mainly used in the resolution of optimization problems by presenting the problem objective function and constraints on the network energy function. The optimization is assured by the existence of network stability and the equilibrium point coincides with the optimal solution of the optimization problem.

Furthermore, different neural networks exist and developed in the literature, we have chosen to mention the artificial neural network related and the more relevant to the theses contributions. The next chapters are dedicated to the contributions parts.

Chapter 4

Neural-Metaheuristics approach for portfolio selection problem

4.1 Introduction

Portfolio management is a very important problem in the econometric field. He gives the answer to the investor for his most difficult question, how to allocate his capital between many actives? The first model was the Markowitz model or Mean-variance model in 1952 [75] [51]. Markowitz was the pioneer of the first rigorous treatment of the investor's dilemma. In this chapter, we propose new models, by adding a new constraint to the Markowitz's models, to avoid to consider the assets of a negative return. Considering its effectiveness, a continuous Hopfield network is used to solve the proposed models. In this regard, we construct an original energy function that makes a compromise between the risk, profit and cardinality constraints. To ensure the equilibrium point feasibility, the parameters of the energy function are chosen based on a consistency, mathematical results; besides, the slop of the activation functions is chosen such that the behavior of each neuron is almost leaner. The proposed method produces better solutions. However, the proposed energy function seems to be linear at a point due to the behaviour of the risk aversion parameter. to avoid the aforementioned dilemma, we propose a hybrid approach, basing basing on a continuous Hopfield network (CHN) and Genetic Algorithm (GA). This latter product better solution compared to literature methods, basing on real financial data.

4.2 Portfolio selection problem Models

The most important problem that financial sectors face is selecting an optimal portfolio. this latter can be defined as picking assets providing the maximum return at a specific risk level or the minimum risk at a specific return level. The first model for the portfolio selection problem was proposed by Markovitzch in 1952 [75] called Mean-variance model [6]. Moreover, the model aims were to reach a lower level risk with an expected return. The Markowitz model uses the variance of the assets return

as a risk measure and the mean return to estimate the portfolio profit, thence his name. However, Jarrow in 1988 [51] introduces a general equilibrium model among the level risk and the portfolio return called the mean-variance efficient frontier. During the research, we were more interested in improving the equilibrium model. Since this latter is originally a multiobjective model transformed using the weighted sum method. However, the section discusses different strategies for the portfolio selection problem.

4.2.1 Introduction

4.2.1.1 Rentability

Profitability is a fundamental concept in finance and appears in the expression of most portfolio management models.

Definition 4.2.1 *The profitability measures the relative appreciation (or depreciation) of financial asset value or a financial assets portfolio between two successive instants [4].*

Let t and $t+1$ be two successive moments. We note P_t and P_{t+1} the values (price) of the asset at times t and $t+1$, respectively. The performed rentability in the interval $[t, t+1]$ can be represented by:

Arithmetic Profitability A value called arithmetic profitability all profitability observed in $[t, t+1]$ the value defined by:

$$r_t = \frac{P_{t+1} - P_t}{P_t} \tag{4.2.1}$$

Logarithmic profitability A value called Logarithmic profitability all profitability observed in $[t, t+1]$ the value defined by:

$$r_t = \ln(\frac{P_{t+1} - P_t}{P_t}) \tag{4.2.2}$$

Many financial models use historical data series for stock prices to estimate the stochastic properties of the corresponding profitability such as the average profitability. Moreover, several methods are available. The presented one is the estimation of the subperiod returns, then the following formula is used.

$$\overline{r} = \frac{1}{T}\sum_{1}^{T} r_t \tag{4.2.3}$$

The hypothesis that the returns on financial assets are normally distributed is often made in the financial literature. But it does not correspond to reality. In general, profitability in the markets does not follow a Gaussian distribution. In this case, two other stochastic properties come into play, these are Skewness and Kurtosis. In this chapter, we consider the stock price returns possess normal and identically distributed probability densities.

4.2.1.2 Risk measure

Variance The variance, Var, is a classical measure of risk. It is well known that Markowitz is the first researcher to use the variance as the measure of risk [75], [76], for the first time, the problem of portfolio selection was clearly developed and solved. The proposed model was based on two notions of probability, the mean returns and the variance between returns. Moreover, The Markowitz model tries to share the richness among assets while reducing portfolio variance.

Definition 4.2.2 *Let f be the density function of a random variable r, we define the variance of r by*

$$Var(r) = \sigma^2(r) := \int_{-\infty}^{\infty} (r - E(r))^2 f(r) dr \tag{4.2.4}$$

Where E is the mathematical expectation operator and r is a continuous variable. For the case where r is a discrete variable, the variance of r is defined by

$$Var(r) = \sigma^2(r) := \sum_{r} (r - E(r))^2 f(r) \tag{4.2.5}$$

Or the sum is performed on all possible values of r.

If r is the portfolio return, the return variance will be the standard deviation square performance against the mean return. Assume a list of historical returns is available. The list contains the returns historical records of n assets during m periods. Considering the aforementioned list and Equ. 4.2.3 we define the average return of each asset using the following formula:

$$r_i := \frac{1}{m} \sum_{j=1}^{m} r_{ij} \tag{4.2.6}$$

Where r_{ij} is the return of asset i in period j such as $i = 1, ..., n$ and $j = 1, ..., m$. Let $x_1, x_2, ..., x_n$ the n assets at risk that form the portfolios. Moreover, we suppose that $x_i \geq 0$ and $\sum_{i=1}^{n} x_i = 1$ and the variables x_i coresponds to the randoms return $r = (r_1, r_2, ..., r_n)^t$. The portfolio return is then $R = x^t r$ and the portfolio risk is defined by $\sigma^2 = x^t Q x$. such as Q is the variance-covariance matrix, were the element (i,j) is estimated by the formula 4.2.7.

$$\sigma_{ij} = \frac{1}{m} \sum_{k=1}^{m} (r_{ik} - r_i)(r_{jk} - r_j) \tag{4.2.7}$$

The use of variance as a risk measure has its advantages. The simplicity can be the first advantage of this measure, besides, using the matrix of variance-covariance, we can build a quadratic programming model convex [65] to calculate efficient portfolios. However, approaching the risk by the variance presents some disadvantages. Since the variance penalizes similarly gains and losses that are too far from the average [65]. In this context, alternative models have been developed to outcome the MV model disadvantages.

Semi-variance In 1959, Markowitz realizes the limits of the variance s risk measure, he then proposed the semi-variance. This measure takes into consideration only deviations below a goal. The aforementioned measure is classified as a downside risk measure, this class of risks measure are also useful when returns follow a non-Gaussian distribution. Mathematically the semi-variance measure is defined as follows [34]:

$$\Sigma_T := \sum_r ([r - T]^-)^2 f(r) \tag{4.2.8}$$

Such as T is the target et f is the density function of random return. Markowitz assumes that T is the mean return, E(r), hence the name semivariance, in this context the formula 4.2.8 becomes:

$$\Sigma_T := \sum_r ([r - E(r)]^-)^2 f(r) \tag{4.2.9}$$

The semivariance takes into account only the deviations associated with the values of r which are lower than the expectation of returns. However, The semi-covariance matrix is more difficult the define, Hogan and warren [42] define the semicovarience between assets i and j as:

$$\Sigma_{ij}^{HW} = E[(r_i - r_f).min(r_j - r_f, 0)] \tag{4.2.10}$$

Where the superscript HW indicates that this definition proposed by Hogan and Warren and r_f is the expected return. Nevertheless, this definition has two drawbacks:

- The benchmark return is limited to the risk-free rate and cannot be tailored to any desired benchmark;
- the semi-covariance matrix is usually asymmetric ($\Sigma_{ij}^{HW} \neq \Sigma_{ji}^{HW}$), This characteristic is particularly limiting. Besides, it is not clear the contribution interpretation of assets i and j to the portfolio risk.

to come over the aforementioned characteristics, Estrada [28] proposed to define the semicovariance between assets i and j with respect to benchmark T as:

$$\Sigma_{ijB} = E[Min(r_i - B, 0).min(r_j - B, 0)] = \frac{1}{T}\sum_{t=1}^{T}[min(r_{it} - B, 0).min(r_{jt} - B, 0)] \tag{4.2.11}$$

According to Estrada, this definition can be tailored to any desired B and generates a symmetric and exogenous semicovariance matrix.Or, by using the semi-variance as risk measure, the portfolio risk well be presented by:

$$\Sigma_{pB}^2 = \frac{1}{T}\sum_{t=1}^{T}[Min(R_{pt} - B, 0)]^2 \tag{4.2.12}$$

Where R_{pt} denotes the returns of portfolio and Σ_{pB}^2 their semivariance. Considering the formula 4.2.12, the semi-covariance matrix, according to Estrada, is endogenous;

that is, a change in weights affects the periods in which the portfolio underperforms the benchmark, which in turn affects the elements of the semi-covariance matrix. In 2008, Estrada proposed an alternative heuristic [29] to use the semivariance risk measure supporting the variance characteristics. The proposed heuristic allows solving all mean-semivariance problems with the same well-known closed-form solutions widely-used to solve mean-variance problems. More precisely, the semivariance of a portfolio with respect to a benchmark B can be approximated with the following expression:

$$\Sigma_{pB}^2 \approx \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \Sigma_{ijB} \tag{4.2.13}$$

This expression 4.2.13 returns symmetric and exogenous semivariance matrix, according to Estrada, the formula can be used likewise the covariance matrix employed in the mean-variance problems.

4.2.2 Mean-variance Model

Markowitz approach, so-called medium-variance approach, is aiming to minimize the risk of the portfolio by fixing the minimum return expected by the investor or vice versa, in other words: Maximize the expected return by setting the minimum risk desired by this investor.Through the notations previously presented, the portfolio return and level risk can be presented by:

Portfolio Risk

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \sigma_{ij} \tag{4.2.14}$$

portfolio Return

$$\sum_{i=1}^{n} x_i r_i \tag{4.2.15}$$

Such as x_i refers to the proportion of capital C invested in the asset a_i identified by its uncertain performance r_i. In this context, the mean-variance model is defined as:

$$(MV)\begin{cases} \min\limits_{x} & \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \sigma_{ij} & & & \\ \text{s.t.} & \sum_{i=1}^{n} x_i r_i & \geq & R & \\ & \sum\limits_{i=1}^{n} x_i & = & 1 & \\ & x_j & \geq & 0 & \forall i \in 1,..,n \end{cases} \tag{4.2.16}$$

Such as R is the minimum return expected on the portfolio, the hyperparameter is controlled by the investor. Besides, the constraint $\sum_{i=1}^{n} x_i = 1$ is the assignment constraint, since the capital proportion is limited by the invested capital.
The (MV) model is a quadratic programming problem that creates a feasible mean-variance combination.

4.2.3 Efficient Frontier

efficient portfolio every portfolio that allows the highest expected return for a given level of risk, or reversely, each portfolio that presents the lowest risk for a given expected return.
The set of efficient portfolios form a border that Markowitz called "Efficient frontier". Fig. 4.1 illustrates the efficient frontier shape. However, the portfolio selection is an

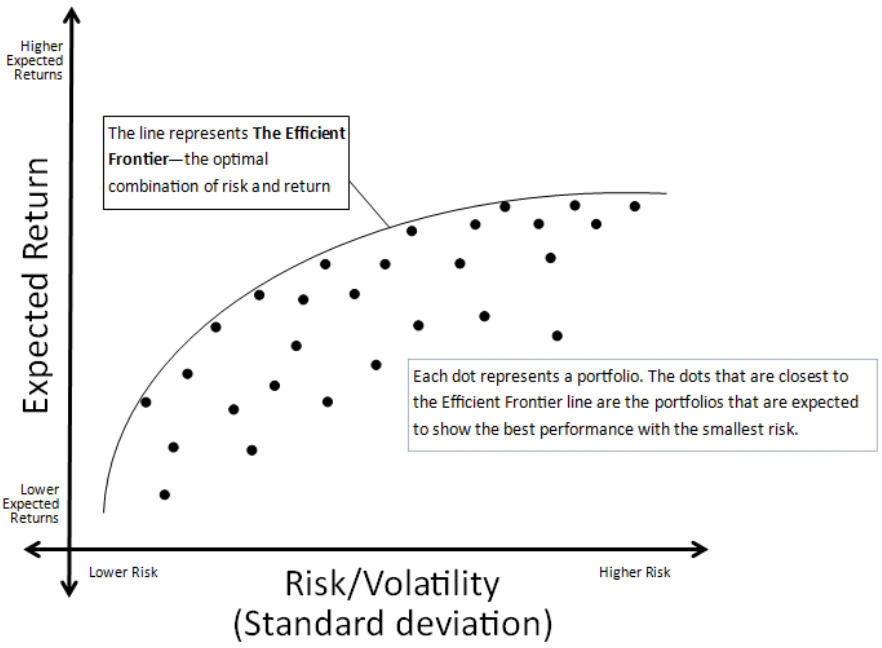


Figure 4.1: Portfolio Efficient Frontier [89]

instance of a multiobjective optimization problem family, aiming to balance the risk level and the portfolio return. This balance is more knows as Pareto front. Moreover, Jarow [51] proposed to use a parameter λ called aversion risk to achieve the balance such as the two objectives (min risk and max return) are combinate on a mono-objective model, this technic is known as the weighted sum method [77]. The resulting model is presented as :

$$\begin{cases} \min\limits_{x} & \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \sigma_{ij} & - & (1-\lambda) \sum_{i=1}^{n} x_i r_i & \\ \text{s.t.} & & & & \\ & \sum\limits_{i=1}^{n} x_i & = & 1 & \\ & x_j & \geq & 0 & \forall i \in 1,..,n \end{cases} \tag{4.2.17}$$

Where λ is aversion risk parameter such as $\lambda \in [0,1]$. for $\lambda = 1$ the situation corresponds to the risk-averse investor. However, for $\lambda = 0$, the situation becomes an investor having no fair to take the risk. Furthermore, for $\lambda \in [0,1]$ the efficient frontier can be performed.

4.3 Proposed Rational Models

In this part, we are interested in the efficient frontier model, since this latter is a multiobjective and quadratic model. However, we propose to add a new constraint to the model aiming to reduce the negative assets return and avoiding the negative influence in the portfolio, besides to avoid losing in the investment capital. Then the return can reach his maximum and the risk the lowest level. The constraint is expressed by :

$$\sum_{j \in I} x_j = 0 \tag{4.3.1}$$

Such as I is the set of negative assets return formulated by:

$$I = \{j \mid r_j \leq 0\} \tag{4.3.2}$$

The portfolio selection problem can be expressed as:

$$\begin{cases} \min\limits_{x} & \lambda\theta_r & - & (1-\lambda)\sum_{i=1}^{n} x_i r_i & \\ \text{s.t.} & & & & \\ & \sum\limits_{i=1}^{n} x_i & = & 1 & \\ & \sum_{j \in I} x_j & = & 0 & I = \{j \mid r_j \leq 0\} \\ & x_j & \geq & 0 & \forall i \in 1, .., n \end{cases} \tag{4.3.3}$$

Such as θ_r present the risk measure. Firstly, we use the variance as a risk measure to solve the problem, then we adopted the Estrada semi-variance [29]. The emerging models are called rational models [103], [107]. Nevertheless, the problem was solved with different methods. In the next sections, we propose different approaches for the proposed rational model, basing on a neural tool approach and metaheuristics.

4.4 Continuous Hopfield Network for Rational Models

In this part, we propose an adaptative continuous Hopfield network, detailed in section 3.4, to solve the proposed models for the portfolio selection problem. The approach must make a compromise between the objective function and the constraints. For that, different steps are recused, the complete proposed adaptation is detailed in the following:

4.4.1 Energy function

To solve the (PM) via the CHN, we propose the following energy function:

$$E(x) = \alpha\frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \sigma_{ij} - \frac{\alpha}{2}(1-\lambda)\sum_{i=1}^{n} r_i x_i + \beta_1 \tau^t x + \beta_2 (\tau^t x)^2 + \varphi_1 \tau_I^t x + \varphi_2 (\tau_I^t x)^2 \tag{4.4.1}$$

Where

$$\tau^t = (1, ..., 1)^t \in \mathbb{R}^n \tag{4.4.2}$$
$$\tau_I^t = (\delta_1, ..., \delta_n)^t \in \mathbb{R}^n \tag{4.4.3}$$

such that

$$\delta_i = \begin{cases} 1 & if \quad i \in I \\ 0 & else \end{cases} \tag{4.4.4}$$
$$I = \{ i \mid r_i \leq 0 \ \} \tag{4.4.5}$$

The variables α, β_1, β_2, φ_1 and φ_2 are the penalty parameters of the CHN energy function. As well, The gradient function of the CHN energy function can be expressed by:

$$\nabla E(x) = \alpha\lambda Qx - (1-\lambda)\alpha r + \beta_1\tau + \beta_2\tau(\tau^t x) + \varphi_1\tau_I + \varphi_2\tau_I(\tau_I^t x) \tag{4.4.6}$$

from the obtained formula (Equ. 4.4.6), the bias is given by:

$$-\nabla E(0) = \alpha(1-\lambda)r - \beta_1\tau - \varphi_1\tau_I \tag{4.4.7}$$

However, we have also:

$$\nabla^2 E(x) = \alpha\lambda Q + \beta_2\tau\tau^t + \varphi_2\tau_I\tau_I^t \tag{4.4.8}$$

Basing on the Equ. 4.4.8, the weight matrix is geven by:

$$T = -\alpha\lambda Q - \beta_2\tau\tau^t - \varphi_2\tau_I\tau_I^t \tag{4.4.9}$$

The weight matrix and the bias vector depend on the penalty parameters; to ensure the feasibility of the equilibrium point, we use a mathematical method, seen in section 4.4.3, to select adequate weights and bias.

4.4.2 Activation function

As the portfolio problem looks for continuous values, the activation function must be chosen such that every value in $[0, 1]^n$ has the same probability to be taken. Fig. 4.2 gives the plot of the function $f(x) = \arctan(Tx)$ for several values of the slope T under $[a, b]$. The best slop is the one overspending to

- Almost linear plot
- Almost surjective under $[a, b]$

Firstly, we must define the set $[a, b]^n$ that contains every possible value of the neuron states noted U_i. As it is known, the state U_i is given by $U_i = T_i x + I_i$ where T_i is the vector line of the matrix T. In this sense, verify the following equality

$$\sum_{j=1}^{n} min(0, T_{ij}) + I_i \leq U_i \leq \sum_{j=1}^{n} max(0, T_{ij}) + I_i \tag{4.4.10}$$

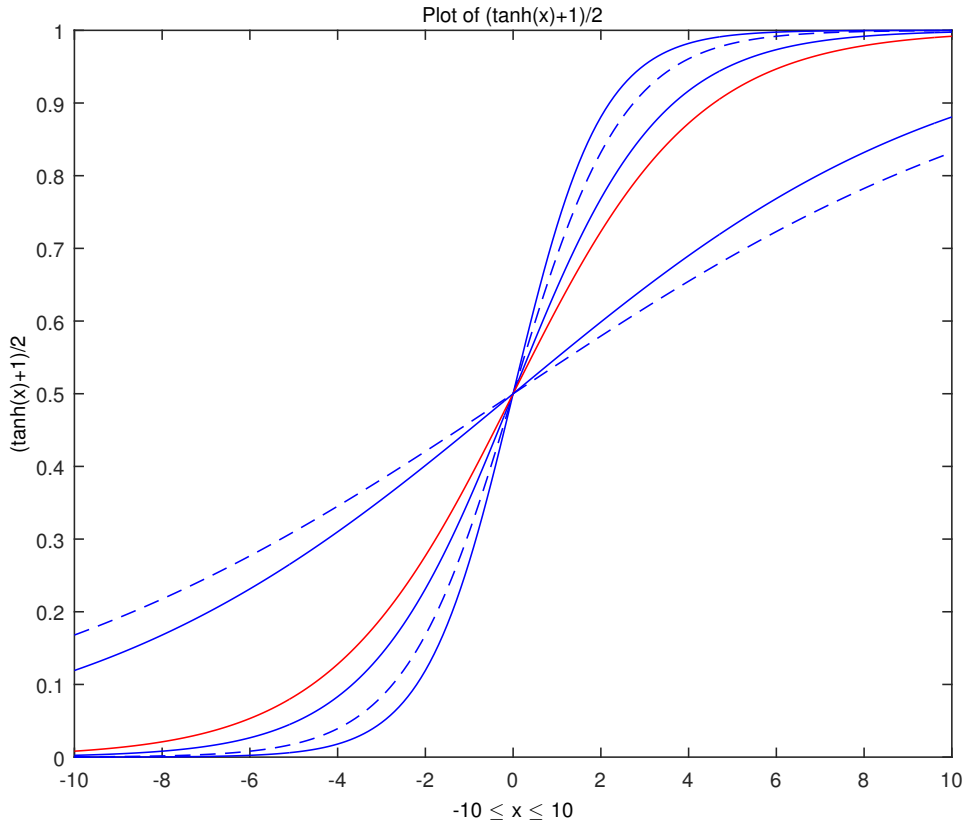


Figure 4.2: The slope activation function for slope values (0.5, 0.4, 0.3, 0.24, 0.1 and 0.08).

Let $\underline{U}$ and $\overline{U}$ be two instants defend by:

$$\underline{U} = min \left\{ \sum_{j=1}^{n} min(0, T_{ij}) + I_i \right\} \tag{4.4.11}$$

$$\overline{U} = max \left\{ \sum_{j=1}^{n} max(0, T_{ij}) + I_i \right\} \tag{4.4.12}$$

In this sense, we have

$$\forall i \in \{1, ..., n\} \ \underline{U} \leq U_i \leq \overline{U} \tag{4.4.13}$$

Since the function $f(x)$ is an increasing function, we have:

$$\forall i \in \{1, ..., n\} \ f(\underline{U}) \leq f(U_i) \leq f(\overline{U}) \tag{4.4.14}$$

As results, the parameters a and b are given by the formulas:

$$a = f(\underline{U}) \ and \ b = f(\overline{U}) \tag{4.4.15}$$

Basing on the parameters a and b, we picked the slope of the activation function, such that, this latter is almost linear and subjective from its start set to the interval $[a, b]$. By doing this, the activation function doesn't promote any asset.

4.4.3 Parameter setting

The parameter setting step aims to find the correct parameter to guarantee the network stability, as we aim to minimize the quantity:

$$\frac{1}{2}(1-\lambda)x^tQx - \lambda r^t x \tag{4.4.16}$$

The parameter α must be strictly positive $\alpha > 0$. If for x_i exist a neuron in $i \in I$ such as $x_i > 0$. Then, $E_i = \frac{\delta E(x)}{\delta x_i}$ must decresase: $E_i(x) < \epsilon$ ($\epsilon \in \mathbb{R}^{*+}$). Moreover, the sufficient condition is:

$$(1-\lambda)\sum_{j=1}^{n} q_{ij}\alpha\lambda r_i + \beta_1 + \varphi_1\delta_i\varphi_2\delta_i\,|I| \leq \epsilon \tag{4.4.17}$$

Such that

$$\varphi_2 > 0\, and\, \beta_1 < 0 \tag{4.4.18}$$

Or the sufficient free condition is:

$$\alpha(1-\lambda)\overline{\Sigma Q} - \alpha\lambda\underline{r} + \beta_1 + \varphi_1 + \varphi_2\,|I| \leq \epsilon \tag{4.4.19}$$

In such way that

$$\varphi_1 > 0\,;\overline{\Sigma Q} = min\sum_{j=1}^{n} q_{ij}\,;\underline{r} = minr_i \tag{4.4.20}$$

Eventually, any solution in the system (S) (Eq. 4.4.21) leads to a feasible portfolio.

$$(S)\begin{cases} \epsilon > 0 \\ \alpha > 0 \\ \beta_1 < 0 \\ \varphi_1 > 0 \\ \varphi_2 > 0 \\ \alpha(1-\lambda)\overline{\Sigma Q} - \alpha\lambda\underline{r} + \beta_1 + \varphi_1 + \varphi_2\,|I| \leq \epsilon \end{cases} \tag{4.4.21}$$

Considering the second model (PM'), the mean-semivariance rational model, we construct a similar continuous Hopfield network by replacing the variance formula by the Estrada semivariance one. By doing this, only the weight matrix is changed.

4.5 Hybrid Genetic Algorithm for Rational Mean-variance Model

The proposed aims to improve the CHN answers using the Genetic Algorithm as an accelerator. We are confident that such a combination well leads to great results. GA will help the CHN to overcome the energy function linearity problem since the balance parameter converges to zero at a point, the energy function, admitted as a quadratic function, accords with a linear function. Moreover, CHN will speed up the GA by getting a suitable solution to the initial population.

4.5.1 Algorithm

The adapted GA algorithm steps are presented as follows:
Step 1 : Initialize the population by CHN program solutions
Step 2: Evaluate the fitness of an individual in the population
Step 3: Select the best solution to reproduction
Step 4: Apply the two-points Crossover
Step 5: Apply the mutation, rotate between two points randomly chosen
Step 6: The children individuals go through a constraints verification process. If the current obtained solutions don't suffice the problem constraints, then the process requires a correction step. Else, go directly to step 7.
Step 7: Replace the current generation by the new generation.
Step 8: Verify the stopping criteria, ones the criteria is reached, stop the process, else go to Step 2.
The flowchart corresponding to the hybrid GA is presented by Fig. 4.3.

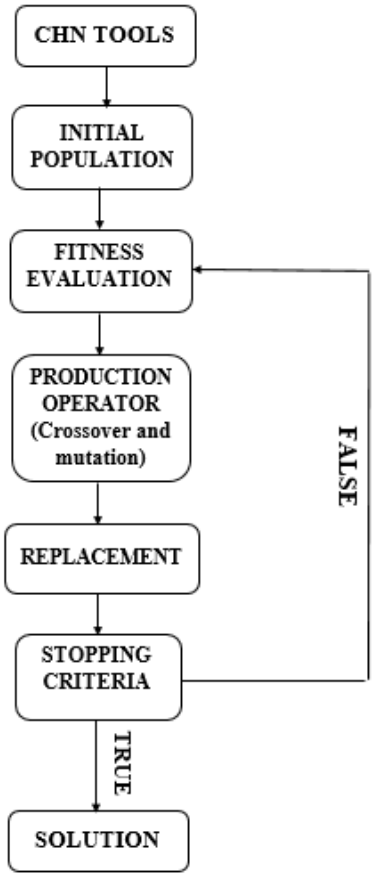


Figure 4.3: Hybrid GA flowchart

4.5.2 Genetic Algorithm Adaptation

As the Genetic Algorithm requires an adaptation to the portfolio problem, in this part we present the different modifications used to adapt the algorithm.

Individual coding: Each individual generated in the initial population is a potential solution to the portfolio problem coded corresponding to the problem variable: x_{ij} a continuous vector referring to the proportion of invested capital. Fig. 4.4 represents a possible individual formed by several genes, such as each gene is produced in $[0, 1]$ range. The GA population is often generated randomly. However, the generated solutions must meet the problem constraints.

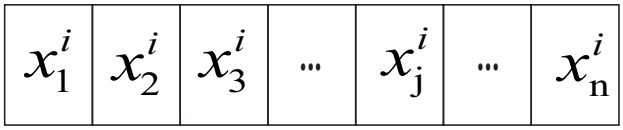


Figure 4.4: Individual Coding

Fitness assignment: The individual fitness defines the ability of an individual to compete among other individuals in the population. The probability that an individual will be selected for reproduction is based on its fitness score. Since each problem has a specific fitness function, we propose to use the proposed objective function of the portfolio problem as a fitness score, considering, this latter is has a continuous score that will not cause any implementation problem. Such as that the individual with a minimized fitness is more suitable to be the problem solution.

Parent Selection: A selection mechanism is used to select individuals to serve as parents during the next-generation production. In this context, Several methods were proposed [52]. However, we propose to use the tournament selection [40] to select the parents. This method is a variant of rank-based selection methods. It consists of randomly selecting a set of k-individuals, then, the k-choosing solutions are sorted according to their relative fitness and the most relevant individual is selected for reproduction. This process is repeated until the selection proportion is reached. Therefore, the select probability of each individual can be presented by:

$$p(i) = \begin{cases} \frac{C_{n-1}^{k-1}}{C_n^k} & if i \in [1, n-k-1] \\ 0 & if i \in [n-k, n] \end{cases} \tag{4.5.1}$$

Crossover: The role of the crossover operator is to provide mixing individuals that converge to a subspace. The operator recombines information from two good 'parent' into two new individuals called children. Such as the selected population is split into two-part, according to the crossover probability p_c, a part is elected to go on crossover operator. In literature, several possible representations [88] was proposed,

we choose to use the Crossover in Position Listing representation method. This latter, Randomly choose two crossover points and exchange all the genes between the crossover points. Fig. 4.5 represents the used crossover operator.

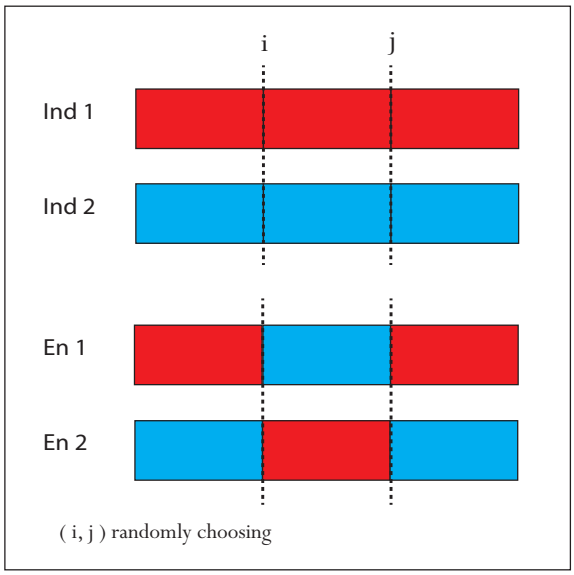


Figure 4.5: Crossover Operator

Mutation: the main purpose of using a mutation operator consists of maintaining diversity in the population as well as preventing premature convergence and local optima. The operator generates an offspring population by randomly changing one or several genes in a selected individual. Fig. 4.6 describes the used mutation mechanism.

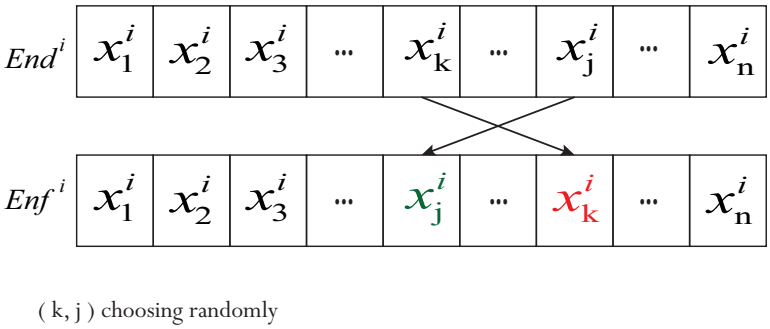


Figure 4.6: Mutation Operator

Correction: Since the production operators create the next-generation population, it's important to verify the produced solution quality. Each new individual goes through a constraints check, if the produced individual doesn't meet the problem constraint, a correction step is required. Therefore, we propose an adaptative heuristic for the GA correction step, answering to the portfolio problem constraints. The alg. 5 presents the proposed hybrid GA Correction heuristic.

Algorithm 5: Hybrid GA Correction heuristic

Require: Ind^i produced individual, nb : number of asset a_j^i such as $x_j^i \neq 0$
Ensure:

if $\sum_{j=1}^{n} x_j^i == 1$ **then**
  break
else
  while $\sum_{j=1}^{n} x_j^i \neq 1$ **do**
    if $\sum_{j=1}^{n} x_j^i < 1$ **then**
      $temp = 1 - \sum_{j=1}^{n} x_j^i$
      $temp = \frac{temp}{nb}$
      for $j = 1$ to n **do**
        if $x_j^i \neq 0$ **then**
          $x_j^i = x_j^i + temp$
        end if
      end for
    else
      $temp = \sum_{j=1}^{n} x_j^i - 1$
      $temp = \frac{temp}{nb}$
      for $j = 1$ to n **do**
        if $x_j^i \neq 0$ **then**
          if $x_j^i > temp$ **then**
            $x_j^i = x_j^i - temp$
          else
            $temp = temp - x_j^i$
            $x_j^i = 0$
            choose randomly k such as $k \notin I$
            $x_k^i += temp$
          end if
        end if
      end for
    end if
  end while
end if
if $\sum_{i \in I} x_i \neq 0$ **then**
  for $j = 1$ to n **do**
    if $j \in I$ and $x_j^i \neq 0$ **then**
      $temp = x_j^i$
      $x_j^i = 0$
      choose randomly k such as $k \notin I$
      $x_k^i += temp$
    end if
  end for
end if

4.6 Hybrid Multiobjective GA for Rational Mean-semivariance Model

Since we look for a problem solution that balance between risk level and profitability of the proposed model, the model aims to meet several objectives, we can consider a multiobjective modeling [25]. To solve such a problem, many methods were proposed and classified, such as Weighted Aggregation methods, non-aggregated non-Pareto methods and Pareto-based approach (see chapter for more details). The first section considers a weighted aggregation approach using the risk aversion parameter (λ). In this section, we propose to use a Pareto-based to enhance the obtained solutions and avoid the risk aversion parameter influence. Moreover, we apply a multiobjective GA developed to solve multiobjective optimization problems. Various MGA-variation was developed like VEGA [101] (non-aggregated non-Pareto methods), MOGA [30], NSGA [113], NPGA [46] (based on Pareto non-elitist); SPEA [131], PAES [62], PESA [23] NSGA II [26] (Pareto elitist). Since the problem constraints are linear, we can use the multiobjective genetic algorithm of Matlab solver, the option "gamultiobj" is used, this function attempts to create a set of Pareto optima for a multiobjective minimization. You may optionally set bounds and linear constraints on variables. "gamultiobj" uses the genetic algorithm for finding local Pareto optima. As in the "ga" function, you may specify an initial population, or have the solver generate one automatically. This function considers the following options:

- Does not support nonlinear constraints, so its syntax has fewer inputs,
- Takes an option "DistanceMeasureFcn", a function that assigns a distance measure to each individual for its neighbours,
- Takes an option "ParetoFraction", a number between 0 and 1 that specifies the fraction of the population on the best Pareto frontier to be kept during the optimization. If there is only one Pareto frontier, this option is ignored,
- Uses only the Tournament selection function,
- Uses elite individuals differently than "ga". It sorts no inferior individuals above inferior ones, so it uses elite individuals automatically,
- Has only one hybrid function, "fgoalattain",
- Does not support a stall time limit,
- Has different plot functions available,
- Does not support a choice of scaling function.

4.7 Experimental Results

To point out the efficiency of chosen risk measure, the portfolio selection problem is solved using different measures. firstly, the proposed rational mean-variance model; then, the semivariance is required as a risk measure of the rational mean-semivariance portfolio. Since this measure has asymmetric form, the symmetric of the Hopfield weight matrix is symmetric too; consequently, the stability of the proposed neural network is guaranteed [45]. The results of the Continuous Hopfield network are then ejected as the initial population of adapted genetic algorithm in sec. 4.5 for the rational mean-variance model; moreover, the mean-semivariance results are improved by multiobjective Genetic algorithm. The used Data are real financial data from yahoo finance database [126]. The data includes three portfolios, each one included 20 active over 20 periods; the study will cover the period from 29 September 2014 to 17 February 2015 in weekly.

4.7.1 Rational Mean-Variance Model Results

4.7.1.1 Continous hopfield network solver

In follow, the problem of Portfolio selection is resolve using variance risk measure. The continuous Hopfield network results are reported in table 4.1 In this sense, the CHN solver is run several times starting from different initial solutions and for different risk aversion parameter values $\lambda \in [0, 1]$; Then, the best solutions are kiped. It should be noted that the results are presented in terms of proportion. Eventually, a comparison between the CHN and genetic algorithm is presented to evaluate the performance of the proposed approach.

In table 4.1 The sign $P_i \quad (i = 1, ..., 6)$ refers to the obtained portfolio, moreover, the vector (ACA.PA,. . . , TEF.MC) is the assets abbreviated names as in yahoo finance website. From table 4.1 observation, the difference between each presented portfolio risk level isn't very significant, nevertheless, this difference is representative at a big-budget (billion).

The portfolio selection solutions that form the efficient frontier are mathematically beyond compare, including the solutions shown in tab 4.1, in such case, it is essential to know what investor face us. Indeed, it is important to remember that several investor types exist, well the most known are as follow:

- Only considering the level risk
- Only considering the return result
- Be satisfied with a compromise between the risk level and the return

As explained in the previous paragraph, we can graphically represent the efficient frontiers basing on the CHN obtained results. In this regard, we have generated various solutions using different risk-aversion parameter values $\lambda \in [0, 1]$; Fig. 4.7 represent the obtained efficient frontier. The red colour presents the composed shape of the rational mean-variance rational model efficient front, or the blue points are

Table 4.1: CHN Optimal Portfolios

Assets \ Obtained portfolios	P_1	P_2	P_3	P_4	P_5	P_6
ACA.PA	**8.09**	**6.41**	0	0	0	0
BN.PA	**23.55**	**20.5**	**16.7**	**5.85**	**1.95**	**7.93**
OR.PA	**31.23**	**34**	**28**	**28**	**30**	**9.96**
CA.PA	0	0	0	0	0	0
RNO.PA	0	0	0	0	0	0
VIE.PA	0	0	0	0	0	0
EN.PA	**0.14**	**5.09**	**14.65**	**28**	**30**	**36.11**
KER.PA	0	0	0	0	0	0
GLE.PA	0	0	0	0	0	0
DIE.BR	0	0	0	0	0	0
UCB	0	0	0	0	0	0
YOOX.MI	0	0	0	0	0	0
BPE.MI	0	0	0	0	0	0
ADS.DE	0	0	0	**10.15**	0	0
IFX.DE	0	0	0	0	**8.05**	0
MUV2.DE	0	0	**12.65**	0	0	0
KBC.BR	0	0	0	0	0	0
ENI.MI	0	0	0	0	0	0
AURS.PA	**36.99**	**34**	**28**	**28**	**30**	**46**
TEF.MC	0	0	0	0	0	0
Risk level	**0.0297**	**0.0288**	**0.0276**	**0.0285**	**0.0282**	**0.0312**
Mean return	**0.99**	**1.04**	**1.11**	**1.22**	**1.27**	**1.32**

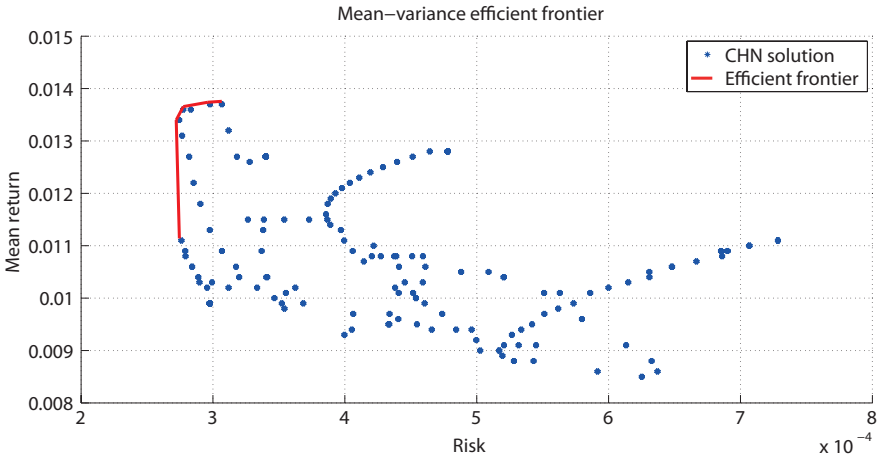


Figure 4.7: Rational Mean-variance Model Efficient Frontier

the obtained solutions for each time simulation.
From Fig. 4.1 observation, the low-risk levels, corresponding to high values of the risk aversion parameter λ, built an adequate efficient frontier representation. Whereas, for low values of λ, when the objective function converges to the mean return, regard-

less of the risk, the obtained solutions are not enough adequate to trace a complete efficient frontier. This phenomenon can be explained basing of the risk aversion parameter behaviour, Indeed, when the parameter converges to zero, the energy function formula becomes a leaner equation, in consequence, the proposed function for the portfolio problem via continuous Hopfield network don't satisfy the criteria of an energy function.This fact will be reinforced in the coming section comparing the genetic algorithm solutions and CHN results. Since the objective function of the Rational model present risk level and mean return management, we propose to compare the results by fixing each objective and check the corresponding results of each CHN and GA methods. Table 4.2 represents a comparison by fixing the mean return level. From table 4.2, we notice that the proposed continuous Hopfield net-

Table 4.2: Comparison mean return result between CHN and GA for a fixed mean return value

RISK		MEAN RETURN	
CHN	AG	CHN	AG
0.000297	0.000351	0.0099	
0.000288	0.000300	0.0104	
0.000276	0.000399	0.0111	
0.000285	0.000328	0.0122	
0.000282	0.000323	0.0127	
0.000312	0.000534	0.0132	

work gives the lowest risk values. Therefore it is recommended to use the proposed method for the problem under study considering $\lambda \geq 0.5$.
Otherwise, we propose to fix the risk value and compare the obtained results by CHN and GA, for the mean return. Table 4.3 represents a comparison of different solutions obtained for a low-risk aversion parameter λ. From table 4.3 we can see

Table 4.3: Comparison mean return result between CHN and GA for a fixed risk level

MEAN RETURN		RISK	
CHN	AG	CHN	AG
0.0101	**0.0162**	0.000586	
0.0105	**0.0138**	0.000508	
0.0107	**0.0145**	0.000663	
0.0085	**0.0134**	0.000625	
0.0110	**0.0152**	0.000707	
0.0093	**0.0140**	0.000527	

that for a fixed risk, meeting the lewder risk aversion parameter, the GA performs better that CHN, such as the GA obtain better mean return for all fixed risk level.

As conscience, the genetic algorithm and the continuous Hopfield network are, experimentally, uncomparable. However, to overcome the energy function linearity, we propose to use both, CHN and GA, for the proposed model, the idea is detailled in section 4.5. the next section presents the results of hybrid Genetic algorithm basing on continuous Hopfield network.

4.7.1.2 Hybrid Genetic Algorithm Results

As seen in section 4.5, the continuous Hopfield network (CHN) results are used as an initial population for the adapted genetic algorithm. In this context, the CHN program is launched several times. The parameters of used GA are presented in table 4.4. According to the parameters in table 4.4, the hybrid GA program

Table 4.4: Hybrid GA parameter setting

The parameter	**The setting**
Initial population	30 CHN solutions
Selection	Tournament
Tournament rate	80% of population
Crossover rate	60% of selected individual
Stopping criteria	$10000 * population_size$ iteration

is launched, several solutions are generated for each value of λ: the risk-aversion parameter, where λ take deferent value from 0 to 1 by a pitch of 0.05. Consequently, the efficient frontier is constricted from the generated solutions. Table 4.5 represent examples of obtained solutions for different λ values. We can notice, from table 4.5 that for a small value of λ the risk level is important. Alike, when λ take high value the mean returns seems to be smaller. For more interpretation, we present the different obtained results by a point cloud, presented in Fig. 4.8. From Fig. 4.8,

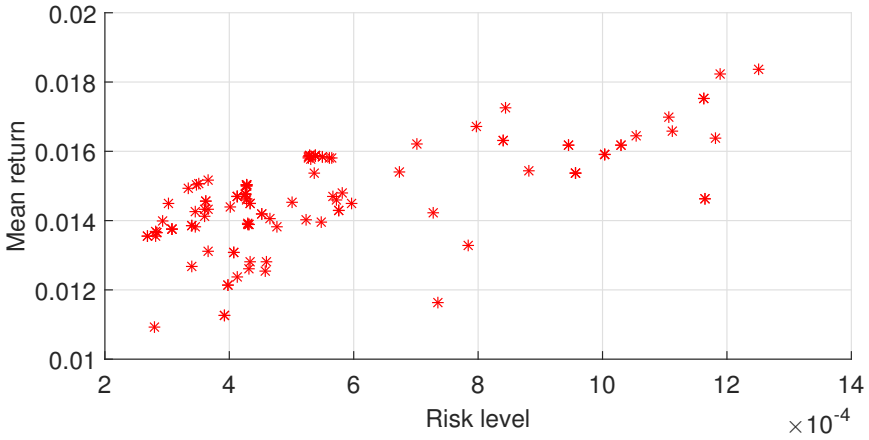


Figure 4.8: Hybrid GA Obtained Results

we can observe a possible efficient frontier. This latter allows acquiring optimal

Table 4.5: Exemple of Obtained Hybrid GA Results

Assets \ λ Values	0.3	0.5	0.7	0.9
ACA.PA	0	0	0	0
BN.PA	0	0	0	0
OR.PA	0	0	**0.1785**	**0.2512**
CA.PA	0	0	0	0
RNO.PA	**0.4822**	0	0	0
VIE.PA	0	0	0	0
EN.PA	**0.2999**	**0.4925**	**0.3815**	**0.3688**
KER.PA	0	0	0	0
GLE.PA	0	0	0	0
DIE.BR	0	0	0	0
UCB	0	0	0	0
YOOX.MI	0	0	0	0
BPE.MI	0	0	0	0
ADS.DE	**0.2179**	0	0	0
IFX.DE	0	0	0	0
MUV2.DE	0	0	0	0
KBC.BR	0	0	0	0
ENI.MI	0	0	0	0
ENI.MI	0	0	0	0
AURS.PA	0	**0.5075**	**0.4400**	**0.3800**
TEF.MC	0	0	0	0
Risk level	$9.5742e-04$	$3.6468e-04$	$3.0674e-04$	$2.8320e-04$
Mean return	0.0154	0.0143	0.0137	0.0136

portfolios, which refer to the highest possible return for a given level of risk or the lowest possible risk for a given level of return. Fig. 4.9 illustrated the obtained efficient frontier. To evaluate the proposed approach we propose to use a graphic comparing the obtained hybrid GA efficient frontier to the CHN efficient frontier. Such that, the two efficient frontiers are performed in the same figure evaluating the quality of the obtained solutions. Fig. 4.10 is a representation of both efficient frontiers. In Fig. 4.10, the CHN results and efficient frontier are represented by blue colour. However, the red colour presents the proposed approach solutions and the obtained efficient frontier. We can notice, from Fig. 4.10, that The CHN efficient frontier is positioned lewder than the proposed approach and, as suspected, the obtained results are better than CHN solutions. Moreover, we may conclude that the CHN solutions were improved by the Genetic Algorithm.

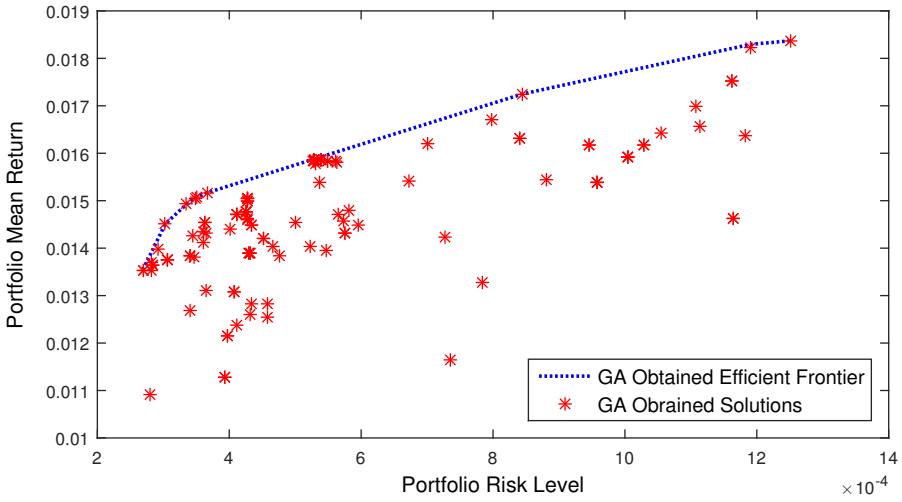


Figure 4.9: hybrid GA Efficient Frontier

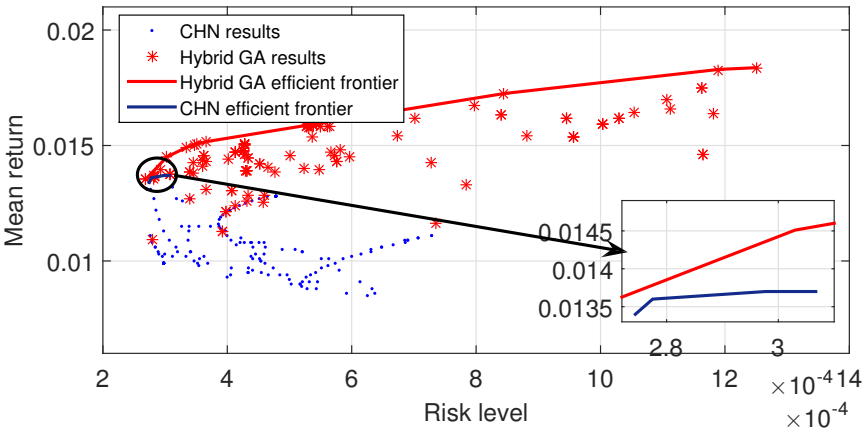


Figure 4.10: CHN and hybrid GA performance comparison

4.7.2 Rational Mean-Semivariance Model Results

4.7.2.1 Continous hopfield network solver

In this section, we present the portfolio selection problem solutions, using the Estrada semi-variance [28], [29] as a risk measure through the proposed approach, using, each time, a deferent value of the risk-aversion parameter λ. In this regard, the systems (mean-variance, CHN) and (Estrada mean-semivariance, CHN) are compared. The obtained solutions are presented in terms of efficient frontier; see Fig. 4.11. The red colour in Fig. 4.11. Is the constructed shape of the mean-semivariance rational model efficient front, well the blue colour is the CHN obtained solutions for each time simulation. We can say that the obtained shape Simulates the expected efficient frontier, as known in the literature. Moreover, table 4.6 gives an example of an efficient portfolio produced by the proposed continuous Hopfield network.

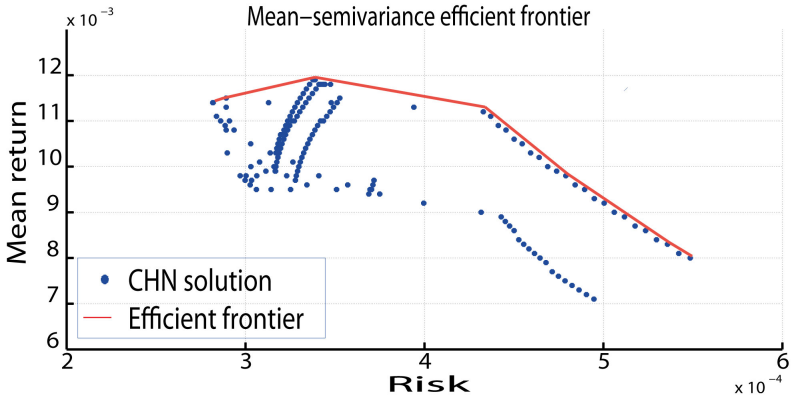


Figure 4.11: (Estrada Mean-Semivariance, CHN) Efficient frontier

Table 4.6: Example of Obtained Efficient Portfolio

Assets	$x_i(\%)$
ACA.PA	0
BN.PA	32
OR.PA	32
CA.PA	0
RNO.PA	0
VIE.PA	0
EN.PA	2.30
KER.PA	0
GLE.PA	0
DIE.BR	0
UCB.BR	0
YOOX.MI	0
BPE.MI	0
ADS.DE	0
IFX.DE	0
MUV2.DE	1.70
KBC.BR	0
ENI.MI	0
AURS.PA	32
TEF.MC	0

In our knowledge, it is the first time that the system (Estrada semivariance, continuous Hopfield network) is used, for the portfolio selection problem. To point out to effectiveness of this strategy, the results are compared to mean-variance and CHN one. To this end, we fix the mean-return of both systems then the risk levels are compared. Fig. 4.12 illustrated the said comparison. From Fig. 4.12 observation, the system (Estrada mean-semivariance, CHN) produces decisions with the smallest

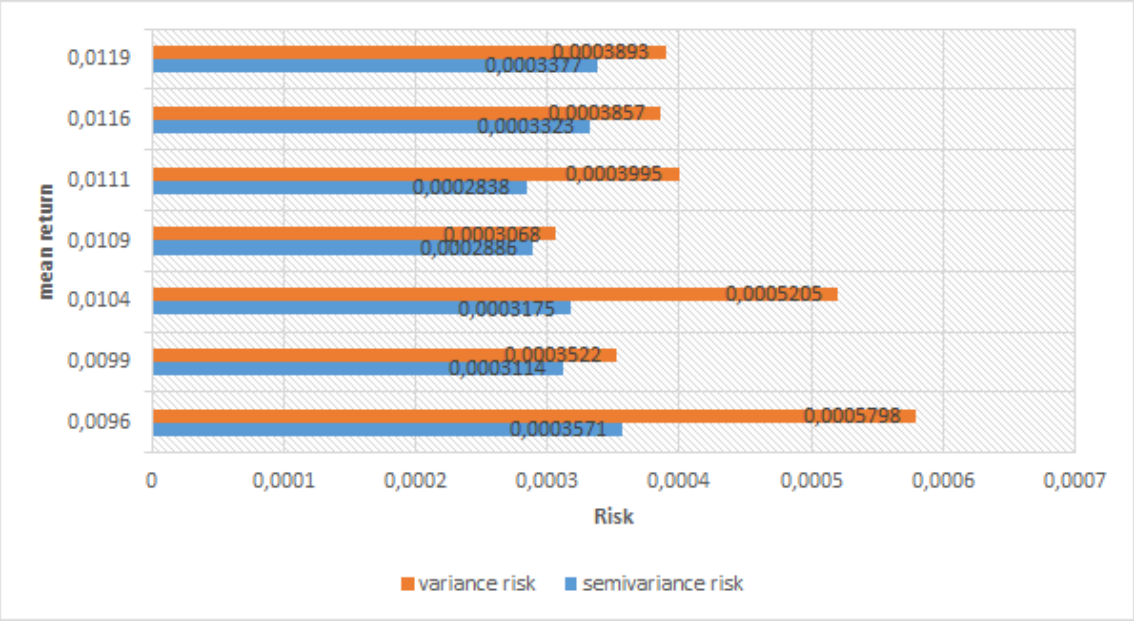


Figure 4.12: (Estrada mean-Semivariance CHN) and (mean-variance, CHN) Comparing

risk level comparing with one produced by the system (mean-variance, CHN). From Fig. 4.11 and 4.12, we can confirm that the Mean-semivariance rational model has proved his efficiencies. However, the linearity problem of the CHN energy function seems to be persistent. By observing the efficient frontier in Fig. 4.11, a quality decreasing of solutions concerning the $\lambda \in [0, 0.5]$ is noticed, it seems that the risk-aversion parameter is affecting the (Estrada mean-semivariance, CHN) decision. To avoid the risk-aversion parameter influence, we proposed to use a Pareto-based approach using continuous Hopfield decisions. The following section presents the obtained results.

4.7.2.2 Hybrid Multiobjective Genetic Algorithm Results

Basing on the obtained results in section, we have generated several solutions by the CHN, by altering the values of λ the risk-aversion parameter, to prepare a diverse initial population for multi-objective genetic algorithm. Then the algorithm parameters are set as in table 4.7 Any other parameters are used by default.

According to the parameters in table 4.7 and after having launched the Matlab MGA solver, we obtain the resulted Pareto front, presented by Fig. 4.13. table 4.8 presents a solution among the obtained Efficient frontier in Fig. 4.13.

The table 4.8 left side presents the names of the assets as given in the Yahoo database. Well, the right side represents the Investissement rate of the corresponding asset. Regarding the proposed approach, it is difficult, even impossible to decide among the obtained efficient frontier (in general case: Pareto front). The approach generates a set of optimal solutions, it is the role of the decision-maker to choose from

Table 4.7: Multiobjective Genetic Algorithm Parameters adjustment

Genetic algorithm setting	
Number of variables	20
Population type	Double vector
Population size	186
Initial population	CHN solution
Selection	Tournament
Tournament size	37 (20% of the population)
Crossover	80% of population
Mutation function	Constraint dependent
Crossover	Two-point
Multiobjective problem setting	
Distance measure function	Crowding distance
Pareto front population fraction	Default : 0.35
Stopping criteria	
Generations	100*number of variables

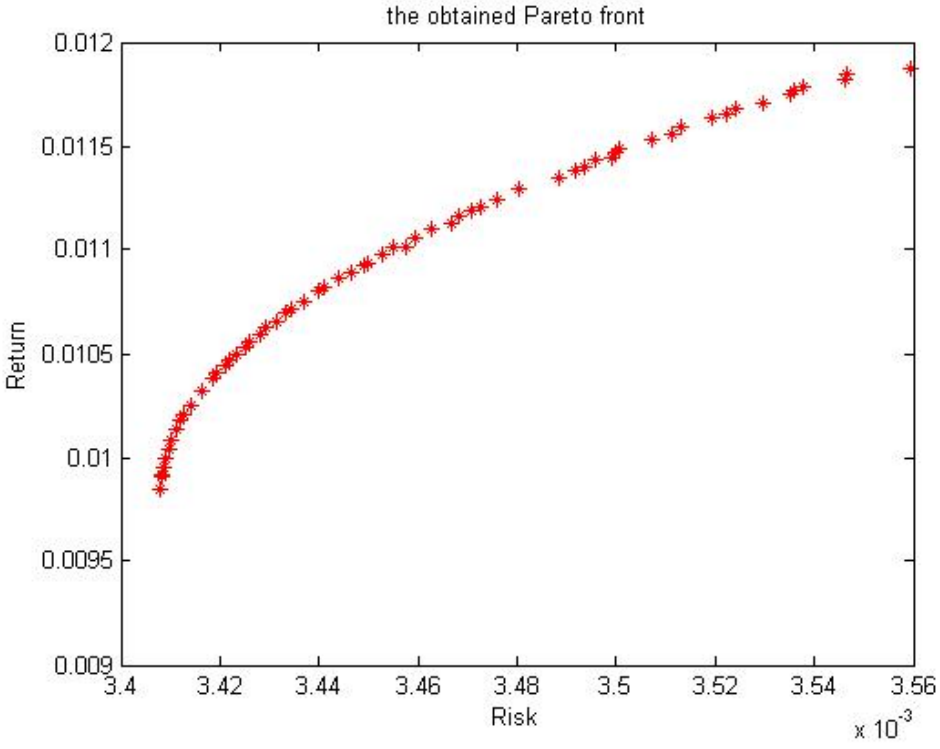


Figure 4.13: Obtained Efficient Front of Hybrid Multiobjctive Genetic Algorithm

the Pareto front. However, to evaluate the effectiveness of the proposed approach,

Table 4.8: Example of an Optimized Solution

The assets of the Portfolio	x_i
ACA.PA	0
BN.PA	0,141783048249336
OR.PA	0,552911488975139
CA.PA	0
RNO.PA	0
VIE.PA	0
EN.PA	0
KER.PA	0
GLE.PA	0
DIE.BR	0
UCB.BR	0
YOOX.MI	0
BPE.MI	3,81469726562500e-06
ADS.DE	0
IFX.DE	1,52587890625000e-05
MUV2.DE	0,00615351210627622
KBC.BR	4,19801642196498e-07
ENI.MI	0
AURS.PA	0,298540150299675
TEF.MC	0

we propose to compare the obtained results to multiobjective genetic algorithm results and CHN results. Firstly, the comparison of the proposed approach and the multiobjective genetic algorithm results is performed. Fig. 4.14 present a graphic comparison by illustrating the obtained efficient frontier (CHN + MGA) and the multiobjective genetic algorithm solver. The red colour denotes the proposed approach front and the blue one represent the multiobjective genetic algorithm front. From Fig. 4.14 observation, the obtained results can be considerate better, since, with the proposed approach, the same level of multiobjective genetic algorithm return is obtained with a lower risk level. Given the comparing results between the proposed approach and multiobjective genetic algorithm, we propose to compare the obtained results with initial continuous Hopfield network results. Conceding that the continuous Hopfield network is a mono-objective method, therefore we evaluate the proposed approach by fixing every criterion and compare the results. At first, the portfolio mean-return criteria is fixed and the risk level is compared,besides, the number of assigned assets is compared to. Accordingly, table 4.9 presents the described comparison.

From table 4.9 We may notice that the risk level generated by the proposed approach and CHN are substantially equal, we may notice a difference of 0.0001 to 0.0005. This difference is due to the number of affected assets since the number

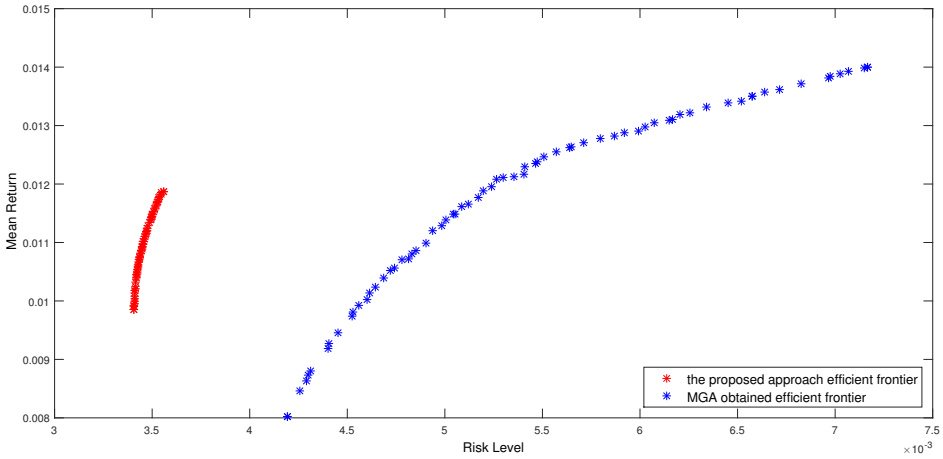


Figure 4.14: Graphic comparing of the proposed approach and multiobjctive genetic algorithm

of assets affected by the proposed approach exceeds that of CHN. By reviewing table 4.9, we can notice that the maximum and a minimum number of assignments for the proposed approach are respectively 7 and 4, well, the maximum and the minimum number of assignments for continuous Hopfield network are respectively only 4 and 2. As conclusion, the obtained results for the proposed approach can be considered as outperforming.

The second step will be to fix the level of risk and compare the obtained return. Table 4.10 Represents the comparison in term of risk-level. The fixed risk is the result obtained by the two methods (our approach and CHN). From table 4.10, the first observation is that the results mark the superiority of the CHN, an exception to the first table line (marked in bold).

Considering table 4.9 and table 4.10 outcomes, we can estimate that the results are unreliable. Since, for each new generation of solutions, we are faced with new conclusions. Moreover, isn't possible to decide which one of the two methods (the proposed approach and CHN) provide the best results.

4.8 Conclusion

In this chapter, we proposed an additional constraint to complete the Markowitz models for portfolio selection problem. These latter promoted the selection of assets with a good mean-return. Moreover, Due to the effectiveness of the neural network tools, we proposed to use the continuous Hopfield network to solve the obtained model through an original energy function with an adequate parameter that ensures the equilibrium point feasibility. The proposed system produces a portfolio with no negative return. Furthermore, the representation of the return respect to risk, for several values of λ, leads to an acceptable efficient frontier, especially when the mean-semivariance is used as a risk measure. However, the proposed energy

Table 4.9: Table Comparing the proposed Approach and CHN in term of Mean-return criteria

Mean Return	Risk Level		Affectation	
	(CHN+MGA)	CHN	(CHN+MGA)	CHN
0.0098	0.003407	0.003284	4	3
0.0099	0.003408	0.003291	5	3
0.0100	0.003409	0.003298	5	3
0.0101	0.003410	0.003306	4	3
0.0102	0.003412	0.003315	4	3
0.0103	0.003414	0.003325	5	3
0.0104	0.003418	0.003235	4	3
0.0105	0.003421	0.003348	5	3
0.0106	0.003425	0.003360	5	3
0.0107	0.003431	0.003374	5	3
0.0108	0.003439	0.003388	6	3
0.0109	0.003443	0.003403	6	3
0.0110	0.003452	0.003419	7	4
0.0111	0.003459	0.003454	5	3
0.0112	0.003468	0.003473	5	2
0.0113	0.003480	0.003492	5	4
0.0114	0.003491	0.003479	6	4
0.0115	0.003499	0.003306	6	3
0.0116	0.003511	0.003374	5	3
0.0117	0.003522	0.003394	6	3
0.0118	0.003535	0.003475	5	2
0.0119	0.003546	0.003394	6	3

Table 4.10: Table Comparing the proposed Approach and CHN in term of risk-level

Portfolio Risk	Portfolio Return	
	(CHN+MGA)	CHN
0.003410	**0.0100**	**0.0098**
0.003411	0.0101	0.0118
0.003512	0.0105	0.0114
0.003414	0.0103	0.0118
0.003419	0.0104	0.0110
0.003429	0.0106	0.0118
0.003436	0.0107	0.0110
0.003454	0.0110	0.0111
0.003475	0.0112	0.0118

function seems to be linear at a point due to the behavior of the risk aversion parameter. What limits the tools fully exploration. for that consideration, we propose a new approach to solve the model, basing on continuous Hopfield network (CHN) response and genetic algorithm process (GA). The proposed hybrid approach will speed up the GA and avoid the CHN energy function linearity problem. The approach has been applied for the rational mean-variance model, this latter has proved his efficiency according to the obtained shape of the efficient frontier. Well, for the rational mean-Semivariance model, using the Estrada semivariance, we have proposed a multiobjective approach based on neural network tools to avoid the explained energy function default. Such as, the multiobjective genetic algorithm initial population is provided by the continuous Hopfield network. Since the model constraints are linear, We have to use the MGA generate by Matlab. The proposed hybridization provides enhanced solutions according to the evaluation section.

Chapter 5

Multiobjective learning approaches for multilayer perceptron

5.1 Introduction

The multilayer perceptron neural network (MLPNN) supervised learning problem has been widely discussed in recent years. Respectively, several approaches were proposed such as classical learning [99] or optimization methods [93] [2] and their efficiency were widely discussed. So why converging to a Pareto-based multiobjective approach? Once the different objectives are recused to be optimized, these algorithms turn out to be difficult, since the only objective can be minimized directly and can produce an overfitted MLPNN. The classical approach seems to be weak handling competitive objectives in learning [79]. This latter does not necessarily yield to good generalization models [72], his method has several disadvantages, as slow convergence, local minima problem, and model selection uncertainty. To deal with the aforementioned, the classical methods added, to balance the two objectives, a penalty parameter (λ) to the model objective function. This technic is known as network regularization [16]. Unfortunately, the mentioned parameter turn to be problematic ones it needs to be fixed. Instead, we propose to use a multi-objective approach based on the Pareto concept [26]. One of the important approach advantages can be presented on the number of Pareto-optimal solutions, this previous can achieve a large number of solutions called Pareto front, from which the user is capable to derive the problem information and make an improved decision one the single final solution is recused. From there, the tradeoff (training-complexity), using the Pareto-based methodology, has been examined. In this respect, earlier research approaching supervised learning by Pareto-based concept can generally be partitioned into three categories based on their motivations, such as generating interpretable models or model selection for generalization and ensemble generation. In this chapter, we propose a new approaches to select the right model for a better MLPNN generalization. The proposed aims to regularize the network complexity during the MLPNN training; finding the tradeoff between accuracy and complexity of the network. Moreover, we propose a multi-objective methodology including

two objectives: to minimize the general error of the perceptron and chosen regularisation terms supported by a decision variable, under constraints representing the communication between the network layers. Furthermore, we solve the emerging models using NSGA II (Non-dominated Sorting Genetic Algorithm II) [128] one of the most efficient multi-objective genetic algorithms. The Pareto-based methodology was considered, in addition to feed-forward neural networks, for generation of radial-basis neural networks [36], support vector machines [10][50][83], decision trees [60], and classifier systems [73]. The Pareto-based multiobjective concept was also have been reported for unsupervised learning, mainly multiobjective data clustering [61].

5.2 Multiobjective Training Model for MLPNN

Using a multiobjective model in machine learning is, in fact, an essential idea, since we are looking for a tradeoff between accuracy and complexity. As reasons that an unsuitable topology expands the training time, as well as can cause non-convergence. Mainly, the mentioned reasons are usually responsible for MLPNN generalization performance deficiency [110]. In general, once the complexity is reduced, the model is more controlled and the overfitting is prevented. Therefore, we propose to optimize, addition to MLPNN learning error, two different regularization term: a modified LASSO and the modified $L_{1/2}$ norm to regularise the network complexity. The aforementioned regularization will yield to better control on weight values, furthermore, this matter yield to suitable MLPNN topology. Since he is an important criterion influencing MLPNN efficiency. The models support the introduction of a new decision variable to the objectives formula, who plays the role of a pruning technique, aiming to reduce unfunctional neuron in the MLPNN architecture. This technique will yield to a large weights elimination, which ends up with great results. Moreover, the constraints ashore large communication between layers.

Basing on O. Ludwig and U. Nunes [71], it was found that one hidden layer usually performs well enough for most problems. For this specific reason, the network studied, in this chapter, is composed of an individual hidden layer. The proposed models will be explained in the following sections.

5.2.1 Notation

N : The hidden layer size.
n_0 : The input layer size.
n : The output layer size.
X : Input data of the neural network.
Y : The MLPNN calculated output.
d : Desired output.
f : Activation function.
h : The hidden layer output.

w : network weights.

C_{ij}^{k} : Binary variable as:

$$C_{ij}^{k} = \begin{cases} 1 \text{ if } \textit{the connection between the neuron } i \textit{ and } j \textit{ is kaped} \\ 0 \text{ if not} \end{cases} \tag{5.2.1}$$

k : is the connection index between two successive layers.
U_i : Binary variable as:

$$U_i = \begin{cases} 1 \text{ if } \textit{the neuron } i \textit{ is kaped in the hidden layer} \\ 0 \text{ if not} \end{cases} \tag{5.2.2}$$

5.2.2 Training Objective

Obviously, the first objective function is the most common using error function for neural network training, the mean squared error (MSE), such as J is the example set size:

$$E\left(w, C, U\right) = \frac{1}{2J} \sum_{j=1}^{J} ||F_j\left(U, C, X, w\right) - d_j||^2 \tag{5.2.3}$$

Concerning the second objective: network complexity, we propose two different functions that will be detailed in the next section.

5.2.3 Modified LASSO regularization

The proposed model aims to increase the training ability of the network with a reduced network complexity. However, several works proposed different complexity realisation, in this part, we propose to present this latter by a modified LASSO regularization. Moreover, the model is developed.

5.2.3.1 Output of the hidden layer

The MLPNN hidden layer is directly connected to the input layer, so the suppression of the hidden layer neurons and the weights connecting the input layer to the hidden layer is represented in the output of each neuron by:

$$h = \begin{pmatrix} h_1 \\ \vdots \\ h_i \\ \vdots \\ h_N \end{pmatrix} = \begin{pmatrix} U_1 \left(1 - \prod_{j=1}^{n_0} \left(1 - C_{j1}^{0}\right)\right) f\left(\sum_{j=1}^{n_0} C_{j1}^{0} w_{j1}^{0} x_j\right) \\ \vdots \\ U_i \left(1 - \prod_{j=1}^{n_0} \left(1 - C_{ji}^{0}\right)\right) f\left(\sum_{j=1}^{n_0} C_{ji}^{0} w_{ji}^{0} x_j\right) \\ \vdots \\ U_N \left(1 - \prod_{j=1}^{n_0} \left(1 - C_{jN}^{0}\right)\right) f\left(\sum_{j=1}^{n_0} C_{jN}^{0} w_{jN}^{0} x_j\right) \end{pmatrix} \tag{5.2.4}$$

5.2.3.2 Output of the Neural Network

The network output is calculated using the weights connecting the hidden layer to the output layer since, at this point, only the connections are pruned, so the network output is calculated by:

$$F(U,C,X,w) = \begin{pmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_n \end{pmatrix} = \begin{pmatrix} \left(1 - \prod_{j=1}^{N}\left(1 - C_{j1}^{1}\right)\right) f\left(\sum_{j=1}^{N} C_{j1}^{1} w_{j1}^{1} h_j\right) \\ \vdots \\ \left(1 - \prod_{j=1}^{N}\left(1 - C_{ji}^{1}\right)\right) f\left(\sum_{j=1}^{N} C_{ji}^{1} w_{ji}^{1} h_j\right) \\ \vdots \\ \left(1 - \prod_{j=1}^{N}\left(1 - C_{jn}^{1}\right)\right) f\left(\sum_{j=1}^{N} C_{jn}^{1} w_{jn}^{1} h_j\right) \end{pmatrix} \tag{5.2.5}$$

5.2.3.3 Network Complexity Objectif Function

The network complexity is presented by a modified LASSO regularization [33] such as we introduce a binary variable to the known formula. This presentation aims to reduce the network weights number in order to control the variations of the weight during the training for a better generalization. The resulting formula is presented as :

$$\Phi_1(w,C,U) = \sum_{i=0}^{n_0}\sum_{j=0}^{N}\left|C_{ij}^{0} w_{ij}^{0}\right| + \sum_{i=0}^{N}\sum_{j=0}^{n}\left|C_{ij}^{1} w_{ij}^{1}\right| \tag{5.2.6}$$

5.2.3.4 Constraints

The first constraint guarantees the existence of hidden layer by the existence of; at least, one neuron in it, the constraint is expressed by

$$\sum_{i=1}^{N} U_i \geq 1 \tag{5.2.7}$$

Since we remove connections and neurons of the network, the communication between them must be guaranteed. A neuron, which has no connections, should be removed from the network architecture. The same, as we delete a neuron, we must remove its connections too. The constraint, then, is expressed by:

- The communication between the input layer and the hidden layer :

$$U_j \prod_{i=1}^{n_0}\left(1 - C_{ij}^{0}\right) = 0 \qquad \forall j = 1, ..., N \tag{5.2.8}$$

- The communication between the hidden layer and the output layer

$$U_i \prod_{j=1}^{n}\left(1 - C_{ij}^{1}\right) = 0 \qquad \forall j = 1, ..., N \tag{5.2.9}$$

5.2.3.5 Obtained Model

The obtained multi-layer perceptron optimization model [105] can be formulated as:

$$\begin{cases} \min\ E(w,C,U) \\ \min\ \Phi_1(w,C,U) \\ Subject\ to \\ \sum_{i=1}^{N} U_i \geq 1 \\ U_j . \prod_{i=1}^{n_0} \left(1 - C_{ij}^0\right) = 0 \qquad \forall j = 1 \ldots N \\ U_i . \prod_{j=1}^{n} \left(1 - C_{ij}^1\right) = 0 \qquad \forall i = 1 \ldots N \\ w = \left(w_{ij}^k\right)_{\substack{k=\{1,2\} \\ 1 \leq i \leq N \\ 1 \leq j \leq n}} \quad where \quad w_{ij}^k \in \mathbb{R} \\ C = \left(C_{ij}^k\right)_{\substack{k=\{1,2\} \\ 1 \leq i \leq N \\ 1 \leq j \leq n}} \quad where \quad C_{ij}^k \in \{0,1\} \\ U = (U_i)_{1 \leq i \leq N} \quad where \quad U_i \in \{0,1\} \end{cases} \tag{5.2.10}$$

5.2.4 Non-Smooth $L_{1/2}$ regularization

In this section, we will present a new multiobjective model basing on Non-Smooth $L_{1/2}$ regularization such as this later present the complexity of the network. Furthermore, new constraints are introduced. more details are given in the section.

5.2.4.1 Hidden Layer Output

Considering the network structure, the network input is directly connected to the only perceptron-hidden layer. Regarding the proposed suppression (weights and neurons) process, the output eq. 5.2.5 is kept.

5.2.4.2 Neural Network Output

At this stage, only the elimination of weights is taken into consideration since we must avoid removing the network outputs. For this reason, the network output is calculated by:

$$F(U,C,X,w) = \begin{pmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_n \end{pmatrix} = \begin{pmatrix} f\left(\sum_{j=1}^{N} C_{j1}^1 w_{j1}^1 h_j\right) \\ \vdots \\ f\left(\sum_{j=1}^{N} C_{ji}^1 w_{ji}^1 h_j\right) \\ \vdots \\ f\left(\sum_{j=1}^{N} C_{jn}^1 w_{jn}^1 h_j\right) \end{pmatrix} \tag{5.2.11}$$

5.2.4.3 Network Complexity presentation

In this part, the network complexity is presented by modified $L_{1/2}$ regularization [124], [125], which is a nonconvex penalty and have many promising properties. Basing on experiments [121], $L_{1/2}$ achieve better pruning than weight Decay [67] and weight Elimination [122]. The coresponding formula is:

$$\Phi_2(w,C,U) = \sum_{j=1}^{N} U_j \left(\sum_{i=1}^{n_0} C_{ij}^0 \left|w_{ij}^0\right|^{\frac{1}{2}} + \sum_{k=1}^{n} C_{jk}^1 \left|w_{jk}^1\right|^{\frac{1}{2}}\right) \tag{5.2.12}$$

5.2.4.4 Constraints

As always we look to avoid the hidden layer total elimination, it is necessary to keep at least one neuron in the hidden layer. The constraint 5.2.7 is kept in the model. However, we propose new constraints to keep communication between layers. Since the network goes through a random pruning process. Besides, neurons without any relation with the layers must be removed from the architecture. The constraints, then, is expressed by:

- The following family constraint present the complete removal of any connection between the pruned neuron and input-output layers :

$$(1-U_j)\left(\sum_{i=1}^{n_0} C_{ij}^0 + \sum_{k=1}^{n} C_{jk}^1\right) = 0 \qquad \forall j = 1 \ldots N \tag{5.2.13}$$

- Well the communication between the delated connection and attached neuron is ensured by :

$$U_j . \left(\prod_{i=1}^{n_0}\left(1-C_{ij}^0\right) . \prod_{k=1}^{n}\left(1-C_{jk}^1\right)\right) = 0 \qquad \forall j = 1 \ldots N \tag{5.2.14}$$

5.2.4.5 Obtained Model

The multiobjective multilayer perceptron (MLPNN) training model [106] can be expressed by:

$$\begin{cases} \min E(w, C, U) \\ \min \Phi_2(w, C, U) \\ Subject\ to \\ \sum_{i=1}^{N} U_i \geq 1 \\ (1 - U_j)\left(\sum_{i=1}^{n_0} C_{ij}^0 + \sum_{k=1}^{n} C_{jk}^1\right) = 0 \qquad \forall j = 1 \ldots N \\ U_j . \left(\prod_{i=1}^{n_0}\left(1 - C_{ij}^0\right) . \prod_{k=1}^{n}\left(1 - C_{jk}^1\right)\right) = 0 \qquad \forall j = 1 \ldots N \\ w = \left(w_{ij}^k\right)_{\substack{k=\{0,1\} \\ 1 \leq i \leq \{n_0, N\} \\ 1 \leq j \leq \{N, n\}}} \quad where \quad w_{ij}^k \in \mathbb{R} \\ C = \left(C_{ij}^k\right)_{\substack{k=\{0,1\} \\ 1 \leq i \leq \{n_0, N\} \\ 1 \leq j \leq \{N, n\}}} \quad where \quad C_{ij}^k \in \{0, 1\} \\ U = (U_i)_{1 \leq i \leq N} \quad where \quad U_i \in \{0, 1\} \end{cases} \tag{5.2.15}$$

To solve the proposed multiobjective MLPNN Training models several methods and approaches are available. For our case, we propose to use dominated sorting genetic algorithm II (NSGA II) [64] for the described multiobjective models.

5.3 Non-dominated Sorting Genetic Algorithm for Multiobjective Training Models

NSGA II is a multiobjective evolutionary algorithm proposed by Deb et al [26], it was classified as one of the most efficient multiobjective evolutionary algorithms, besides, NSGA II has the features required for a good multi-objective GA for his selection method. The algorithm uses a heuristic to evaluate invidious of the same front, called crowding distance [64], such as the one situated in a less crowded area is picked. The crowded tournament selection guarantees the selection of the best rank possible.

Since we perform two different multiobjective models using NSGA II as a solver, the section is distributed as three parts, the first one present the similar part of NSGA II to models and the remind parts will discuss the specific details of each adaptation.

5.3.1 NSGA II General Used Algorithm

Step 1: Divide the data example set into training D_{train} and testing D_{test} subsets.
Step 2: Set n=0 (number of iteration). Create the initial population P_n ,randomly, of size A. Such as, each individual represent a different multi-layer perceptron.
Step 3: for each individual in the population, compute the output of the MLPNN for each x input vector in the training set D_train.
Step 4: Compute the two objectives: Training $E(w, C, U)$ and complexity ($\Phi(w, C, U)$, $\Phi_1(w, C, U)$ for each individual.
Step 5: Identify the non-dominated fronts F_1,F_2,...,F_k of population P_n, using the non-dominated rank concept.
Step 6: Compute the crowding distance of each individual in the different population front.
Step 7: Apply to the population P_n the crowded tournament selection operator to generate an intermediate population S_n of size M.
Step 8: Apply the crossover and mutation operators to S_n, to create the offspring population Q_n of size A.
Step 9: Combine P_n and Q_n to obtain a union population $R_n = P_n \cup Q_n$.
Step 10: Apply steps 3-6 on to R_n to obtain a sorted union population.
Step 11: Select the A best solutions from the sorted union to create the next population P_{n+1}.
Step 12:Verify if the stopping criteria are reached. Otherwise, set n=n+1, back to step 7 and reply the procedure.

Finally, since NSGA II is an elitist algorithm; the best front in regards to non-dominance and diversity is kept. Moreover, the testing set is performed using the obtain solutions of the Pareto front (the MLPNNs trained with the optimal weight values). Then the MLPNN with the best results is the best solution for the MLPNN generalization problem.

5.3.2 General NSGA II Adaptation

Since the used method is based on the genetic algorithm process, this part considers the common NSGA II adaptation of the proposed models. Moreover, considering that the models treat the same problem, the adaptation takes several common parts. the following operators are used for the considered models.

Individual Coding. The coding step is very important; each in the population simulates an possible MLPNN structure to the problem, Fig. 5.1 present an example of an individual and the corresponding MLPNNs structure. The coding is proposed according to the problem variables wich can be an integer, continuous, binary or mixed chromosome. Considering the proposed models, a single individual contains three chromosomes presenting the three problem variables: W_{ij}^k weights matrix containing continuous variables, C_{ij}^k connections matrix containing binary variables and U_i neurons decision vector; so the NSGA II individual is structured as

Fig. 5.2. Moreover, each individual simulates an MLPNN structure, Fig. 5.3 present an example of an individual and the corresponding MLPNNs structure.

Initial population. The GA initial population is the pool of the initial solution, it's often achieved randomly, we adopt the same procedure. However, the generated individual must satisfy the problem constraints. The vector U_i and the matrix C_{ij}^k are binary; zero or one presents them. Moreover, W_{ij}^k the weights matrix contain continuous variables, they are generated in the interval [-0.5,0.5].

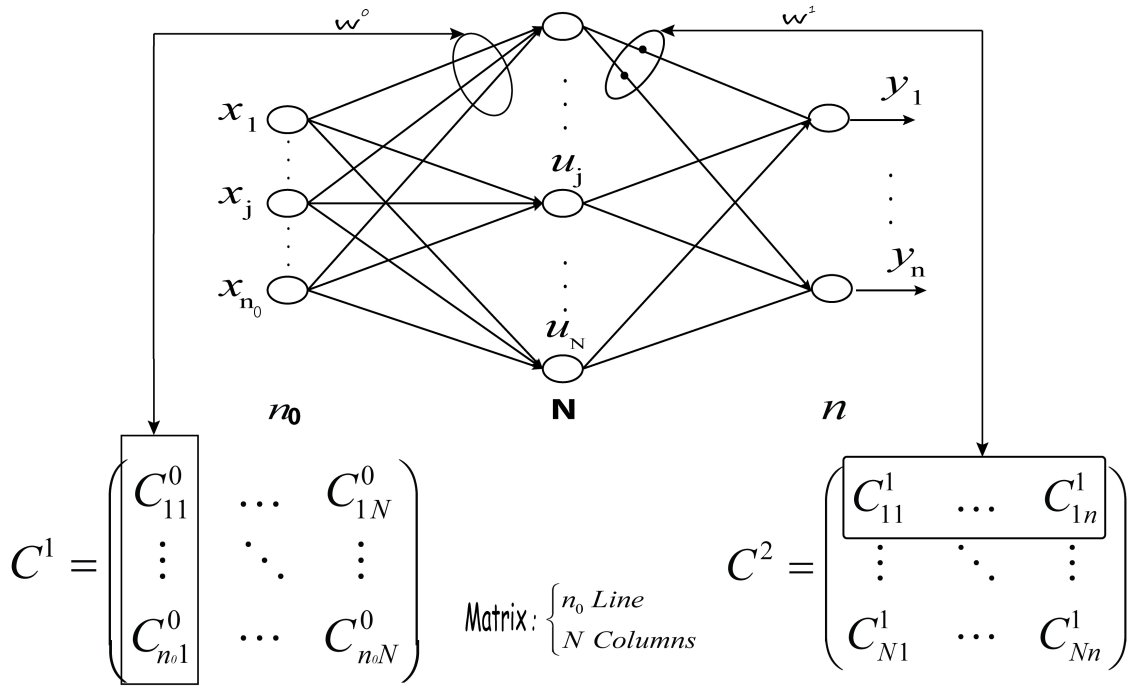


Figure 5.1: MLPNN structure coding using the connections matrix (C_{ij}^k), weight matrix (W_{ij}^k) and neurons decision vector (U_i).

Fitness assignment. The population of individuals are evaluated according to the Pareto domination theory. Moreover, each individual is characterised by the rank and crowding distance values. Such as, the best individuals have the smaller rank and the largest crowding distance value.

Parent selection. The selection phase ensures the creation of the offspring population for the next generation. The best individuals (solutions) are chosen to create a better and fitter next-generation individuals. In the literature, many selection methods are proposed [31], [52]. However, since we use the NSGA II algorithm, the crowded tournament selection operator is employed for picking the next generation parents. The method consists to select randomly a set of k individuals, then, ranked according to their relative fitness (rank and crowding distance) and the fittest individual is selected for reproduction. The whole process is repeated n times until the selection proportion is reached.

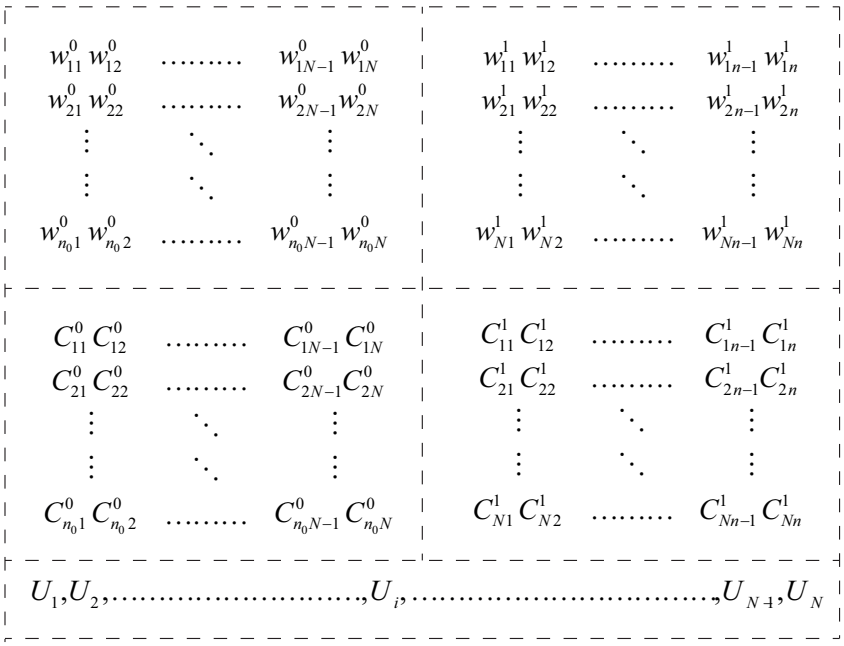


Figure 5.2: The Coresponding Individual Coding

Correction. At this phase, each child produced is a potential solution to the proposed model, accordingly, a verification-correction step is necessary. Each resulting individual of production operators goes through a model constraints verification, ones an individual is detected violates the problem constraints, the individual goes to the correction step to fit the proposed model constraints.

Replacement. Since the NSGA II is an elitist algorithm, the steady-state population replacement strategy is used. The offspring population is combined with the current population. Moreover, the N (population size) best individuals are selected, using the non-dominated ranking strategy, to create the next population generation.

Stopping criteria. We use a maximum number iteration as stopping criteria.

5.3.3 Modified LASSO regularization model adaptation

However, for each model, we propose a specific adaptation presented by different operators setting. In this part, we detail the used operators for the modified LASSO regularization model.

Crossover. This step is a convergence operation, which is intended to pull the population towards local optimality [88]. We applied the operation in two points chosen randomly. *Mutation.* Is a divergence operation. It is intended to occasionally break one or more members of a population out of a local minimum/maximum space and potentially discover a better optimal space. The mutation point is randomly chosen.

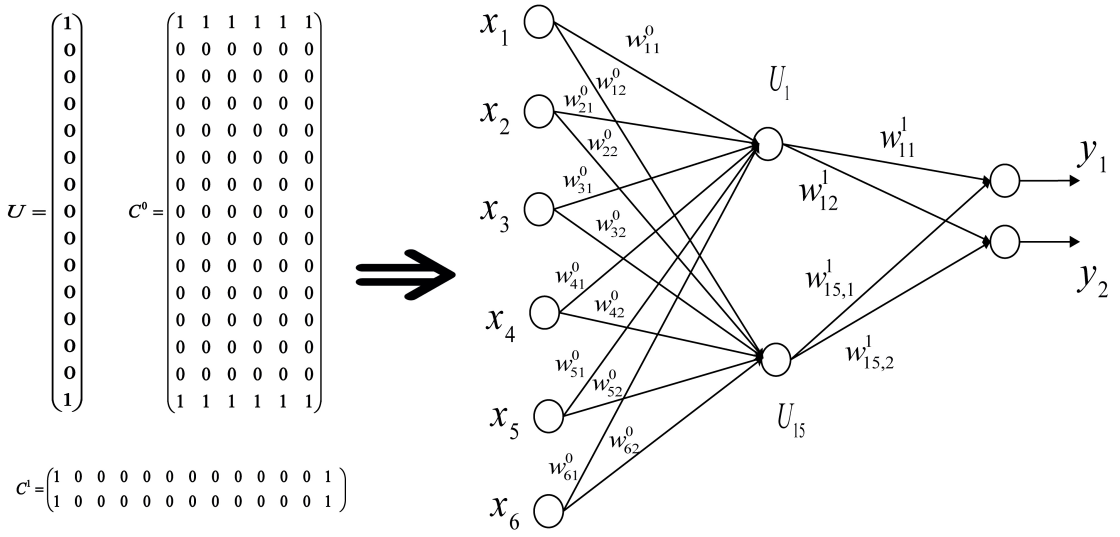


Figure 5.3: Corresponding MLPNN network to an individual coding.

5.3.4 Non-Smooth $L_{1/2}$ regularization model Adaptation

For this part-section, the particular operators for the Non-Smooth $L_{1/2}$ regularization model are developed.

Production Operator: The population resulting from parent selection, offspring population, are used to create new individuals called children. Moreover, the offspring population is divided into two part; each part will undergo crossover or mutation operators. Many different types of crossover and mutation exist in the literature, as follow; we describe the operators adapted for our specific problem:

- Crossover: The crossover scheme employed in the current algorithm simulates the binary crossover attended in nature. We apply the crossover technique for each variable individually.

  Crossover for binary variables(C_{ij}^k connections matrix and U_i neurons decision vector): The crossover used is given as bellow:

$$Y_{1,k} = \alpha_k X_{1,k} + (1 - \alpha_k) X_{2,k} \qquad \forall k = 1, \ldots, m \tag{5.3.1}$$

$$Y_{2,k} = \alpha_k X_{2,k} + (1 - \alpha_k) X_{1,k} \qquad \forall k = 1, \ldots, m \tag{5.3.2}$$

  Such as $X_{(i,k)}$ is the selected parent, $Y_{(1,k)}$ is the i^{th} child with k^{th} component. Where α_k is a binary variable randomly chosen. The formula can be seen as a multipoint crossover choose randomly, as explained above in Fig. 5.4;

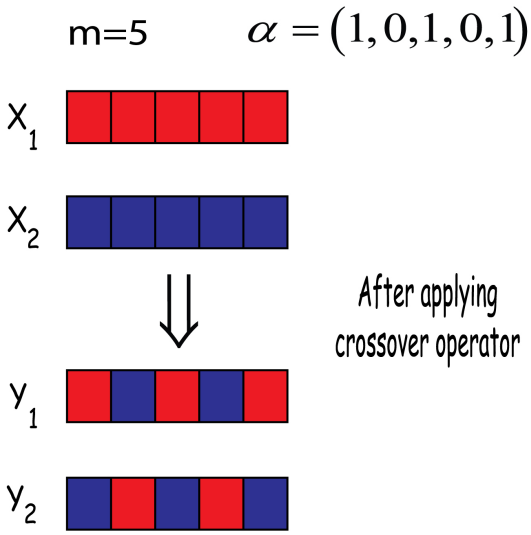


Figure 5.4: The binary crossover process.

Crossover for continuous variable (W_{ij}^k weights matrix): we adopt the same equations used for the binary crossover, with α_k is a continuous parameter, chosen by a uniform distribution in [0,1].

- Mutation: The mutation employed is described as follow:
  The mutation for the continuous variable: the genes to mutate, of the selected parent, are chosen according to a mutation rate. Ones the genes are selected we apply to the following formula:

$$Y(i) = X(i) + (\alpha - 0.5) * 14 \tag{5.3.3}$$

  Such as i is the selected gene, X is the selected parent and Y is the obtained child. Moreover, α is a continuous parameter chosen by a uniform distribution in [0,1].
  Mutation for binary variables: For the binary variable, we choose to apply a simple mutation, which is, commuting between several genes values chose randomly. The genes to permute are selected according to a mutation rate. Fig. 5.5 represents the described process.

5.3.5 Models identification

The present approach proposes the NSGA II as a solver. Accordingly, the process is systematically followed in purpose to determine the adequate (accuracy, complexity) MLPNN. The above-mentioned MLPNN structure is obtained by computing the Pareto front solutions using a significant parameter decision, regarding the application field (classification, approximation,...). Fig 5.6 presents the general flowchart methodology for the proposed process.

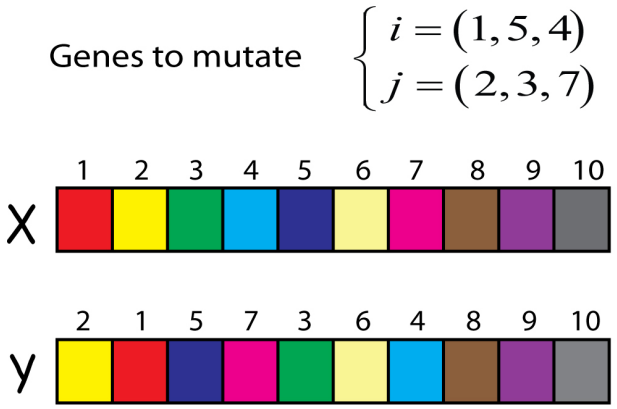


Figure 5.5: The mutation process for binary variables.

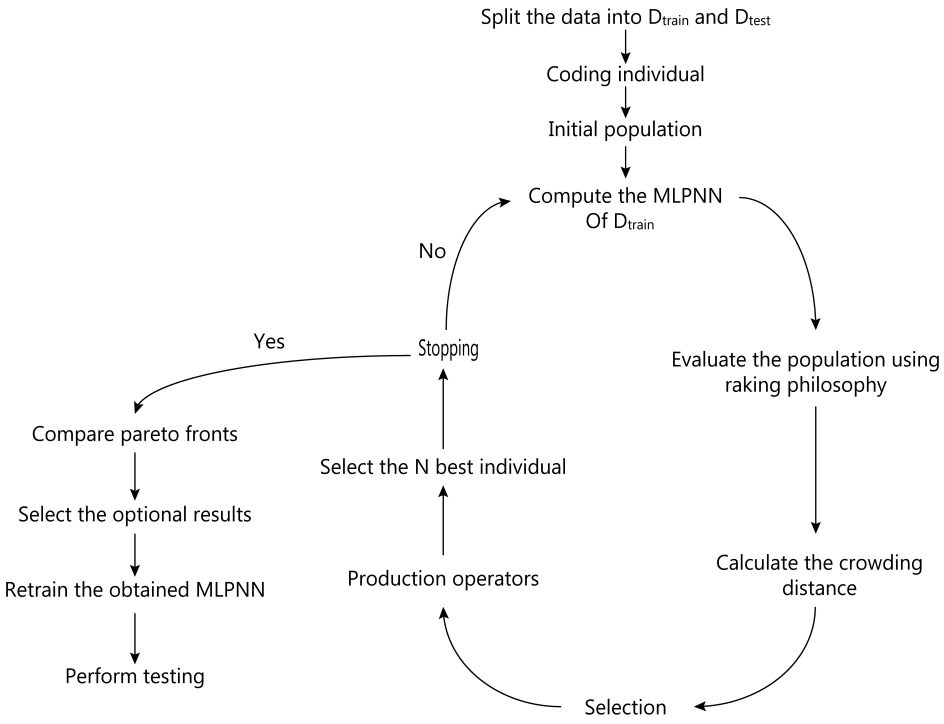


Figure 5.6: The flowchart of the methodology.

5.4 Experiment Results

5.4.1 Description of used Data set

In this part of the chapter, we perform a range of tests to the proposed approach using standard benchmark data sets from the University of California Irvine (UCI) machine-learning repository [119]. We chose four classification problems: Fisher's

iris data set, Wine data, Breast cancer Wisconsin and Seed data set. The table 5.1 present an overview of the used data set characteristics: the number of examples, attributes and class are mentioned.

Table 5.1: Characteristics of used database.

Database	Examples	Attributes	Class
Iris	150	4	3
Wine	178	13	3
Cancer	699	9	2
Seed	210	7	3

Iris database: consists of three target classes: Iris Setosa, Iris Virginica and Iris Versicolor, each species contains 50 data samples. Each sample has four real-valued features: Sepal Length, Sepal width, Petal Length and Petal width, all in centimetres.

The wine database consists of three target classes: the first class corresponds to 59, the second corresponds to 71 and the last one corresponds to 48 examples, each sample has 13 real-valued features. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The database includes 178 instances.

Breast Cancer Wisconsin database, collected by Dr W. H. Wolberg at the University of Wisconsin Madison Hospitalists consists of 699 cases, of which 458 are diagnosed as benign and the remaining 241 are known to be malignant. There are no missing attributes in the data set, and in this case, we are interested in classifying the breast tumour as benign and malignant. These two-class problems determine if a patient suffers from breast cancer based on several characteristics of the nuclei's cells. We are used the first 349 of the patterns as the training set (i.e, for optimizing the NN weights), and the remaining 350 as the test set.

Seed database represents measurements of geometrical properties of kernels belonging to three different varieties of wheat. The examined group comprised kernels belonging to three different varieties of wheat: the Kama, Rosa and Canadian, 70 elements each, randomly selected for the experiment. The data, also, have 7 attributes.

5.4.2 Network parameter

To outline the chosen network parameter, we specify that the algorithm, at first, starts with a proposed MLPNN architecture to converge to an adequate architecture (Fig 5.7), the initial architecture consists of a single hidden layer containing 10 hidden layers for Iris, 15 for Wine, Cancer and Seed. Moreover, the MLPNN weights are randomly generated in [-0.5,0.5]. Furthermore, since we study the case of classification problem (data described above), we use the Softmax function described below, as MLPNN activation function:

Table 5.2: NSGA II Parameter setting for each tested data.

Parameter	Iris	Wine	Cancer	Seed
Population size	25	20	25	20
Crossover rate	70%	70%	70%	70%
Mutation rate	40%	40%	40%	40%
Stopping criteria (it)	100000	30000	100000	30000

$$f_j(x,w) = \frac{\exp(a_j(x,w))}{\sum_k \exp(a_k(x,w))} \tag{5.4.1}$$

The error function of the proposed model became:

$$E_D(w,C,U) = -\sum_{j=1}^{J}\sum_{i=1}^{n} d_j^i \log\left(F_j^i(U,C,X,w)\right) \tag{5.4.2}$$

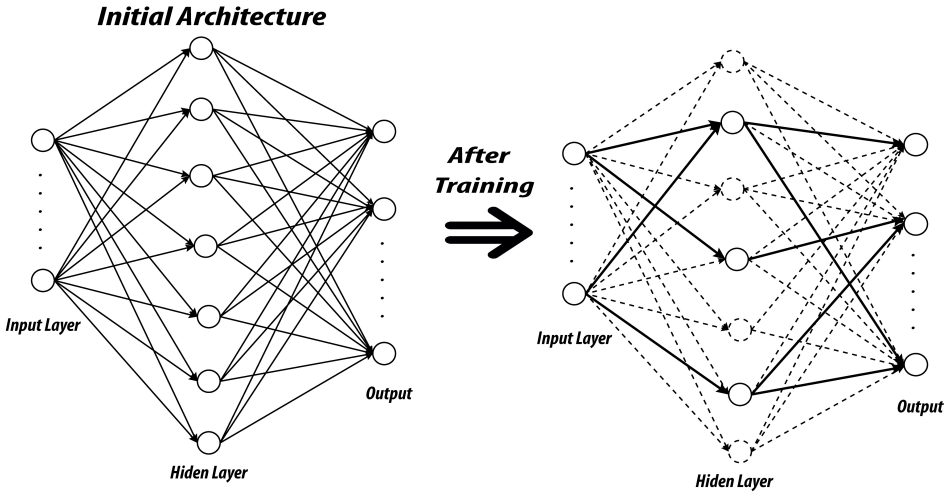


Figure 5.7: MLPNN architecture before and after training.

5.4.3 NSGA II parameter

The NSGA II algorithm application requires, as well, a parameter setting. For each used data, the different setting was proposed; the following table (5.2) shows the proposed parameters:

5.4.4 Performance measurement

To measure the performance of a multiobjective optimization method, we mainly adopt the Waves measure [120], used to visualize the population convergence to the compromised surface.

This metric corresponds to the maximum rank of the Pareto classification. Each iteration, the maximum rank obtained is represented; the shape of the curve thus obtained is that of waves. If the max rank value approximate one during optimization, the individuals can be considerate as converging to the trade-off surface. The following algorithm can be obtained by the Pareto maximum rank:

Algorithm 6: Algorithm : Max Pareto Rank

```
S is the solution set
d=0
while card(S) = 0 do
    S = S − Pareto(S)
    d=d+1
end while
MaxRank=d
```

The function Pareto determines a set of no-dominated solutions of the set given as a parameter. The waves measure allows measuring the spread of the calculated compromise surface (PPT), such as:

$$W = MaxRank\,(PPT) \tag{5.4.3}$$

If W is large, it means that the population is not grouped at the compromised area, many individuals are not "efficient". Otherwise, if the metric value approximates 1, it means that the individuals tend to converge towards the compromised surface.

5.4.5 Modified LASSO regularization Numerical Results

Firstly, the Pareto front for the proposed model is obtained, however, it isn't enough, for the discussed problem (classification, interpolation), the users need only a single solution. It is, from the algorithm point of view, all the solution in the Pareto front are equally important. For this, many approaches was proposed [49]. We have chosen the must adequate solution between the obtained optimal solutions by an approach basing on the crowding distance. The solution obtained is illustrated in 5.3.

The P. Method in table 5.3 represent the proposed model and Back Propagation algorithm (BP)+LASSO represent the Backpropagation method with a regularized error. Well, to prove the efficiency of our approach, we compare it to BP+LASSO. The backpropagation is applied to minimize the MLPNN error with a second term, aiming to regulate the current error, the error is presented by Equ. 5.4.4.

$$E(w) = \phi_1(w, C, U) + \beta \sum |w_i| \tag{5.4.4}$$

β is a parameter fixed by testing several values $(0,05; 0,1; 0,15\ldots)$, the best results are found at$\beta = 0.1$. For which presented results in table 5.3.
Moreover, The rate Class (%) presented in table 5.3 is calculated by Equ. 5.4.5.

Table 5.3: Modified LASSO Model Obtained Results

Method	Data	MLP Error	$\sum \lvert C_i w_i \rvert$	$\sum \lvert w_i \rvert$	n_w	$n_n euron$	Class (%)
BP+LASSO	Iris	0,8	-	30,25	105	15	2,66
	Wine	0,38	-	51,29	240	15	3,37
	Cancer	0,91	-	42,38	165	15	2,28
	Seed	0,56	-	49,21	150	15	5,38
	Thyroid	0,74	-	69,33	360	15	3,53
P. Method	Iris	0.85	16,43	-	30	6	1,33
	Wine	0,39	34,56	-	80	8	2
	Cancer	0.94	10,12	-	6	8	2
	Seed	0,64	16,58	-	36	7	4,71
	Thyroid	0,76	22,19	-	208	10	2,87

$$Class\,(\%) = \frac{Misclassified\ \times 100}{Card\,(Testing\ set)} \tag{5.4.5}$$

From the presented results, the performed pruning is considerably good. Considering that, for data iris, *23.8%* connexions network were removed (25 connexions). Well, for data Wine, *66.66%* of the network connexions were pruned (160 connexions). However, for data Cancer, *95.15%* of the network was simply displaced. Moreover, *76%* were pruned for data seed, in addition to *42.22%* for data Thyroid.

From table 5.3 we can see that our approach gets better results. The number of connections and neurons are considerably reduces compared to the BP+LASSO Method. Table 5.3, also shows a better miss-classified rate for the proposed model. In this fact, the results are obtained through the added decision variable to the regularization term. Moreover, the proposed model is able to provide good generalization by reducing the MLPNN topology.

5.4.6 Non-Smooth $L_{1/2}$ Regularization Numerical Results

Ones the parameters are settled, we run the program several times, the obtained Pareto Fronts for different data are presented by Fig. 5.8.

From figure 5.8, we can observe a correct Pareto front shape for all uses data; unfortunately, it isn't enough to judge the algorithm performance. Fig. 5.9. Emphases the waves measure presentation, from this latter; we can notice the fast convergence of the algorithm from the first iterations. Ones the algorithm is getting closer to the maximum number of iterations, the maximum rank number some to be stable at 1. That can be explained by the efficiency of the final population. It means that individuals converge certainly to the compromised surface.

The Pareto-optimality concept is a primary step in approaching a multi-objective model optimization. The selection of an appropriate solution compromising between accuracy and complexity from the Pareto front (all non-dominate solution) is not

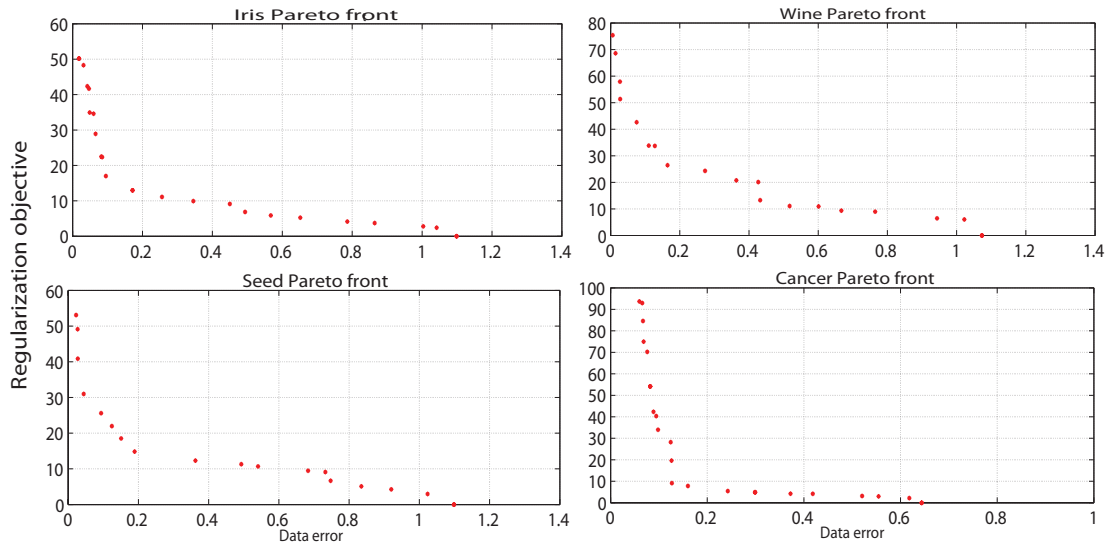


Figure 5.8: The obtained Pareto front.

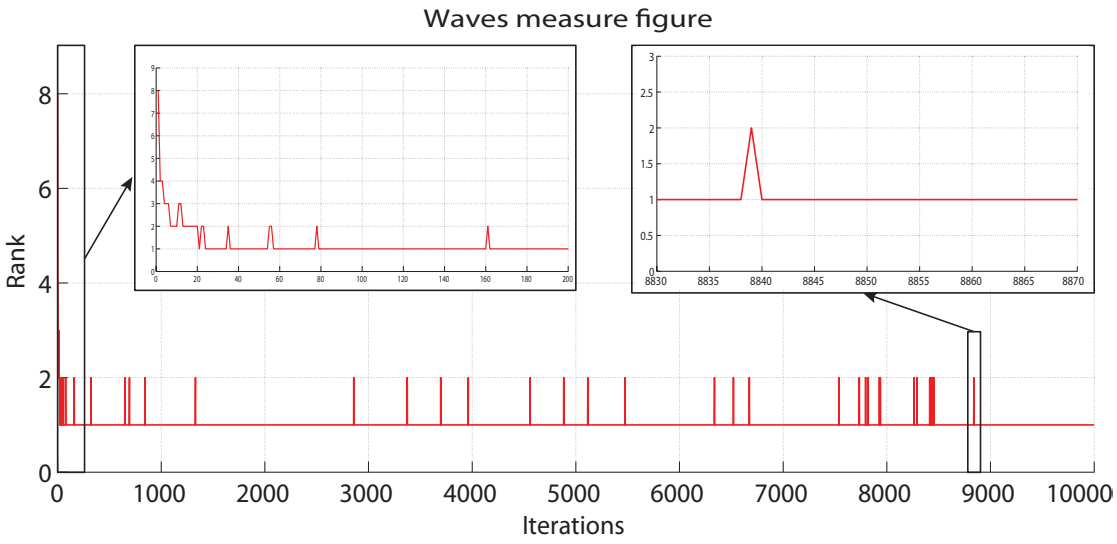


Figure 5.9: The waves performance measure.

just a problem-dependent, the choice is usually determined regarding the personal preferences of the decision-maker. Therefore, the problem final solution is the result of both an optimization and a decision process. In purpose to make the right choice among the obtained Pareto front, we propose to use the misclassified rate as decision parameter; the solution with the best miss-classified rate will be selected as the best-obtained solution, presented by Equ. 5.4.5.

In general, the pruning of the network achieved can be considerate as remarkable. The removed weights number were 71 weights for data iris, which is 85.54% of the MLPNN initial weights, 175 weights for data wine (67.82%), in addition to 127

weights for data cancer (77.91%), and data seed. Accompanying to this large weights elimination, hidden neurons was pruned too, we record 6 hidden neurons (40%) for data iris, against 9 hidden neurons (60%) for data wine, 10 hidden neurons (66%) for cancer data and finally for Seed data. However, Fig.5.10 emphases the resulting MLPNN for the data iris.

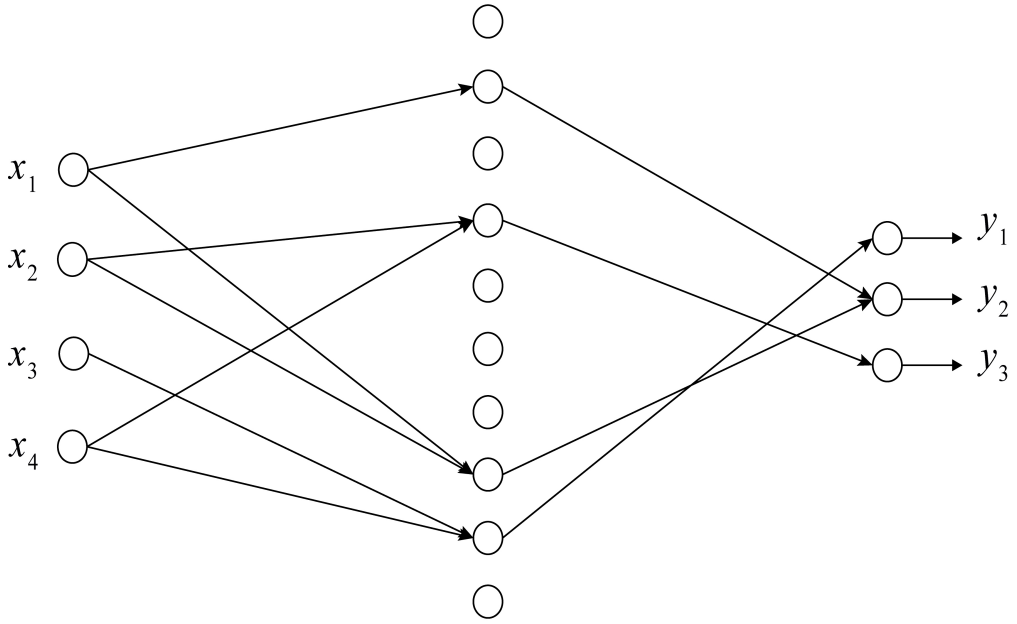


Figure 5.10: Obtained neural network for Iris Data.

We propose to compare the obtained results to four different methods, the first method is BP+LASSO, the backpropagation algorithm with a LASSO regularization function. We compare the obtained results to our intern work, Senhaji et al. [108] the multiobjective model presented in section 5.2.3. In addition to NN-MOPSPCD [129], the paper proposes to optimize the weights and architecture, through a Particle Swarm Optimization (Partical Swarm Optimization (PSO))-based multiobjective algorithm, for MLPNN training. The MOBJ (Multi-OBJective training) [13] approach is also compared; the model minimizes the squared error and the norm of weights, using vector optimization methods. The last method is LASSO (Last Absolute Shrinkage and Selection Operator) [24] a bi-objective algorithm proposed for minimizing the residual sum of squares subject to the sum of the absolute weights being less than a constant t.

The table 5.4 present a descriptive of the obtained solution compared to four different methods (mono and multiobjective). The MLPNN error presented in table 5.4 is the error obtained by training the data test set of each data. n_w and n_{neuron} are respectively the number of weights and neurons after pruning. The misclassified rate is also included in table 5.4. The complexity objective is presented by three different functions depending on the compared paper. Our proposed function is presented by $\Phi(w, C, U)$. We base our performance comparison on the mis-classified rate, number

Table 5.4: Table of compared results.

Methods	Data	MLP Error	$\sum \lvert C_i w_i \rvert$	$\sum \lvert w_i \rvert$	$\Phi(w, C, U)$	n_w	n_{neuron}	Class (%)
BP+ LASSO	Iris	0,8	-	30,25	-	105	15	2,66
	Wine	0,38	-	51,29	-	240	15	3,37
	Cancer	0,91	-	42,38	-	165	15	2,28
	Seed	0,56	-	49,21	-	150	15	5,38
Senh et al.	Iris	0.85	16,43	-	-	30	6	1,33
	Wine	0,39	34,56	-	-	80	8	2
	Cancer	0.94	10,12	-	-	6	8	2
	Seed	0,64	16,58	-	-	36	7	4,71
Our Approach	Iris	0,06	-	-	28,90	11	4	2.66
	Wine	0.006	-	-	75.36	83	6	1.12
	Cancer	0.08	-	-	54.14	36	10	1.42
	Seed	0.09	-	-	25.56	41	5	2.85
NN-MOPSOCD	Iris	0.045	-	-	-	66.02	-	6.24
MOBJ	Cancer	0.9575	-	18.22	-	55	10	4.3
LASSO	Cancer	0.9625	-	13.78	-	10	2	3.76

of resulting weights and neurons.

Basing on the obtained results in table 5.4, we can report our approach superiority for all datasets, regarding the misclassified rate, compared to the suggested method in the table. Considering each data severally, such as Data iris, we report four hidden neurons left after pruning against 6 for our previous work [108] and unchanged (15 hidden neurons) for BP+LASSO. The weight number after pruning decreased to 12 against 30 for [108], 66.08 for NN-MOPSOCD method [129] and 105 for BP+LASSO. Considering Cancer data, we report 10 hidden neurons left against two for the LASSO model, 8 for [108] and 10 for MOBJ model [13]. When it comes to the weights number left, we record 36 for our approach alongside with six weights left for [108], 55 for MOBJ model [13] and 10 for LASSO [13]. For the remaining datasets, we compute the obtained results with BP+LASSO and our previous work results; since the other proposed methods have not considered it as reported results. Even if we can record better pruning than the proposed approach for some datasets such as cancer dataset using the LASSO model, the method still reporting better classification compared to the aforementioned methods.

5.5 Conclusion

The present chapter delivered a brand optimized MLPNN basing on multiobjective learning models. The proposed training models balance between MLPNN accuracy and complexity by controlling the variation of the weight of the network during the learning process presented as two objectives under connexion constraints. We propose, as well, to adapt the NSGA II algorithm, an efficient variation of the multiobjective genetic algorithm, to the current models. To outline the results efficiencies of the proposed methods, we compute it to different methods, multiobjective one along with classical methods; the method was applied to a variety of data classi-

fication; judging on the obtained results, the multiobjective MLPNN models allow to reduce effectively unnecessary network nodes and connexions, compared to others algorithm. To concludes, we can confirm that the approach can reach higher generalization solutions for different Data problems, as well as a reduced topology.

General conclussion and perspectives

Multi-objective optimization offers the user more freedom to model their problem. It also highlights certain modeling difficulties that are often avoided in mono-objective optimization. Indeed, it often makes a compromise between certain objectives, without the knowledge of the designer of the model. Multi-objective optimization brings this compromise to light and this is also the difficulty of multi-objective optimization. It must be kept in mind that multi-objective optimization will not treat the compromise with a magic wand. It will always be to the decision-maker to decide and choose one solution over another. Still, multi-objective optimization offers tools to help the decision-maker to decide. In this thesis, the multiobjective optimization approach was used for two different problems; Portfolio selection problem is an interesting economic problem, many researchers proposed different models to approach the risk/profitability balance. However, the proposed work was inspired by the Markovitz model. This later, use the variance and semivariance as risk measurement. In this context, we proposed multiobjective models using the Markowitz models adding new constraints. Moreover, for the model resolving, different approaches were operated. Considering the variance as risk measurement, we found the continuous Hopfield neural network (CHN) is more adequate. Indeed, the quadratic form of the objective function makes the Hopfield neural network implementation easier. This latter has provided good empirical results. However, we noticed that the level risk is minimized at the expense of profitability, which is expected since we used a scalarization method to express the two objectives as one function. Moreover, the behaviour of the used risk aversion parameter weakens the CHN energy function, it seems to be linear ones $\lambda \leq 0.5$. For this reason, we proposed to avoid the explained energy default by a CHN-Genetic algorithm hybridization. According to the obtained efficient frontier, the approach can be considered efficient. However, the energy function default is caused by using an aversion parameter, the more appropriate approach to avoid such default by using a Pareto approach. In fact, for the mean-semivariance model, we proposed to perform the results using a multiobjective genetic algorithm with a CHN-initial population. the second part of the thesis addresses a different type of multiobjective problem. Naturally, the learning is multicriteria, especially, the artificial neural network learning. Since the complexity of networks plays an important role in network training performance. Accordingly, two different multiobjective models were proposed taking into account: the MPLNN complexity and accuracy. In this context, Non- dominated genetic algorithm was adapted to each model. Basing on the obtained results, the approach

can record higher accuracy with reduced complexity. The performance of all contributions have been demonstrated by several experiments, which have been divided into visual, graphical and numerical experiments.

Bibliography

[1] *7 types of neural network activation functions: how to choose?* `https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/`, Accessed: 2020-10-23.

[2] E. Abdelatif, H. Fidae, and E. Mohamed, "Optimization of the organized kohonen map by a new model of preprocessing phase and application in clustering," *Journal of Emerging Technologies in Web Intelligence*, vol. 6, no. 1, pp. 80–84, 2014.

[3] H. Abdi, *Les réseaux de neurones*. Presses universitaires de Grenoble Grenoble, 1994.

[4] F. Aftalion, *La nouvelle finance et la gestion des portefeuilles*. Economica, 2004.

[5] R. de Albuquerque Teixeira, A. P. Braga, R. H. Takahashi, and R. R. Saldanha, "Improving generalization of mlps with multi-objective optimization," *Neurocomputing*, vol. 35, no. 1-4, pp. 189–194, 2000.

[6] R AlHalaseh, A Islam, and R Bakar, "Portfolio selection problem: models review," *The Social Sciences*, vol. 11, no. 14, pp. 3408–3417, 2016.

[7] V. Barichard, "Approches hybrides pour les problèmes multiobjectifs," Ph.D. dissertation, Université d'Angers, 2003.

[8] H. B. Barlow, "Unsupervised learning," *Neural computation*, vol. 1, no. 3, pp. 295–311, 1989.

[9] M. S. Bartlett, "An inverse matrix adjustment arising in discriminant analysis," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 107–111, 1951.

[10] J. Bi, "Multi-objective programming in svms," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 35–42.

[11] P. Borne, M. Benrejeb, and J. Haggège, *Les réseaux de neurones: présentation et applications*. Editions OPHRYS, 2007, vol. 15.

[12] A. Bouzerdoum and T. R. Pattison, "Neural network for quadratic optimization with bound constraints," *IEEE transactions on neural networks*, vol. 4, no. 2, pp. 293–304, 1993.

[13] A. P. Braga, R. H. Takahashi, M. A. Costa, and R. de Albuquerque Teixeira, "Multi-objective algorithms for neural networks learning," in *Multi-objective machine learning*, Springer, 2006, pp. 151–171.

[14] L. Burke and J. P. Ignizio, "A practical overview of neural networks," *Journal of Intelligent Manufacturing*, vol. 8, no. 3, pp. 157–165, 1997.

[15] P. A. Castillo, J. Merelo, A. Prieto, V Rivas, and G. Romero, "G-prop: global optimization of multilayer perceptrons using gas," *Neurocomputing*, vol. 35, no. 1-4, pp. 149–163, 2000.

[16] Z. Chen and S. Haykin, "On different facets of regularization theory," *Neural Computation*, vol. 14, no. 12, pp. 2791–2846, 2002.

[17] A. Cheriet, "Métaheuristique hybride pour l'optimisation multi-objectif," Ph.D. dissertation, Université Mohamed Khider-Biskra, 2016.

[18] A Cochocki and R. Unbehauen, *Neural networks for optimization and signal processing*. John Wiley & Sons, Inc., 1993.

[19] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, "Mop evolutionary algorithm approaches," *Evolutionary Algorithms for Solving Multi-Objective Problems: Second Edition*, pp. 61–130, 2007.

[20] C. Coello Coello, "An update survey of ga-based multiobjective optimization techniques. xalapa, veracruz, laboratorio nacional de informática avanzada (lania): 45," Technical Report Lania-RD-98-08, Tech. Rep., 1998.

[21] Y. Collette and P. Siarry, *Optimisation multiobjectif: Algorithmes*. Editions Eyrolles, 2011.

[22] A. Corhay and M. Mbangala, *Fondements de gestion financière: manuel et applications*. Editions du CEFAL, 2008.

[23] D. W. Corne, J. D. Knowles, and M. J. Oates, "The pareto envelope-based selection algorithm for multiobjective optimization," in *International conference on parallel problem solving from nature*, Springer, 2000, pp. 839–848.

[24] M. A. Costa and A. P. Braga, "Optimization of neural networks with multiobjective lasso algorithm," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, IEEE, 2006, pp. 3312–3318.

[25] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[26] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: nsga-ii," in *International conference on parallel problem solving from nature*, Springer, 2000, pp. 849–858.

[27] M. Ehrgott, "A characterization of lexicographic max-ordering solutions," 1999.

[28] J. Estrada, "Downside risk in practice," *Journal of Applied Corporate Finance*, vol. 18, no. 1, pp. 117–125, 2006.

[29] ——, "Mean-semivariance optimization: a heuristic approach," *Journal of Applied Finance (Formerly Financial Practice and Education)*, vol. 18, no. 1, 2008.

[30] C. M. Fonseca, P. J. Fleming, *et al.*, "Genetic algorithms for multiobjective optimization: formulationdiscussion and generalization.," in *Icga*, Citeseer, vol. 93, 1993, pp. 416–423.

[31] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms*, vol. 1, Elsevier, 1991, pp. 69–93.

[32] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[33] C. Goutte and L. K. Hansen, "Regularization with a pruning prior," *Neural Networks*, vol. 10, no. 6, pp. 1053–1059, 1997.

[34] N. Gülpınar and B. Rustem, "Worst-case robust decisions for multi-period mean–variance portfolio optimization," *European Journal of Operational Research*, vol. 183, no. 3, pp. 981–1000, 2007.

[35] H. Hachimi, "Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications," Ph.D. dissertation, 2013.

[36] T. Hatanaka, N. Kondo, and K. Uosaki, "Multi-objective structure selection for radial basis function networks based on genetic algorithm," in *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, IEEE, vol. 2, 2003, pp. 1095–1100.

[37] M. X. He, S. V. Petoukhov, and P. E. Ricci, "Genetic code, hamming distance and stochastic matrices," *Bulletin of mathematical biology*, vol. 66, no. 5, pp. 1405–1421, 2004.

[38] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[39] D. O. Hebb and D. Hebb, *The organization of behavior*. Wiley New York, 1949, vol. 65.

[40] K. Hingee and M. Hutter, "Equivalence of probabilistic tournament and polynomial ranking selection," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, IEEE, 2008, pp. 564–571.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] W. W. Hogan and J. M. Warren, "Toward the development of an equilibrium capital-market model based on semivariance," *Journal of Financial and Quantitative Analysis*, vol. 9, no. 1, pp. 1–11, 1974.

[43] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[44] ——, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[45] J. J. Hopfield and D. W. Tank, ""neural" computation of decisions in optimization problems," *Biological cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

[46] J. rey Horn, N. Nafpliotis, and D. E. Goldberg, "Multiobjective optimization using the niched pareto genetic algorithm," *IlliGAL report*, vol. 93005, 1993.

[47] K. Hornik, M. Stinchcombe, H. White, *et al.*, "Multilayer feedforward networks are universal approximators.," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[48] Z.-H. Huang, X.-H. Miao, and P. Wang, "A revised cut-peak function method for box constrained continuous global optimization," *Applied Mathematics and Computation*, vol. 194, no. 1, pp. 224–233, 2007.

[49] C.-L. Hwang and A. S. M. Masud, *Multiple objective decision making—methods and applications: a state-of-the-art survey*. Springer Science & Business Media, 2012, vol. 164.

[50] C. Igel, "Multi-objective model selection for support vector machines," in *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 2005, pp. 534–546.

[51] R. A. Jarrow, *Finance theory*. Prentice Hall, 1988.

[52] K. Jebari and M. Madiafi, "Selection methods for genetic algorithms," *International Journal of Emerging Sciences*, vol. 3, no. 4, pp. 333–344, 2013.

[53] Y. Jin, *Multi-objective machine learning*. Springer Science & Business Media, 2006, vol. 16.

[54] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: an overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[55] N.-e. Joudar and M. Ettaouil, "Mathematical mixed-integer programming for solving a new optimization model of selective image restoration: modelling and resolution by chn and ga," *Circuits, Systems, and Signal Processing*, vol. 38, no. 5, pp. 2072–2096, 2019.

[56] S. Kaoutar and E. Mohamed, "Multi-criteria optimization of neural networks using multi-objective genetic algorithm," in *2017 Intelligent Systems and Computer Vision, ISCV 2017*, 2017.

[57] S. Kaoutar and E. Mohamed, "Multi-criteria optimization of neural networks using multi-objective genetic algorithm," in *2017 Intelligent Systems and Computer Vision (ISCV)*, IEEE, 2017, pp. 1–4.

[58] E. I. A. Kareem, W. A. A. Alsalihy, and A. Jantan, "Multi-connect architecture (mca) associative memory: a modified hopfield neural network," *Intelligent Automation & Soft Computing*, vol. 18, no. 3, pp. 279–296, 2012.

[59] R. L. Keeney and H. Raiffa, *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press, 1993.

[60] D. Kim, "Structural risk minimization on decision trees using an evolutionary multiobjective optimization," in *European Conference on Genetic Programming*, Springer, 2004, pp. 338–348.

[61] Y. Kim, W. N. Street, and F. Menczer, "Evolutionary model selection in unsupervised learning," *Intelligent data analysis*, vol. 6, no. 6, pp. 531–556, 2002.

[62] J. Knowles and D. Corne, "The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation," in *Congress on Evolutionary Computation (CEC99)*, vol. 1, 1999, pp. 98–105.

[63] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[64] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: a tutorial," *Reliability Engineering & System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[65] H. Konno, K. Akishino, and R. Yamamoto, "Optimization of a long-short portfolio under nonconvex transaction cost," *Computational Optimization and Applications*, vol. 32, no. 1-2, pp. 115–132, 2005.

[66] J. Koski, "Defectiveness of weighting method in multicriterion optimization of structures," *Communications in applied numerical methods*, vol. 1, no. 6, pp. 333–337, 1985.

[67] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in neural information processing systems*, 1992, pp. 950–957.

[68] A. Land, "Doig. ag,"an automatic method for solving discrete programming problems"," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[69] E. G. Learned-Miller, "Introduction to supervised learning," *I: Department of Computer Science, University of Massachusetts*, 2014.

[70] G. Liu and V Kadirkamanathan, "Learning with multi-objective criteria," in *1995 Fourth International Conference on Artificial Neural Networks*, IET, 1995, pp. 53–58.

[71] O. Ludwig and U. Nunes, "Novel maximum-margin training algorithms for supervised neural networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 6, pp. 972–984, 2010.

[72] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing qos data via aggregating nonnegative latent factor models," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 3, pp. 524–537, 2015.

[73] E. B. i Mansilla and J. M. G. i Guiu, "Molecs: using multiobjective evolutionary algorithms for learning," in *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 2001, pp. 696–710.

[74] S Marcos, C Vignat, and O Macchi, "L'algorithme de rétropropagation du gradient en réseaux de neurones comparé aux algorithmes adaptatifs du traitement du signal'," *Internal Rep., Laboratoire des Signaux et Systèmes, Gif sur Yvette*, 1990.

[75] H. Markowitz, "Portfolio selection," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[76] ——, "Portfolio selection: efficient diversification of investments, new york: johnwiley and sons, inc.," *MarkowitzPortfolio Selection: Efficient Diversification of Investments*, 1959.

[77] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[78] M. Marvin and P. Seymour, *Perceptrons*, 1969.

[79] Y. Matsuyama, "Harmonic competition: a self-organizing multiple criteria optimization," *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 652–668, 1996.

[80] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[81] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[82] T. Murata and H. Ishibuchi, "Moga: multi-objective genetic algorithms," in *IEEE international conference on evolutionary computation*, vol. 1, 1995, pp. 289–294.

[83] H. Nakayama, "Support vector machines formulated as multi objective linear programming," *Proceedings of ICOTA2001*, vol. 3, pp. 1171–1178, 2001.

[84] P. Ngatchou, A. Zarei, and A El-Sharkawi, "Pareto multi objective optimization," in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, IEEE, 2005, pp. 84–91.

[85] P. M. Pardalos, A. Žilinskas, and J. Žilinskas, *Non-convex multi-objective optimization*. Springer, 2017.

[86] M. Parizeau, "Réseaux de neurones gif-21140 et gif-64326," *Université Laval*, 2004.

[87] M. Pescador-Rojas, R. H. Gómez, E. Montero, N. Rojas-Morales, M.-C. Riff, and C. A. C. Coello, "An overview of weighted and unconstrained scalarizing functions," in *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 2017, pp. 499–513.

[88] P. W. Poon and J. N. Carter, "Genetic algorithm crossover operators for ordering applications," *Computers & Operations Research*, vol. 22, no. 1, pp. 135–147, 1995.

[89] *Porfofio efficient frontier*, `http://www.smart401k.com/Content/retail/resource-center/advanced-investing/modern-portfolio-theory-and-the-efficient-frontier`, Accessed: 2020-10-23.

[90] L. Prechelt, "Early stopping-but when?" In *Neural Networks: Tricks of the trade*, Springer, 1998, pp. 55–69.

[91] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941*, vol. 7, 2017.

[92] H Ramchoun, M. A. J. Idrissi, Y Ghanou, and M Ettaouil, "New modeling of multilayer perceptron architecture optimization with regularization: an application to pattern classification," *network*, vol. 1, no. 2, p. 4, 2017.

[93] H. Ramchoun, M. A. J. Idrissi, Y. Ghanou, and M. Ettaouil, "Multilayer perceptron: architecture optimization and training.," *IJIMAI*, vol. 4, no. 1, pp. 26–30, 2016.

[94] F Rosenblatt, "The perceptron-a perceiving and recognizing automaton (85-460-1)," *Ithica: Cornell Aeronautical Laboratory*, 1957.

[95] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[96] B. Roy, *Méthodologie multicritère d'aide à la décision*, BOOK. Economica, 1985.

[97] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[98] ——, "Learning representations by backpropagating errors, nature, 323 (6088), 533," *Volume*, vol. 1, pp. 1555–1565, 1986.

[99] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[100] S. SAIT, "Iterative computer algorithms with applications in engineering," *IEEE CS Press*, pp. 352–355, 1999.

[101] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications, 1985*, Lawrence Erlbaum Associates. Inc., Publishers, 1985.

[102] J. D. Schaffer, "Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition)," 1984.

[103] K. Senhaji, K. El Moutaouakil, and M. Ettaouil, "A robust neural network approach for the portfolio selection problem basing on new rational models," *Journal of Engineering and Applied Sciences*, vol. 14, no. 3, pp. 675–683, 2019.

[104] ——, "Portfolio selection problem: new multicriteria approach for the mean-semivariance model," in *Proceedings of the 3rd IEEE International Conference on Logistics Operations Management, GOL 2016*, 2016.

[105] K. Senhaji, H. Ramchoun, and M. Ettaouil, *Multilayer perceptron: nsga ii for a new multi-objective learning method for training and model complexity*, Springer, 2019.

[106] ——, "Training feedforward neural network via multiobjective optimization model using non-smooth l<inf>1/2</inf> regularization," *Neurocomputing*, vol. 410, pp. 1–11, 2020.

[107] K. Senhaji and M. Ettaouil, "The portfolio selection problem: hybrid approach to solve the mean-variance rational model," *Boletin de Estadistica e Investigacion Operativa*, vol. 36, no. 1, pp. 25–40, 2020.

[108] K. Senhaji, H. Ramchoun, and M. Ettaouil, "Multilayer perceptron: nsga ii for a new multi-objective learning method for training and model complexity," in *First International Conference on Real Time Intelligent Systems*, Springer, 2017, pp. 154–167.

[109] H.-S. Shih, U.-P. Wen, S Lee, K.-M. Lan, and H.-C. Hsiao, "A neural network approach to multiobjective and multilevel programming problems," *Computers & Mathematics with Applications*, vol. 48, no. 1-2, pp. 95–108, 2004.

[110] J. Sietsma and R. J. Dow, "Creating artificial neural networks that generalize," *Neural networks*, vol. 4, no. 1, pp. 67–79, 1991.

[111] S. Sivanandam and S. Deepa, "Genetic algorithms," in *Introduction to genetic algorithms*, Springer, 2008, pp. 15–37.

[112] K. Smith, M. Palaniswami, and M. Krishnamoorthy, "Neural techniques for combinatorial optimization with applications," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1301–1318, 1998.

[113] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[114] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*, 4. MIT press Cambridge, 1998, vol. 2.

[115] P. M. Talavan and J. Yanez, "A continuous hopfield network equilibrium points algorithm," *Computers & operations research*, vol. 32, no. 8, pp. 2179–2196, 2005.

[116] ——, "Parameter setting of the hopfield network applied to tsp," *Neural Networks*, vol. 15, no. 3, pp. 363–373, 2002.

[117] ——, "The generalized quadratic knapsack problem, a neuronal network approach," *Neural networks*, vol. 19, no. 4, pp. 416–428, 2006.

[118] A. N. Tikhonov and V. I. Arsenin, *Solutions of ill-posed problems*. Winston, Washington, DC, 1977, vol. 14.

[119] UCI, *333 data sets for machine learning*, `https://www.ics.uci.edu/mlearn/MLRepository.html`, 2019 (accessed February 3, 2019).

[120] D. A. Van Veldhuizen, "Multiobjective evolutionary algorithms: classifications, analyses, and new innovations," Ph.D. dissertation, AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING, 1997.

[121] J. Wang, G. Yang, S. Liu, and J. M. Zurada, "Convergence analysis of multilayer feedforward networks trained with penalty terms: a review," *Journal of Applied Computer Science Methods*, vol. 7, no. 2, pp. 89–103, 2015.

[122] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with application to forecasting," in *Advances in neural information processing systems*, 1991, pp. 875–882.

[123] P. Werbos, "Beyond regression: new fools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard University, 1974.

[124] Z. Xu, H. Zhang, Y. Wang, X. Chang, and Y. Liang, "$L_{1/2}$ Regularization," *Science China Information Sciences*, vol. 53, no. 6, pp. 1159–1169, 2010.

[125] Z. Xu, X. Chang, F. Xu, and H. Zhang, "$L_{1/2}$ Regularization: a thresholding representation theory and a fast solver," *IEEE Transactions on neural networks and learning systems*, vol. 23, no. 7, pp. 1013–1027, 2012.

[126] *Yahoo finance. financial data*, `https://fr.finance.yahoo.com/`.

[127] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[128] G. G. Yen and H. Lu, "Hierarchical rank density genetic algorithm for radial-basis function neural network design," *International Journal of Computational Intelligence and Applications*, vol. 3, no. 03, pp. 213–232, 2003.

[129] J. P. T. Yusiong and P. C. Naval, "Training neural networks using multiobjective particle swarm optimization," in *International Conference on Natural Computation*, Springer, 2006, pp. 879–888.

[130] G Zames, N. Ajlouni, N. Ajlouni, N. Ajlouni, J. Holland, W. Hills, and D. Goldberg, "Genetic algorithms in search, optimization and machine learning.," *Information Technology Journal*, vol. 3, no. 1, pp. 301–302, 1981.

[131] E. Zitzler and L. Thiele, "An evolutionary algorithm for multiobjective optimization: the strength pareto approach," *TIK-report*, vol. 43, 1998.