REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE Dr. MOULAY TAHAR – SAIDA

FACULTE DE TECHNOLOGIE

DEPARTEMENT D'INFORMATIQUE

N° D'ORDRE : .....

THESE

Présentée par

BOUSMAHA RABAB

Pour l'obtention du diplôme de

DOCTORAT «L. M. D» en INFORMATIQUE

Spécialité: Informatique

Option: Informatique

Les méta-heuristiques, les méthodes bio-inspirées et le datamining pour l'extraction des connaissances en vue des Big data

---

Défendu publiquement, en 15/07/2021

Devant le jury composé de:

AMINE Abdelmalek	Professeur	Université de Saida	Président
BELALEM Ghalem	Professeur	Université d'Oran 1	Examinateur
BENDAOUD Zakaria	M.C.A	Université de Saida	Examinateur
YAHLALI Mebarka	M.C.A	Université de Saida	Examinateur
HAMOU Reda Mohamed	Professeur	Université de Saida	Directeur de thèse

Année Universitaire 2020-2021

Laboratoire GeCoDe, Université de Saida

Dédicace

J'adresse en premier lieu ma reconnaissance notre DIEU tout puissant, de m'avoir permis d'en arriver là, car sans lui rien n'est possible.

Je dédie ce travail,

À mon très cher père

À celui qui m'a aidé découvrir le savoir aucune dédicace ne saurait exprimer l'amour, l'estime, et le respect que j'ai toujours eu pour vous. Merci d'avoir été toujours la pour moi, un grand soutien tout au long de mes études. Que dieu le tout puissant te préserve, t'accorde santé, bonheur et te protège de tout mal.

À ma très chère mère

À une personne qui m'a tout donné sans compter Tu n'as pas cessé de me soutenir et de m'encourager, Merci pour tous les efforts que tu as fait pour moi Puisse Dieu, le tout puissant, te préserver et t'accorder santé, longue vie et bonheur afin que je puisse te rendre un minimum de ce que je te dois.

À AMINA et HADJER mes soeurs adorées

Amina, Tu as toujours été un exemple à suivre pour moi, la grande soeur modèle, Je te remercie pour tout ce que tu as fait pour moi, Je vous dédie ce travail avec tous mes voeux de bonheur, de santé et de réussite.

À Hadjer, dont je suis fière d'être ma soeur, pour l'affection et la complicité qui nous unissent. Votre aide et votre générosité extrêmes ont été pour moi une source de courage, de confiance et de patience, je te remercie également. Je te souhaite un avenir plein de joie, de bonheur, de réussite.

Á ma chère nièce Ilhem et mon cher neveu Iyed, mes plus grandes sources de bonheur, j'espère que la vie vous réserve le meilleur. J'implore DIEU le tout-puissant de vous garder pour votre tante qui vous adore. Je vous aime de tout mon cœur.

À mon oncle Abbes

m'as beaucoup soutenu, et aidé, je te remercie infiniment Que Dieu te bénisse et te guide vers le meilleur inchaellah

À mon beau frère Abdelkader Je te dédie ce travail en témoignage de mon profond respect,

À ma famille, qui a toujours était la pour mois,

À mes amis.

Rabab

Remerciments

Voila enfin, après de longues années de travail, avec l'aide d'ALLAH, le tout puissant, qui mène toujours à bonne fin, j'ai réussi à mettre en forme la thèse que vous avez entre les mains.

Un énorme remerciement que dois présenter en premier lieu à mon directeur de thèse Pr. HAMOU Reda Mohamed qui ma a soutenue et m'a guidé au cours de la réalisation de cette thèse, merci pour leur temps qu'il m'a consacré ; pour ses convictions ; son énergie et ses conseils.

Mes remerciements vont ensuite aux membres du jury :
À Pr. AMINE Abdelmalek, Professeur à l'université de SAIDA Dr. MOULAY Taher qui a accepté de présider cette thèse.

À Pr. BELALEM Ghalem, Professor à l'Université d'Oran Ahmed Ben Bella, Dr. BENDAOUD Zakaria, Docteur à l'université de SAIDA Dr. MOULAY Taher et Dr. YAHLALI mebarka, Docteur à l'université de SAIDA Dr. MOULAY Taher qui m'ont fait l'honneur de participer à ce jury.

J'adresse des remerciements très respectueux à tous mes enseignants qui m'ont apporté leurs aides, leurs conseils, leurs précieux soutiens tout au long de mon cursus.

J'aimerais remercier du fond du coeur mes parents pour leur soutien moral et matériel. Ma famille qui ont toujours porté un intérêt à ce que je faisais

Merci également à mes amis qui m'ont apportée soutien et détente durant cette thèse...

Résumé

Le processus de l'extraction des connaissances comprend plusieurs étapes qui sont la sélection attributs, le prétraitement, la transformation, l'exploration des données et l'interprétation des résultats. Tous ces problèmes peuvent être formulés comme des problèmes d'optimisation combinatoire or, de nombreux problèmes d'optimisation combinatoire sont NP-difficile et ne pourront pas être résolus de manière exacte dans un temps raisonnable. D'où plusieurs travaux utilisent des méthodes d'optimisation pour résoudre ces problèmes. Aujourd'hui, les données à analyser sont non seulement volumineuses, mais ils sont composés de différents types de données, et comprennent même des données continues. Comme les méthodes traditionnelles d'analyse de données ne sont pas conçues pour des données complexes. Cette thèse a pour but de réunir les forces des métaheuristiques, des méthodes bio inspirées ainsi que le data mining pour effectuer l'extraction des connaissances robustes, et d'optimiser les méthodes existantes. Nous nous sommes particulièrement intéressés à la réalisation et la conception des algorithmes basés sur l'hybridation des algorithmes évolutionnaires et les algorithmes d'intelligence des essaims pour l'optimisation globale et pour résoudre différents problèmes d'extraction de connaissances à savoir la sélection d'attributs et l'optimisation des algorithmes de classification à savoir les réseaux de neurones a propagation avant, les réseaux de neurones récurrents et les machines à vecteurs de support (SVM) appliqué sur plusieurs domaines.
Mots clés Extraction des connaissances, Métaheuristiques, Data Mining, Algorithmes évolutionnaires, Algorithmes d'intelligence des essaims, Optimisation combinatoire, Sélection d'attributs, Classification.

Abstarct

The knowledge discovery process includes several steps which are feature selection, preprocessing, transformation, data mining and interpretation of results. All these problems can be formulated as combinatorial optimization problems, but many combinatorial optimization problems are NP-hard and will not be solved accurately in a reasonable time. Hence several works use optimization methods to solve these problems. Today, the data to be analyzed are not only voluminous, but they are composed of different types of data, and even include continuous data. Traditional data analysis methods are not designed for complex data. This thesis aims to combine the strengths of metaheuristics, bio-inspired methods and Data Mining to perform robust knowledge discovery, and to optimize existing methods. We are particularly interested in the realization and design of new algorithms based on the hybridization of evolutionary algorithms and swarm intelligence algorithms for global optimization and to solve various knowledge discovery problems such as feature selection and optimization of classification algorithms : feed forward neural networks (FFNN), recurrent neural networks (RNN) and support vector machines (SVM) applied on several domains.
Keywords knowledge discovery, Metaheuristics, Evolutionary algorithms, Data mining, Swarm intelligence, Combinatorial optimization, Feature selection, Classification.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

KDP	Knowledge Discovery Process
KDD	Knowledge Discovery in Databases
IR	Information Research
DE	Differential Evolution
BA	BAT
BAT-SDE	AUTO-ADAPTATIF HYBRIDE BAT
PSO	Particle Swarm Optimization
MVO	Multi-Verse Optimizer
MFO	Moth-Flame optimization
WOA	Whale Optimization Algorithm
PMVO	HYBRID PSO-MVO
PLMVO	HYBRID PSO-LMVO
MVGWO	HYBRID GWO-MVO
HC	Hill Climbing
GP	Genetic Programming
ES	Evolution Strategy
EP	Evolutionary Programming
ACO	Ant colony Optimization
TS	Tabu search
AIS	Artificial Immune System
FA	Firefly algorithm
LM	Levenberg-Marquardt
BBO	Biogeography-Based Optimiser
MFO	Moth Flame Optimizer
IMBO	Improved monarch butterfly optimization
GCMBO	IMBO using Greedy strategy and self-adaptive Crossover operator
GWO	Grey Wolf Optimizer
LPSONS	Hybrid Particle Swarm Optimization, Mantegna Lévy flight and neighborhood search
HACPSO	Hybrid Accelerated Cuckoo Particle Swarm Optimization
APSO	Adaptive Particle Swarm Optimization
GSA	Gravitational Search Algorithm
SSD	Social Ski Driver
RS	Random search
SEOA	Social Emotional Optimization Algorithm
FOA	Forest Optimization Algorithm
SCA	Sine Cosine Algorithm
ALO	Ant Lion Optimizer
CSO	Competitive Swarm Optimizer
BGSA	Binary Gravitational Search Algorithm
NFL	No-Free-Lunch
jDE	Self-adaptive Differential Evolution
HBA	Hybrid BAT Algorithm
ABC	Artificial Bee Colony Algorithm

CGWO	Chaotic GWO
MGWO	Modified GWO
BGWO	Binary GWO
PGWO	Power GWO
IGWO	Intelligent GWO
CS	Cuckoo Search
FFNN	Feed-Forward Neural Network
ANN	Artificial Neural Network
ReLU	Rectified Linear Units
MLP	Multilayer Perceptron
RBF	Radial Basis Function
DBN	Deep Belief Network
RNN	Recurrent Neural Network
DNN	Deep Neural Networks
CNN	Convolutional Neural Network
LSTM	Long short-term memory
BP	Backpropagation
MSE	Mean Squared Error
MCC	Matthews Correlation Coefficient
G-mean	Geometric Mean
FS	Feature Selection
ML	Machine Learning
SVM	Support Vector Machine
J48	Decision Tree Algorithm
MaxEnt	Maximum
MNB	Multinomial/ Multimodal Naive Bayes
SMOTE	Synthetic Minority Oversampling Technique
ENN	Extended Nearest Neighbor
GPU	Graphics Processing Unit
CPU	Central Processing Unit
Avg	Average value (la valeur moyenne)
STD	Standard Deviation (l'écart type)
Best	Best value (la meilleure valeur)
Worst	Worst Value (la pire valeur)
CPU time	Temps de traitement

Introduction

L'extraction des connaissances (KDD) et l'exploration de données (datamining) est un domaine interdisciplinaire axé sur les méthodologies permettant d'extraire des connaissances utiles à partir de données. La croissance rapide et continue des données en ligne due à l'Internet et l'utilisation généralisée des bases de données a créé un immense besoin de méthodologies de l'extraction de connaissance. Le défi de l'extraction des connaissances à partir des données s'appuie sur la recherche dans les statistiques, les bases de données, la reconnaissance de formes, l'apprentissage automatique, la visualisation des données, l'optimisation et le calcul haute performance, pour fournir des solutions avancées de business Intelligente.
L'extraction de connaissances à partir des Données (ECD). L'ECD, où Knowledge Discovery in databases en anglais, est le processus de découverte de connaissances utiles à partir d'une collection de données. Plus précisément, l'ECD est un "processus non trivial d'identification dans les bases de données de structures inconnues, valides et potentiellement exploitables" (FAYYAD, PIATETSKY-SHAPIRO et SMYTH, 1996). Cette technique d'extraction de connaissances largement utilisée est un processus qui comprend quatre phases : acquisition et stockage des données, prétraitement des données (la préparation, la sélection et le nettoyage des données), l'exploration de données (datamining), post-traitement (l'intégration des connaissances préalables sur les ensembles de données et l'interprétation de solutions à partir des résultats observés). Nos travaux se situent principalement au niveau de l'étape deux et l'étape trois : le prétraitement et l'exploration des données (fouille de données).
Lors de l'extraction de connaissance, dans laquelle des modèles inconnus doivent être découverts, l'analyse peut être très complexe en raison de la nature des données manipulées. C'est ce qui est au cœur de l'exploration de données. Une façon de résoudre les problèmes d'exploration de données consiste à les modéliser comme des problèmes d'optimisation combinatoire. L'exploration de données peut se décomposer en trois tâches majeures : la discrimination (classification supervisée), la catégorisation ou le clustering et la recherche de règles d'association. Ces tâches peuvent, par leur formalisation, être modélisées en problèmes d'optimisation combinatoire. À partir de la formalisation des tâches d'extraction de connaissances en problèmes d'optimisation combinatoire, il faut mettre en œuvre des méthodes de résolution (JOURDAN, 2003). Habituellement, les méthodes utilisées pour résoudre ces problèmes sont de trois types : les méthodes exactes, les méthodes heuristiques spécifiques à une tâche et les métaheuristiques.
Un problème d'optimisation combinatoire est généralement caractérisé par un ensemble fini de solutions admissibles (D) (espace de décision), et une fonction objective (f) associant une valeur à chaque solution admissible. Par conséquent, un problème d'optimisation consiste à minimiser ou maximiser la fonction objective (f) sous un ensemble de contraintes permettant de décrire l'ensemble de solutions réalisables.
Il existe de nombreux exemples de problèmes d'optimisation combinatoire (GAREY, 1979). En effet, ils peuvent se retrouver dans la gestion de la production, la conception de réseaux de télécommunications, la bio-informatique, la programmation etc. Nous nous intéressons dans notre thèse aux problèmes relatifs à l'extraction de connaissances et à leur modélisation en problèmes d'optimisation.

Cette thèse a pour but de réunir les forces des métaheuristiques et des méthodes bio-inspirées ainsi que le datamining pour effectuer l'extraction des connaissances robustes, et d'optimiser les méthodes existantes.
Nous nous sommes particulièrement intéressés à la réalisation et la conception des algorithmes basés sur l'hybridation des algorithmes évolutionnaires et les algorithmes d'intelligence des essaims pour l'optimisation globale et pour résoudre différents problèmes d'extraction de connaissances : la sélection d'attributs, et l'optimisation des algorithmes de classification à savoir les réseaux de neurones à propagation avant, les réseaux de neurones récurrents et les machines à vecteurs de support (SVM) appliqué sur plusieurs domaines afin d'augmenter l'exactitude de la classification. Ces deux problèmes peuvent être modélisés comme des problèmes d'optimisation NP-difficiles.
Dans notre première approche, une nouvelle hybridation entre l'évolution différentielle auto-adaptative et l'algorithme des chauves-souris a été proposée pour entrainer les réseaux de neurones propagation avant. L'algorithme proposé est comparé à l'évolution différentielle et l'algorithme des chauves-souris pour résoudre un ensemble de cinq fonctions de tests afin de trouver la solution globale. Dans la deuxième expérience, la performance de l'approche proposée a été comparée à huit algorithmes utilisés pour entrainer les réseaux de neurones dans la littérature. La comparaison a été étalonnée et évaluée à l'aide de sept ensembles de données biomédicales standard provenant des dépôts d'ensembles de données (UCI) et d'un grand ensemble de données de détection des fraudes par carte de crédit. Cet ensemble de données est très déséquilibré ; la méthode combinée SMOTE + ENN a été utilisée pour résoudre ce problème en tant que technique de prétraitement utilisée pour traiter des ensembles de données déséquilibrées. L'ensemble de données étant très bruyant ; l'élimination récursive des caractéristiques avec validation croisée est utilisée comme méthode de sélection des caractéristiques.
Dans les réseaux de neurones, optimiser le nombre de neurones cachés et les poids de connexion simultanément, il est considéré comme une tâche difficile. En effet, la modification des neurones cachés affecte considérablement la structure d'un réseau de neurones et augmente la difficulté du processus d'entraînement qui nécessite des considérations spéciales. L'optimisation de l'essaim de particules (PSO) est l'un des algorithmes métaheuristiques les plus importants en raison de sa vitesse de convergence et sa simplicité de mise en œuvre. L'optimisation de multi-verse (MVO) basée sur le vol de Lévy est un algorithme récent et rapide qui peut éviter la convergence prématurée et peut atteindre un meilleur équilibre entre l'exploration et l'exploitation. Dans notre deuxième approche, nous avons proposé une nouvelle méthode de formation basée sur l'optimisation des essaims de particules hybrides avec l'optimisation de multi-verse basée sur le vol de Lévy nommé (PLMVO) pour optimiser le nombre de neurones cachés et les poids de connexion simultanément dans les réseaux de neurones à propagation avant. L'algorithme hybride est utilisé pour mieux chercher dans l'espace des solutions qui prouve son efficacité à réduire les problèmes de piégeage en minima locaux. Pour évaluer l'algorithme proposé, nous avons utilisé trois séries expérimentales, dans la première, l'algorithme PLMVO proposé est comparé aux algorithmes MVO et PSO pour résoudre un ensemble de 15 fonctions de tests afin de trouver la solution globale. Dans la deuxième expérience, la performance de l'approche proposée a été comparée à cinq techniques utilisées pour entrainer les réseaux de neurones dans la littérature. La comparaison a été évaluée à l'aide de neuf ensembles de données biomédicales. Dans la troisième expérience, le PLMVO-MLP proposé est utilisé pour prédire les fichiers exécutables malveillants de Linux.
Dans la troisième approche, un nouvel algorithme hybride nommé MVGWO basé sur l'optimiseur loup gris en exploitation avec l'optimiseur multi-vers en exploration en posant un coefficient d'équilibre est proposé pour résoudre des problèmes d'optimisation à grande dimension. L'algorithme hybride utilise pour fusionner les avantages et pour surmonter

les problèmes de ces deux algorithmes afin d'atteindre un optimum global. Une version améliorée de GWO basée sur un poids d'inertie adaptatif est proposée pour améliorer la capacité de recherche globale de cet algorithme et pour maintenir la diversité des solutions. Pour évaluer l'algorithme proposé, nous avons utilisé deux expériences. Dans la première, vingt-deux fonctions de test de différents types et dimensions sont utilisées et le MVGWO est comparé au GWO et au MVO. Dans la deuxième expérience, MVGWO est utilisé pour la sélection d'attributs et l'optimisation des paramètres du SVM afin d'obtenir une exactitude de classification élevée. Les résultats de MVGWO-SVM sont comparés à la recherche de grille et à quatre algorithmes métaheuristiques : GWO, MVO, WOA et BAT en utilisant quinze ensembles de données étiquetés.
Dans la quatrième approche, un modèle basé sur les réseaux de neurones récurrents optimisé par l'algorithme MVGWO nommé (M-RNNMVGWO) a été proposé pour la prédiction des anomalies dans le réseau IoT en utilisant la technique de sur-échantillonnage des minorités synthétiques (SMOTE). Les approches proposées sont simulées avec des données DS2OS et les performances sont comparées à d'autres approches d'apprentissage automatique. Les paramètres d'évaluation tels que la précision, la sensibilité, la spécificité, et la F-mesure sont utilisés pour confirmer la supériorité de notre approche proposée.

Ce mémoire s'articule en cinq parties.

Le premier chapitre fournit d'abord quelques informations de base sur les problèmes d'optimisation combinatoires et leurs méthodes de résolution.
La deuxième partie se concentre sur les métaheuristiques classées en deux groupes : les méthodes à population de solutions et les méthodes à solution unique. Ensuite, la dernière partie du chapitre fournit quelques informations sur les méthodes d'hybridation des algorithmes métaheuristiques. Dans ce but on a présenté deux taxinomies, elles permettent de comprendre quels sont les différents moyens d'hybridation possibles et comment ceux-ci ont été exploités dans la littérature scientifique.

Dans le deuxième chapitre, nous décrivons le processus de l'extraction de connaissances, nous introduirons une présentation générale de chaque étape, et nous détaillerons ensuite plus précisément l'étape de datamining (l'exploration de données).

Le troisième chapitre, présente un aperçu général sur les réseaux de neurones, l'apprentissage profond et les réseaux de neurones profonds.

Le quatrième chapitre fournit d'abord une description de la tâche de classification et une présentation des méthodes de classification standard. Ensuite, il aborde l'utilisation des métaheuristiques pour optimiser ces méthodes de classification et nous proposons ensuite de montrer comment la sélection d'attributs peut être réalisée avec les métaheuristiques.

Dans le cinquième chapitre, nous présentons nos approches proposées basées sur l'hybridation des métaheuristiques pour la sélection d'attributs et pour optimiser quelques algorithmes de classification à savoir les réseaux de neurones et les machines à vecteurs de support (SVM) appliqué sur plusieurs domaines.

Nous terminerons ce mémoire par différentes perspectives de recherche qui nous semblent intéressantes pour continuer ce travail.

Chapitre 1

Métaheuristiques et bio-inspirations

1.1 Introduction

Les problèmes d'extraction de connaissances peuvent être formulés comme des problèmes d'optimisation combinatoire or, de nombreux problèmes d'optimisation combinatoire sont NP-difficile et ne pourront pas être résolus de manière exacte dans un temps raisonnable. Des méthodes dédiées à ce genre de problème. comme les métaheuristiques peuvent être utilisées.
L'extraction de connaissances en particulier la phase de datamining implique la construction et l'évaluation de nombreux modèles, certains de ces modèles semblent être très bons prédicateurs, tandis que certains modèles peuvent être mauvais. La qualité de ces modèles peut être mesurée en fonction du contexte et de l'objectif du processus de l'extraction de connaissances. Une approche d'optimisation considérerait chaque modèle comme une solution possible du problème de datamining qui consiste à expliquer les relations entre les données. La qualité d'une solution dépend de la qualité du modèle.
Ce chapitre, fournit d'abord quelques informations de base sur les problèmes d'optimisation combinatoires et leurs méthodes de résolution.
La deuxième partie se concentre sur les métaheuristiques classées en deux groupes : les méthodes à population de solutions et les méthodes à solution unique. Ensuite, la dernière partie du chapitre fournit quelques informations sur les méthodes d'hybridation des algorithmes métaheuristiques. Dans ce but on a présenté deux taxinomies, elles permettent de comprendre quels sont les différents moyens d'hybridation possibles et comment ceux-ci ont été exploités dans la littérature scientifique.

1.2 Les problèmes combinatoires

1.2.1 Optimisation

Un problème d'optimisation peut être définit par :
D, un ensemble de solutions qui représente l'espace de recherche (espace de décision).
F, une fonction objective qui associe à chaque solution une valeur représentative de sa qualité (la plupart du temps une valeur réelle) (DHAENENS, 2016).
Selon le problème, les solutions $S \in D$ peuvent être de natures différentes et peuvent être définies par des contraintes déterminant des solutions réalisables. Dans le cas d'un ensemble fini de solutions discrètes D, on parle alors de problèmes d'optimisation combinatoire.
Par conséquent, un problème d'optimisation consiste à minimiser ou maximiser un critère nommé fonction objectif donné sous un ensemble de contraintes permettant de décrire l'ensemble de solutions réalisables (DHAENENS, 2016).
Un problème de minimisation peut être définis par :

$$Min\, f(S)$$

$$S \in D$$

Un problème de maximisation peut être définis par :

$$Max\ f(S)$$

$$S \in D$$

La grande variété des problèmes liés à l'optimisation combinatoire est due aux nombreuses applications. En effet, les problèmes d'optimisation combinatoire peuvent se retrouver dans, la conception de réseaux de télécommunications, la bio-informatique et l'extraction de connaissances (DHAENENS, 2016), etc..

1.2.2 Résolution

La résolution d'un problème d'optimisation combinatoire nécessite trois points principaux :

— Définition de l'ensemble des solutions réalisables ;
— Détermination de la fonction objective à optimiser ;
— Choix de la méthode d'optimisation.

Les deux premiers points concernent la modélisation du problème, tandis que le troisième concerne la résolution du problème. Pour définir l'ensemble des solutions réalisables, il est nécessaire d'exprimer l'ensemble des contraintes du problème qui nécessite la connaissance du problème étudié et de son domaine d'application.
De même, la détermination de la fonction objective nécessite également la connaissance du problème, car il est nécessaire de pouvoir qualifier ce que serait une bonne solution. Enfin, le choix de la méthode d'optimisation dépendra souvent de la complexité du problème. En effet, selon sa complexité, il peut ou non être possible de résoudre le problème de manière optimale (DHAENENS, 2016).

1.3 Méthodes d'optimisation

Le but des méthodes d'optimisation est de trouver une solution optimale ou quasi optimale avec un faible effort de calcul. L'effort d'une méthode d'optimisation peut être mesuré comme le temps (temps de calcul) et l'espace (mémoire d'ordinateur) consommé par la méthode. Pour de nombreuses méthodes d'optimisation, et en particulier pour les méthodes heuristiques, il existe un compromis entre la qualité de la solution et l'effort, comme avec l'augmentation de l'effort, la qualité de la solution augmente (DHAENENS, 2016).
Nous pouvons distinguer deux types différents de méthodes d'optimisation : les méthodes d'optimisation exactes qui garantissent de trouver une solution optimale et les méthodes d'optimisation heuristiques où nous n'avons aucune garantie qu'une solution optimale est trouvée. Le choix de la méthode à utiliser pour résoudre un problème d'optimisation combinatoire peut dépendre de sa complexité. Dans le cas de problèmes de classe P, un algorithme d'optimisation polynomiale peut être utilisé pour résoudre le problème de manière optimale. En cas de problèmes de classe NP pas d'algorithme d'optimisation polynomiale est trouvé, et deux approches sont possibles (voir figure 1.1) (DHAENENS, 2016).
Habituellement, une méthode d'optimisation exacte est la méthode de choix si elle peut résoudre un problème d'optimisation avec un effort qui croît polynomialement avec la taille du problème. La situation est différente si les problèmes sont NP-difficiles car les méthodes d'optimisation exactes nécessitent alors un effort exponentiel (DHAENENS, 2016). Ensuite, même les instances de problème de taille moyenne deviennent souvent insolubles et ne

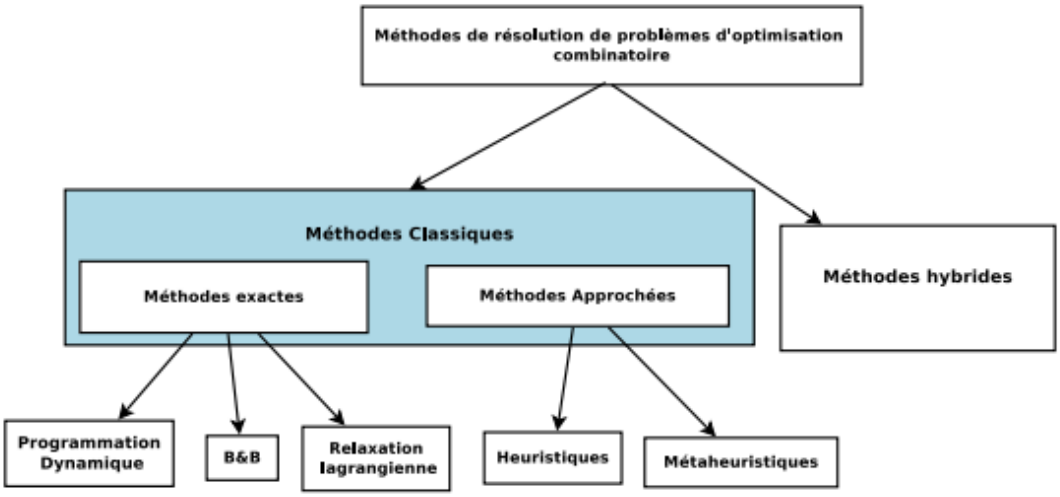

FIGURE 1.1 – Vue globale des approches de résolution des problèmes d'optimisation combinatoire(TALBI, 2002)

peuvent plus être résolues à l'aide de méthodes exactes. Pour surmonter ces problèmes, il est recommandé d'utiliser des méthodes heuristiques pour trouver de bonnes solutions dans un délai raisonnable, même si l'optimalité n'est pas garantie (ROTHLAUF, 2011). Parmi ces méthodes, il existe soit des méthodes heuristiques spécifiques développées pour un problème dédié, soit des métaheuristiques qui proposent des schémas de résolution génériques pouvant potentiellement être adaptés à tout type de problème d'optimisation. En effet, une métaheuristique peut être définie comme un algorithme conçu pour résoudre approximativement un large éventail de problèmes d'optimisation difficile sans être profondément adapté à chaque problème (DHAENENS, 2016).
Le but d'une telle métaheuristique est d'explorer efficacement l'espace de recherche, sans énumérer toutes les solutions. Ensuite, une métaheuristique réussira sur un problème d'optimisation donné si elle peut fournir un équilibre entre l'exploration (diversification) et l'exploitation (intensification) (HAMOU, AMINE et BOUDIA, 2013). Une exploration est nécessaire pour identifier les régions de l'espace de recherche avec des solutions de haute qualité. L'exploitation est importante pour intensifier la recherche dans ces régions prometteuses. Les principales différences entre les métaheuristiques existantes sont dans la manière particulière dont elles tentent d'atteindre cet équilibre.
La modélisation est donc une phase importante dans l'analyse d'un problème, car les problèmes de classe P ou NP ne peuvent pas être abordés exactement de la même manière. En outre, la définition de la fonction objective est cruciale mais peut-être difficile à réaliser, en particulier pour les problèmes du monde réel. (BARTZ-BEIELSTEIN, 2006).

1.4 Les métaheuristiques

Les métaheuristiques d'optimisation sont des algorithmes stochastiques généraux qui permettent d'approximer les solutions de problèmes d'optimisation difficiles et réels, qui sont souvent des problèmes NP-difficiles pour lesquels on ne connaît pas des méthodes classiques plus efficaces pour les résoudre dans un temps raisonnable.
Les métaheuristiques peuvent être classifiées en deux classes : les métaheuristiques à solution unique et les métaheuristiques à population de solutions (GUENDOUZ, AMINE et

HAMOU, 2017) (voir figure 1.2). Dans ce chapitre, nous décrirons brièvement les méthodes associées à chacune de ces classes.

1.4.1 Concepts des métaheuristiques

Il existe deux concepts de base communs à tout type de métaheuristique : la représentation des solutions traitées par des algorithmes et la définition de la fonction objective qui guidera la recherche.

Représentation

La conception de tout métaheuristique itérative nécessite un encodage (représentation) d'une solution. C'est une question de conception fondamentale dans le développement de la métaheuristique. L'encodage joue un rôle majeur dans l'efficience et l'efficacité de toute métaheuristique. En effet, pour un problème donné, plusieurs encodages peuvent être utilisés(DHAENENS, 2016).
Chaque codage peut être manipulé différemment par la métaheuristique à travers des mécanismes d'optimisation (opérateurs, fonction d'évaluation, etc.) qui peuvent être plus ou moins efficaces pour le problème étudié. De plus, l'efficacité d'une représentation est également liée aux opérateurs de recherche appliqués sur cette représentation (voisinage, recombinaison, etc..). En effet, lors de la définition d'une représentation, il faut garder à l'esprit comment la solution sera évaluée et comment les opérateurs de recherche fonctionneront.
Concernant les problèmes d'exploration de données, de nombreux encodages ont été proposés(DHAENENS, 2016).
Parmi les plus célèbres, on peut citer :

- **Codage binaire :** la solution est représentée par un vecteur de n valeurs binaires, représentant les variables de décision du problème. L'espace de recherche est de taille 2^n.
- **Vecteur de valeurs discrètes :** les variables ne sont pas limitées aux valeurs binaires, mais elles peuvent prendre des valeurs discrètes.
- **Permutation :** la solution est décrite par une permutation de taille n. Chaque permutation décode une solution unique. L'espace de solution est représenté par l'ensemble de toutes permutations de taille $(n-1)!$.
- **Vecteur de valeurs réelles :** les variables peuvent avoir des valeurs réelles.

Enfin, il est important de rappeler que l'efficacité de tout codage est fortement liée aux mécanismes qui seront appliqués à ce codage spécifique.

Fonction objectif

La détermination du critère d'optimisation - qui mesure la qualité des solutions - est cruciale car la performance du processus d'optimisation en dépend. En effet, développer une méthode efficace qui n'utilise pas le bon critère nous conduira à obtenir la bonne réponse à la mauvaise question. L'une des phases les plus difficiles pour transformer une tâche d'extraction de connaissances en un problème d'optimisation est de définir ce critère d'optimisation qui peut-être soit spécifique à la tâche de datamining, soit dépendant de l'application. Une fois le critère d'optimisation défini, la fonction objective doit être formulée(DHAENENS, 2016).

La fonction d'objectif f formule le but à atteindre. Elle associe à chaque solution de l'espace de recherche une valeur réelle qui décrit la qualité de la solution, $f : S \rightarrow R$. Elle

représente alors une valeur absolue et permet un ordonnancement complet de toutes les solutions de l'espace de recherche. À partir de l'espace de représentation des solutions R, certaines fonctions de décodage d peuvent être appliquées, $d : R \rightarrow S$, pour générer une solution qui peut être évaluée par la fonction f. La fonction objective est un élément important dans la conception d'une métaheuristique. Elle va guider la recherche vers de "bonnes" solutions de l'espace de recherche. Si la fonction objective est mal définie, elle peut conduire à des solutions inacceptables, quelle que soit la métaheuristique utilisée(DHAENENS, 2016). Par conséquent, l'utilisation de méthodes d'optimisation, en particulier la métaheuristique, requiert une attention particulière sur la manière de définir les critères d'optimisation pour traiter les problèmes de l'extraction de connaissances(DHAENENS, 2016).

Analyse de performance

Les métaheuristiques sont des méthodes stochastiques et l'analyse des performances de ces méthodes est une tâche nécessaire doit être effectuée de manière équitable. Une approche théorique n'est généralement pas suffisante pour évaluer une métaheuristique (BARTZ-BEIELSTEIN, 2006). Une attention particulière doit être portée à la comparaison de plusieurs métaheuristiques.
Cette section aborde quelques lignes directrices pour évaluer expérimentalement une métaheuristique et / ou comparer les métaheuristiques de manière rigoureuse.
Pour évaluer les performances d'une métaheuristique de manière rigoureuse, les trois étapes suivantes doivent être prises en compte(DHAENENS, 2016).

Conception expérimentale : dans la première étape, les objectifs des expériences (la qualité des solutions, le temps de calcul, la robustesse), les instances sélectionnées (benchmarks réels, benchmarks aléatoires) et les facteurs doivent être définis(DHAENENS, 2016).

Mesures de performance : dans la deuxième étape, les mesures de performance doivent être identifiés. Après avoir exécuté les différentes expériences, une analyse statistique est appliquée aux résultats obtenus. Cette analyse statistique doit être utilisée pour effectuer l'évaluation des performances des métaheuristiques. Par conséquent, des tests statistiques sont effectués pour estimer si la confiance des résultats est valide. L'analyse des performances doit être effectuée avec des algorithmes d'optimisation de l'état de l'art qui sont dédiés au problème.
Pour évaluer le temps de calcul, la mesure peut être, par exemple, le temps nécessaire pour atteindre la solution optimale. La difficulté avec cette mesure est que le temps de calcul dépend de l'architecture de l'ordinateur et les résultats de la littérature sont difficiles à comparer.

Rapports : les résultats sont présentés de manière globale, et une analyse est effectuée en fonction des objectifs définis. Un autre point important est de garantir la reproductibilité des expériences de calcul.
Une difficulté supplémentaire pour évaluer les performances de la métaheuristique face à des problèmes d'exploration de données (datamining) est de mesurer l'intérêt de la solution, c'est-à-dire les connaissances extraites, lorsqu'elles sont appliquées à de nouvelles données(DHAENENS, 2016).
Il peut être intéressant de différencier deux points de vue pour l'analyse des performances :

— Point de vue datamining : le but de l'analyse de performance est d'évaluer l'intérêt des connaissances extraites et, en particulier, sa capacité à traiter des données inconnues)(DHAENENS, 2016).

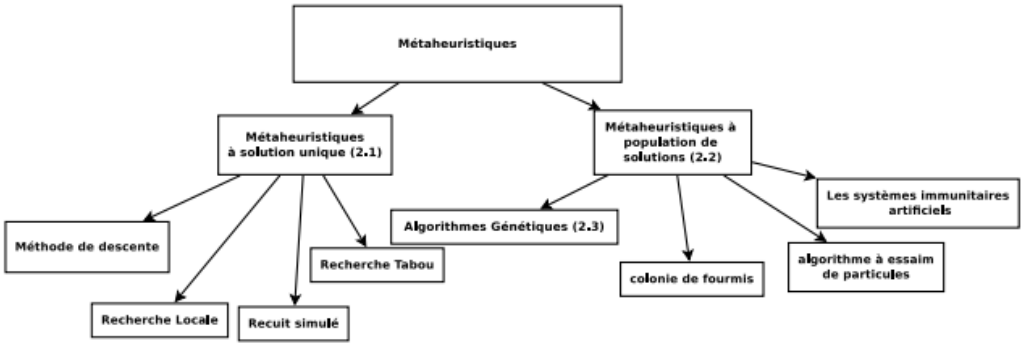

FIGURE 1.2 – Classification des métaheuristiques (TALBI, 2002)

— Point de vue optimisation : le but de l'analyse de performance est d'évaluer la capacité de la méthode à explorer l'espace de recherche et à trouver une solution de bonne qualité (la plupart du temps par rapport à d'autres méthodes)(DHAENENS, 2016).

1.4.2 Classification des métaheuristiques

Les métaheuristiques à solution unique / recherche locale

Tout en résolvant les problèmes d'optimisation, les métaheuristiques basées sur une solution unique (S-métaheuristiques) améliorent une solution unique.
La S-métaheuristique applique la procédure de génération et de remplacement à une solution unique. Au stade de la génération, un ensemble de solutions candidates est généré à partir de la solution actuelle. Dans la phase de remplacement, une solution appropriée, parmi l'ensemble généré, est choisie pour remplacer la solution actuelle. Ce processus se poursuit jusqu'à l'obtention d'un résultat satisfaisant (solution de bonne qualité).
Plusieurs métaheuristiques de recherche locale et leurs extensions ont été proposées. Le principal concept commun à ces méthodes est l'application d'une transformation locale sur la solution actuelle, appelée opérateur de voisinage.
Ainsi, ce concept est défini dans cette section, puis plusieurs méthodes de recherche locales bien connues sont présentées.

Le voisinage d'une solution

La définition du voisinage est une étape requise pour la conception de toute S-métaheuristique. La structure du voisinage joue un rôle crucial dans la performance d'une S-métaheuristique. Si la structure du voisinage n'est pas adaptée au problème, toute métaheuristique S ne résoudra pas ce problème (DHAENENS, 2016).
Le voisinage de s, désigné par $N(s)$, est l'ensemble des solutions qui peuvent être obtenues à partir de s en utilisant une transformation locale (ou élémentaire) appelée l'opérateur de voisinage N (ROTHLAUF, 2011). Une transformation sera considérée comme locale (ou élémentaire) si elles ne perturbent pas globalement la structure de la solution qu'elles changent (FOUILHOUX, 2015).

$$N : s \rightarrow N(s)$$

Pour illustrer cette notion de voisinage, considérons un problème de voyageur de commerce avec sept villes à visiter une seule fois (A, B, C, D, E, F et G). Les solutions peuvent être encodées avec la liste ordonnée des villes visitées. Plusieurs opérateurs de voisinage peuvent être définis :

— déplacer : déplacer la position d'une ville ;

— échange : échange la position de deux villes ;
— inverse : inverse une sous-séquence de villes.

Soit une solution initiale : A D C B E G F :

— en déplaçant F, on peut obtenir s1 : A D C B E F G ;
— en échangeant A et E, on peut obtenir s2 : E D C B A G F ;
— en inversant la sous-séquence D C B, on peut obtenir s3 : A B C D E G F

En plus de cette définition du voisinage, il faut définir le concept d'optimum local.

Local optimum

Définition

Étant donné un opérateur de voisinage N, une solution $s \in D$ est un optimum local si aucun de ses voisins n'a une meilleure qualité. Dans un contexte de minimisation, cela peut s'exprimer par : s est un optimum local, si $\forall s' \in N(s)$, $f(s) \leq f(s')$.

Dans le reste de cette section plusieurs méthodes de recherche locale sont présentées (DHAENENS, 2016).

Les méthodes de descente (Hill Climbing)

La méthode de descente (appelée "hill-climbing") Il commence avec une solution initiale, explore son voisinage et elle sélectionne ensuite une solution qui améliore strictement la solution actuelle. Cette sélection peut se faire de différentes manières. La solution retenue peut-être la première solution réalisable qui améliore la fonction objective ou la meilleure solution réalisable de tout le voisinage (DHAENENS, 2016).

En fonction de ce choix, la méthode est respectivement appelée méthode de descente simple, descentes itérées ou descente stochastique (DHAENENS, 2016).

Concernant la manière dont le voisin est sélectionné. Plusieurs stratégies de sélections peuvent être envisagées :

— **Descente simple (la meilleure amélioration) :** dans cette stratégie, le meilleur voisin est sélectionné parmi l'ensemble du voisinage. Cela nécessite de générer et d'évaluer de manière exhaustive tous les voisins de la solution actuelle. L'algorithme s'arrête donc quand il n'est plus possible d'améliorer la solution. Cette descente simple n'est intéressante que dans le cas où le voisinage est suffisamment petit.
— **Descentes itérées (La première amélioration) :** dans cette stratégie, le premier voisin qui améliore la solution est choisi parmi l'ensemble du voisinage. Cela peut éviter l'exploration de l'ensemble voisinage. Cette descente aléatoire évite de visiter systématiquement un voisinage qui serait trop grand.
— **Descente stochastique (l'amélioration aléatoire) :** plusieurs voisins sont générés et une sélection aléatoire est effectuée parmi ceux qui améliorent la solution actuelle.

Algorithm 1 Méthode de descente générique

Procédure : φ fonction de cout
Variable locale : S solution courante
Choix d'une solution initiale S_0 ;
Solution courante $S \leftarrow S_0$;
(a.) Génération des condidats par voisinage ;
Choix du meilleur candidat C ;
if $\varphi(C) < \varphi(S)$ **then**
  $S \leftarrow C$;
  Aller en (a.) ;
end if
return S

La méthode de descente est l'une des méthodes les plus simples trouvées dans la littérature ; cependant, il présente une limitation importante. Il peut se retrouver facilement piégé

dans un minimum local et encourage donc l'exploitation et non l'exploration. Afin de surmonter ce problème certains mécanismes ont été proposés pour échapper à l'optimum local. Ils conduisent à la proposition d'autres méthodes de recherche locale, telles que la recherche taboue (TS), le recuit simulé et autres (DHAENENS, 2016).

Le recuit simulé (Simulated Annealing)

Le recuit simulé (FRÉDÉERIC et al., 2017) est la première métaheuristique à proposer un processus permettant d'échapper aux optima locaux. Il s'agit d'une méthode inspirée du processus de recuit pratiqué en métallurgie, qui consiste à fondre le métal à haute température pour être ensuite refroidi jusqu'à l'obtention d'un état stable, appelé équilibre thermodynamique. Cet état stable peut être "bon" ou "mauvais", c'est-à-dire avec une énergie minimale ou non. En effet, lorsque le métal est rapidement refroidi, des déformations peuvent apparaître, alors qu'un métal "parfait" est obtenu si le processus de refroidissement est adéquat. La température correspond au paramètre de contrôle de la stabilité du métal (FRÉDÉERIC et al., 2017).

Par analogie, sur la base d'une solution aléatoire s , une solution $\acute{s}$ est générée dans le voisinage de s. Si cette solution voisine améliore la solution actuelle, s est mis à jour. Sinon, $\acute{s}$ peut également être accepté selon la probabilité $\exp(\frac{\Delta f}{T})$. Cette probabilité permet d'accepter une solution dégradante dans le cas où la solution $\acute{s}$ présente une faible dégradation $\exp(\Delta f)$ par rapport à s ou lorsque la température T est suffisamment élevée. L'exploration est donc préférable. Cependant, cette probabilité devient plus faible lorsque l'on sait que la température suit une fonction décroissante et est actualisée à chaque itération, ce qui rend l'exploitation plus appropriée.

La recherche Taboue (Tabu Search)

Algorithm 2 Le recuit simulé

Initialiser la température initiale T
Générer une solution initiale aléatoire s
while le critère d'arrêt n'est pas satisfait **do**
  Générer une voisine $s\prime$ de s
  Évaluer $s\prime$
  if $f(s\prime) \leq f(s)$ or $\exp(\frac{\Delta f}{T}) > rand(0,1)$ **then**
    Mettre à jour s avec $s\prime$
  end if
  Mettre à jour la température T
end while
return s

TS a été initialement proposé par Glover (EL GHAZALI, 2009). Le principe est d'accepter, dans certains cas, des solutions qui semblent moins intéressantes pour éviter de tomber dans un optimum local. Cette méthode fonctionne d'abord comme l'algorithme HC. Cependant, lorsqu'un optimum local est atteint, la méthode accepte un voisin non améliorant (en général, la meilleure solution du voisinage) comme la prochaine solution actuelle. Remarquons que dans ce cas, la solution actuelle à la fin de la recherche peut ne pas être la meilleure solution rencontrée lors de la recherche et qu'il faut mémoriser $s*$ comme une meilleure solution. Après avoir sélectionné un voisin qui semble moins intéressant comme une solution actuelle, le prochain mouvement peut créer un cycle entre ce voisin et l'optimum local. Pour éviter de tels cycles, TS interdit de revenir à des solutions récemment visitées à l'aide d'une mémoire. Cette mémoire, appelée liste Taboue, peut stocker soit les dernières solutions visitées, soit les derniers mouvements visités. Dans ce dernier cas, un mouvement tabou peut encore être appliqué s'il permet d'atteindre une meilleure solution que $s*$. Ce mécanisme est géré par le critère d'aspiration. La conception de la méthode doit spécifier la gestion de

la liste Taboue.
Cette méthode ne s'arrête pas d'elle-même et il faut déterminer un critère d'arrêt en fonction du temps de recherche que l'on s'octroie. Ce critère peut être, par exemple, la non-amélioration de la meilleure solution pendant un certain nombre d'itérations ou l'exécution d'un certain nombre d'itérations (EL GHAZALI, 2009).

Algorithm 3 Méthode générique du Tabou

Procédure : φ fonction de cout
Variable locale : S solution courante, Liste Taboue L, meilleure solution M,itération courante K, Nombre d'iterations N
Paramétrages : Taille de la liste taboue, critère d'aspiration, choix d'une solution initiale S_0.
Choix d'une solution initiale S_0 ;
Solution courante $S \leftarrow S_0$;
Meilleure solution $M \leftarrow S$;
$K \leftarrow 0$;
while $K < N$ **do**
  $K \leftarrow K + 1$;
  Mise à jour de L
  Génération des condidats E par opération de voisinage ;
  $C \leftarrow best(E)$
  Choix du meilleur candidat C ;
  if $\varphi(S) < \varphi(M)$ OU C n'est pas tabou OU C vérife l'aspiration **then**
    $S \leftarrow C$
  else
    $E \leftarrow E \setminus C$
  end if
end while
return S

Les métaheuristiques à population de solution

Les métaheuristiques basées sur la population (P-métaheuristique) partagent de nombreux concepts communs. Elles peuvent être considérées comme une amélioration itérative d'une population de solutions. Tout d'abord, la population est initialisée. Ensuite, une nouvelle population de solutions est générée. Enfin, cette nouvelle population est intégrée à la population actuelle à l'aide de procédures de sélection. Le processus de recherche est arrêté lorsqu'une condition donnée est satisfaite (critère d'arrêt) (FREITAS, 2011).
Les métaheuristiques basées sur la population offrent une bonne occasion d'explorer l'espace de recherche. La plupart de ces approches sont basées sur des analogies avec des concepts naturels. Les algorithmes de calcul évolutif sont donc inspirées de la théorie de l'évolution de Darwin et de la capacité de la nature à faire évoluer les êtres vivants en les adaptant à leur environnement. C'est l'objet de la première partie de cette section. Dans la deuxième partie, des algorithmes inspirés de l'intelligence en essaim sont présentés (FREITAS, 2011).
Les algorithmes évolutionnaires
Les algorithmes évolutionnistes ce sont des algorithmes de recherche stochastiques qui sont basés sur des abstractions des processus de l'évolution darwinienne. Les idées de base de ce paradigme sont les suivantes. Un algorithme évolutif maintient une population d'"individus", chacun d'entre eux étant une solution candidate à un problème donné.

Chaque individu est évalué par une fonction objective, qui mesure la qualité de sa solution candidate correspondante. Les individus évoluent vers dés meilleurs individus par le biais d'une procédure de sélection basée sur la sélection naturelle, et des opérateurs basés sur la génétique, par exemple les opérateurs de croisement (recombinaison) et de mutation (COELLO, 2005). L'algorithme suivant décrit le schéma général des algorithmes évolutionnaires(DHAENENS, 2016).

Algorithm 4 Les approches générales du calcul évolutif

Initialiser la *population* initiale P_i
while critère d'arrêt non satisfait **do**
  Sélectionner des *parents* dans P_i
  Appliquer les opérateurs de recherche (recombinaison, mutation)
  Évaluer les nouvelles individus
  construire la prochaine *population* P_{i+1}
end while

Les algorithmes génétiques (Genetic Algorithm)
Les algorithmes génétiques ont été introduits par John H. Holland au début des années 1960 (HOLLAND, 1962a ; HOLLAND, 1962b). La principale motivation pour la création de l'algorithme génétique était la résolution des problèmes d'apprentissage automatique. La conception d'un tel algorithme nécessite de définir la représentation des solutions (appelée chromosomes), la stratégie de sélection, les opérateurs de recherche (cross-over et mutation). L'objectif de la stratégie de sélection est de choisir les individus (appelés parents) qui seront utilisés pour la construction de la prochaine génération. Ceci est principalement basé sur les valeurs de la fonction objective des solutions qui représentent leur capacité à répondre au problème à résoudre. L'une des spécificités des GA est l'opérateur de croisement (cross-over) le processus ou de nouveaux individus sont formés à partir des parents. L'objectif est de partager de bonnes caractéristiques pour obtenir des descendants de meilleure qualité. Étant donné que dans les premièrs GA, les chromosomes étaient principalement codés par des chaînes binaires de longueur fixe, de nombreux opérateurs de croisement classiques adaptés à cette représentation ont été proposés, tels que le croisement à 1-point (ou plus généralement le croisement à n-points), le croisement uniforme (HOLLAND, 1962a), etc.
De nos jours, le développement des GA ne se limite pas aux codages de chaînes de bits, mais d'autres représentations ont été proposées pour traiter différents types de problèmes d'optimisation. Ce sera le cas pour les problèmes de l'extraction de connaissances (RECHENBERG, 1965).
Programmation génétique (Genetic Programming)
GP regroupe des approches plus récentes. Elles ont été proposées par Koza au début des années 1990 (KOZA et KOZA, 1992). Les GP sont très similaires aux GA, mais la principale différence est que les individus sont eux-mêmes des programmes. Une représentation arborescente est utilisée. Les feuilles de l'arbre (appelées terminaux dans les GP) représentent les variables et les constantes et les nœuds internes représentent les opérations arithmétiques (appelées fonctions). Comme dans Les GA, la sélection des parents est principalement proportionnelle à l'aptitude (fitness) et un remplacement générationnel est adopté. Des opérateurs spécifiques gérant les arbres sont nécessaires. Les opérateurs de croisement peuvent être basés sur des échanges d'arbres (ou de sous-arbres) et les opérateurs de mutation peuvent être basés sur des changements aléatoires dans l'arbre l'une des difficultés de la GP est la représentation à longueur variable, car les programmes encodés peuvent être de plusieurs tailles. Au niveau le plus abstrait, la PG est une méthode systématique indépendante du domaine qui permet de résoudre des problèmes à partir d'une déclaration de haut

niveau de ce qui doit être fait. Les GP sont largement utilisés dans les tâches d'apprentissage automatique et de datamining (KOZA et KOZA, 1992).

La stratégie d'évolution (Evolution Strategy)

La stratégie d'évolution est un paradigme particulièrement bien adapté pour les taches d'optimisation continue où les représentations sont basées sur des vecteurs de valeur réelle (RECHENBERG, 1973 ; FOGEL, OWENS et WALSH, 1966). Similaire aux autres algorithmes évolutionnaires, les opérateurs génétiques en SE sont appliqués dans une boucle jusqu'à ce qu'un critère d'arrêt soit atteint. Elles utilisent généralement un remplacement élitiste et une mutation spécifique normalement distribuée (gaussienne). Le croisement (cross-over) est rarement utilisé. Dans un ES, il y a une distinction entre la population des parents de taille μ et la population des descendants de taille $\lambda \geq \mu$. L'opérateur de sélection est déterministe et elle est basée sur le classement de l'aptitude. Une caractéristique importante de la SE est l'utilisation des mécanismes d'auto-adaptation pour optimiser non seulement les solutions du problème étudié, mais aussi certains paramètres pour muter ces solutions. Leur principal avantage est leur efficacité en termes de complexité temporelle (KENNEDY et EBERHART, 1995a).

Programmation évolutionnaire (Evolutionary Programming)

La programmation Évolutionnaire (EP), initialement proposée par Fogel et al. (BEYER, 2001), utilise des opérateurs de mutation exclusivement pour générer des descendants. Aucune recombinaison n'est appliquée et le mécanisme de sélection des parents est déterministe. Les opérateurs de mutation consistent à ajouter aux parents un nombre aléatoire des certaines distributions. EP est très similaire à ES et, par conséquent, n'est pas largement utilisé.

Évolution différentielle (Differential Evolution)

L'évolution différentielle de Stom & Prince (1995) (STORN et PRICE, 1997a) est un algorithme d'optimisation qui a connu énormément de succès depuis son apparition et qui fut initialement créé pour résoudre des problèmes continus. Chaque individu est codé par un vecteur de valeurs réelles. L'idée principale derrière l'évolution différentielle est l'utilisation d'un opérateur de recombinaison ternaire pour la création des nouvelles générations. Les spécificités de ces approches sont les opérateurs. L'opérateur de croisement ne combine pas une partie du chromosome des parents comme dans les EA classiques, mais fait une combinaison linéaire de trois solutions choisies au hasard. De même, l'opérateur de mutation sélectionne aléatoirement trois solutions et ajoute le vecteur de différence pondérée entre deux solutions à la troisième solution. DE est une méthode simple qui a l'avantage de ne nécessiter que quelques paramètres de contrôle et ensuite elle est facile à régler (STORN et PRICE, 1997a). Nous détaillerons plus amplement cet algorithme dans le chapitre 06.

L'intelligence en essaim

Les algorithmes inspirés du comportement collectif d'espèces telles que les fourmis, les abeilles, les guêpes, les termites, les poissons et les oiseaux sont appelés algorithmes d'intelligence des essaims (BONABEAU, DORIGO et THERAULAZ, 1999 ; PINTO, RUNKLER et SOUSA, 2005).

L'intelligence des essaims est issue du comportement social des espèces en compétition pour l'accès à la nourriture (DHAENENS, 2016). Les principales caractéristiques des algorithmes basés sur l'intelligence en essaim sont les suivantes : les particules sont des agents simples et non sophistiqués, elles coopèrent par un moyen de communication indirect et effectuent des mouvements dans l'espace de décision.

Parmi les algorithmes d'optimisation inspirés de l'intelligence en essaim, on trouve l'optimisation des colonies de fourmis et l'optimisation par essaims de particulaires (RUNKLER, 2008).

Algorithmes à essaim de particules (Particle Swarm Optimizer)

L'optimisation des essaims de particules (PSO) est une technique d'optimisation globale

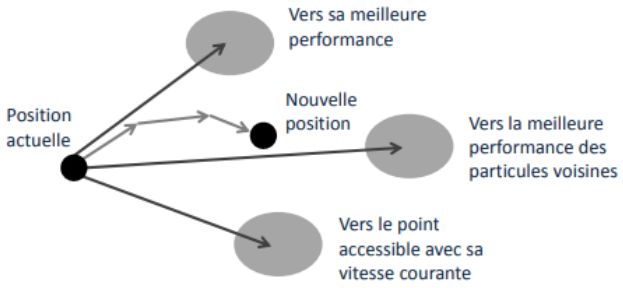

FIGURE 1.3 – Déplacement d'une particule

développée par Kennedy et Eberhart en 1995 (FOGEL, OWENS et WALSH, 1966). Ces algorithmes sont inspirés des essaims d'insectes (ou des bancs de poissons ou des nuées d'oiseaux) et de leurs mouvements coordonnés. Les individus de l'algorithme sont appelés particules et la population appelée essaim. Chaque particule est considérée comme une solution du problème, où elle possède une position (le vecteur de solution) et une vitesse. De plus, chaque particule possède une mémoire lui permettant de se souvenir de sa meilleure performance $(Pbest)$ et de la meilleure performance atteinte par les particules « voisines » $(Gbest)$. Les nouvelles vitesses et direction de la particule seront définies en fonction de trois composantes : une composante d'inertie ; la propension à suivre son propre chemin, une composante cognitive ; sa tendance à revenir vers sa meilleure position atteinte et une composante sociale ; sa tendance à aller vers son meilleur voisin (KENNEDY et EBERHART, 1995a) (voir figure 1.3). Un schéma général de l'algorithme est donné dans l'algorithme 5 (KENNEDY et EBERHART, 1995a).

Algorithm 5 Algorithme d'optimisation de l'essaim de particules

Initialiser les particules initials
while critère d'arrêt non satisfait **do**
  for tous les particules i **do**
    Mettre à jour la vitesse
    Déplacer la nouvelle position x_i
  end for
  if $f(x_i) < f(pbest_i)$ **then**
    $pbest_i \leftarrow x_i)$
    if $f(x_i) < f(gbest)$ **then**
      $gbest \leftarrow x_i)$
    end if
  end if
end while

Les colonies de fourmis (Ants System)

Les algorithmes des colonies de fourmis (ACO) ont été proposés par M. Dorigo dans les années 1990 (EL DOR, 2012). Dans les ACO, une population de fourmis artificielles coopère entre elles pour trouver le meilleur chemin dans un graphe, représentant une solution candidate au problème cible, de manière analogue à la façon dont les fourmis naturelles coopèrent pour trouver le chemin le plus court entre deux points ; leur nid est une source de nourriture. En effet, en se déplaçant du nid à la source de nourriture et vice-versa, les fourmis déposent au passage sur le sol une substance odorante appelée phéromone, ce qui a pour effet de

créer une piste chimique (DORIGO, 1992). Lorsqu'une fourmi choisit son chemin, elle choisit la piste qui porte la plus forte concentration de phéromone, ce qui entraîne l'apparition des chemins plus courts.
Les colonies artificielles de fourmis simulent ce comportement pour construire des solutions à un problème d'optimisation. Par conséquent, le problème étudié est modélisé comme un graphe complètement connecté, dont les nœuds sont des composants de solutions. Une valeur de phéromone est associée à chaque composante de la solution et guide la construction de cette solution (DHAENENS, 2016). Le système de fourmis a été employé avec succès sur des nombreux problèmes par exemple : voyageur de commerce, affectation quadratique, etc. Un aperçu global est présenté dans l'algorithme 6.

Algorithm 6 algorithme d'optimisation des colonies de fourmis

```
Initialiser les valeurs de phéromone
while critère d'arrêt non satisfait do
  for tous les fourmis i do
    Construire des solutions en utilisant le chemin des phéromones
    Mettre à jour les chemins de phéromones (évaporation, renforcement)
  end for
end while
```

Systèmes immunitaires artificiels (Artificial Immune System)
Les systèmes immunitaires artificiels (AIS) sont des systèmes informatiques métaphoriques apparus dans les années 90 et sont inspirés du fonctionnement du système immunitaire naturel pour ce qui est de la mémorisation et l'apprentissage comme moyens de résolution de problèmes d'optimisation (JOURDAN, 2003). Les fonctionnements des systèmes immunitaires ont un grand intérêt pour les informaticiens comprennent notamment :

- chaque corps possède son propre système immunitaire avec ses faiblesses et forces,
- le système immunitaire reconnaît et élimine les molécules qui n'appartiennent pas à l'individu,
- le système immunitaire peut détecter et réagir aux antigènes que le corps n'a jamais rencontrés,

Les AIS utilisent la métaphore de sécrétion d'anticorps où un anticorps va représenter une solution potentielle au problème (JOURDAN, 2003).

1.5 Hybridation entre métaheuristiques

Le développement des métaheuristiques hybrides est susciter l'intérêt académique. Les méthodes hybrides combinent différents concepts de différentes métaheuristiques. L'hybridation des métaheuristiques tentent de fusionner les points forts et éliminent les faiblesses des métaheuristiques (JOURDAN, 2003).
Toutes les métaheuristiques ont été hybridées avec succès dans plusieurs applications. Selon la taxonomie proposée par Talbi (JOURDAN, 2003) l'hybridation des métaheuristiques entre elles se fait en deux classifications principales. Une classification hiérarchique et une classification à plat.

1.5.1 Classification hiérarchique des métaheuristiques

La classification hiérarchique (JACQUIN, 2015) est caractérisée par le niveau et le mode de l'hybridation. L'hybridation par niveau se divise en deux classes : l'hybridation de bas

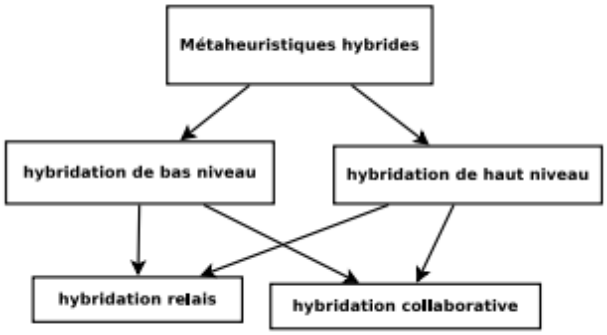

FIGURE 1.4 – Classification hiérarchique des métaheuristiques hybride (TALBI, 2002)

niveau et l'hybridation de haut niveau . Dans le niveau bas, une métaheuristique h1 remplace une fonction ou un opérateur d'une autre méthode h1 qui l'englobe. À l'inverse, dans l'hybridation de haut niveau, chaque métaheuristique garde sa propriété au cours de l'hybridation. Chacun des deux niveaux d'hybridation se subdivise en deux modes de coopération à savoir, le mode collaboratif et le mode relais. L'hybridation collaborative se fait lorsque les différentes méthodes fonctionnent en parallèle pour explorer l'espace de solutions. Dans le mode relais, les métaheuristiques sont exécutées de façons séquentielles, c'est-à-dire le résultat de la méthode précédente est l'entrée de la méthode suivante.
La combinaison des classes précédente (niveaux et mode) donne quatre classes d'hybridation comme illustrée la figure 1.4. A savoir l'hybridation relais de bas niveau, l'hybridation collaborative de bas niveau, l'hybridation relais de haut niveau et l'hybridation collaborative de haut niveau.

L'hybridation relais de bas niveau

Cette classe d'hybridation représente des algorithmes dans lesquels une métaheuristique donnée est incorporée dans une métaheuristique à solution unique. Les méthodes doivent ne s'exécutent que séquentiellement. L'exécution de la méthode globale doit dépendre du résultat de la méthode intégrée (HACHIMI, 2013). Pour résoudre le problème du voyageur de commerce et celui de la partition de graphe, Martin (MARTIN, 1990) est incorporé une recherche locale dans un algorithme de recuit simulé. Cette méthode a surpassé les méthodes traditionnelles basées sur la recherche locale.

L'hybridation collaborative de bas niveau

Consiste à intégrer une métaheuristique à base de solution unique dans une métaheuristique à population de solutions. L'avantage de ce type d'hybridation est de compenser la puissance d'exploration d'une recherche globale et l'exploitation d'une recherche locale (STÜTZLE et HOOS, 2000). Pour résoudre le problème du voyageur de commerce et la partition de graphes Stûtzle et Hoos (AZIMI, 2005) insèrent une fonction de recherche locale dans un algorithme de colonie de fourmis. Pour résoudre le problème de partition de graphes Gambardella et al. (TANESE, 1987) incorporent un algorithme de descente dans un algorithme de colonie de fourmis. Taillard et al. Remplacent l'algorithme de descente par l'algorithme du recuit simulé. Fleurent et Ferland (ELBENANI, FERLAND et BELLEMARE, 2012)

remplacent l'opérateur de mutation d'un algorithme génétique par une recherche taboue pour résoudre le problème de partition de graphe.

L'hybridation relais de haut niveau

On parle de l'hybridation relais de haut niveau lorsque les métaheuristiques sont s'exécuter de manière séquentielle c'est-à-dire les solutions finales de la première métaheuristique sont des entrées de la deuxième. Ce type d'hybridation est le plus utilisé dans la littérature. Pour résoudre le problème d'agencement d'horaires d'examens (LIN et YAMASHITA, 2002) utilisent deux hybridations de ce type, la première, consiste à initialiser les phéromones des colonies de fourmis par une recherche locale. Dans la deuxième, il a utilisé la solution finale de l'algorithme de colonie de fourmis en tant qu'une solution initiale pour une recherche taboue. Dans (BOUHLEL et al., 2007), une hybridation entre l'algorithme génétique et le recuit simulé est proposée où la population initiale de l'algorithme génétique est créé par la méthode du recuit simulé.

L'hybridation collaborative de haut niveau

Cette hybridation consiste à combiner deux méthodes métaheuristiques qui ne s'imbriquent pas l'une dans l'autre et qui s'exécutent parallèlement. Ces deux algorithmes travaillent sur le même problème d'optimisation mais sur des parties différentes de l'espace de recherche et échangent des informations. Dans (GAMBARDELLA, MONTEMANNI et WEYLAND, 2012), pour résoudre le problème du partitionnement de graphe Ghédira et al, utilisent le recuit simulé et la recherche Taboue, les deux méthodes travaillent parallèlement et échangent des informations à intervalles réguliers pour trouver la meilleure solution. Notons qu'il existe d'autres travaux basés sur ce type d'hybridation en utilisant des recuits simulés, des recherches taboues.

Classification à plat des métaheuristiques

La classification à plat des métaheuristiques comporte 3 critères (HACHIMI, 2013) de classification :

Homogènes/Hétérogènes

On trouve des méthodes hybridées homogènes lorsque les algorithmes hybridés sont identiques. C.-à-d. se basent sur la même métaheuristique, si les métaheuristiques utilisées sont différentes l'hybridation est dite hétérogène. Une méthode collaborative de haut niveau dans (TAILLARD, 1993) est proposée ou plusieurs algorithmes génétiques sont combinés. Ces algorithmes travaillent en parallèle sur des petites sous populations.

Globales/Partielles

L'hybridation globale a lieu lorsque toutes les méthodes combinées agissent sur l'ensemble de l'espace de recherche. À l'opposé, l'hybridation partielle découpe un problème en sous problèmes où chaque problème a son propre espace de recherche. Comme exemple de l'hybridation partielle, dans (ABBATTISTA, ABBATTISTA et CAPONETTI, 1995), une décomposition du problème du routage de véhicules en divisant l'ensemble des villes à visiter en secteurs dont chacun représente un espace de recherche, l'algorithme utilisé pour la résolution de ce problème est la recherche taboue.

Généralistes/Spécialistes

On parle de l'hybridation générale quand toutes les méthodes hybridées résolvent le même problème d'optimisation. À l'inverse, les hybridations spécialistes ont lieu lorsque chaque méthode traite un problème d'optimisation différent. Par exemple, l'utilisation d'une métaheuristique pour optimiser les paramètres d'une autre métaheuristique. L'utilisation d'une métaheuristique pour initialiser les paramètres d'une autre métaheuristique est un exemple de ce type. On peut citer, Abbattista (KRUEGER, 1994) optimise un algorithme de colonie de fourmis à l'aide d'un l'algorithme génétique (AG). Dans (GRINSTEIN et WIERSE, 2002) les paramètres d'un recuit simulé sont optimisés à l'aide d'un AG.

1.6 Conclusion

Dans ce chapitre, nous avons présenté quelques méthodes d'optimisation en s'appuyant sur les métaheuristiques. Ces dernières sont très efficaces pour la résolution d'un problème d'optimisation combinatoire sans avoir besoin de modifier la structure de base de l'algorithme utilisé. Elles sont devenues très populaires grâce à leur simplicité, diversité et flexibilité. Il est à noter qu'une bonne performance nécessite souvent une formalisation adéquate du problème posé et une adaptation intelligente d'une métaheuristique.
Nous nous sommes ensuite intéressés aux techniques d'hybridations permettant d'hybrider plusieurs approches de résolutions pour objectif d'obtenir une nouvelle approche plus rapide et / ou plus efficace. Dans ce but on a présenté deux taxinomies, elles permettent de comprendre quels sont les différents moyens d'hybridation possibles et comment ceux-ci ont été exploités dans la littérature scientifique.
Les métaheuristiques sont des méthodes génériques capables de traiter de nombreux problèmes d'optimisation. Leur diversité et leur flexibilité rendent cette classe de méthodes très attrayante pour s'attaquer aux problèmes difficiles qui apparaissent dans l'extraction de connaissances. Ceci est l'objet des chapitres suivants.

Chapitre 2

L'extraction de connaissances

2.1 Introduction

Nous vivons dans un monde où de grandes quantités de données sont collectées quotidiennement. Cette quantité est stockée dans des bases de données réelles, et continue de croître rapidement, cela permet de créé à la fois une opportunité et un besoin de méthodes automatiques qui découvrent les connaissances "cachées" dans telles bases de données. Si une telle activité d'extraction de connaissances est réussie, les connaissances découvertes peuvent être utilisées pour améliorer le processus décisionnel d'une organisation par exemple.
Il existe une distinction entre les termes datamining (exploration de données) et l'extraction de connaissances. Le terme exploration de données se réfère à l'étape centrale du processus de l'extraction de connaissances dans des bases de données.
Le processus de l'extraction de connaissances comprend plusieurs étapes de prétraitement (ou de préparation des données) et de post-traitement (ou d'affinement des connaissances). Le but des méthodes de préparation des données est de transformer les données pour faciliter l'application d'un (ou plusieurs) algorithme (s) donné (s) de datamining, alors que le but des méthodes de raffinement des connaissances est de valider et d'affiner les connaissances découvertes.
Dans ce chapitre, nous décrivons le processus de l'extraction de connaissances, nous introduirons une présentation générale de chaque étape, et nous détaillerons ensuite plus précisément l'étape de datamining (l'exploration de données).

2.1.1 L'extraction de connaissance

Le processus de l'extraction de connaissances (ECD) (knowledge discovery in databases, KDD) (FAYYAD et al., 1996 ; FREITAS, 2002a), également appelé l'extraction des connaissances à partir des données, recherche de nouvelles connaissances dans certains domaines d'application. Il est défini comme un processus non trivial consistant à identifier des modèles de données valides, nouveaux, potentiellement utiles et finalement compréhensibles. Ce processus est constitué de plusieurs étapes, qui couvre la préparation des données, l'application de méthodes de fouille (datamining) et enfin la validation ou la visualisation des résultats, plus précisément l'extraction de connaissances concerne l'ensemble du processus d'extraction de connaissances, y compris la manière dont les données sont stockées et accessibles, la manière d'utiliser des algorithmes efficaces et évolutifs pour analyser des ensembles de données massives, la manière d'interpréter et de visualiser les résultats, et la manière de modéliser et de soutenir l'interaction entre l'homme et la machine (voir Figure 2.1). L'automatisation d'une de ces étapes se révèle une tâche difficile (CIOS et al., 2007).

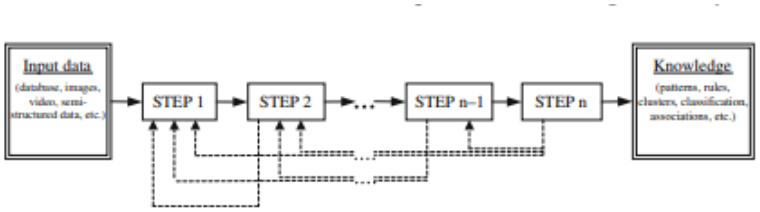

FIGURE 2.1 – Structure séquentielle du modèle KDP

Présentation du processus de l'extraction des connaissances

Le processus de l'extraction de connaissance (Knowledge Discovery Process (KDP)) est un modèle constitué d'un ensemble d'étapes de traitement à suivre par les praticiens lors de l'exécution d'un projet de l'extraction de connaissances. Le modèle décrit les procédures qui sont exécutées dans chacune de ses étapes. Il est principalement utilisé pour planifier, mener à bien et réduire le coût d'un projet donné. Depuis les années 1990, plusieurs KDP ont été développés. Les premiers efforts ont été menés par la recherche universitaire, mais ont été rapidement suivis par l'industrie (CIOS et al., 2007). La première structure de base du modèle a été proposée par Fayyad et al. (FAYYAD et al., 1996) puis améliorée/modifiée par d'autres. Le processus se compose de plusieurs étapes, qui sont exécutées séquentiellement. Chaque étape est lancée après l'achèvement de l'étape précédente et requiert le résultat généré par l'étape précédente comme entrée. Une autre caractéristique commune des modèles proposés est la gamme d'activités couvertes, qui s'étend de comprendre le domaine du projet et les données, en passant par la préparation et l'analyse des données, jusqu'à l'évaluation, la compréhension et l'application des résultats générés. Tous les modèles proposés mettent également l'accent sur la nature itérative du modèle, en termes de nombreuses boucles de rétroaction qui sont déclenchées par un processus de révision. Un schéma est présenté à la figure 2.1. Les principales différences entre les modèles décrits ici résident dans le nombre et la portée de leurs étapes spécifiques. Une caractéristique commune à tous les modèles est la définition des entrées et des sorties. Les entrées typiques incluent des données dans différents formats, tels que des données numériques et nominales stockées dans des bases de données ou des fichiers plats ; images ; vidéo ; des données semi-structurées, telles que XML ou HTML ; etc. Le résultat est la nouvelle connaissance générée - généralement décrite en matière de règles, modèles, modèles de classification, associations, analyse statistique, etc (HAND et ADAMS, 2014).

Aperçu du processus de l'extraction des connaissances

Les efforts pour établir un modèle du processus de l'extraction de connaissances ont été lancés dans le milieu universitaire. Au milieu des années 1990, lorsque le domaine de datamining était en cours d'élaboration, les chercheurs ont commencé à définir des procédures en plusieurs étapes pour guider les utilisateurs de datamining dans le monde complexe de l'extraction des connaissances. L'accent principal était de fournir une séquence d'activités qui aideraient à exécuter un processus de l'extraction de connaissances dans un domaine arbitraire. Les deux modèles de processus développés en 1996 et 1998 sont le modèle en neuf étapes de Fayyad et al.(FAYYAD et al., 1996) et le modèle en huit étapes d'Anand et Buchner (BUCHNER et al., 1999). Nous présentons ci-dessous le premier d'entre eux, qui est perçu comme le modèle de recherche le plus important.

Le processus de l'extraction de connaissances proposé par Fayyed et al. (FAYYAD et al., 1996) est une séquence itérative et interactive des principales étapes suivantes :

1. Développer et comprendre le domaine d'application. Cette étape permettra d'abord de comprendre le domaine d'application, les connaissances antérieures pertinentes,

et les objectifs de l'utilisateur final des connaissances extraites (FAYYAD et al., 1996).

2. Création d'un ensemble de données cible. Dans cette étape à partir de données brutes, certaines informations sont sélectionnées pour faire face aux objectifs de l'extraction de connaissance qui sont identifiés. Cette étape comprend généralement l'interrogation des données existantes pour sélectionner le sous-ensemble souhaité (FAYYAD et al., 1996).
3. Nettoyage et prétraitement des données. L'étape de nettoyage peut consister à gérer les valeurs manquantes et à supprimer le bruit ou les valeurs aberrantes, par exemple. Des méthodes statistiques complexes, ainsi que des algorithmes d'exploration de données, ont été proposés à cet effet. Il s'agit d'une étape cruciale car ces données représentent la matière première pour les étapes suivantes (FAYYAD et al., 1996).
4. Réduction et projection des données. Cette étape a pour but de préparer des données à exploiter. Il peut s'agir de la réduction des dimensions (sélection d'attributs, échantillonnage) et la transformation des attributs (par exemple la discrétisation des attributs numériques). Cela peut également être une étape cruciale pour la réussite du projet de l'extraction de connaissances car elle dépend du contexte et est liée directement aux objectifs du projet l'extraction de connaissances (FAYYAD et al., 1996).
5. Choix de la tâche d'exploration de données. Ici, le data mineur fait une correspondance entre les objectifs définis à l'étape une et une tache d'exploration de données (datamining) particulières, telle que la classification, la régression, le clustering, etc (FAYYAD et al., 1996).
6. Choix de l'algorithme d'exploration de données (datamining). Le data mineur sélectionne des méthodes (algorithmes) pour créer des modèles pertinents (FAYYAD et al., 1996).
7. Exploration de données. C'est le cœur du processus de l'extraction des connaissances. Cette étape génère des modèles permettre de l'extraction d'informations utiles à partir de grands ensembles de données ou de bases de données. Plusieurs tâches d'exploration de données peuvent être identifiées selon le type de modèles attendus (FAYYAD et al., 1996).
8. Interpréter les modèles extraits. Les modèles extraits de l'étape d'exploration des données (datamining) sont transformés en connaissances, grâce à l'interprétation. Une évaluation est réalisée pour déterminer si la connaissance extraite il s'agit d'une nouvelle connaissance et si elle répond aux objectifs identifiés. Si ce n'est pas le cas, certains ajustements doivent être faits et le processus est répété soit depuis le début, soit depuis une étape intermédiaire (FAYYAD et al., 1996).
9. Consolider les connaissances extraites. La dernière étape consiste à intégrer les connaissances extraites dans le système de performance, à les documenter et à les communiquer aux parties intéressées. Cette étape peut également inclure la vérification et la résolution de conflits potentiels avec des connaissances précédemment crues (FAYYAD et al., 1996).

2.2 Datamining (exploration de données)

Le terme de datamining signifie littéralement forage de données, ou la fouille de données est l'ensemble des méthodes et techniques destinées à l'exploration et l'analyse de grandes bases de données informatiques. Et de transformer ces données en informations utiles, en établissant des relations entre les données ou en repérant des patterns. Ces informations

peuvent ensuite être utilisées par les entreprises pour augmenter un chiffre d'affaires ou pour réduire des coûts. (FALTINGS et SCHUMACHER, 2009).
Plusieurs définitions ont été proposées dans (FALTINGS et SCHUMACHER, 2009),

— Le datamining serait : - " la découverte de nouvelles corrélations, tendances et modèles par le tamisage d'un grand nombre de données " ;
— " l'extraction d'informations originales, auparavant inconnues, potentiellement utiles à partir des données " ;
— " un processus d'aide à la décision où les utilisateurs cherchent des modèles d'interprétation dans les données" ;
— "un processus de mise à jour de nouvelles corrélations, tendances et de modèles significatifs par un passage au crible des bases de données volumineuses, et par l'utilisation de modèles d'identification technique aussi bien statistiques que mathématiques." .

2.2.1 Les tâches principales de datamining (exploration de données)

Les tâches d'exploration de données peuvent être classées en deux catégories : tâches supervisées et non supervisées (prédictives ou descriptives). Les tâches supervisées apprennent sur les données disponibles pour faire des prédictions sur des nouvelles données, tandis que les tâches non supervisées impliquent une description des données et des relations existantes. Les principales tâches d'exploration de données sont la classification (supervisée), le clustering (également appelé classification non supervisée), l'extraction de règles d'association et la sélection des attributs, comme illustré la Figure 2.2. En effet, même si la sélection d'attributs peut être utilisée dans l'étape d'intégration, pour préparer des données, elle peut également être utilisée conjointement avec d'autres tâches d'exploration de données telles que la classification ou le clustering (DHAENENS, 2016). Pour donner un aperçu général, chacune de ces tâches est brièvement décrite ci-après.

La classification

La classification est probablement la tâche d'exploration de données la plus étudiée. Nous présentons ci-dessous un aperçu des concepts de base et des problèmes liés à cette tâche. Dans la tâche de classification, chaque instance de données (ou enregistrement de base de données) appartient à une classe, qui est indiquée par la valeur d'un attribut cible (DHAENENS, 2016). Cet attribut peut prendre un petit nombre de valeurs discrètes, chacune d'entre elles correspondent à une classe. Chaque instance se compose de deux parties, à savoir un ensemble de valeurs d'attributs prédicteurs et une valeur d'attribut cible (classe). Les premières sont utilisées pour prédire la valeur de la seconde. Notez que les attributs prédicteurs doivent être pertinents pour prédire la classe (valeur d'attribut cible) d'une instance de données (DHAENENS, 2016). Par exemple, si l'attribut cible indique si un patient a ou va développer une certaine maladie, les attributs prédicteurs doivent contenir des informations médicales pertinentes pour cette prédiction, et non pas des attributs non pertinents tels que le nom du patient.
Le modèle est construit à partir des données disponibles (observations disponibles), puis pour les nouvelles observations, le modèle est appliqué pour déterminer la valeur de la variable cible (DHAENENS, 2016).
Il existe de nombreuses applications de la classification. Il peut être utilisé, par exemple :

— Dans la détection de fraude, pour déterminer si une carte de crédit particulière est frauduleuse ;
— Dans le diagnostic médical des maladies, pour déterminer si un patient peut contracter une maladie à l'avenir ;

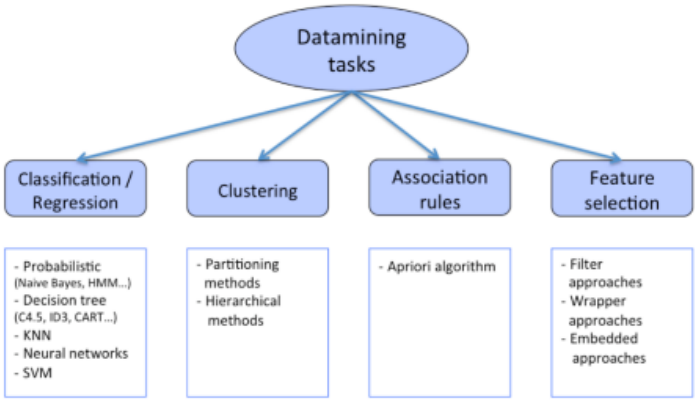

FIGURE 2.2 – Vue globale des taches et approches de l'exploration de données (DHAENENS, 2016)

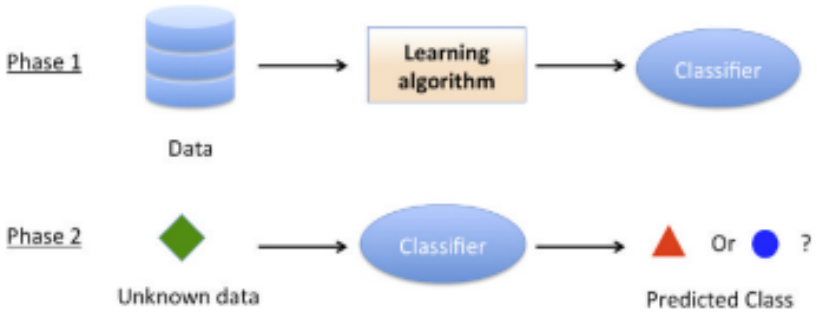

FIGURE 2.3 – La tache de classification (DHAENENS, 2016)

— En marketing, pour identifier les clients susceptibles d'être intéressés par un produit donné ;
— Dans l'analyse des réseaux sociaux, pour prédire les propriétés utiles des acteurs d'un réseau social.

Comme mentionné précédemment, l'objectif de cette tâche d'exploration de données est de construire un modèle qui prédit la valeur d'une variable, appelée la «classe», à partir des valeurs connues d'autres variables. Le modèle est construit à partir d'observations connues, puis le modèle est appliqué pour déterminer la valeur de la variable cible (la classe) pour des nouvelles observations. La Figure 2.3 illustre ce processus en deux phases (DHAENENS, 2016).
Dans sa forme de base, la variable prédite (la classe) est catégorique sans aucun classement. Cependant, des extensions ont été proposées pour considérer les cas où la classe est décrite par un ensemble fini de valeurs avec une relation d'ordre. De plus, lorsque la variable prédite est numérique, la tâche devient une régression.
Plusieurs approches standard ont été proposées pour traiter la classification supervisée (DHAENENS, 2016).

La régression

La régression est une méthode statistique utilisée dans la finance, et d'autres disciplines qui tente de déterminer la force de la relation entre une variable dépendante (généralement désignée par Y) et une série d'autres variables (appelées variables indépendantes).
Les deux types de régression de base sont la régression linéaire simple et la régression linéaire multiple. La régression linéaire simple utilise une variable indépendante pour expliquer ou prédire le résultat de la variable dépendante Y, tandis que la régression linéaire multiple utilise deux variables indépendantes ou plus pour prédire le résultat (DHAENENS, 2016).
Comme la classification, la régression peut être considérée comme une tâche d'exploration de données impliquant la prédiction de la valeur d'un attribut cible (classe) défini par l'utilisateur, compte tenu des valeurs des autres attributs (prédicteurs). La principale différence entre la classification et la régression est la suivante. Dans la classification, l'attribut cible est catégorique (nominal). En revanche, dans la régression, l'attribut cible est continu (valeur réelle). Dans les deux tâches, l'ensemble des attributs peut inclure des attributs catégoriels et continus. Cependant, dans la tâche de régression, il est probablement plus courant d'avoir un ensemble d'attributs ne contenant que des attributs continus. Cela permet l'utilisation d'un large éventail de méthodes de régression statistique / numérique (DHAENENS, 2016).

Le clustering

En substance, la tâche de clustering vise à décomposer ou à partitionner un ensemble de données en groupes (ou clusters) d'instances de données, de telle sorte que : (a) chaque cluster possède des instances très similaires (ou "proches") les unes des autres (la distance entre les points des groupes est minimisée). Et (b) les instances de chaque cluster soient aussi différentes (ou "éloignées") que possible des points des autres clusters (la distance entre les points de différents groupes est maximisée) (DHAENENS, 2016).
En d'autres termes, un algorithme de clustering devrait maximiser la similitude intra-cluster (ou intra-cluster) et minimiser la similitude intercluster (entre-clusters). Cependant, la satisfaction de ces deux objectifs de base ne suffit pas pour obtenir une bonne solution de clustering, car ces deux objectifs peuvent être satisfaits de manière triviale en attribuant simplement chaque instance de données vers un cluster singleton différent. Par conséquent, il est également important de privilégier un nombre relativement faible de clusters, augmentant ainsi le nombre d'instances de données affectées à un cluster. Un défi majeur de la tâche de regroupement est de trouver un bon compromis entre les trois objectifs ci-dessus (CIOS et al., 2007).
Parmi les différents types d'algorithmes de clustering, nous mentionnons ici deux les plus populaires : le partitionnement itératif et le clustering hiérarchique (ALDENDERFER et BLASHFIELD, 1984 ; BACKER, 1995). Chacun de ces deux types peut être divisé en deux sous-types (DHAENENS, 2016).

- les méthodes de partitionnement partitionnent les données en ensemble afin que les ensembles soient aussi homogènes que possibles. La méthode la plus connue est k-means ;
- les méthodes hiérarchiques fusionnent ou divisent les clusters pour construir des clusters homogènes.

Plus précisément, les méthodes hiérarchiques produisent une hiérarchie de clusters, tandis que les méthodes de partitionnement itératives produisent une solution de cluster "plate". Les méthodes hiérarchiques peuvent être subdivisées en méthodes d'agglomération et de division. Les méthodes agglomérées commencent à affecter chaque instance de données à un cluster, puis fusionnent de manière itérative les deux clusters les plus proches jusqu'à

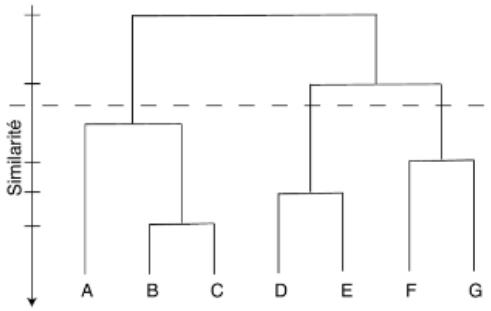

FIGURE 2.4 – Un dendrogramme construit par un algorithme de clustering d'agglomération (ZHANG et ZHANG, 2003)

ce qu'il n'y ait qu'un seul cluster, contenant toutes les instances des données en cours d'exploration. Les méthodes de divisons fonctionnent en sens inverse. Ils commencent à affecter toutes les instances de données à un cluster. Ensuite, ce cluster est itérativement divisé en clusters de plus en plus petits (ZHANG et ZHANG, 2003).
Les méthodes de regroupement par division sont en général beaucoup plus coûteuses en termes de calcul que les méthodes d'agglomération. C'est probablement l'une des raisons de la plus grande popularité des méthodes d'agglomération.
Les méthodes de partitionnement itératives peuvent être subdivisées en méthodes qui produisent des clusters qui ne se chevauchent pas et en méthodes qui produisent des clusters qui se chevauchent. Dans ce dernier, une instance de données peut appartenir à deux clusters ou plus en même temps, ce qui n'est pas autorisé dans le premier (ZHANG et ZHANG, 2003).
Dans ce qui suit, seuls les algorithmes standards sont décrits.
Clustering hiérarchique Il existe deux types distincts de méthodes hiérarchiques : les méthodes agglomératives, qui commencent par un nombre de clusters et les fusionnent progressivement, et les méthodes de division, qui commencent avec toutes les données dans un seul cluster et les divisent progressivement en clusters plus petits. Les méthodes d'agglomération sont les méthodes les plus utilisées. Les méthodes hiérarchiques d'analyse des clusters permettent un affichage graphique pratique dans lequel la séquence entière de fusion (ou de fractionnement) des clusters sont affichés. En raison de sa structure arborescente, le résultat l'affichage s'appelle un dendrogramme, comme illustrait la figure 2.4.

Les méthodes d'agglomération sont basées sur des mesures de distance entre clusters et sont décrits dans l'algorithme 7. Ils fusionnent de manière itérative les deux clusters les plus proches pour réduire le nombre de clusters. Habituellement, la configuration de départ du processus se compose de chaque point de son propre cluster. Ensuite, la fusion est réalisée jusqu'à ce qu'un seul cluster contenant tous les points de données soit obtenu. Différentes mesures de distance entre les clusters ont été proposées et conduisent à différents algorithmes : par exemple, l'algorithme de clustering hiérarchique à lien unique (single-link) ou la distance entre deux clusters est le minimum des distances entre toutes les paires d'exemples de ces deux clusters. Une autre variante est l'algorithme de clustering hiérarchique à lien complet (complete-link), où on prend le maximum des distances (ZHANG et ZHANG, 2003).

Les méthodes de division commencent par un seul cluster qui contient tous les points de données et le divise en sous-ensembles. Le processus de division est répété autant de fois que nécessaire et se termine par des clusters singleton.

Algorithm 7 Méthodes d'agglomération

for $i = 1, ..., n$ **do**
  while Il reste plus d'un cluster à gauche **do**
    for $k = 1, ..., K$ **do**
      Laisser C_i et C_j être les clusters minimisant la distance $D(C_k, C_h)$ entre n'importe deux clusters
      $C_i = C_i \cup C_j$
      Supprimer le cluster C_j
    end for
  end while
end for

Clustering de partitionnement

Dans le clustering par partition, l'objectif est de partitionner un ensemble de données en K ensembles disjoints de points, de sorte que les points d'un ensemble soient aussi homogènes que possibles. L'homogénéité est calculée à l'aide d'une fonction de score qui est souvent basée sur une notion de similarité ou de distance dénotée par d. L'objectif est de minimiser une fonction (moyenne, somme, etc.) sur la dissimilitude entre chaque point et le centroïde du cluster à laquelle il est attribué. Le centroïde d'un ensemble pourrait être un point de données réelles, ou une "position" dans l'espace des caractéristiques. L'algorithme le plus connu de cette catégorie est le K-means (MACQUEEN et al., 1967). Dans sa version la plus basique, cet algorithme commence par choisir au hasard K centres de grappes, puis il étiquette chaque point en fonction du centre de grappe le plus proche. L'algorithme itéré ensuite jusqu'à ce qu'il n'y ait plus de changement des centres de grappe (DHAENENS, 2016).

Les règles d'association

Le problème des règles d'association a été formulé pour la première fois par Agrawal et al. (MAFARJA et MIRJALILI, 2018) prenons un ensemble de données dans lequel une instance de données se compose d'un ensemble d'attributs binaires appelés éléments. Chaque instance de données représente une transaction client et chaque élément de cette transaction peut prendre la valeur oui ou non, indiquant si le client correspondant a acheté cet article dans cette transaction ou non. Une règle d'association est une relation de la forme $X \sim Y$, où X et Y sont des ensembles d'éléments disjoints, c'est-à-dire $X \cap Y = 0$ (MITRA et CHAUDHURI, 2000 ; DHAENENS, 2016). Chaque règle d'association est généralement évaluée par un support et une mesure de confiance (KABLI, HAMOU et AMINE, 2018). La prise en charge d'une règle d'association est le rapport du nombre d'instances (transactions) ayant la valeur oui pour tous les éléments dans l'ensemble X et l'ensemble Y diviser par le nombre total d'instances. La confiance d'une règle d'association est le rapport du nombre d'instances ayant la valeur oui pour tous les éléments dans l'ensemble X et l'ensemble Y diviser par le nombre d'instances ayant la valeur oui pour tous les éléments dans l'ensemble X (MITRA et CHAUDHURI, 2000 ; DHAENENS, 2016). Les règles avec un support et une confiance supérieurs ou égaux aux seuils spécifiés par l'utilisateur (appelés support minimum et confiance minimum). L'algorithme le plus connu pour extraire des règles d'association est a priori proposé par Agrawal et Srikant (AGRAWAL, SRIKANT et al., 1994 ; DHAENENS, 2016). Cet algorithme en deux phases trouve d'abord tous les ensembles d'éléments fréquents, puis génère des règles de confiance élevée à partir de ces ensembles. Des nombreuses améliorations de la méthode initiale, y compris des implémentations parallèles, ont été proposées pour nous permettre de traiter des très grandes bases de données (BORGELT, 2003 ; YE et CHIANG, 2006 ; ZAKI, 2001 ; MITRA et CHAUDHURI, 2000 ; DHAENENS, 2016).

La sélection d'attributs

Une difficulté dans l'exploration de données, en particulier dans le cas du Big Data est liée à la taille énorme des ensembles de données et à la présence de trop d'attributs (DHAENENS, 2016).
En classification, toutes les variables qui sont stockées dans la base de données ne sont pas toutes nécessaire à une discrimination précise. Leur inclusion dans la classification peut même réduire les performances du modèle (DHAENENS, 2016). La sélection des caractéristiques, également connue sous le nom de sélection de variables, sélection d'attributs ou sélection de sous-ensembles de variables, vise à sélectionner un ensemble optimal de caractéristiques ou d'attributs pertinents qui sont nécessaires à la classification. Par exemple, certains attributs peuvent être redondants ou non liés à la variable prédictive (DHAENENS, 2016). Par conséquent, la sélection de certains attributs pourrait être nécessaire pour réduire le temps de calcul des algorithmes d'exploration de données, pour simplifier le modèle obtenu afin d'avoir une discrimination précise entre les observations (DHAENENS, 2016). Ensuite, l'objectif est de trouver un sous-ensemble de variables pertinentes p_1, où $p_1 << p$. Par conséquent, l'objectif principal de la sélection des attributs dans l'apprentissage supervisé est de trouver un sous-ensemble d'attributs qui produit une précision de classification plus élevée. D'autre part, dans l'apprentissage non supervisé, la sélection des attributs vise à trouver un bon sous-ensemble de variables qui forme des clusters de haute qualité pour un nombre donné de clusters. Une sélection appropriée des attributs peut améliorer l'efficacité d'un modèle d'inférence (DHAENENS, 2016).
En apprentissage supervisé, trois approches existent selon l'interaction avec la procédure de classification :

- les approches de filtrage évaluent les attributs en fonction de leurs caractéristiques pour les sélectionner (ou non) (DHAENENS, 2016) ;
- les approches wrapper évaluent la qualité d'un sous-ensemble des attributs à l'aide d'un algorithme d'apprentissage (DHAENENS, 2016) ;
- les approches intégrées combinent les deux approches susmentionnées en incorporant dans une approche wrapper une interaction plus profonde entre la sélection des attributs et la construction du classificateur (DHAENENS, 2016).

2.3 La différence entre l'exploration de données et l'extraction de connaissances (KDD)

Le terme KDD doit être employé pour décrire l'ensemble du processus de l'extraction de connaissances à partir de données. Dans le contexte, la connaissance signifie les relations et les modèles entre les éléments de données. L'exploration de données (datamining) doit être utilisée exclusivement pour l'étape de l'extraction de connaissances dans le processus de KDD. Plus précisément,l'exploration de données (datamining) est le processus de recherche ou d'analyse de données à partir d'une grande quantité de données, tandis que l'extraction de connaissances est le processus de recherche de connaissances à partir d'une grande quantité de données en utilisant l'exploration de données comme montre la figure 2.6.

La partie de l'extraction de connaissances (KDD) qui traitant l'analyse des données a été appelée "datamining" et c'est une étape très cruciale du processus KDD.

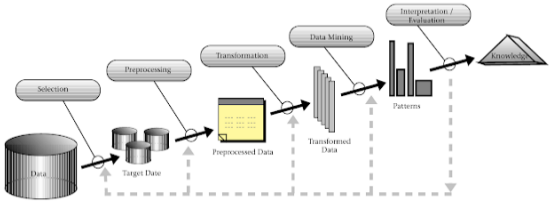

FIGURE 2.5 – Processus de l'extraction de données (KDD).

FIGURE 2.6 – La différence entre l'extraction de données (KDD) et l'exploration de données (datamining).

2.4 Les tâches d'exploration de données comme des problèmes d'optimisation

Comme nous l'avons vu précédemment, les tâches d'exploration de données portent sur des opérations telles que la classification, le clustering, la sélection d'attributs, etc. Tous ces problèmes peuvent être formulés comme des problèmes d'optimisation combinatoire (DHAENENS, 2016). C'est pourquoi plusieurs travaux utilisant des méthodes d'optimisation pour résoudre des problèmes d'exploration de données (OLAFSSON, 2006 ; OLAFSSON, LI et WU, 2008 ; MEISEL et MATTFELD, 2010 ; CORNE, DHAENENS et JOURDAN, 2012 ; MAIMON et ROKACH, 2007).
Le contexte des Big Data rend difficile la résolution de ces problèmes en utilisant des approches exactes. La métaheuristique sera donc une solution intéressante. Maimon et al se concentre sur les métaheuristiques, en particulier les algorithmes évolutionnaires et l'intelligence en essaim (MAIMON et ROKACH, 2005) pour l'extraction des connaissances (DHAENENS, 2016).
De plus, Freitas se concentre dans son livre sur l'exploration de données (datamining) et l'extraction de connaissances avec des algorithmes évolutifs, qui représentent une partie de la métaheuristique (FREITAS, 2002b ; AGGARWAL et al., 2018). En particulier, cette thèse révèle comment les métaheuristiques peuvent être utilisées pour la sélection d'attributs et la classification (DHAENENS, 2016).

2.5 La recherche d'informations (IR)

La recherche d'informations (IR) est la science de la recherche d'informations dans des documents. Les documents peuvent être textuels ou multimédias et peuvent résider sur le Web (FALTINGS et SCHUMACHER, 2009). Les différences entre les systèmes traditionnels de

recherche d'informations et les systèmes de bases de données sont les suivants : la recherche d'informations suppose que :

1. les données recherchées ne sont pas structurées ;
2. Les requêtes sont formées principalement par des mots-clés, qui n'ont pas de structures complexes (contrairement aux requêtes SQL dans les systèmes de base de données).

En bref, la tâche d'un système de recherche d'informations (IR) est la suivante : à partir d'une collection de documents, trouver des informations utiles correspondant à la requête d'un utilisateur. Plusieurs facteurs rendent cette tâche difficile (KARASOZEN, RUBINOV, WEBER et al., 2006) :

— les informations contenues dans la base de données de documents ne sont généralement pas structurées ;
— les documents sont généralement écrits dans un langage naturel sans contrainte ;
— Très souvent, les documents couvrent un large éventail de sujets.

Les approches typiques de la recherche d'informations adoptent des modèles probabilistes. Par exemple, un document texte peut être considéré comme un sac de mots, c'est-à-dire un ensemble multiple de mots apparaissant dans le document. Le modèle de langage du document est la fonction de densité de probabilité qui génère le sac de mots dans le document (FALTINGS et SCHUMACHER, 2009). La similitude entre deux documents peut être mesurée par la similitude entre leurs modèles linguistiques correspondants (FALTINGS et SCHUMACHER, 2009). De plus, un sujet dans un ensemble de documents textuels peut être modélisé comme une distribution de probabilité sur le vocabulaire, appelé modèle de sujet (un topic model). Un document texte, qui peut porter sur un ou plusieurs sujets, peut être considéré comme un mélange de plusieurs modèles de sujets (FALTINGS et SCHUMACHER, 2009).

D'intégrant des modèles de recherche d'informations et des techniques d'exploration de données (datamining), on peut trouver les principaux sujets dans une collection de documents et, pour chaque document de la collection, les principaux sujets impliqués. De plus en plus des données textuelles et multimédias ont été accumulées et mises à disposition en ligne en raison de la croissance rapide du Web et d'applications telles que les bibliothèques numériques, et les systèmes d'informations sur les soins de santé. Leur recherche et leur analyse efficaces ont soulevé de nombreux problèmes difficiles dans l'exploration de données. Par conséquent, l'exploration de texte et l'exploration de données multimédias, intégrées aux méthodes de recherche d'informations, sont devenues de plus en plus importantes (FALTINGS et SCHUMACHER, 2009).

2.6 Big Data

Récemment, le terme de Big Data a été inventé se référant à ces défis et avantages provenant de la collecte et le traitement de grandes quantités de données . Ce thème est apparu depuis que les organisations doivent faire face à plusieurs pétaoctets de données à grande échelle. Les sources de cette énorme quantité d'informations sont les applications qui collectent des données à partir des traces de transactions, des capteurs, et d'ailleurs. Cependant, le premier problème pour la définition exacte de "Big Data" est le nom lui-même, comme on pourrait penser que c'est juste en rapport avec le volume de données (DHAENENS, 2016).

La structure hétérogène, diversifiée et dimensionnalité, variété de la représentation de données, s'est également portée sur cette question cela dépend aussi bien sûr sur le temps de calcul, c. -à-d. l'efficacité et la vitesse à la fois dans la réception et le traitement des données (DHAENENS, 2016).

2.6.1 Les 5 v de Big Data

Pour exprimer le Big Data, c'est souvent l'image des 5 V, ou plus précisément de 4 V à l'origine augmentés plus récemment d'un cinquième V.

— Volume : fait référence à la masse des données générées chaque seconde. Ce volume augmente à un rythme exponentiel, Nous ne parlons plus en Téraoctets mais en Zettabytes ou Brontobytes. Ce volume important de données est désormais trop important pour être stocké ou analysé de façon « traditionnelle », c'est-à-dire avec des bases de données. Avec le Big Data, nous pouvons stocker et utiliser ces jeux de données à l'aide de systèmes distribués dans lesquels les différentes parties des données sont stockées dans différents endroits mais rassemblées grâce à un logiciel.

— Vélocité : fait référence à la vitesse à laquelle la donnée est créée et se déplace. Pensez juste aux messages sur les réseaux sociaux qui deviennent viraux en quelques secondes, les transactions bancaires frauduleuses détectées en quelques minutes ou encore le temps que prennent les logiciels pour analyser les réseaux sociaux et capter les comportements qui déclenchent l'achat, doit des millisecondes !Et le big data se doit d'être performant pour analyser la donnée, même si elle n'est pas dans nos bases de données.

— Variété : faite référence aux différents types de données que nous pouvons utiliser. Dans le passé, nous nous sommes appuyés principalement sur des données structurées. Aujourd'hui, nous avons la possibilité d'utiliser et d'analyser une grande variété de données, Le Big Data offre la capacité de réunir toutes ces données et de les analyser.

— Véracité : faite référence à la fiabilité de la donnée. Avec autant de types de grosse donnée, la précision et la qualité sont moins vérifiables. L'une des missions du big data est d'apporter un peu d'ordre à tout cela non pas en organisant la donnée, mais plutôt en organisant son accès et en permettant d'y associer les analytiques qui correspondent aux besoins des utilisateurs. Le manque de qualité et d'exactitude résulte souvent des gros volumes.

— Valeur : c'est le dernier V à prendre en compte quand on parle de Big Data, La notion de valeur s'est très rapidement associée aux quatre précédents 'V'. Un projet big data et son accès aux utilisateurs n'a d'intérêt que s'il apporte de la valeur.

2.7 Conclusion

Dans ce chapitre, nous avons présenté le processus de l'extraction de connaissances, nous avons introduire une présentation générale de chaque étape, nous avons détaillé ensuite plus précisément l'étape de l'exploration de données (datamining), après nous avons présenté les principaux différences entre l'exploration de données et l'extraction de connaissances (KDD). Nous nous sommes ensuite intéressés à présenter une vue d'ensemble sur la recherche d'informations.
Plusieurs tâches de l'exploration de données (datamining) peuvent être considérées comme des problèmes d'optimisation combinatoire ou des approches d'optimisation, et, en particulier, les métaheuristiques sont de bons candidats pour traiter ces problèmes.

Chapitre 3

Réseaux de neurones et apprentissage profond

3.1 Introduction

Les réseaux de neurones artificiels sont des techniques d'apprentissage automatique populaires qui simulent le mécanisme d'apprentissage dans les organismes biologiques. Le système nerveux humain contient des cellules, appelées neurones. Les neurones sont connectés les uns aux autres à l'aide d'axones et de dendrites, et les régions de connexion entre les axones et les dendrites sont appelées synapses. Ce changement est la façon dont l'apprentissage se déroule dans les organismes vivants. Ce mécanisme biologique est simulé dans des réseaux de neurones artificiels, qui contiennent des unités de calcul appelées neurones (DA SILVA et al., 2017). Les unités de calcul sont connectées les unes aux autres par des poids, qui jouent le même rôle que les forces des connexions synaptiques dans les organismes biologiques. Chaque entrée d'un neurone est mise à l'échelle avec un poids, qui affecte la fonction calculée à cette unité. Un réseau de neurones artificiel calcule une fonction des entrées en propageant les valeurs calculées des neurones d'entrée aux neurones de sortie et en utilisant les poids comme paramètres intermédiaires.
Les poids entre les neurones sont ajustés dans un réseau de neurone en réponse à des erreurs de prédiction. Le but de changer les poids est de modifier la fonction calculée pour rendre les prédictions plus correctes dans les futures itérations. Par conséquent, les poids sont soigneusement modifiés d'une manière mathématiquement justifiée afin de réduire l'erreur de calcul. En ajustant successivement les poids entre les neurones sur de nombreuses paires entrée-sortie, la fonction calculée par le réseau de neurones est affinée au fil du temps afin de fournir des prédictions plus précises.
Ce chapitre, présente un aperçu général sur les réseaux de neurones, l'apprentissage profond et les réseaux de neurones profonds.

3.2 Réseau de neurones artificiels

Un réseau de neurones artificiels (ANN) est un modèle de calcul basé sur la structure et les fonctions du cerveau biologiques et du système nerveux. Les composants de calcul ou unités de traitement, appelés neurones artificiels, sont des modèles simplifiés de neurones biologiques. Ces modèles ont été inspirés par l'analyse de la façon dont une membrane cellulaire d'un neurone génère et propage des impulsions électriques (MCCULLOCH et PITTS, 1943). Les neurones artificiels utilisés dans les réseaux de neurones artificiels sont non linéaires, fournissant généralement des sorties continues et exécutant des fonctions simples, telles que la collecte des signaux disponibles sur leurs entrées, leur assemblage en fonction de leurs fonctions opérationnelles et la production d'une réponse tenant compte de leurs fonctions d'activation (MCCULLOCH et PITTS, 1943).

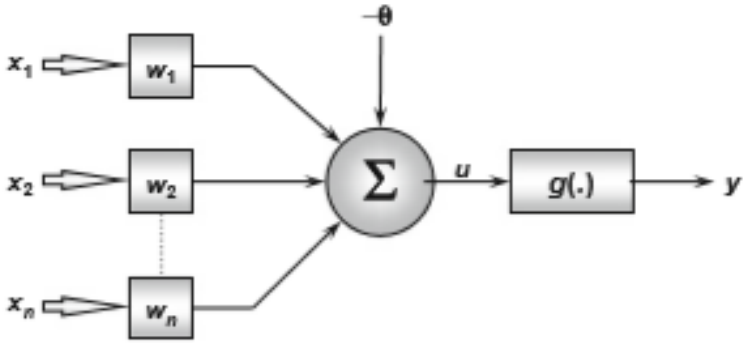

FIGURE 3.1 – Réseau artificiel (HODGKIN et HUXLEY, 1952)

Dans ce modèle, chaque neurone dans le réseau peut être mis en œuvre comme montre la figure 3.1. Les multiples signaux d'entrée provenant de l'environnement externe sont représentés par l'ensemble $\{x_1, x_2, x_3, ...x_n\}$. Les pesées effectuées par les jonctions synaptiques du réseau sont mises en œuvre sur le neurone artificial sous la forme d'un ensemble de poids synaptiques $\{w_1, w_2, w_3, ...w_n\}$. De même, la pertinence de chacune des entrées du neurone $\{x_i\}$ est calculée en les multipliant par leur poids synaptique correspondant $\{w_i\}$, ce qui permet de pondérer toutes les informations externes arrivants sur le neurone. Par conséquent, il est possible que la sortie du corps cellulaire artificial, désignée par u, soit la somme pondérée de ses entrées (HODGKIN et HUXLEY, 1952).

À partir de la figure 3.1, il est possible de voir que le neurone artificiel est composé de sept éléments de base, à savoir ;

1. Les signaux d'entrée ($\{x_1, x_2, x_3, ...x_n\}$) sont les signaux ou les échantillons provenant de l'environnement externe. Les signaux d'entrée sont généralement normalisés afin d'améliorer l'efficacité de calcul des algorithmes d'apprentissage.
2. Les poids synaptiques ($\{w_1, w_2, w_3, ...w_n\}$) sont les valeurs utilisées pour pondérer chacune des variables d'entrée, ce qui permet de quantifier leur pertinence par rapport à la fonctionnalité du neurone.
3. L'agrégateur linéaire ($\sum$) rassemble tous les signaux d'entrée pondérés par les poids synaptiques pour produire une tension d'activation.
4. Seuil d'activation ou biais (θ) est une variable utilisée pour spécifier le seuil approprié que le résultat produit par l'agrégateur linéaire devrait avoir pour générer une valeur de déclenchement vers la sortie du neurone.
5. Le potentiel d'activation (u) est le résultat produit par la différence entre l'agrégateur linéaire et le seuil d'activation. Si cette valeur est positive, c'est-à-dire si $u \geq 0$, alors le neurone produit un potentiel excitateur ; sinon, il sera inhibiteur.
6. Fonction d'activation (g) dont le but est de limiter la sortie des neurones dans une plage raisonnable de valeurs, assumée par sa propre image fonctionnelle.
7. Le signal de sortie (y) consiste à la valeur finale produite par le neurone à partir d'un ensemble particulier de signaux d'entrée, et peut également être utilisé comme entrée pour d'autres neurones interconnectés séquentiellement.

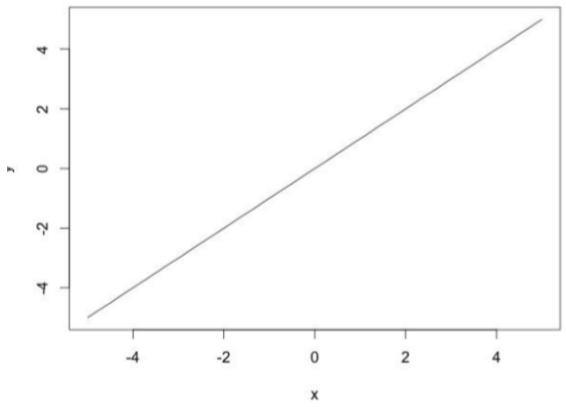

FIGURE 3.2 – Fonction d'activation linéaire

Les deux expressions suivantes synthétisent le résultat produit par le neurone artificiel proposé par (TEH et HINTON, 2001) :

$$u = \sum_{i=1}^{n} w_i \cdot x_i - \theta \tag{3.1}$$

$$y = g(u) \tag{3.2}$$

3.2.1 Fonctions d'activation

Dans la programmation de réseaux de neurones, les fonctions d'activation ou de transfert établissent des limites pour la sortie des neurones. Les réseaux de neurones peuvent utiliser de nombreuses fonctions d'activation différentes (RUMELHART, HINTON et WILLIAMS, 1988). Nous discuterons dans cette section les fonctions d'activation les plus courantes.
Le choix de la fonction d'activation est un élément essentiel dans la conception du réseau de neurones car cela peut affecter la façon dont vous devez formater les données d'entrée (RUMELHART, HINTON et WILLIAMS, 1988).

Fonction d'activation linéaire

La fonction d'activation la plus fondamentale est la fonction linéaire car elle ne modifie pas du tout la sortie des neurones (RUMELHART, HINTON et WILLIAMS, 1988). L'équation 3.3 montre comment le programme implémente généralement une fonction d'activation linéaire (RUMELHART, HINTON et WILLIAMS, 1988).

$$\phi(x) = x \tag{3.3}$$

Cette fonction d'activation renvoie simplement la valeur que les entrées du neurone lui ont transmise (RUMELHART, HINTON et WILLIAMS, 1988). La figure 3.2 montre le graphe d'une fonction d'activation linéaire.

Fonction d'activation pas ou par seuil

La fonction d'activation par seuil est une fonction d'activation simple. Les réseaux de neurones étaient à l'origine perceptrons (RUMELHART, HINTON et WILLIAMS, 1988). McCulloch (1943) a introduit le premier perceptron et a utilisé une fonction d'activation par

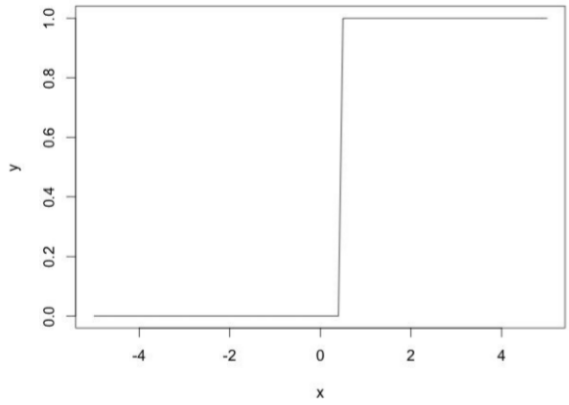

FIGURE 3.3 – Fonction d'activation par pas ou par seuil

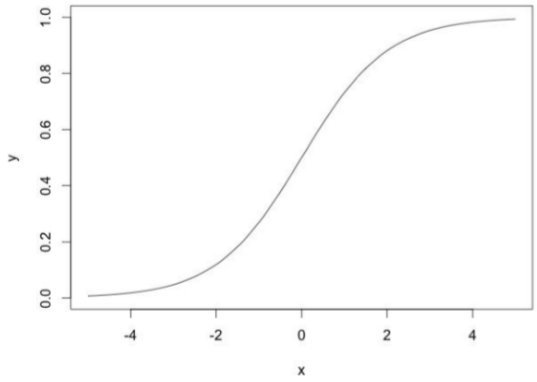

FIGURE 3.4 – Fonction d'activation sigmoïde

seuil comme montre l'équation 3.4 (TEH et HINTON, 2001).

$$(\phi(x) = \begin{cases} 1 & x \geq 0.5 \\ 0 & otherwise \end{cases}) \tag{3.4}$$

L'équation 3.4 génère une valeur de 1,0 pour les valeurs entrantes de 0,5 ou plus et 0 pour toutes les autres valeurs (RUMELHART, HINTON et WILLIAMS, 1988). Les fonctions de pas sont souvent appelées fonction de seuil car elles ne renvoient que 1 (vrai) pour les valeurs supérieures au seuil spécifié, comme le montre la figure 3.3.

Fonction d'activation sigmoïde

La fonction d'activation sigmoïde ou logistique est un choix très courant pour les réseaux de neurones à propagation avant (RUMELHART, HINTON et WILLIAMS, 1988). qui n'ont besoin de produire que des nombres positifs (RUMELHART, HINTON et WILLIAMS, 1988). L'équation 3.5 montre la fonction d'activation sigmoïde. La figure 3.4 montre le graphe de la fonction d'activation sigmoïde.

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

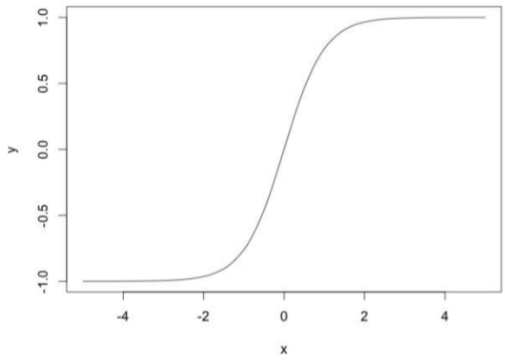

FIGURE 3.5 – Fonction d'activation de la tangente hyperbolique

Fonction d'activation de la tangente hyperbolique

La fonction de tangente hyperbolique est également une fonction d'activation très courante pour les réseaux de neurones qui doivent produire des valeurs comprises entre -1 et 1 (RUMELHART, HINTON et WILLIAMS, 1988). Cette fonction d'activation est simplement la fonction de tangente hyperbolique (tanh), comme le montre l'équation 3.6.

$$\phi(x) = \tanh(x) \tag{3.6}$$

Le graphique de la fonction tangente hyperbolique a une forme similaire à la fonction d'activation sigmoïde, comme le montre la figure 3.5.

Unités linéaires rectifiées (ReLU)

Introduite en 2000 par Teh & Hinton (HEATON, 2015), l'unité linéaire rectifiée (ReLU) a été adoptée très rapidement au cours des dernières années. La plupart des recherches actuelles recommandent désormais le ReLU en raison de résultats d'entrainement supérieurs (RUMELHART, HINTON et WILLIAMS, 1988). En conséquence, la plupart des réseaux de neurones devraient utiliser le ReLU sur les couches cachées et la fonction softmax ou linéaire sur la couche de sortie. L'équation 3.7 montre la fonction ReLU.

$$\phi(x) = \max(0, x) \tag{3.7}$$

Pourquoi ReLU fonctionne généralement mieux que d'autres fonctions d'activation pour les couches cachées. Une partie de l'augmentation des performances est due au fait que la fonction d'activation ReLU est une fonction linéaire non saturante (RUMELHART, HINTON et WILLIAMS, 1988).
La figure 3.6 montre le graphe de la fonction d'activation ReLU.

3.3 Les architectures de base des réseaux de neurones

L'architecture d'un réseau neuronal artificiel définit la façon dont ses différents neurones sont disposés ou placés les uns par rapport aux autres. Ces arrangements sont structurés essentiellement en dirigeant les connexions synaptiques des neurones (HODGKIN et HUXLEY, 1952).
Dans cette section, nous présenterons les réseaux de neurones monocouches et multicouches. Dans le réseau monocouche, un ensemble d'entrée est directement mappé à une sortie en

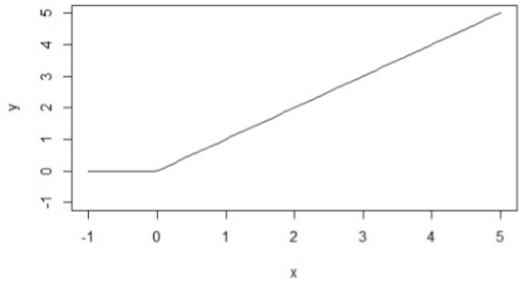

FIGURE 3.6 – Fonction d'activation ReLU

utilisant une variation généralisée d'une fonction linéaire. Cette simple instanciation d'un réseau de neurones est appelée perceptron (DA SILVA et al., 2017). Dans les réseaux de neurones multicouches, les neurones sont disposés en couches, dans lesquelles les couches d'entrée et de sortie sont séparées par un groupe de couches cachées. Cette architecture par couches du réseau de neurones est également appelée réseau de neurones à propagation avant. Cette section abordera les réseaux de neurones monocouches et multicouches (DA SILVA et al., 2017).
En général, un réseau neuronal artificiel peut être divisé en trois parties, appelées couches, qui sont connues comme :

— **Couche d'entrée :** cette couche est chargée de recevoir des informations (données), des signaux, des attributs. Ces entrées sont généralement normalisées dans les limites des valeurs produites par les fonctions d'activation. Cette normalisation se traduit par une meilleure précision numérique pour les opérations mathématiques effectuées par le réseau.
— **Couches cachées :** intermédiaires ou invisibles : ces couches sont composées de neurones qui sont chargés d'extraire les schémas associés au processus ou au système analysé. Ces couches effectuent la plupart du traitement interne à partir d'un réseau.
— **Couche de sortie :** cette couche est également composée de neurones et est donc chargée de produire et de présenter les sorties finales du réseau, qui résultent du traitement effectué par les neurones dans les couches précédentes.

Les principales architectures des réseaux de neurones artificiels, compte tenu de la disposition des neurones, ainsi que de la façon dont ils sont interconnectés et de la façon dont ses couches sont composées.
Les principales architectures des réseaux de neurones peuvent être divisées comme suit :

— réseau de neurones à propagation avant à une seule couche,
— réseau de neurones à propagation avant à plusieurs couches,
— réseaux récurrents
— réseaux maillés.

3.3.1 Réseau de neurones à propagation avant à une seule couche

L'architecture de réseau de neurones la plus simple (DA SILVA et al., 2017). Ce réseau de neurones artificiel ne comporte qu'une seule couche d'entrée et une seule couche de sortie. La figure 3.7 illustre un réseau à propagation avant à couche simple composé de n entrées et m sorties (HODGKIN et HUXLEY, 1952). Les informations circulent toujours dans une seule direction (donc unidirectionnelle), qui va de la couche d'entrée à la couche de sortie. À partir de la figure 3.7, il est possible de voir que dans les réseaux appartenant à cette architecture, le nombre de sorties de réseau coïncidera toujours avec sa quantité de

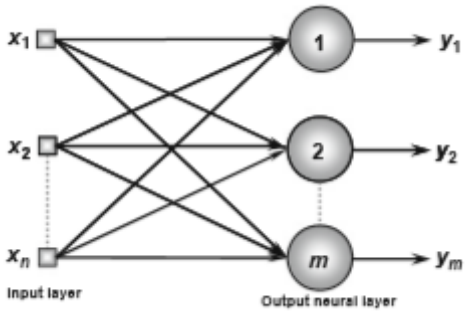

FIGURE 3.7 – Exemple de réseau de neurones à propagation avant à une seule couche (HODGKIN et HUXLEY, 1952)

neurones. Ces réseaux sont généralement utilisés dans les problèmes de classification de motifs et de filtrage linéaire (HODGKIN et HUXLEY, 1952). Parmi les principaux types de réseaux appartenant à cette architecture sont le perceptron et l'ADALINE.

3.3.2 Réseau de neurones à propagation avant multicouches

Contrairement aux réseaux appartenant à l'architecture précédente, les réseaux de neurones multicouches contiennent plusieurs couches de calcul ; les couches intermédiaires supplémentaires (entre entrée et sortie) sont appelées couches cachées car les calculs effectués ne sont pas visibles pour l'utilisateur (DA SILVA et al., 2017 ; AMINE et al., 2011). Les réseaux à propagation avant multicouches sont composés d'une ou plusieurs couches neuronales cachées (Figure 3.8). Ils sont utilisés pour résoudre divers problèmes, comme ceux liés à l'approximation des fonctions, à la classification des modèles, à l'identification du système, au contrôle des processus, à l'optimisation, à la robotique, etc. La figure 3.8 montre un à propagation avant avec plusieurs couches composées d'une couche d'entrée avec n échantillons de signaux, de deux couches cachées constituées respectivement n_1 et n_2 neurones et, enfin, d'une couche neuronale de sortie composée de m neurones représentant les valeurs de sortie respectives du problème en cours d'analyse (HODGKIN et HUXLEY, 1952). Parmi les principaux réseaux utilisant des architectures multicouches se trouvent le perceptron multicouche (MLP) et la fonction de base radiale (RBF) (HODGKIN et HUXLEY, 1952). La figure 3.8 permet de comprendre que la quantité de neurones composant la couche cachée est généralement différente du nombre de signaux composant la couche d'entrée du réseau. En fait, le nombre de couches cachées et leur nombre de neurones dépendent de la nature et de la complexité du problème cartographié par le réseau, ainsi que de la quantité et de la qualité des données disponibles sur le problème (HODGKIN et HUXLEY, 1952).

3.3.3 Réseau de neurones récurrents

Dans ces réseaux, les sorties des neurones sont utilisées comme entrées de rétroaction pour d'autres neurones. La fonction de rétroaction qualifie ces réseaux pour le traitement dynamique de l'information, ce qui signifie qu'ils peuvent être utilisés sur des systèmes variant dans le temps, tels que la prédiction de séries chronologiques, l'identification et l'optimisation du système, le contrôle de processus (HODGKIN et HUXLEY, 1952). Parmi les principaux réseaux de rétroaction, il y a réseau de neurones d'Hopfield et le perceptron avec rétroaction entre les neurones de couches distinctes, dont les algorithmes d'apprentissage utilisés dans leurs processus d'entrainement sont respectivement basés sur la minimisation

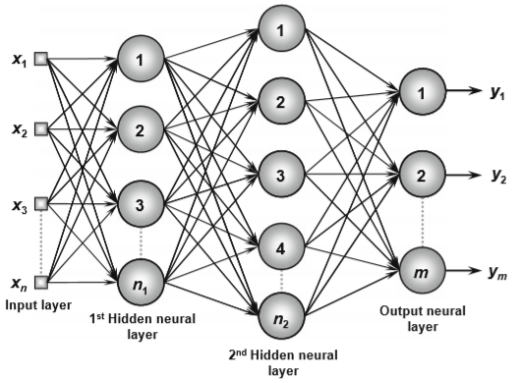

FIGURE 3.8 – Exemple de réseau de neurones à propagation avant multi-couches (HODGKIN et HUXLEY, 1952)

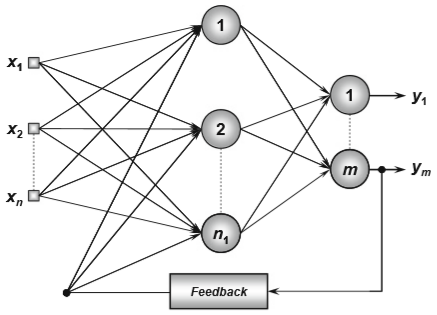

FIGURE 3.9 – Exemple de réseau de neurones récurrents (HODGKIN et HUXLEY, 1952)

des fonctions énergétiques et la règle delta généralisée (HODGKIN et HUXLEY, 1952). La figure 3.9 illustre un exemple de réseau perceptron avec rétroaction, où l'un de ses signaux de sortie est renvoyé à la couche intermédiaire. Ainsi, en utilisant le processus de rétroaction, les réseaux avec cette architecture produisent des sorties de courant en tenant également compte des valeurs de sortie précédentes.

3.3.4 Réseau de neurones d'architectures maillées

Les principales caractéristiques des réseaux à structure maillée résident dans la prise en compte de la disposition spatiale des neurones à des fins d'extraction de motifs, c'est-à-dire que la localisation spatiale des neurones est directement liée au processus d'ajustement de leurs poids et seuils synaptiques (HODGKIN et HUXLEY, 1952). Ces réseaux servent un large éventail des applications et sont utilisés dans des problèmes impliquant le regroupement de données (le clustering), la reconnaissance des formes, l'optimisation de systèmes (HODGKIN et HUXLEY, 1952).
Le réseau Kohonen est le principal réseau représentant cette architecture (HODGKIN et HUXLEY, 1952). La figure 3.10 illustre un exemple du réseau Kohonen.

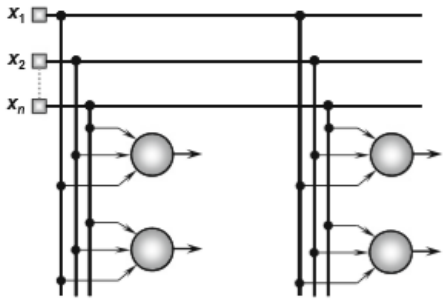

FIGURE 3.10 – Exemple de réseau de neurones d'architectures maillées (HODGKIN et HUXLEY, 1952)

3.4 L'apprentissage profond

L'apprentissage profond est une branche de l'apprentissage automatique basée sur un ensemble d'algorithmes qui tentent de modéliser des abstractions de haut niveau dans les données en utilisant plusieurs couches de traitement avec des structures complexes, ou autrement composées de multiples transformations non linéaires (AREL, ROSE et KARNOWSKI, 2010 ; MINAR et NAHER, 2018). L'apprentissage profond fait partie d'une famille plus large de méthodes d'apprentissage automatique basées sur des représentations d'apprentissage des données. Une observation (par exemple, une image) peut être représentée de nombreuses façons, comme un vecteur de valeurs d'intensité par pixel, ou par une manière plus abstraite comme un ensemble de bords, de régions de forme particulière, etc. (COŞKUN et al., 2017).
L'apprentissage profond pourrait être caractérisé comme une classe d'algorithmes d'apprentissage automatique qui utilisent une cascade de nombreuses couches d'unités de traitement non linéaires pour l'extraction et la transformation des caractéristiques (COŞKUN et al., 2017). Chaque couche utilise la sortie de la couche précédente comme entrée. Les algorithmes profonds peuvent être supervisés ou non supervisés (COŞKUN et al., 2017).
L'apprentissage automatique traditionnel repose sur des réseaux de neurones peu profonds composés d'une couche d'entrée et d'une couche de sortie, et pas plus d'une couche cachée. L'apprentissage profond est qualifié lorsqu'il existe plus de trois couches dans un réseau, y compris les couches d'entrée et de sortie (BENGIO, 2009).
Diverses architectures d'apprentissage profond telles que les réseaux de neurones profonds, les réseaux de neurones profonds convolutionnels, et les réseaux de neurones récurrents ont été appliquées à des domaines tels que la vision par ordinateur, la reconnaissance vocale, le traitement du langage naturel, la reconnaissance audio et la bio-informatique où il a été démontré qu'ils produisent des résultats très satisfaisants pour diverses tâches (COŞKUN et al., 2017).

3.4.1 Réseau de neurones profonds

Dans cette section, nous aborderons brièvement les réseaux de neurones profonds (DNN), ainsi que leurs améliorations. Les réseaux de neurones fonctionnent avec des fonctionnalités similaires au cerveau humain. Ceux-ci sont composés principalement de neurones et de connexions (GOODFELLOW et al., 2016). Lorsque nous parlons de réseau de neurones profonds, nous pouvons supposer qu'il devrait y avoir un certain nombre de couches cachées, qui peuvent être utilisées pour extraire des caractéristiques à partir des entrées et pour calculer des fonctions complexes (GOODFELLOW et al., 2016). Bengio (WITTEN et al., 2005) a

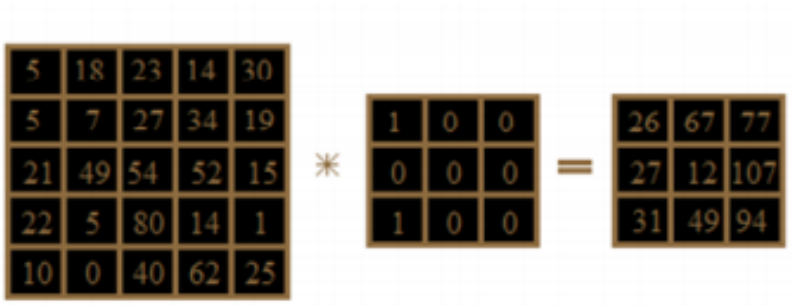

FIGURE 3.11 – Une représentation simple de l'opération de convolution en 2 dimensions (BENGIO, 2009)

expliqué les réseaux de neurones pour les architectures profondes, par ex. Réseaux de neurones convolutifs (CNN), auto-encodeurs (AE), etc. Deng et Yu (GUPTA et RAZA, 2019) ont détaillé certaines architectures de réseaux de neurones, par exemple AE et ses variantes.

Réseaux de neurones convolutifs

Le réseau de neurone convolutif (CNN), est un type spécialisé de réseau neuronal pour le traitement de données ayant une topologie connue en forme de grille (YUVARAJ et THANGARAJ, 2019). Par exemple, les exemples incluent les données de séries temporelles, qui peuvent être considérées comme une grille 1D, et les données d'image, qui peuvent être considérées comme une grille 2D de pixels. Le nom «réseau neuronal convolutif» indique que le réseau utilise une opération mathématique appelée convolution (YUVARAJ et THANGARAJ, 2019). La convolution est une opération linéaire spécialisée. Les réseaux convolutifs sont simplement des réseaux de neurones qui utilisent la convolution à la place de la multiplication matricielle générale dans au moins une de leurs couches (YUVARAJ et THANGARAJ, 2019).

Opération de convolution

La convolution n'est qu'une opération mathématique qui décrit une règle de mélange de deux fonctions et produit une troisième fonction. Cette troisième fonction est une intégrale qui exprime le degré de chevauchement d'une fonction lorsqu'elle est déplacée sur l'autre fonction. En d'autres termes, une donnée d'entrée et un noyau de convolution sont soumis à une opération mathématique particulière pour générer une carte de caractéristiques (features map) transformée (BENGIO, 2009).

La convolution est souvent interprétée comme un filtre, où le noyau filtre la carte des caractéristiques (features map) pour des informations d'un certain type (BENGIO, 2009). La convolution est décrite formellement comme suit :

$$h(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{3.8}$$

CNN fonctionne généralement avec une opération de convolution bidimensionnelle comme montre la figure 3.11. La matrice à gauche correspond aux données d'entrée. La matrice au milieu est le noyau de convolution et la matrice à droite est une carte de caractéristiques. La carte des caractéristiques (feature map) est calculée en glissant le noyau de convolution sur toute la matrice d'entrée (BENGIO, 2009).

L'opération de convolution est généralement connue sous le nom de noyaux. Par différents choix de noyaux, différentes opérations pour des images peuvent être obtenues. Les

opérations incluent généralement la détection des contours, le flou, la netteté, etc. (BENGIO, 2009).

Réseaux de neurones récurrents

Les réseaux de neurones récurrents sont des réseaux avec des connexions qui forment des cycles dirigés. En d'autres termes, sont des réseaux de neurones dans lesquels l'information peut se propager dans les deux sens. En conséquence, ils ont un état interne, ce qui en fait des candidats de choix pour résoudre les problèmes d'apprentissage impliquant des séquences de données, comme la reconnaissance de l'écriture manuscrite, la reconnaissance vocale et la traduction automatique (FARIS et al., 2018).
Les RNN sont appelés récurrents car ils effectuent la même tâche pour chaque élément d'une séquence, dans lequel les sorties de neurones sont utilisées comme rétroactions aux neurones de la couche précédente (BENGIO, 2009). En d'autres termes, la sortie courante est considérée comme une entrée pour la sortie suivante. Ces réseaux possèdent des connexions récurrentes au sens où elles conservent des informations en mémoire : ils peuvent prendre en compte à un instant t un certain nombre d'états passés (FARIS et al., 2018). Plusieurs chercheurs ont proposé des versions améliorées de réseaux de neurones récurrents par exemple le LSTM (Long Short Term Memory).
Mémoire à long terme (LSTM)
Les réseaux de longue mémoire à court terme (LSTM) sont une extension des réseaux neuronaux récurrents (RNN), qui étend leur mémoire. Les unités d'un LSTM sont utilisées comme unités de construction pour les couches d'un RNN, qui est alors souvent appelé un réseau LSTM (VAPNIK, 1999).
Les LSTM permettent aux RNN de se souvenir de leurs intrants sur une longue période de temps. C'est parce que les LSTM contiennent leurs informations dans une mémoire (VAPNIK, 1999).
Cette mémoire peut être vue comme une cellule gated, où gated signifie que la cellule décide de stocker ou de supprimer des informations, en fonction de l'importance qu'elle attribue à l'information (VAPNIK, 1999). L'attribution de l'importance se fait à travers des poids, qui sont également appris par l'algorithme. Cela signifie simplement qu'il apprend avec le temps quelle information est importante et laquelle ne l'est pas (VAPNIK, 1999).
Dans un LSTM il y a trois portes : la porte d'entrée, la porte d'oublie et la porte de sortie. Ces portes déterminent s'il faut laisser entrer une nouvelle entrée ou non (porte d'entrée), supprimer l'information car elle n'est pas importante (oublier la porte) ou la laisser influencer la sortie au pas de temps courant (porte de sortie) (VAPNIK, 1999). Depuis 1997, la recherche liée au LSTM est un domaine très actif et de nombreuses variations ont été proposées.

3.5 Formation et évaluation d'un réseau de neurones

La formation est le processus où les poids d'un réseau de neurones sont ajustés pour produire la sortie souhaitée. Le processus de formation peut ajuster les poids afin que le réseau neuronal produise une sortie utile. La plupart des algorithmes de formation du réseau neuronal commencent par initialiser les poids à un état aléatoire. La formation progresse ensuite à travers une série d'itérations qui améliorent continuellement les poids pour produire une meilleure sortie (RUMELHART, HINTON et WILLIAMS, 1988). La formation utilise l'évaluation, qui est le processus où la sortie du réseau de neurones est évaluée par rapport à la sortie attendue (RUMELHART, HINTON et WILLIAMS, 1988). Cette section couvrira l'évaluation et introduira la formation d'un réseau de neurones. Parce que les réseaux de neurones peuvent être formés et évalués de différentes manières, nous avons besoin d'une méthode

cohérente pour les juger. Une fonction objective évalue un réseau neuronal et renvoie un score. La formation ajuste le réseau neuronal de manière à obtenir de meilleurs résultats. En règle générale, la fonction objective souhaite des scores inférieurs. Le processus consistant à tenter d'obtenir des scores inférieurs est appelé minimisation. On peut établir des problèmes de maximisation, dans lesquels la fonction objective souhaite des scores plus élevés. Par conséquent, nous pouvons utiliser la plupart des algorithmes de formation pour les problèmes de minimisation ou de maximisation. On peut optimiser les poids d'un réseau de neurones avec n'importe quel algorithme d'optimisation continue, comme le recuit simulé, l'optimisation de l'essaim de particules, les algorithmes génétiques (RUMELHART, HINTON et WILLIAMS, 1988).

3.5.1 Formation par la rétropropagation du gradient

Rumelhart, Hinton et Williams (1986) (LI et al., 2012) ont introduit la rétropropagation. La rétropropagation est l'une des méthodes les plus courantes pour former un réseau de neurones. La rétropropagation est un type de descente de gradient. La descente de gradient fait référence au calcul d'un gradient sur chaque poids dans le réseau neuronal pour chaque élément d'entraînement. Si le réseau de neurones ne fournit pas la valeur attendue pour un élément d'entraînement, le gradient de chaque poids vous donnera une indication sur la manière de modifier chaque poids pour obtenir la valeur attendue. Si le réseau neuronal fournit exactement ce qui est attendu, le gradient de chaque poids sera égal à 0, ce qui signifie qu'il n'est pas nécessaire de modifier le poids (RUMELHART, HINTON et WILLIAMS, 1988).

Algorithme rétropropagation

Le processus de la rétropropagation peut être décrit comme suit :

— **Phase de propagation :** dans cette phase, les données d'une instance de formation sont introduites dans le réseau neuronal. Il en résulte une cascade de calculs à travers les couches, en utilisant l'ensemble des poids actuels. La sortie prédite peut être comparée à celle de l'instance de formation et la dérivée de la fonction de perte par rapport à la sortie est calculée. La dérivée de cette perte doit maintenant être calculée par rapport aux poids dans toutes les couches de la phase de retour (DA SILVA et al., 2017).

— **Phase de rétropropagation :** l'objectif principal de cette phase est d'apprendre le gradient de la fonction de perte (fonction objectif) par rapport aux différents poids en utilisant la règle de la chaîne de calcul de differential. Cette phase est appelé la phase inverse (DA SILVA et al., 2017).

Description de l'algorithme rétropropagation

L'idéologie guidant les règles d'apprentissage du réseau BP est la suivante : la modification de la valeur de poids et de la valeur seuil du réseau doit être effectuée le long de la direction du gradient négatif.

$$x_{k+1} = x_k - \eta_k g_k \tag{3.9}$$

Dans la formule mentionnée ci-dessus, x_k représente la matrice de la valeur de poids actuelle et de la valeur seuil ; g_k représente le gradient de la fonction actuelle ; η_k représente le taux d'apprentissage. Ici, le réseau BP à trois couches est pris comme exemple pour décrire en détail l'algorithme BP (KUBAT, 1999). Comme pour le réseau BP à trois couches, supposons que son nœud d'entrée est x_i, le nœud de la couche cachée est y_j et le nœud de la couche de sortie est z_l (KUBAT, 1999). La valeur de poids du réseau entre le nœud d'entrée et le

nœud de la couche cachée est w_{ij}, et la valeur de poids du réseau entre les nœuds de la couche cachée et la couche de sortie est v_{lj}. Lorsque la valeur attendue du nœud de sortie est t_l, $f(\bullet)$ est la fonction d'activation (KUBAT, 1999). La formule de calcul du modèle est exprimée comme suit :

Propagation vers l'avant : sortie du réseau informatique

Sortie du nœud de la couche cachée :

$$y_j = f(w_{ji}x_i - \theta_j) = f(net_j) \tag{3.10}$$

$$net_j = \sum_i w_{ji}x_i - \theta_j \tag{3.11}$$

Sortie de calcul du nœud de sortie :

$$z_l = f(\sum_j v_{lj}y_j - \theta_l) = f(net_l) \tag{3.12}$$

$$net_l = \sum_j v_{lj}y_j - \theta_l \tag{3.13}$$

Erreur du nœud de sortie :

$$E = \frac{1}{2}\sum_l (t_l - z_l)^2 = \frac{1}{2}\sum_l (t_l - f(\sum_j v_{lj}y_j - \theta_l))^2 \tag{3.14}$$

$$\frac{1}{2}\sum_l (t_l - f(\sum_j v_{lj}f(w_{ji}x_i - \theta_j) - \theta_l))^2 \tag{3.15}$$

La rétropropagation : la méthode de descente de gradient est adoptée pour réguler la valeur de poids de toutes les couches, et l'algorithme d'apprentissage de la valeur de poids est exprimé comme suit (KUBAT, 1999) :

Modification de la valeur de poids

1. Dérivation du nœud de sortie au moyen d'une fonction d'erreur :

$$\frac{\partial E}{\partial v_{lj}} = \sum_{k=1}^{n} \frac{\partial E}{\partial z_k} \cdot \frac{\partial z_k}{\partial v_{lj}} = \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial v_{lj}} \tag{3.16}$$

E est une fonction contenant plusieurs z_k, mais un seul z_l est lié à v_{lj} et tous les z_k sont indépendants les uns des autres, dans cette formule,

$$\frac{\partial E}{\partial z_l} = \frac{1}{2}\sum_k \left[-2(t_k - z_k) \cdot \frac{\partial z_k}{\partial z_l}\right] = -(t_l - z_l) \tag{3.17}$$

$$\frac{\partial z_l}{\partial v_{lj}} = \frac{\partial z_l}{\partial net_l} \cdot \frac{\partial net_l}{\partial v_{lj}} = f\prime(net_l) \cdot y_j \tag{3.18}$$

De cette façon,

$$\frac{\partial E}{\partial v_{lj}} = -(t_l - z_l) \cdot f\prime(net_l) \cdot y_j \tag{3.19}$$

Supposons que l'erreur du nœud d'entrée soit,

$$\delta_l = (t_l - z_l) \cdot f\prime(net_l) \tag{3.20}$$

De cette façon,

$$\frac{\partial E}{\partial v_{lj}} = -\delta_l \cdot y_j \tag{3.21}$$

2. Déviation du nœud de la couche cachée par fonction d'erreur :

$$\frac{\partial E}{\partial w_{ji}} = \sum_l \sum_j \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial y_j} \cdot \frac{\partial y_j}{\partial w_{ji}} \tag{3.22}$$

E est une fonction contenant plusieurs z_l ; il est ciblé sur certains w_{ji}, correspondant à un y_j, et liés à tous les z_l, dans cette formule,

$$\frac{\partial E}{\partial z_l} = \frac{1}{2} \sum_k \left[-2(t_k - z_k) \cdot \frac{\partial z_k}{\partial z_l} \right] = -(t_l - z_l) \tag{3.23}$$

De cette façon,

$$\frac{\partial E}{\partial w_{ji}} = -\sum_l (t_l - z_l) \cdot f\prime(net_l) \cdot v_{lj} \cdot f\prime(net_j) \cdot x_i = -\sum_l \delta_l v_{lj} f\prime(net_j) \cdot x_i \tag{3.24}$$

Supposons que l'erreur du nœud de la couche de cachée soit,

$$\delta\prime_j = f\prime(net_j) \cdot \sum_l \delta_l v_{lj} \tag{3.25}$$

De cette façon,

$$\frac{\partial E}{\partial w_{ji}} = -\delta\prime_j x_i \tag{3.26}$$

Comme la modification du poids Δv_{lj} et Δw_{ji} est proportionnelle aux fonctions d'erreur et descend le long du gradient, la formule montrant la modification du poids de la couche cachée et de la couche de sortie est exprimée comme suit :

$$\Delta v_{lj} = -\eta \frac{\partial E}{\partial v_{lj}} = \eta \delta_l y_j \tag{3.27}$$

Dans cette formule, η représente le taux d'apprentissage. La formule montrant la modification entre la couche d'entrée et la couche cachée est exprimée comme suit :

$$\Delta w_{ji} = -\eta\prime \frac{\partial E}{\partial w_{ji}} = \eta\prime \delta\prime_j x_i \tag{3.28}$$

$$\delta\prime_j = f\prime(net_j) \cdot \sum_l \delta_l v_{lj} \tag{3.29}$$

Dans cette formule, $\eta\prime$ représente le taux d'apprentissage ; $\sum_l \delta_l v_{lj}$ l'erreur de nœud de la couche cachée, $\delta\prime_j$ exprime l'erreur δ_l du nœud de sortie z_l est propagé à nouveau à travers la valeur des poids v_{lj} vers le nœud y_j pour devenir l'erreur de nœud de la couche cachée.

Modification de la valeur de seuil

La valeur de seuil θ est également une valeur de variation et elle doit également être modifiée pendant que la valeur de poids est modifiée ; la théorie appliquée est la même que celle utilisée pour la modification de la valeur pondérale (KUBAT, 1999).

1. Dérivation du seuil du nœud de sortie par la fonction d'erreur :

$$\frac{\partial E}{\partial \theta_l} = \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial \theta_l} \tag{3.30}$$

Dans cette formule,

$$\frac{\partial z_l}{\partial \theta_l} = \frac{\partial z_l}{\partial net_l} \cdot \frac{\partial net_l}{\partial \theta_l} = f\prime(net_l) \cdot (-1) \tag{3.31}$$

De cette façon, la formule exprimant la modification de la valeur de seuil est :

$$\Delta\theta_l = \eta \frac{\partial E}{\partial \theta_l} = \eta\delta_l \tag{3.32}$$

À savoir,

$$\theta_l(k+1) = \theta_l(k) + \Delta\theta_l = \theta_l(k) + \eta\delta_l \tag{3.33}$$

2. Dérivation du seuil de nœud de la couche cachée par la fonction d'erreur :

$$\frac{\partial E}{\partial \theta_j} = \sum_l \frac{\partial E}{\partial z_l} \cdot \frac{\partial z_l}{\partial y_j} \cdot \frac{\partial y_j}{\partial \theta_j} \tag{3.34}$$

Dans cette formule,

$$\frac{\partial z_l}{\partial \theta_l} = \frac{\partial z_l}{\partial net_l} \cdot \frac{\partial net_l}{\partial \theta_l} = f\prime(net_l) \cdot (-1) \tag{3.35}$$

De cette façon,

$$\frac{\partial E}{\partial \theta_j} = \sum_l (t_l - z_l) \cdot f\prime(net_l) \cdot v_{lj} \cdot f\prime(net_j) = \sum_l \delta_l v_{lj} f\prime(net_j) = \delta\prime_j \tag{3.36}$$

La formule exprimant la modification de la valeur seuil est :

$$\Delta\theta_j = \eta\prime \frac{\partial E}{\partial \theta_j} = \eta\prime\delta\prime_j \tag{3.37}$$

À savoir,

$$\theta_j(k+1) = \theta_j(k) + \Delta\theta_j = \theta_j(k) + \eta\prime\delta\prime_j \tag{3.38}$$

Amélioration de l'algorithme rétropropagation

Les avantages de la mise en œuvre de l'algorithme de la rétropropagation sont sa simplicité et son efficacité. Cependant, les inconvénients de cet algorithme sont par exemple : tomber dans le minimum local et sa convergence lente, en particulier dans les tâches complexes qui nécessitent un réseau massif (DENG et YU, 2014).
Les chercheurs ont proposé de nombreux algorithmes qui améliorent cet algorithme (BENUWA et al., 2016) pour résoudre ces défauts. Ses méthodes améliorées peuvent généralement être classées en trois catégories : l'une consiste à améliorer la vitesse de la formation du réseau neuronal ; la seconde est d'améliorer la précision de la formation ; et la troisième est d'éviter de tomber dans le minimum local. Parmi ces méthodes, les plus typiques sont la méthode de momentum et la méthode du taux d'apprentissage variable (KUBAT, 1999).
Méthode de Momentum
La méthode de momentum est formée en introduisant le coefficient de momentum α basé sur l'algorithme de descente de gradient (KUBAT, 1999). La formule qui montre l'ajustement

de la valeur de poids et inclut le coefficient de moment est exprimée comme suit :

$$\Delta w(t+1) = \alpha \Delta w(t) + \eta(1-\alpha)\frac{\partial E}{\partial w} \tag{3.39}$$

Dans cette formule, $\Delta w(t+1)$ et $\Delta w(t)$ représentent les corrections de poids après la $(t+1)^{th}$ et $(t)^{th}$ itération ; la valeur du coefficient de moment doit être comprise entre 0 et 1, la valeur 0.9 est généralement sélectionnée (KUBAT, 1999).

Méthode adaptative du taux d'apprentissage

La surface d'erreur du réseau varie considérablement en fonction du paramètre variable ; le taux d'apprentissage le plus élevé doit être sélectionné pour les zones dont les surfaces d'erreur sont très régulières ; le taux d'apprentissage le plus faible doit être sélectionné pour les zones dont les surfaces d'erreur sont très précipitées (KUBAT, 1999). La méthode du taux d'apprentissage variable est utilisée pour l'ajustement auto-adaptatif du taux d'apprentissage en fonction du changement d'erreur. La formule ci-dessous montre l'ajustement du taux d'apprentissage (KUBAT, 1999) :

$$\eta(t+1) = \begin{cases} k_{inc}\eta(t) & E(t+1) < E(t) \\ k_{dec}\eta(t) & E(t+1) > E(t) \\ \eta(t) \end{cases} \tag{3.40}$$

Dans cette formule, le facteur incrémental du taux d'apprentissage $k_{inc} > 1$, généralement $k_{inc} = 1.05$; le facteur de réduction du taux d'apprentissage $0 < k_{dec} < 1$, généralement $k_{dec} = 0.7$; $E(t+1)$ et $E(t)$ représentent la somme totale des erreurs quadratiques après la $(t+1)^{th}$ et la $(t)^{th}$ itération respectivement ; η représente le taux d'apprentissage. Si $E(t+1) < E(t)$, cela représente que la $(t)^{th}$ itération est efficace, alors la multiplication du facteur incrémental augmente le taux d'apprentissage ; si $E(t+1) > E(t)$, cela représente que la $(t)^{th}$ itération est inefficace, alors la multiplication du facteur de réduction pour réduire le taux d'apprentissage afin de réduire l'itération inefficace et accélérer le taux d'apprentissage du réseau (KUBAT, 1999).

3.6 Problèmes pratiques dans la formation des réseaux neuronaux

Malgré la formidable réputation des réseaux de neurones, des défis considérables demeurent en ce qui concerne la formation (l'entrainement) effective des réseaux neuronaux. Ces défis sont principalement liés à plusieurs problèmes pratiques liés à la formation, dont le plus important est le sur-ajustement (DA SILVA et al., 2017).

3.6.1 Le problème du sur-ajustement

Le problème du sur-ajustement est lié au fait que l'ajustement d'un modèle à un ensemble de données d'entraînement particulier ne garantit pas qu'il fournira de bonnes performances de prédiction sur des données de test invisibles, même si le modèle prédit parfaitement les cibles sur les données d'entraînement (DA SILVA et al., 2017). En d'autres termes, il existe toujours un écart entre les performances des données d'entraînement et de test, ce qui est particulièrement important lorsque les modèles sont complexes et que l'ensemble de données est petit (DA SILVA et al., 2017).

3.6.2 Difficultés de convergence

Une convergence suffisamment rapide du processus d'optimisation est difficile à réaliser avec des réseaux très profonds, car la profondeur entraîne une résistance accrue au processus d'entraînement en ce qui concerne la fluidité des gradients dans le réseau. Par conséquent, certaines "astuces" ont été proposées dans la littérature pour ces cas, y compris l'utilisation de réseaux résiduels (DA SILVA et al., 2017 ; KOHAVI et JOHN, 1997).

3.6.3 Défis informatiques

Un défi important dans la conception des réseaux de neurones est le temps de fonctionnement nécessaire pour former le réseau. Il n'est pas rare qu'il faille des semaines pour former des réseaux de neurones dans les domaines d'images et du texte. Ces dernières années, les progrès de la technologie matérielle, comme les unités de traitement graphique (GPU), ont contribué dans une large mesure. Les GPU sont des processeurs matériels spécialisés qui peuvent accélérer considérablement les types d'opérations couramment utilisées dans les réseaux neuronaux.

3.6.4 Optima locaux fallacieux

La fonction d'optimisation d'un réseau de neurones est hautement non linéaire, avec de nombreux optima locaux. Lorsque l'espace des paramètres est grand et qu'il existe de nombreux optima locaux, il est logique de faire un effort pour choisir de bons points d'initialisation. Un tel procédé pour améliorer l'initialisation du réseau de neurones est appelé pré-entraînement (DA SILVA et al., 2017). L'idée de base est d'utiliser une formation supervisée ou non supervisée sur des sous-réseaux peu profonds du réseau d'origine afin de créer les poids initiaux. Ce type de pré-entrainement est effectué de manière gourmande et par couches dans laquelle une seule couche du réseau est formée à la fois afin d'apprendre les points d'initialisation de cette couche (DA SILVA et al., 2017). Ce type d'approche fournit des points d'initialisation qui ignorent des parties radicalement non pertinentes de l'espace des paramètres. De plus, un pré-entraînement non supervisé a souvent tendance à éviter les problèmes associés au sur-ajustement (DA SILVA et al., 2017). L'idée de base ici est que certains des minima de la fonction de perte sont des optima faux car ils ne sont présentés que dans les données d'apprentissage et non pas dans les données de test. L'utilisation d'un pré-entraînement non supervisé tend à rapprocher le point d'initialisation du bassin des «bons» optima dans les données de test. C'est un problème associé à la généralisation du modèle (DA SILVA et al., 2017).

3.7 Conclusion

Les réseaux neuronaux artificiels (ANN) sont largement utilisés pour la classification supervisée et non supervisée et constituent une alternative aux autres méthodes de classification (ZHANG, 2000).
Dans ce chapitre, nous avons présenté un aperçu sur les réseaux de neurones et les réseaux de neurones profonds.
Depuis le début des réseaux de neurones, ces méthodes sont utilisées dans plusieurs domaines et plusieurs optimisations sont proposées dans la littérature pour améliorer leurs fonctionnements.
L'optimisation des ANN peut porter sur plusieurs aspects. Elle peut concerner soit l'optimisation de la topologie des réseaux de neurones, soit dans leur formation ou entrainement. Les algorithmes métaheuristiques ont été largement utilisés pour traiter ces deux aspects.

Chapitre 4

Métaheuristiques pour la classification supervisée et la sélection d'attributs

4.1 Introduction

L'une des tâches les plus importantes de l'exploration de données (datamining) est la classification supervisée. Une telle tâche prend en entrée une collection d'observations (ou objets), chacune appartenant à un petit nombre de classes et décrite par ses valeurs pour un ensemble fixe d'attributs (également appelés variables ou caractéristiques) (DHAENENS, 2016). L'objectif est de construire un classificateur qui peut prédire avec précision la classe à laquelle appartient une nouvelle observation. Par conséquent, le but de cette tâche d'exploration de données est de créer un modèle qui prédit la valeur de la classe à partir des valeurs connues des autres variables (DHAENENS, 2016).
Dans la classification, toutes les variables qui sont stockées dans la base de données ne sont pas toutes nécessaires pour une discrimination précise. Les inclure dans la classification peut même réduire les performances du modèle. La sélection d'attributs, également appelée sélection de sous-ensembles de variables, vise à sélectionner un ensemble optimal d'attributs pertinents et nécessaires à la classification. Une sélection d'attributs appropriée peut améliorer l'efficacité d'un modèle d'inférence. Dans (NDIAYE et al., 2014), les auteurs indiquent que les effets de la sélection d'attributs sont les suivants : 1) améliorer les performances ; 2) visualiser les données pour la sélection du modèle ; 3) réduire la dimensionnalité et supprimer le bruit.
La classification supervisée et l'optimisation mathématique ont des liens étroits (NARENDRA et FUKUNAGA, 1977). Ce chapitre fournit d'abord une description de la tâche de la classification et une présentation des méthodes de classification standard. Ensuite, il aborde l'utilisation des métaheuristiques pour optimiser ces méthodes de classification et nous proposons ensuite de montrer comment la sélection d'attributs peut être réalisée avec les métaheuristiques.
Dans ce chapitre, on se concentre sur deux méthodes de classification à savoir les réseaux de neurones et les machines à vecteurs de support (SVM).

4.2 Métaheuristiques pour la classification supervisée

4.2.1 Description du problème

L'objectif de cette tâche d'exploration de données est de construire un modèle qui prédit la valeur d'une variable, appelée "classe", à partir des valeurs connues d'autres variables. Le modèle est construit à partir d'observations connues, puis, pour les nouvelles observations, le modèle est appliqué pour déterminer la valeur de la variable cible. Dans sa forme de base, la variable prédite (la classe) est catégorielle. En outre, lorsque la variable prédite est

	Predicted	
	Negative	Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

FIGURE 4.1 – La matrice de confusion

numérique, la tâche devient une régression (DHAENENS, 2016).
Plusieurs approches standard ont été proposées pour traiter de la classification supervisée.

4.2.2 Modèle d'optimisation

Un problème combinatoire

La représentation descriptive des solutions du problème de classification est directement liée à la manière dont le modèle de classification est exprimé. Il n'est donc pas possible ici de donner une manière générale de représenter la solution. Au contraire, quel que soit le modèle de classification utilisé, les mesures de qualité, qui indiquent la capacité du modèle à bien classer les nouvelles observations, sont les mêmes (DHAENENS, 2016).

Mesures de qualité

Les mesures de qualité visent à évaluer les performances d'un classificateur. Plusieurs mesures de qualité ont été proposées dans la littérature. Cette partie présente les plus utilisées.
Matrice de confusion
Un grand nombre de mesures de qualité sont basées sur la matrice de confusion. La figure 4.1 présente une telle matrice pour un cas de classification binaire (deux classes). Les deux classes sont souvent appelées négatives et positives. Les lignes présentent la classe réelle, tandis que les colonnes présentent la classe prédite par l'algorithme de classification. La matrice de confusion contient quatre valeurs. TP et TN représentent le nombre d'observations de l'ensemble de test qui sont correctement classées, tandis que FN et FP représentent les observations mal classées. Notons N comme le nombre total d'observations.
Mesures de performance
Exactitude :l'exactitude mesure le ratio des observations bien classées

$$Accuracy = \frac{TP + TN}{N} \tag{4.1}$$

Précision : la précision mesure le ratio des observations positives bien classées par rapport à toutes les observations positives prévues

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

Sensibilité :la sensibilité (ou rappel) mesure le rapport entre les observations positives bien classées et toutes les observations positives réelles

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.3}$$

Spécificité :la spécificité mesure le rapport entre les observations négatives bien classées et l'ensemble des observations négatives réelles

$$Specificity = \frac{TN}{TN + FP} \tag{4.4}$$

F-mesure : calcule une moyenne harmonique entre la précision et le rappel

$$F - measure = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)} \tag{4.5}$$

Les mesures de performances les plus utilisées avec les jeux de données déséquilibrés sont les suivants :

Le coefficient de corrélation de Matthews (MCC)

Le coefficient de corrélation de Matthews (MCC) est utilisé dans l'apprentissage automatique comme une mesure de la qualité des classifications binaires (à deux classes). Il est défini en termes de vrai positif (TP), vrai négatif (TN), faux positif (FP) et faux négatif (FN). En général, il est considéré comme une mesure équilibrée qui peut être utilisée dans des ensembles de données déséquilibrés. Il peut être également écrit en termes de TP, γ et π comme suit (BOUGHORBEL, JARRAY et EL-ANBARI, 2017) :

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{((TP + FP)(FP + FN)(TN + FP)(TN + FN))}} = \frac{TP - \gamma\pi}{\sqrt{\gamma(1-\gamma)\pi(1-\pi)}} \tag{4.6}$$

$$\pi = P = (Y = 1) \ \ and \ \ \gamma(\theta) = P(\theta = 1) \tag{4.7}$$

Il est important de mentionner que π correspond à la classe minoritaire si la petite classe est considérée comme ayant l'étiquette 1. Le MCC prend des valeurs dans l'intervalle [-1, 1]. Un coefficient de +1 représente une prédiction parfaite, - 1 une prédiction inverse et 0 est une prédiction aléatoire moyenne.

G-Moyenne :le score moyen géométrique correspond à la moyenne géométrique de la sensibilité et de la spécificité est principalement utilisé dans les problèmes déséquilibrés. La moyenne géométrique essaie de maximiser la précision sur chacune des classes tout en gardant ces précisions équilibrées.

$$G - moyenne = \sqrt{(Sens \times Spec)} \tag{4.8}$$

Le choix de la mesure de qualité est fortement corrélé au contexte de l'application et peut-être considéré comme la fonction objective par une méthode d'optimisation utilisée pour traiter la classification.

4.2.3 Métaheuristiques pour optimiser des algorithmes de classification

Optimisation des réseaux de neurones artificiels (ANN)

L'optimisation des réseaux de neurones artificiels peut porter sur plusieurs aspects. Elle peut concerner soit l'optimisation de la topologie des réseaux de neurones, soit la formation du réseau de neurones. Plusieurs métaheuristiques ont été proposées dans la littérature pour traiter ces deux aspects.

Les métaheuristiques présentent de nombreux avantages : elles s'appliquent à tout type de réseau de neurones avec toute fonction d'activation (PALMES, HAYASAKA et USUI, 2005), fournissent des solutions acceptables dans un délai raisonnable pour résoudre des

problèmes complexes et difficiles (BLUM et al., 2011), et sont particulièrement utiles pour traiter de grands problèmes complexes qui génèrent de nombreux optima locaux (MARTI et EL-FALLAHI, 2004 ; HAMM, BRORSEN et HAGAN, 2002). Les métaheuristiques peuvent être divisées en deux catégories : les algorithmes basés sur une seule solution (NANDY, SARKAR et DAS, 2012) et les algorithmes basés sur la population (NAWI, REHMAN et KHAN, 2014a). Pour les algorithmes basés sur une solution unique, certaines études ont utilisé la recherche taboue et le recuit simulé (SHAW et KINSNER, 1996 ; SEXTON et al., 1998) pour la formation de réseaux de neurones à propagation avant. Les algorithmes basés sur la population peuvent être divisés en deux groupes : les algorithmes d'intelligence en essaim et les algorithmes évolutionnaires (FEDOROVICI et al., 2012). Pour les algorithmes évolutionnaires, P.P. Palms (PALMES, HAYASAKA et USUI, 2005) a utilisé des algorithmes génétiques (GA). Les auteurs ont montré que les GA surpassent les BP. Pour les algorithmes d'intelligence en essaim, plusieurs auteurs ont proposé des algorithmes basés sur les essaims comme méthodes de formation de réseau de neurones, par exemple, l'optimisation des essaims de particules (PSO) (CHEN et al., 2007), l'optimisation des colonies de fourmis (ACO) (NANDY, SARKAR et DAS, 2012), l'algorithme de chauve-souris (BA) (NAWI, REHMAN et KHAN, 2014a), l'algorithme de chauve-souris modifié (JADDI, ABDULLAH et HAMDAN, 2015), l'algorithme d'optimisation des baleines (WOA) (FARIS, ALJARAH et MIRJALILI, 2018) etc....

Dans (ALJARAH, FARIS et MIRJALILI, 2018a) un algorithme de formation de réseau de neurones basé sur l'algorithme d'optimisation des baleines (WOA) a été proposé. Il a été prouvé que cet algorithme est capable de résoudre un large éventail de problèmes d'optimisation et de surpasser les algorithmes actuels. Pour la première fois dans la littérature, 20 ensembles de données présentant différents niveaux de difficulté ont été choisi pour tester le formateur proposé (WOA). Les résultats sont vérifiés par des comparaisons avec l'algorithme de rétropropagation et six techniques évolutives. Les résultats qualitatifs et quantitatifs ont prouvé que ce formateur est capable de surpasser les autres algorithmes sur la majorité des ensembles de données en termes d'évitement des optima locaux et de vitesse de convergence.

Dans (FARIS, ALJARAH et MIRJALILI, 2016a) l'algorithme "multi-verse optimizer" (MVO) est utilisé pour la formation du réseau de neurones a perceptron multicouches (MLP). Cette approche de formation est comparée et évaluée à l'aide de neuf ensembles de données biomédicales. Les résultats sont comparés à cinq algorithmes métaheuristiques évolutionnaires classiques et récents : algorithme génétique (GA), optimisation de l'essaim de particules (PSO), évolution différentielle (DE), algorithme des lucioles (FF) et la recherche du coucou (CS). De plus, les résultats sont comparés à deux méthodes classiques d'entraînement basées sur le gradient : les algorithmes conventionnels de rétropropagation (BP) et de Levenberg-Marquardt (LM). L'étude comparative a démontré que MVO est très compétitif et surpasse les autres algorithmes de formation dans la majorité des ensembles de données en termes d'évitement d'optima locaux et la vitesse de convergence.

(ALJARAH et al., 2018) propose un nouveau mécanisme d'entrainement pour les réseaux de neurones à fonctions de base radiale basé sur "biogeography-based optimiser (BBO)". Pour prouver l'efficacité de cette méthode, 12 ensembles de données bien connus sont utilisés et l'algorithme est comparé à 11 algorithmes de formation utilisés dans la littérature, y compris les approches basées sur les gradients et les approches stochastiques. L'article envisage de modifier le nombre de neurones et d'étudier les performances des réseaux de fonctions de base radiales avec un nombre différent de paramètres. Un test statistique est également effectué pour juger la signification des résultats. Les résultats ont montré que ce formateur est capable de surpasser considérablement les autres algorithmes

d'entraînement sur tous les ensembles de données en termes de précision de classification, de vitesse de convergence et d'évitement d'optima locaux. De plus, la comparaison des entraîneurs sur des réseaux de fonctions de base radiales avec différentes tailles de neurones a révélé que ce formateur est capable d'entraîner efficacement des réseaux de neurones à fonctions de base radiales avec un nombre différent de paramètres.

Un nouvel algorithme d'optimisation appelé "Moth–Flame Optimizer" (MFO) est proposé dans (FARIS, ALJARAH et MIRJALILI, 2017) pour entrainer les réseaux de neurones à fonctions de base radiales. Sept ensembles de données standard sont utilisées pour tester ce formateur. L'entraîneur basé sur le MFO est comparé à l'algorithme des essaims de particules (PSO), l'algorithme génétique (GA), l'algorithme des chauves-souris (BA). Les résultats ont montré que ce formateur a donné des résultats supérieurs dans la majorité des cas. L'observation du comportement de convergence prouve que cet entraîneur bénéficie également de l'accélération de la vitesse de convergence.

Un nouvel algorithme d'optimisation "Improved monarch butterfly optimization" (IMBO) a été proposé par Faris et al. (FARIS, ALJARAH et MIRJALILI, 2018) afin de remédier aux inconvénients de l'algorithme "Monarch butterfly optimization" (MBO) récemment proposé. Pour prouver l'efficacité d'IMBO, un ensemble de 23 fonctions de test est utilisée. Les résultats statistiques montrent qu'IMBO bénéficie d'un évitement élevé des optima locaux et d'une vitesse de convergence rapide, ce qui aide cet algorithme à surpasser le MBO de base et une autre variante récente de cet algorithme (GCMBO). Les résultats de l'algorithme proposé sont comparés à neuf autres approches pour la vérification. L'analyse comparative montre qu'IMBO fournit des résultats très compétitifs et à tendance à surpasser les autres algorithmes. Pour démontrer l'applicabilité de l'IMBO à la résolution de problèmes pratiques difficiles, il est également utilisé pour entraîner les réseaux de neurones. Le formateur basé sur IMBO est testé sur 15 ensembles de données de classification populaires obtenus auprès du référentiel d'apprentissage automatique de l'Université de Californie à Irvine (UCI).). Les résultats sont comparés à une variété de techniques dans la littérature, y compris le MBO original et le GCMBO. Les expérimentations ont montré qu'IMBO améliore considérablement l'apprentissage des réseaux de neurones, prouvant les mérites de cet algorithme pour résoudre des problèmes difficiles.

Dans les réseaux de neurones, trouver simultanément des valeurs optimales pour le nombre de neurones cachés et les poids de connexion est considéré comme une tâche difficile. En effet, la modification des neurones cachés a un impact substantiel sur la structure entière d'un réseau de neurones et augmente la complexité du processus d'entraînement qui nécessite des considérations spéciales. En fait, le nombre de variables change proportionnellement au nombre de nœuds cachés lors de la formation des réseaux de neurones. En tant que l'une des tentatives fondamentales, un schéma de codage hybride est proposé Dans (FARIS, MIRJALILI et ALJARAH, 2019a) pour faire face aux défis susmentionnés. Un ensemble d'algorithmes basés sur la population stochastiques récents et bien considérés est ensuite utilisé pour optimiser le nombre de neurones cachés et les poids de connexion dans un réseau de neurones à propagation avant avec une seule couche cachée. Dans les expériences, vingt-trois ensembles de données standard de classification sont utilisés pour évaluer la technique proposée qualitativement et quantitativement. Les résultats montrent que le schéma de codage hybride permet aux algorithmes d'optimisation de trouver facilement les valeurs optimales pour le nombre de nœuds cachés et les poids de connexion. De plus, l'optimiseur de loup gris "Grey – Wolf Optimizer " (GWO) a surpassé les autres algorithmes.

Dans (TARKHANEH et SHEN, 2019), Tarkhaneh et al. ont proposé un nouvel algorithme de

formation évolutif appelé LPSONS, qui combine les opérateurs de vitesse dans l'optimisation des essaims de particules (PSO) avec la distribution de Mantegna Lévy pour produire des solutions plus diverses en divisant la population et la génération entre différentes sections de l'algorithme. Il combine en outre la recherche du voisinage avec la distribution de Mantegna Lévy pour atténuer la convergence prématurée et éviter les minima locaux. Cet algorithme peut trouver des résultats optimaux et en même temps éviter la stagnation dans les solutions optimales locales ainsi que la convergence prématurée dans la formation des réseaux de neurones a propagation avant. Les expériences menées avec quatorze ensembles de données standard provenant du référentiel d'apprentissage automatique de l'UCI confirment que l'algorithme LPSONS a surpassé largement une approche basée sur les gradients ainsi que certains algorithmes évolutifs bien connus qui sont également des améliorations de PSO.
Dans (KHAN et al., 2019) une nouvelle méthode appelée algorithme hybride (HACPSO), basée sur deux algorithmes métaheuristiques APSO et CS est proposé pour entrainer les réseaux de neurones à propagation avant. Dans cet algorithme (HACPSO), l'APSO permet de communiquer pour rechercher le meilleur endroit ayant le meilleur nid avec une plus grande capacité de survie pour les coucous. Différentes simulations ont été réalisées sur des ensembles de données standard et l'efficacité de l'algorithme proposé est comparée à CS, l'optimisation des colonies de fourmis (ACO) et d'autres variantes hybrides similaires. Les résultats de la simulation montrent que l'algorithme HACPSO est plus performant par rapport aux autres algorithmes en termes de précision, MSE, STD avec un taux de convergence rapide.

Rashid et al. (RASHID, ABBAS et TUREL, 2019) Ont proposé un réseau de neurones récurrents modifié avec un optimiseur de loup gris modifié pour identifier les faiblesses des étudiants universitaires. L'optimiseur de loup gris modifié est utilisé pour optimiser le réseau de neurones récurrent. Les résultats ont montré que le réseau de neurones récurrents modifié avec un optimiseur de loup gris modifié à la meilleure précision par rapport à d'autres modèles.

L'auteur dans (FEDOROVICI et al., 2012) a incorporé "Gravitational Search Algorithm" GSA dans ConvNet pour améliorer ses performances et éviter d'être coincé dans les minima locaux. Le GSA est utilisé pour la formation du ConvNet en conjonction avec l'algorithme de rétropropagation (GSA-ConvNet). Le GSA-ConvNet est évalué sur l'application OCR. Le GSA-ConvNet améliore les performances du ConvNet conventionnel.

L'EA "Evolutionary Algorithm" est utilisé pour optimiser les paramètres du DBN "Deep Belief Network". L'auteur de (ZHANG et al., 2018a) a appliqué l'EA adaptative dans le DBN pour régler automatiquement les paramètres sans avoir besoin de connaissances préalables sur le DBN. L'EADBN est évalué à la fois sur des ensemble de données de référence et du monde réel. Le résultat de l'évaluation montre que l'EADBN améliore les performances des variantes standard du DBN.

Optimisation de machine à vecteurs de support (SVM)

SVM est un modèle proposé et développé par Vapnik (KIRA et RENDELL, 1992). SVM est l'un des types d'algorithmes d'apprentissage basé sur la théorie de l'apprentissage statistique (HE et al., 2016). SVM peut être utilisé pour les tâches de classification et de régression. Récemment, dans une étude comparative approfondie (CARRIZOSA et MORALES, 2013), il a été montré que SVM figure parmi les meilleurs classificateurs (LIU et MOTODA, 2007).

Comme dans la majorité des classificateurs, SVM dépend du processus de formation pour construire son modèle. En utilisant l'astuce du noyau, SVM transforme les données d'apprentissage par des fonctions de mappage non linéaires en un espace dimensionnel supérieur où les données peuvent être séparées linéairement, ou pour trouver les meilleurs hyperplans (vecteurs de support) avec une marge maximale normalisée par rapport aux points de données. Par conséquent, le but du processus d'apprentissage de SVM est de rechercher les hyperplans linéaires optimaux dans cette dimension (LIU et MOTODA, 2007). Bien que l'algorithme d'apprentissage SVM ait généralement de bonnes performances et une base statistique robuste, la qualité d'un SVM est largement influencée par les paramètres des fonctions du noyau. Les paramètres de la fonction noyau, ainsi que le paramètre de régularisation, sont les hyper-paramètres du SVM. En pratique, la méthode standard pour déterminer les hyper-paramètres est la recherche par grille. Dans une simple recherche de grille, les hyper-paramètres varient avec une taille de pas fixe à travers une large gamme de valeurs et les performances de chaque combinaison sont mesurées. En raison de sa complexité de calcul, des approches d'optimisation ont été proposées pour traiter la sélection du modèle SVM (sélection d'hyper-paramètres) et en particulier les approches basées sur les métaheuristiques évolutionnaires (DHAENENS, 2016).

Dans la littérature, les algorithmes métaheuristiques ont montré une grande efficacité dans la génération de solutions acceptables lorsque le problème est très complexe et l'espace de recherche est extrêmement large. Dans (THARWAT, MOEMEN et HASSANIEN, 2017) L'algorithme d'optimisation des baleines (WOA) a été proposé pour optimiser les paramètres de SVM, afin que l'erreur de classification puisse être réduite. Les résultats expérimentaux ont prouvé que le modèle proposé atteignait une sensibilité élevée.

Dans (THARWAT et GABEL, 2019) les auteurs ont proposé un algorithme d'optimisation SSD (social ski driver) pour optimiser les paramètres des SVM, dans le but d'améliorer les performances de classification. Dans cette étude, huit ensembles de données déséquilibrés ont été utilisés pour tester l'algorithme proposé. Pour la vérification, les résultats de l'algorithme SSD-SVM sont comparés à la recherche de grille, et à l'optimisation de l'essaim de particules (PSO). Les résultats expérimentaux montrent que l'algorithme SSD-SVM est capable de trouver des valeurs quasi optimales pour les paramètres de SVM. Les résultats ont également démontré des performances de classification élevées par rapport à l'algorithme PSO.

Dans (THARWAT, HASSANIEN et ELNAGHI, 2017), l'algorithme des chauves-souris (BA) a été proposé pour optimiser les paramètres de SVM, de sorte que l'erreur de classification puisse être réduite. Pour évaluer le modèle proposé (BA-SVM), l'expérience a adopté neuf ensembles de données standard. Pour la vérification, les résultats de l'algorithme BA-SVM sont comparés à la recherche de grille, et à deux algorithmes d'optimisation : l'algorithme génétique (GA) et l'optimisation des essaims de particules (PSO). Les résultats expérimentaux ont prouvé que le modèle proposé est capable de trouver les valeurs optimales pour les paramètres de SVM et évite le problème des optima locaux. Les résultats ont également démontré des taux d'erreur de classification inférieures par rapport aux algorithmes PSO et GA.

Une étude suggère qu'une technique d'optimisation moins complexe, telle que la recherche aléatoire (RS), pourrait être suffisante pour optimiser les paramètres de SVM (MANTOVANI et al., 2015). Les expériences ont utilisé un grand nombre d'ensemble de données, et ils ont comparé RS avec trois métaheuristiques couramment utilisées pour le réglage des paramètres du SVM, à savoir GA, PSO et EDA, et avec la recherche de

grille (GS). Il a été conclu que, selon les tests, toutes les techniques de réglage ont trouvé de meilleures valeurs des paramètres que les valeurs par défaut utilisées pour le SVM. Cependant, aucune de ces techniques ne montre de meilleures performances en général ; par conséquent, l'utilisation d'une technique simple peut, dans certains cas, être un bon compromis.

Dans (THARWAT, GABEL et HASSANIEN, 2017), les auteurs proposent une approche basée sur l'algorithme Dragonfy (DA) (DA-SVM) qui permet d'optimiser les paramètres du SVM pour améliorer la précision de la classification. Différentes expériences ont été menées pour comparer l'algorithme DA-SVM avec PSO+SVM, et GA+SVM appliquer sur de nombreux ensemble de données de classification. Les résultats ont montré que l'algorithme DA-SVM a obtenu des résultats compétitifs.

Un nouvel algorithme d'optimisation (fourmilion chaotique optimisation) (CALO) a été proposé par (THARWAT et HASSANIEN, 2018) pour optimiser les paramètres du classificateur SVM, de sorte que l'erreur de classification puisse être réduite. Pour évaluer l'algorithme (CALO-SVM), l'expérience a adopté six ensembles de données. Pour la vérification, les résultats de l'algorithme CALO-SVM sont comparés à la recherche de grille, optimisation de fourmilion (ALO-SVM), et trois algorithmes d'optimisation bien connus : l'algorithme génétique (GA), l'optimisation de l'essaim de particules (PSO) et l'algorithme d'optimisation émotionnelle sociale (SEOA). Les résultats expérimentaux ont prouvé que CALO est capable de trouver les valeurs optimales des paramètres SVM et évite le problème des optima locaux. Les résultats ont également démontré des taux d'erreur de classification inférieurs par rapport aux algorithmes : GA, PSO et SEOA.

Les travaux précédents ont proposé différentes techniques pour optimiser les paramètres du SVM et effectuer la sélection des caractéristiques (attributs) simultanément. L'une des premières tentatives a été faite par Huang et Wang (HUANG et WANG, 2006a), dans laquelle l'algorithme génétique GA a été appliqué à ce problème. Les comparaisons avec l'algorithme conventionnel de recherche de grille testé sur différents ensembles de données, ils ont montré que GA est capable d'optimiser le SVM pour atteindre une meilleure précision avec un nombre réduit de caractéristiques (attributs).

Une approche similaire a été proposée par Lin et al. (LIN et al., 2008) où l'algorithme PSO a été utilisé au lieu de GA. Ils ont comparé leurs résultats à ceux obtenus par Huang et Wang (HUANG et WANG, 2006a). Leurs résultats ont montré que PSO était très compétitif par rapport à GA et le surpassant dans six ensembles de données sur dix.

Dans (FARIS et al., 2018), Faris et al. ont proposé une approche robuste basée sur une métaheuristique récente inspirée de la nature appelée "multi-verse optimizer" (MVO) pour sélectionner les attributs optimals et optimiser les paramètres de SVM simultanément. En fait, l'algorithme MVO est utilisé pour optimiser les principaux paramètres de SVM et pour trouver l'ensemble optimal d'attributs pour ce classificateur. L'approche est mise en œuvre et testée sur deux architectures différentes. MVO est comparé à quatre algorithmes métaheuristiques classiques et récents sur dix ensembles de données binaires et multiclasses. Les résultats expérimentaux démontrent que MVO peut réduire efficacement le nombre d'attributs tout en maintenant une grande précision de prédiction.

Afin de classer les défauts électriques dans les systèmes de distribution radiale, un classificateur de machine à vecteurs de support (SVM) basé sur l'optimisation des essaims de particules (PSO) a été proposé par (CHO et HOANG, 2017). Le PSO est capable de

sélectionner les attributs appropriés et d'optimiser les paramètres SVM pour augmenter la précision de la classification. La technique (PSO-SVM) a été testée sur un réseau de distribution radiale typique pour identifier dix types de défauts différents. Le taux de réussite du classificateur SVM est supérieur à 97%, a démontré l'efficacité et la grande efficience de cette méthode.

4.3 Métaheuristiques pour la sélection d'attributs

4.3.1 Description du problème

Il existe trois approches classiques pour réaliser la sélection des caractéristiques dans la classification : approche filtre, approche d'enveloppe et approche embarquée.

Méthodes de filtres (filter methods)

Les modèles de filtres évaluent les attributs dans un ensemble de données sans utiliser un algorithme de classification (THARWAT, MOEMEN et HASSANIEN, 2017). Un algorithme de filtrage comprend deux étapes. Dans la première étape, il classe les attributs en fonction de certains critères. L'évaluation des attributs peut être soit univariée, soit multivariée. Dans le schéma univarié, chaque attribut est classé indépendamment de l'espace des attributs, tandis que le schéma multivarié évalue les attributs par lots.
Les deux phases de la méthode de filtrage sont les suivantes (NDIAYE et al., 2014) :

1. La sélection des attributs à l'aide de mesures telles que la distance, la dépendance ou la cohérence ; aucun classificateur n'est utilisé dans cette phase
2. Un classificateur est appris sur les données d'entrainement avec les attributs sélectionnés et testé sur les données de test.

Méthodes enveloppes (wrapper methods)

Les modèles d'enveloppe utilisent un classificateur spécifique pour évaluer la qualité d'attributs sélectionnés et offrent un moyen simple et puissant de résoudre le problème de la sélection des caractéristiques (THARWAT et GABEL, 2019). Un modèle d'enveloppe se compose de deux phases (DHAENENS, 2016) :

— **Phase 1 :** sélection du sous-ensemble d'attributs, qui sélectionne le meilleur sous-ensemble en utilisant comme critère la précision du classificateur (sur les données d'entrainement) ;
— **Phase 2 :** apprentissage et test, où un classificateur est appris à partir des données d'entrainement avec le meilleur sous-ensemble d'attributs et est testé sur les données de test.

Étant donné un classificateur prédéfini, un modèle d'enveloppe effectuera les étapes suivantes :

1. Recherche d'un sous-ensemble d'attributs ;
2. Évaluation du sous-ensemble d'attributs sélectionné en fonction des performances du classificateur ;
3. Répéter les étapes 1 et 2 jusqu'à ce que la qualité souhaitée soit atteinte.

Méthodes embarquées (embedded methods)

Les méthodes embarquées sont similaires aux méthodes d'enveloppe en ce sens que la recherche d'un sous-ensemble optimal est effectuée pour un algorithme de classification

spécifique, mais elles sont caractérisées par une interaction plus profonde entre la sélection d'attributs et la construction du classificateur. La différence entre Les méthodes embarquées et Les méthodes d'enveloppe ; avec les méthodes enveloppes l'algorithme de classification sert non seulement à évaluer un sous-ensemble candidats mais aussi à guider le mécanisme de sélection (THARWAT, HASSANIEN et ELNAGHI, 2017).

4.3.2 Modèle d'optimisation

Un problème combinatoire

Le problème de sélection d'attributs est facile à représenter comme un problème d'optimisation combinatoire où l'objectif est de sélectionner un sous-ensemble d'attributs pour lequel un critère d'évaluation du sous-ensemble d'attributs est optimisé. Les valeurs binaires des variables x_i sont utilisées pour indiquer la présence $(x_i = 1)$ ou l'absence $(x_i = 0)$ de l'attribut i dans l'ensemble d'attributs optimaux. Ensuite, le problème est formulé comme suit (DHAENENS, 2016) :

$$max_x =_{(x_1,\ldots,x_n)\in\{0;1\}^n} F(x) \tag{4.9}$$

Il a été démontré que le problème de sélection de la fonction optimale est du type NP-hard (THARWAT, HASSANIEN et ELNAGHI, 2017).

Représentation

La représentation la plus trouvée dans la littérature consiste en chaîne de N bits, où N est le nombre d'attributs d'origine et chaque bit peut prendre la valeur 1 ou 0, indiquant si l'attribut est sélectionné ou non. Cette représentation individuelle est simple et les opérateurs peuvent être facilement appliqués car la représentation binaire est très classique pour les métaheuristiques. Cependant, le principal inconvénient de cette représentation est qu'elle pourrait avoir un problème d'évolutivité lorsque le nombre d'attributs augmente (DHAENENS, 2016).

Mesures de performance

Les mesures de qualité sont spécifiques à chaque type d'approche pour la sélection d'attributs : filtre, enveloppe/embarquées.

Approches de filtres

Dans les approches par filtre, la mesure doit être rapide à calculer. Les mesures de filtrage les plus courantes pour les problèmes de classification sont la corrélation et l'information mutuelle qui permettent de calculer la qualité d'une caractéristique (ou d'un ensemble de caractéristiques) candidate. D'autres mesures courantes sont $X^2 - statistique$, $F - statistique$, critère de Fisher, mesure basée sur l'entropie, l'information mutuelle ponctuelle, distance inter/intra-classe ou les scores des tests de signification pour chaque combinaison classe / caractéristique (DHAENENS, 2016).

Approches d'enveloppes

Pour l'apprentissage supervisé, le principal objectif de la classification est de maximiser l'exactitude prédictive (predictive accuracy) ; par conséquent, dans les approches enveloppes, l'exactitude prédictive est généralement acceptée et largement utilisée comme mesure principale par les chercheurs et les praticiens (DHAENENS, 2016).

Pour les modèles d'enveloppe, la majorité des fonctions de fitness (fonctions objectives) essaient de minimiser le taux d'erreur du classificateur utilisé. Le taux d'erreur peut être globalement calculé sur l'ensemble de test ou calculé en utilisant l'exactitude de la validation croisée. Pour la classification binaire, certaines mesures spécifiques comme la

sensibilité et la spécificité peuvent être utilisées (DHAENENS, 2016).

Approches d'agrégation

Deux objectifs principaux sont souvent associés : la réduction du taux d'erreur et la réduction du nombre de caractéristiques (attributs) sélectionnées. Pour agréger les deux objectifs, on utilise des paramètres permettant de contrôler le compromis entre les préférences pour chaque objectif (DHAENENS, 2016).

Diverses métaheuristiques ont été utilisées pour gérer la sélection d'attributs. telles que l'optimisation des essaims de particules (PSO) (BOUBEZOUL et PARIS, 2012 ; CHUANG et al., 2008 ; CHUANG, TSAI et YANG, 2011 ; UNLER, MURAT et CHINNAM, 2011), l'optimisation des colonies de fourmis (ACO) (AGHDAM, GHASEM-AGHAEE et BASIRI, 2009 ; CHEN, CHEN et CHEN, 2013 ; HUANG, 2009 ; HUANG et al., 2012 ; KABIR, SHAHJAHAN et MURASE, 2012), les algorithmes génétiques (GA) (CHATTERJEE et BHATTACHERJEE, 2011 ; DAS et al., 2012 ; KABIR, SHAHJAHAN et MURASE, 2011 ; LI et al., 2011 ; OLIVEIRA et al., 2010 ; ÖZÇIFT et GÜLTEN, 2013 ; AL-ANI, ALSUKKER et KHUSHABA, 2013 ; SAHA, SARKAR et MITRA, 2009), la recherche d'harmonie (HS) (RAMOS et al., 2011) et l'évolution différentielle (DE) (AL-ANI, ALSUKKER et KHUSHABA, 2013).

Dans (GHAEMI et FEIZI-DERAKHSHI, 2016) les auteurs proposent une approche basée sur l'algorithme d'optimisation des forêts (FSFOA) afin de sélectionner les attributs. Le FSFOA proposé est validé sur plusieurs ensembles de données du monde réel et il est comparé avec d'autres méthodes, notamment HGAFS, PSO et SVM-FuzCoc. Les résultats des expériences montrent que le FSFOA peut améliorer la précision de la classification dans certains ensembles de données. Les auteurs également ont comparé la réduction de la dimensionnalité du FSFOA par rapport aux autres méthodes disponibles.

Un système de sélection d'attributs basé sur l'algorithme sinus cosinus (SCA) est proposé dans (HAFEZ et al., 2016). SCA est un nouvel algorithme de recherche stochastique pour les problèmes d'optimisation. La fonction de fitness proposée intègre à la fois la précision de la classification et la réduction de la taille d'attributs. Le système proposé a été testé sur 18 ensembles de données et montre une supériorité par rapport aux autres méthodes de recherche comme l'optimisation des essaims de particules (PSO) et l'algorithme génétique (GA).

Une nouvelle approche de sélection d'attributs est proposée dans (MAFARJA et MIRJALILI, 2018) basée sur l'algorithme d'optimisation des baleines (WOA). Deux variantes de l'algorithme WOA ont été proposées pour rechercher les sous-ensembles optimaux d'attributs. Dans le premier, les auteurs avaient étudié l'influence de l'utilisation des mécanismes de sélection du tournoi et de la roulette au lieu d'utiliser un opérateur aléatoire dans le processus de recherche. Dans la seconde approche, des opérateurs de croisement et de mutation ont été utilisés pour améliorer l'exploitation de l'algorithme WOA. Ces méthodes proposées sont comparées à trois algorithmes : l'optimisation des essaims de particules (PSO), l'algorithme génétique (GA), l'optimisation de fourmilion (ALO) et cinq méthodes de sélection d'attributs de filtrage.

Dans (ZHANG et al., 2018b), Zhang et al. ont proposé une variante de l'algorithme Firefly (FA) pour la sélection d'attributs dans les modèles de classification et de régression. 29 ensembles de données de classification et 11 ensembles de données de régression ont été utilisés pour évaluer l'efficacité du modèle FA. Les résultats ont montré des améliorations statistiquement significatives par rapport à d'autres variantes de FA et des méthodes de recherche classiques pour divers problèmes de sélection d'attributs. En bref, la variante FA proposée offre une méthode efficace pour identifier les sous-ensembles de caractéristiques (attributs) optimaux dans les modèles de classification et de régression.

Dans (GU, CHENG et JIN, 2018), Gu et al. ont proposé d'utiliser une variante très récente du PSO, connue sous le nom d'optimiseur d'essaim compétitif (CSO), qui a été consacrée à l'optimisation à grande échelle, pour résoudre les problèmes de sélection d'attributs dans les ensembles de données à haute dimension. De plus, le CSO, qui a été développé à l'origine pour l'optimisation continue, est adapté pour effectuer une sélection d'attributs qui peut être considérée comme un problème d'optimisation combinatoire. Une technique d'archivage est également introduite pour réduire les coûts de calcul. Des expériences menées sur six ensembles de données de référence ont démontré que le CSO non seulement sélectionne un nombre beaucoup plus petit d'attributs mais entraîne également une meilleure performance de classification.

Dans (MAFARJA et MIRJALILI, 2017), deux modèles d'hybridation sont utilisés pour concevoir différentes techniques de sélection d'attributs basées sur l'algorithme d'optimisation des baleines (WOA). Dans le premier modèle, l'algorithme de recuit simulé (SA) est intégré dans l'algorithme WOA, tandis qu'il est utilisé pour améliorer la meilleure solution trouvée après chaque itération de l'algorithme WOA. Dans le deuxième modèle, L'objectif de l'utilisation du SA est d'améliorer l'exploitation en recherchant les régions les plus prometteuses situées par l'algorithme WOA. La performance de ces approches est évaluée sur 18 ensembles de données de référence provenant d'UCI et comparée à trois méthodes d'enveloppe de sélection d'attributs bien connues dans la littérature. Les résultats expérimentaux confirment l'efficacité de ces approches pour améliorer la précision de la classification par rapport aux autres algorithmes à base d'enveloppe, ce qui garantit la capacité de l'algorithme WOA à rechercher dans l'espace des caractéristiques et à sélectionner les attributs les plus informatifs pour les tâches de classification.

La tendance récente de la recherche est d'hybrider deux algorithmes métaheuristiques et plus pour obtenir une solution supérieure dans le domaine des problèmes d'optimisation. Dans (SINDHU et al., 2019), les auteurs ont proposé une nouvelle méthode de sélection des caractéristiques basées sur l'hybridation de l'optimisation basée sur la biogéographie (BBO) et de l'algorithme sinus-cosinus (SCA) nommé (IBBO) pour traiter les problèmes de sélection d'attributs. Les performances d'IBBO sont étudiées à l'aide de quatorze ensembles de données de référence. Les résultats expérimentaux de l'IBBO sont comparés à huit algorithmes de recherche. Les résultats montrent que l'IBBO est capable de surpasser les autres algorithmes dans la majorité des ensembles de données. En outre, la force de l'IBBO est prouvée par diverses expériences numériques comme l'analyse statistique, les courbes de convergence, les méthodes de classement, et les fonctions de tests. Les résultats de la simulation ont révélé que l'IBBO a produit des résultats très compétitifs et prometteurs, par rapport aux autres algorithmes de recherche.

Une méthode hybride de filtre-enveloppe est proposée dans (MOSLEHI et HAERI, 2020) pour la sélection d'attributs établie avec l'intégration d'algorithme génétique (GA) et l'optimisation des essaims de particules (PSO) appelée smart HGP-FS. Cette méthode vise principalement à réduire la complication du calcul et le temps de recherche nécessaire pour obtenir une solution optimale au problème de sélection d'attributs dans les ensembles de données de grandes dimensions. Les méthodes de filtrage et d'enveloppe sont intégrées afin de tirer parti de l'accélération de la technique de filtrage et de la vigueur de la technique d'enveloppe pour la sélection d'attributs. Certaines caractéristiques de l'ensemble de données sont éliminées par la phase de filtrage, ce qui réduit les calculs complexes et le temps de recherche dans la phase d'enveloppement. Des comparaisons ont été faites pour vérifier l'efficacité de cet algorithme hybride avec l'utilisation de trois méthodes de

filtre-enveloppe, deux algorithmes d'enveloppe, deux méthodes de filtrage et de deux techniques d'enveloppe traditionnelles de sélection d'attributs. Les résultats obtenus sur des ensembles de données du monde réel ont montré l'efficacité de l'algorithme HGP-FS.

Dans (JAYARAMAN et SULTANA, 2019), un algorithme de recherche du coucou gravitationnel artificiel ainsi qu'un réseau de neurone à mémoire associative optimisé par les abeilles sont introduits pour gérer les caractéristiques (attributs) présentes dans le système de classification des maladies cardiaques. Initialement, les informations relatives aux maladies cardiaques sont collectées à partir du dépôt de données sur les maladies cardiaques de l'UCI. Les informations collectées sont d'une dimension énorme qui est difficile à traiter, ce qui réduit l'efficacité du système d'identification des maladies cardiaques. Ainsi, la dimensionnalité des attributs est réduite en fonction du comportement de l'algorithme de recherche du coucou gravitationnel. Ses caractéristiques sélectionnées sont traitées par le classificateur de mémoire associative. Ensuite, l'efficacité du système est évaluée à l'aide de résultats expérimentaux basés sur MATLAB.

Une méthode hybride de sélection d'attributs basés sur un algorithme de recherche gravitationnelle binaire (BGSA) et l'information mutuelle (MI) est présentée dans (BOSTANI et SHEIKHAN, 2017), pour améliorer l'efficacité de la BGSA standard en tant qu'algorithme de sélection d'attributs. Cette méthode, appelée MI-BGSA, utilise le BGSA comme méthode de sélection de caractéristiques (attributs) basées sur l'enveloppe pour effectuer une recherche globale. De plus, l'approche MI a été intégrée dans la BGSA, en tant que méthode basée sur le filtre, pour calculer les informations mutuelles sur les caractéristiques et les classes de caractéristiques dans le but d'élaguer le sous-ensemble de caractéristiques. Cette stratégie a permis de trouver les attributs qui présentent le moins de redondance par rapport aux attributs sélectionnés et qui sont les plus pertinents pour la classe cible. Une fonction à deux objectifs basés sur la maximisation du taux de détection et la minimisation du taux de faux positifs a été définie comme une fonction de fitness. Les résultats expérimentaux sur l'ensemble de données NSL-KDD ont montré que cette méthode peut réduire considérablement l'espace des caractéristiques. De plus, cet algorithme a permis de trouver un meilleur sous-ensemble d'attributs et d'obtenir une précision et un taux de détection plus élevés que certaines méthodes de sélection d'attributs standard basées sur l'enveloppe et le filtre.

4.4 Conclusion

La classification supervisée a été largement utilisée dans la littérature et les métaheuristiques contribuent à la proposition de modèles et d'approches intéressants (DHAENENS, 2016).
Les métaheuristiques sont des méthodes génériques capables de traiter de nombreux problèmes d'optimisation. Leur diversité et leur flexibilité rendent cette classe de méthodes très attractive pour s'attaquer aux problèmes complexes qui apparaissent dans l'exploration de données (datamining) (DHAENENS, 2016).
De nombreuses approches sont proposées dans ce chapitre pour optimiser les algorithmes de classification et la sélection d'attributs utilisant les algorithmes métaheuristiques. De nombreux auteurs notent que la performance de la sélection d'attributs dans les algorithmes d'enveloppe (wrapper) dépend de l'algorithme de classification choisi, mais peut être très coûteuse, surtout lorsque le volume de données est très important (DHAENENS, 2016).
Les fonctions objectives utilisées et leur nombre jouent également un rôle important dans la qualité des résultats obtenus (DHAENENS, 2016).

Chapitre 5

Un nouvel algorithme basé sur l'optimisation des chauves-souris avec l'évolution différentielle auto-adaptative pour l'entrainement de réseaux de neurones à propagation avant

5.1 Introduction

Avec l'explosion exponentielle de la quantité d'informations sur le web essentiellement, et avec le big data et tous ses « V », un tas de connaissances emmagasiné doivent être extraits d'une façon automatique. La tâche d'extraction des connaissances est devenue encore plus délicate et complexe. L'un des problèmes majeurs actuels est l'accès au contenu de ces informations, cela requiert l'utilisation d'outils plus spécifiques autrement dit l'accès au contenu par des moyens rapides et efficaces est devenu une tâche nécessaire.
La plupart des problèmes d'extraction de connaissances sont des problèmes d'optimisation combinatoire or, de nombreux problèmes d'optimisation combinatoire sont NP-difficile et ne pourront être pas résolus de manière exacte dans un temps raisonnable. Des méthodes dédiées à ce genre de problème, comme les métaheuristiques peuvent être utilisées.
Les métaheuristiques sont des algorithmes très puissants pour résoudre des problèmes d'optimisation complexes. Ces méthodes assurent un compromis entre la diversification et l'intensification et ont pour objectif de trouver des solutions dont la qualité est au-delà de ce qu'il aurait été possible de réaliser avec une simple heuristique (DHAENENS, 2016).
Dans les chapitres suivants, nous présentons nos approches proposées basées sur l'hybridation des métaheuristiques pour la sélection d'attributs et pour optimiser quelques algorithmes de classification à savoir les réseaux de neurones et les machines à vecteurs de support (SVM) appliqué sur plusieurs domaines.

5.2 L'entrainement de réseaux de neurones à propagation avant

Le réseau de neurones artificiels (ANN) est l'une des techniques d'exploration de données les plus importantes. Il a été appliqué avec succès à de nombreux domaines. Le perceptron multicouche (MLP) est l'un des réseaux de neurones les plus connus. Le perceptron multicouche (MLP) se compose de trois couches composées de neurones organisés en couches d'entrée, sortie et cachées. La première couche reçoit l'entrée, la deuxième

couche est la couche cachée et la troisième couche produit la sortie. Le succès d'un réseau de neurones dépend généralement du processus de formation ou d'entrainement qui est déterminé par les algorithmes de formation (entrainement). L'objectif des algorithmes d'entrainement est de trouver la meilleure connexion entre les poids et les biais qui minimisent l'erreur de classification.
Les algorithmes de formation peuvent être classés en deux catégories : les méthodes de recherche basées sur les gradients et les méthodes de recherche stochastiques. La rétropropagation (BP) et ses variantes sont des méthodes basées sur les gradients et sont considérées comme des techniques les plus populaires utilisées pour entraîner le réseau neuronal à propagation avant. Les méthodes basées sur les gradients présentent de nombreux inconvénients, tels que la convergence lente, la forte dépendance à la valeur initiale des poids et des biais et la tendance à être piégé dans les minimums locaux (Zhang, Zhang, Lok, & Lyu, 2007). Pour résoudre ces problèmes, des méthodes de recherche stochastiques, telles que les métaheuristiques, ont été proposées comme des méthodes alternatives pour la formation du réseau de neurones. Les métaheuristiques présentent de nombreux avantages : elles s'appliquent à tout type d'ANN avec toute fonction d'activation (KIRANYAZ et al., 2009a), fournissent des solutions acceptables dans un délai raisonnable pour résoudre des problèmes complexes et difficiles (RAIDL, 2006), et sont particulièrement utiles pour traiter de grands problèmes complexes qui génèrent de nombreux optima locaux (KENTER et al., 2017).
Les métaheuristiques se sont avérées utiles dans la formation des réseaux de neurones à propagation avant et plusieurs approches sont proposées dans la littérature pour améliorer leur efficacité sous différents angles, en utilisant diverses mesures telles que l'exactitude de la classification et l'erreur d'entrainement.
Bien qu'une grande variété d'algorithmes évolutifs et basés sur des essaims soit étudiée et déployée dans la littérature pour la formation des réseaux de neurones. La question qui se pose ici est ce que de nouveaux algorithmes de formation devaient encore être développés. La réponse est oui, les problèmes des minimums locaux restent disponibles. Le théorème de no-free-lunch (NFL) stipule qu'il n'y a pas d'algorithme d'optimisation supérieur pour résoudre tous les problèmes d'optimisation. L'entrainement des réseaux de neurones est également un problème d'optimisation qui varie pour chaque ensemble de données (FARIS, ALJARAH et MIRJALILI, 2016b).
Sur la base de ces raisons, On a proposé deux nouveaux algorithmes basés sur l'hybridation des métaheuristiques pour entrainer les réseaux de neurones à propagation avant. Dans la première approche, un nouvel algorithme basé sur l'optimisation des chauves-souris avec l'évolution différentielle auto-adaptative, appelée BAT-SDE, est proposé pour former le réseau de neurones à propagation avant. Huit ensembles de données ont été résolus par l'entraîneur proposé.
De plus, l'application de cet entraîneur a été étudiée dans le domaine biomédical et dans la détection des fraudes. La performance du BAT-SDE a été comparée à huit algorithmes métaheuristiques bien connus utilisés pour entrainer les réseaux de neurones dans la littérature : PSO (MENDES et al., 2002a), CS (YAO, 1993), BAT (NAWI, REHMAN et KHAN, 2014b), MFO (YAMANY et al., 2015), MVO (FARIS, ALJARAH et MIRJALILI, 2016b), GWO (HASSANIN, SHOEB et HASSANIEN, 2016; FARIS, MIRJALILI et ALJARAH, 2019a), WOA (MAFARJA et MIRJALILI, 2018), HACPSO (KHAN et al., 2019).

5.2.1 Réseaux de Neurones artificiels (ANNs)

Un réseau de neurones artificiels (ANN) est un modèle de calcul basé sur la structure et les fonctions du cerveau biologiques et du système nerveux. Le réseau de neurones à propagation avant (FFNN) est l'un des types les plus populaires de réseau de neurones artificiels (FARIS, ALJARAH et MIRJALILI, 2016b). Le FFNN comporte trois couches interconnectées. La première couche est constituée de neurones d'entrée. Ces neurones envoient les données à la deuxième couche, appelée couche cachée, qui envoie les neurones de sortie à la troisième couche. Dans FFNN, l'information voyage dans une direction, de la couche d'entrée à la couche de sortie. Le nœud ou le neurone artificiel multiplie chacune de ces entrées par le poids, comme indiqué en (5.1) :

$$S_j = \sum_{i=1}^{n} w_{i,j} I_i + \beta_j \tag{5.1}$$

Où, n est le nombre total d'entrées neuronales, $w_{i,j}$ est le poids de connexion reliant I_j au neurone j et β_j est un poids de biais (FARIS, ALJARAH et MIRJALILI, 2016b). Ensuite, le nœud ou le neurone artificiel ajoute les multiplications et envoie la somme à une fonction de transfert, par exemple, la fonction sigmoïde présentée en (5.2)

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5.2}$$

La sortie du neurone j peut être décrite comme suit (5.3) :

$$y_j = f_j(\sum_{i=1}^{n} w_{i,j} I_i + \beta_j) \tag{5.3}$$

Après avoir construit le réseau de neurones, l'ensemble des poids du réseau sont ajustés pour se rapprocher à des résultats souhaités. Ce processus est réalisé en appliquant un algorithme d'entraînement pour adapter les poids jusqu'à ce que les critères d'erreur soient satisfaits (FARIS, ALJARAH et MIRJALILI, 2016b).

5.2.2 La méthode proposée pour entrainer les réseaux de neurones à propagation avant

Cette section présente l'approche proposée basée sur nos algorithmes décrivent ci-dessous pour former le réseau de neurones à perceptron multicouche (MLP). Deux points importants sont pris en considération : la fonction de fitness (la fonction objective) et la représentation des solutions.
Dans ce travail, nos algorithmes ont été appliqué pour entraîner le réseau MLP avec une seule couche cachée et chaque solution (poids et biais) a été formée par trois parties : les poids de connexion entre la couche d'entrée et la couche cachée, les poids entre la couche cachée et la couche de sortie, et les poids de biais. La longueur de chaque vecteur de solution est donnée par l'équation (5.4), où n est le nombre de caractéristiques d'entrée (attributs) et m est le nombre de neurones dans la couche cachée (FARIS, ALJARAH et MIRJALILI, 2016b), comme suit :

$$L = (N \times m) + (2 \times m) + 1 \tag{5.4}$$

Les solutions (poids et biais) sont implémentées en tant que vecteurs de nombres réels lorsque chaque vecteur appartient à l'intervalle $[-1, 1]$, comme illustré sur la figure 5.1. L'erreur quadratique moyenne (MSE) a été utilisée pour mesurer la valeur de fitness des solutions. MSE a été calculé sur la base de la différence entre les valeurs estimées et réelles

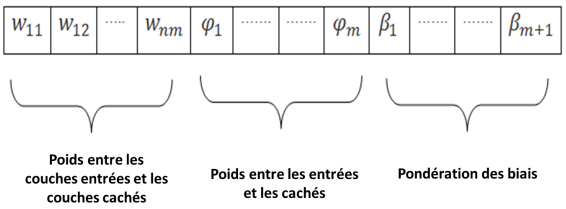

FIGURE 5.1 – Représentation de la structure de solution

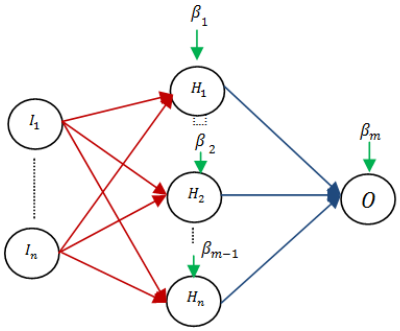

FIGURE 5.2 – Affectation du vecteur de solution au MLP

du réseau neuronal à l'aide des ensembles de données d'apprentissage, comme indiqué dans l'équation (5.5), où n est le nombre d'échantillons dans l'ensemble de données d'entraînement y et $\hat{y}$ sont respectivement les valeurs réelles et prédites :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 \tag{5.5}$$

Dispositif expérimental

Les algorithmes proposés et les autres algorithmes ont été mis en œuvre avec le langage Python et un ordinateur personnel avec un processeur Intel (R) Core (TM) 1,60 GHz 2,30 GHz, un système d'exploitation Windows 7 64 bits et 4 Go (RAM). Les métaheuristiques sont sensibles aux valeurs de leurs paramètres, ce qui nécessite une initialisation soigneuse. Par conséquent, les paramètres de contrôle recommandés dans la littérature ont été utilisés (MENDES et al., 2002a ; NAWI, REHMAN et KHAN, 2014a ; FARIS, ALJARAH et MIRJALILI, 2016b ; MAFARJA et MIRJALILI, 2018 ; FARIS, MIRJALILI et ALJARAH, 2019b) et résumés dans le tableau 5.1. Tous les ensembles de données ont été divisés en 66% pour l'apprentissage et 34% pour le test (la méthode Holdout). De plus, tous les attributs ont été mappées à l'intervalle $[0, 1]$ pour éliminer l'effet des attributs qui ont des échelles différentes.
Dans la littérature, il n'y a pas de méthode standard pour sélectionner le nombre de neurones cachés. Dans ce travail, la méthode proposée dans (MIRJALILI, MIRJALILI et LEWIS, 2014b ; MIRJALILI, 2015 ; FARIS, ALJARAH et MIRJALILI, 2016b) a été utilisée ; le nombre de neurones dans la couche caché est égal à $2N + 1$, Ou, N est le nombre d'attributs dans l'ensemble de données.

TABLE 5.1 – Paramètres initiaux des algorithmes d'optimisation

Paramètre	Définition	Valeur
BAT	Fréquence minimale	0
	Fréquence maximale	1
	Intensité d'émission	0.5
	fréquence du pouls	0.5
DE	Facteur de pondération (F)	0.5
	Probabilité de croisement (CR)	0.9
PSO	Constantes d'accélération	[2.1,2.1]
	Poids d'inertie	[0.9,0.6]
	Nombre de particules	50
MVO	Trou de ver minimum	0.2
	Trou de ver maximum	1
MFO	Taille de la population	50
	Nombre de générations	200
	Nombre d'agents de recherche	50
	b	1
	t	[-1,1]
WOA	Taille de la population	200
	Nombre de générations	200
	r	[0,1]
	Taille de la population	50
HACPSO	B	0.5
	Taille de la population	50
	Nombre de générations	200

5.2.3 BAT-SDE pour l'entrainement du réseau de neurones à propagation avant

Un nouvel algorithme basé sur l'optimisation des chauves-souris avec l'évolution différentielle auto-adaptative, appelée BAT-SDE, est proposé pour former le réseau de neurones à propagation avant. L'application de cet entraîneur a été étudiée dans le domaine biomédical et dans la détection des fraudes. La performance du BAT-SDE a été comparée à huit algorithmes métaheuristiques bien connus utilisés pour entrainer les réseaux de neurones dans la littérature : PSO (MENDES et al., 2002a), MFO (YAMANY et al., 2015), MVO (FARIS, ALJARAH et MIRJALILI, 2016b), WOA (MAFARJA et MIRJALILI, 2018), HACPSO (KHAN et al., 2019).

L'algorithme des chauves-souris (Bat algorithm)

L'algorithme des chauves-souris (Bat algorithm) est un algorithme métaheuristique d'optimisation globale développé par Xin-She Yang en 2010 (YANG, 2010). L'algorithme est basé sur le comportement d'écholocation chez les chauves-souris, chaque mouvement virtuel des chauves-souris est mis à jour en fonction de la fréquence f_i, de la vitesse v_i et de la position x_i de chaque chauve-souris pour trouver une proie. Lorsqu'une nouvelle solution est acceptée, l'intensité sonore A_i et la fréquence du pouls r_i sont mises à jour (YANG, 2010). La position, la vitesse et la fréquence des chauves-souris sont mises à jour sur la base des équations suivantes :

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{5.6}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_{gbest}) f_i \tag{5.7}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{5.8}$$

Où, $\beta \in [0,1]$ est un nombre aléatoire tiré d'une distribution uniforme (YANG, 2011). La meilleure solution actuelle est modifiée selon l'équation suivante :

$$x_{new} = x_{old} + A_i^{(t)} \tag{5.9}$$

Où, ϵ est le facteur d'échelle. Lorsqu'une chauve-souris trouve une proie, la sonie diminue et le taux d'émission des impulsions augmente (YANG, 2011) selon les équations suivantes :

$$A_i^{t+1} = \alpha A_i^t \tag{5.10}$$

$$r_i^{t+1} = r_i^0 [1 - exp(-\gamma\epsilon)] \tag{5.11}$$

Où, α et γ sont des constantes.

L'évolution différentielle

L'évolution différentielle (DE) (STORN et PRICE, 1997a) est un algorithme métaheuristique évolutif pour l'optimisation globale développé par Storn et Price en 1997. DE optimise un problème en maintenant une population de solutions candidates. Il crée de nouvelles solutions candidates en combinant les solutions existantes, puis conserve la solution candidate qui a le meilleur score ou la meilleure adéquation à l'optimisation.
DE soutient une mutation différentielle, un croisement différentiel et une sélection différentielle. En particulier, la mutation différentielle sélectionne au hasard trois solutions et ajoute le vecteur de différence pondérée entre ces deux solutions à une troisième solution. Cette mutation peut être exprimée par l'équation suivante (ZANCHETTIN, LUDERMIR et ALMEIDA, 2011) :

$$u_i^{(t)} = x_{r0}^{(t)} + F \cdot (x_{r1}^{(t)} - x_{r2}^{(t)})_{,i=1..NP} \tag{5.12}$$

Où, $F \in [0;2]$désigne le facteur d'échelle qui échelonne le taux de modification et $r0, r1$ et $r2$ sont des vecteurs choisis au hasard dans l'intervalle $1..NP$. Le croisement uniforme est utilisé comme un croisement différentiel par le DE (ZANCHETTIN, LUDERMIR et ALMEIDA, 2011) Ce processus peut être mathématiquement exprimé comme suit :

$$Z_{i,j}^{(t)} \begin{cases} u_{i,j}^{(t)}, & rand_j \leq CR \vee j = j_{rand} \\ x_{i,j}^{(t)}, & otherwise \end{cases} \tag{5.13}$$

Où, le paramètre $CR \in [0;1]$ représente le taux de croisement. La sélection différentielle peut être exprimée comme suit :

$$x_{i,j}^{(t)} \begin{cases} Z_i^{(t)}, & f(Z_i^{(t)}) \leq f(x_i^{(t)}) \\ x_i^{(t)}, & otherwise \end{cases} \tag{5.14}$$

Auto-Adaptatif hybride BAT

(FISTER JR, FISTER et YANG, 2013) ont montré que l'algorithme de chauve-souris obtient un bon résultat avec les problèmes de faible dimension, mais peut devenir une problématique pour les problèmes de plus grandes dimensions car il a tendance à converger très rapidement au départ. Par conséquent, l'hybridation entre la chauve-souris et l'algorithme d'évolution différentielle proposé par (FISTER JR, FISTER et YANG, 2013) permet de résoudre ce problème et d'améliorer le comportement de l'algorithme de chauve-souris.
Dans notre approche, l'algorithme de chauve-souris a été hybridé avec un algorithme d'évolution différentielle auto-adaptatif (DE) où la solution est modifiée en utilisant la stratégie DE / best / 1 / bin ; où best indique que le vecteur de base est actuellement le meilleur vecteur de la population, 1 une différence de vecteur y est ajoutée et le nombre de paramètres modifiés dans le vecteur de mutation suit la distribution binomiale.
Pour chaque vecteur, les paramètres de contrôle F et CR ont été auto-ajustés au cours de l'analyse selon les équations (5.15), (5.16).

Les paramètres F et CR auto-adaptatifs : algorithme jDE modifié

Brest et coll. (BREST et al., 2006) ont proposé un mécanisme de contrôle auto-adaptatif, dans lequel chaque vecteur de la population a été étendu avec ses propres valeurs F et CR. Les nouveaux paramètres de contrôle F et CR ont été auto-ajustés pendant l'exécution avec les équations suivantes :

$$F_{i,G+1} = \begin{Bmatrix} F_l + rand_1 * F_u, & if \;\; rand_2 < \tau_1 \\ F_{i,G}, & otherwise \end{Bmatrix} \tag{5.15}$$

$$CR_{i,G+1} = \begin{Bmatrix} rand_3, & if \;\; rand_4 < \tau_2 \\ CR_{i,G}, & otherwise \end{Bmatrix} \tag{5.16}$$

Où, $rand_j$, $j \in 1, 2, 3, 4$ sont des nombres aléatoires uniformes dans l'intervalle $[0, 1]$; τ_1 et indiquent les probabilités d'ajustement des paramètres de contrôle F et CR. Pour note étude expérimentale, $\tau_1 = \tau_2 = 0.1$, $F_l = 0.2$ et, $F_u = 0.9$ ont les mêmes valeurs que dans (BREST et al., 2006).
Dans notre approche, l'algorithme jDE a été modifié. $F_{i,G+1}$ Et $CR_{i,G+1}$ ont été calculés comme suit :

$$F_{i,G+1} = \begin{Bmatrix} F_l + rand_1 * F_u, & if \;\; rand_2 < \tau_1 \\ 0.001, & otherwise \end{Bmatrix} \tag{5.17}$$

$$CR_{i,G+1} = \begin{Bmatrix} rand_3, & if \;\; rand_4 < \tau_2 \\ 0.9, & otherwise \end{Bmatrix} \tag{5.18}$$

Il est important de noter que DE / best / 1 / bin a été utilisé pendant la phase d'expérimentation ; où best indique que le vecteur de base est actuellement le meilleur vecteur de la population 1, une différence de vecteur y est ajoutée et le nombre de paramètres modifiés dans le vecteur de mutation suit la distribution binomiale (DAS et SUGANTHAN, 2010).
(*Remarque :* cette hybridation est de type hybridation relais de bas niveau (JOURDAN, 2003))

Les étapes du BAT-SDE pour la formation du réseau de neurones (MLP) peuvent être démontrées comme suit :

1. Initialisation : initialisation de la population de chauves-souris, de la fréquence, de l'intensité sonore et du pouls ;

Algorithm 8 BAT-SDE

Fonction objectif $f(x), x = (x_1, ..., x_d)^T$
Initialiser la population de chauves-souris x_i et v_i $fori = 1...n$
Définir la fréquence d'impulsion $Q_i \in [Q_{min}, Q_{max}]$
Initialiser les fréquences d'impulsions r_i et l'intensité A_i
while $(t < T_{max})$ **do**
  Générez de nouvelles solutions en ajustant la fréquence et
  mise à jour des vitesses et des positions / solutions
  if $(rand(0,1) > r_i)$ **then**
    Modifiez la solution à l'aide de "DE/best/1/bin"
    Générer aléatoirement deux nombres entiers
    $r_1, r_2 \in [1, N]$, où $r_1 \neq r_2 \neq i$

    for $j = 1$ to D **do**
      $v_{(i)j} = best + F.(w_{r_1} - w_{r_2})$
      Générer aléatoirement un nombre réel
      rand $j \in [0; 1]$

      if $randj < CR$ **then**
        then $u_i j := v_i j$
      end if
    end for
    Générer une nouvelle solution en volant aléatoirement
    if $(rand(0,1) < A_i)$ and $(f(x_i) < f(x)$ **then**
      Acceptez les nouvelles solutions
      Incrémenter r_i et réduire A_i
    end if
  end if
  classer les chauves-souris et trouver la meilleure solution courante
end while
Résultats et visualisation

2. Évaluer les solutions aléatoires initiales par la fonction de fitness : affectation les vecteurs de solutions initiales (poids et biais) au réseau MLP et chaque réseau est évalué par la fonction de fitness MSE ;
3. Générer de nouvelles solutions : génération de nouvelles solutions en ajustant la fréquence, la position et la distance ;
4. Modification de la solution : si, la solution est modifiée en utilisant DE / best / 1 / bin et en auto-adaptant les paramètres de contrôle F et CR (équations (5.15), (5.16)) ;
5. Évaluer les solutions : évaluation du réseau MLP à l'aide de MSE ;
6. Trouvez la meilleure solution actuelle : classez les chauves-souris et trouvez la solution actuelle.
7. Les étapes 3 à 6 ont été répétées jusqu'à ce que le nombre maximum d'itérations soit atteint.

La figure 5.3 représente les étapes de l'approche BAT-SDE-MLP.

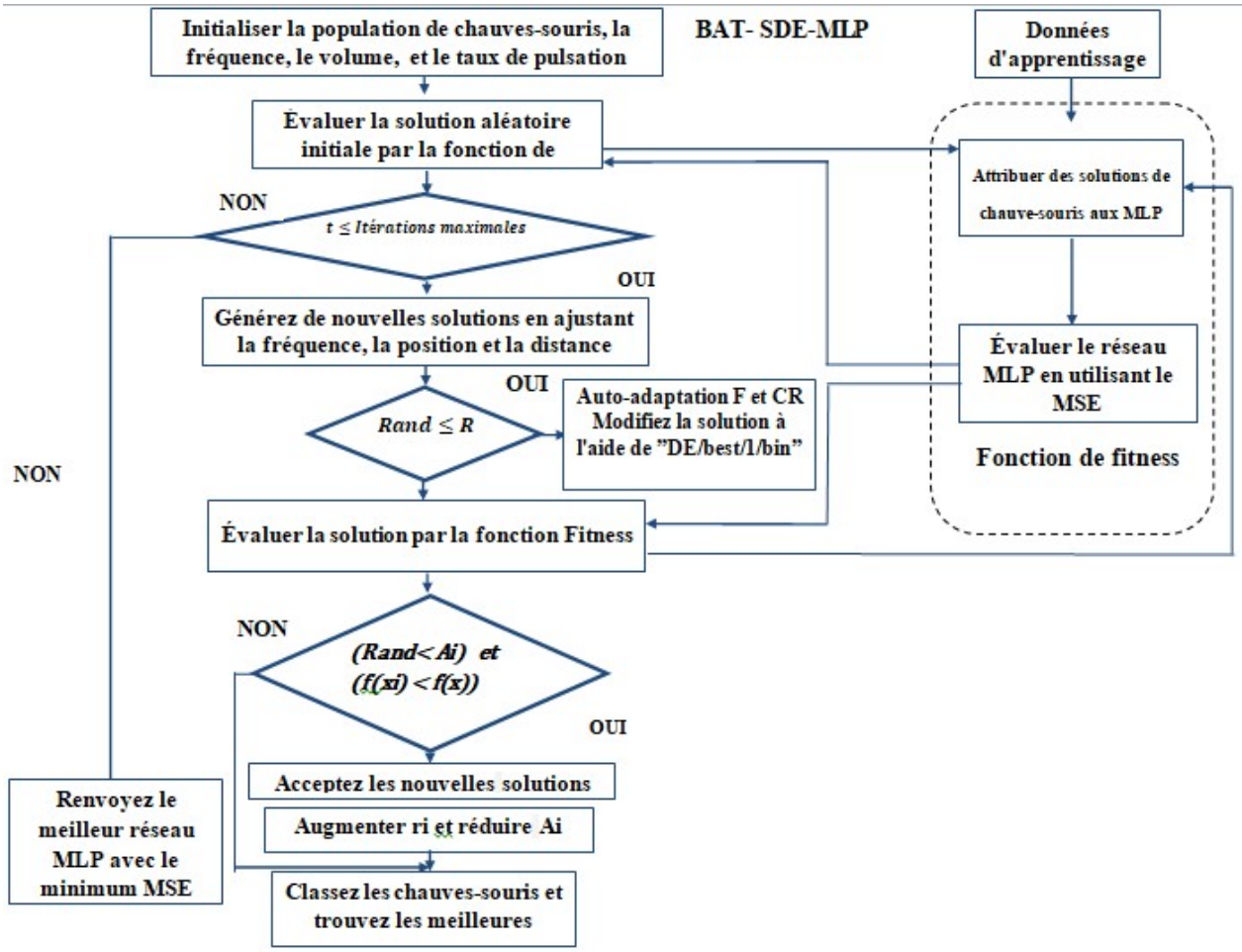

FIGURE 5.3 – Étapes générales de l'approche BAT-SDE-MLP

Expérimentation et résultats

Cette section présente l'évaluation du notre algorithme BAT-SDE pour la formation des réseaux de neurones a propagation avant sur cinq ensembles de données bien connus, qui ont été sélectionnés à partir des dépôts d'ensembles de données (UCI) [1] et Kaggle [2]. Le tableau 5.2 montre la classification de ces ensembles de données en termes de nombre d'attributs, nombre classes, nombre d'instances dans l'ensemble d'entrainement et du test. La comparaison du BAT-SDE a été effectuée avec huit approches utilisées pour former les réseaux neuronaux a propagation avant dans la littérature : PSO (MENDES et al., 2002a), MFO (YAMANY et al., 2015), MVO (FARIS, ALJARAH et MIRJALILI, 2016b), WOA (MAFARJA et MIRJALILI, 2018), HACPSO (KHAN et al., 2019). En outre, l'algorithme proposé a été comparé à la rétropropagation avec moment et le taux d'apprentissage adaptatif, qui est un algorithme basés sur le gradient.

1. http ://archive.ics.uci.edu/ml/
2. https ://www.kaggle.com/datasets/

TABLE 5.2 – Les ensembles de données de classification

Datasets	Attributs	Ensemble d'apprentissage	Ensemble de test
Blood	4	493	255
Breast cancer	8	461	238
Diabetes	8	506	262
Vertebral	6	204	106
Liver	6	79	41
Parkinson	22	128	67
Hepatitis	10	102	53

Vue d'ensemble

Cinq fonctions de test (FISTER JR, FISTER et YANG, 2013) ont été sélectionnées (tableau 5.3) pour évaluer la performance de l'algorithme BAT-SDE proposé. BAT-SDE a été comparé à BAT (YANG, 2010), DE (STORN et PRICE, 1997a) et Hybrid (HBA) (FISTER JR, FISTER et YANG, 2013) pour vérifier les résultats.

TABLE 5.3 – Les fonctions de test

Fonctions		Fmin
Griewangk	$f_1(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	0
Rosenbrock	$f_2(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \quad \right])$	0
Sphere	$f3(x) = \sum_{i=1}^{n} x_i^2$	0
Rastrigin	$f_4(x) = \sum_{i=1}^{n} \left[x_i^2 - 10cos(2\pi x_i) + 10\right]$	0
Ackley	$f_5(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	0

Tous les algorithmes ont été testés 30 fois avec la valeur de dimension D=10 pour chaque fonction de test. Le nombre de générations a été fixé à 1000 (FISTER JR, FISTER et YANG, 2013). Le tableau 5.1 résume les paramètres des algorithmes qui ont été utilisés dans les travaux précédents (STORN et PRICE, 1997a ; FISTER JR, FISTER et YANG, 2013).

TABLE 5.4 – Résultats de l'exactitude de la classification

Algorithmes/fonctions		f_1	f_2	f_3	f_4	f_5
BAT-SDE	Avg	6.40E-01	4.59E+00	6.26E-01	3.65E+01	3.69E+01
	STD	3.07E-01	7.91E-01	7.31E-01	8.07E+00	7.04E+00
	Best	2.15E-01	2.79E+00	1.47E-01	1.54E+01	2.13E+01
	Worst	1.17E+00	5.90E+00	3.96E+00	4.78E+01	5.50E+01
BAT	Avg	2.53E+00	4.40E+00	2.39E+01	3.88E+01	4.00E+01
	STD	6.34E-01	8.62E-01	8.93E+00	8.03E+00	6.58E+00
	Best	1.68E+00	2.78E+00	1.01E+01	2.02E+01	2.92E+01
	Worst	4.03E+00	5.99E+00	4.52E+01	5.91E+01	5.93E+01
HBA	Avg	2.89E+00	4.58E+00	1.86E+01	4.08E+01	3.61E+01
	STD	5.32E-01	1.01E+00	5.06E+00	8.29E+00	1.02E+01
	Best	1.97E+00	2.53E+00	1.10E+01	2.57E+01	1.44E+01
	Worst	4.09E+00	6.62E+00	3.18E+01	5.36E+01	5.68E+01
DE	Avg	1.22E+00	1.91E+05	2.93E+00	6.75E+01	1.18E+01
	STD	8.20E-02	1.51E+05	9.82E-01	7.08E+00	1.31E+00
	Best	1.08E+00	1.26E+04	1.10E+00	4.18E+01	8.93E+00
	Worst	1.37E+00	6.12E+05	5.64E+00	5.52E+01	1.37E+01

Le tableau 5.4 indique clairement que BAT-SDE surpasse les trois autres algorithmes dans trois fonctions de test f_1 ,f_3 ,f_4 . Lorsque les résultats de BAT-SDE sont comparés aux BAT et DE, on peut observer que le BAT-SDE fonctionne mieux que Bat dans les fonctions f_1, f_3, f_4,f_5 et surpasse le DE dans les fonctions f_1, f_2, f_3,f_4 . Les faibles valeurs d'écart-type de BAT-SDE prouvent l'efficacité et la robustesse de cet algorithme.

Résultats

Chaque ensemble de données a été divisé en deux parties pour évaluer le MLP, 66% pour l'ensemble d'apprentissage et 34% pour l'ensemble de test. Tous les algorithmes ont été testés dix fois pour chaque ensemble de données et les performances du MLP ont été évaluées en fonction des valeurs de meilleure, pire, moyenne et écart-type de l'exactitude de la classification et du MSE. La taille de la population et le nombre maximum de générations ont été fixés respectivement à 50 et 200.

TABLE 5.5 – Résultats de l'exactitude de la classification

		BAT-SDE	BAT	PSO	CS	MFO	MVO	GWO	WOA	HACPSO	BP
B.cancer	Avg	0.961	0.813	0.959	0.959	0.958	0.958	0.653	0.956	0.959	0.744
	STD	0.001	0.254	0.003	0.004	0.002	0.002	0.000	0.006	0.004	0.254
	Best	0.963	0.960	0.960	0.960	0.963	0.960	0.653	0.963	0.965	0.945
	Worst	0.958	0.188	0.953	0.952	0.954	0.954	0.653	0.947	0.952	0.680
Blood	Avg	0.765	0.407	0.762	0.761	0.763	0.765	0.653	0.762	0.760	0.744
	STD	0.002	0.193	0.002	0.006	0.005	0.009	0.041	0.004	0.003	0.254
	Best	0.767	0.760	0.765	0.776	0.765	0.784	0.760	0.774	0.760	0.945
	Worst	0.762	0.239	0.760	0.751	0.760	0.760	0.623	0.758	0.758	0.680
Diabetes	Avg	0.782	0.520	0.780	0.704	0.724	0.792	0.340	0.720	0.768	0.619
	STD	0.015	0.178	0.011	0.007	0.008	0.006	0.000	0.028	0.006	0.08
	Best	0.792	0.814	0.796	0.731	0.735	0.802	0.340	0.750	0.774	0.690
	Worst	0.732	0.339	0.758	0.669	0.709	0.782	0.340	0.657	0.764	0.601
Liver	Avg	0.751	0.586	0.722	0.689	0.722	0.737	0.419	0.665	0.679	0.519
	STD	0.002	0.129	0.017	0.017	0.010	0.006	0.000	0.030	0.020	0.055
	Best	0.784	0.735	0.748	0.713	0.744	0.744	0.633	0.700	0.709	0.586
	Worst	0.722	0.418	0.700	0.669	0.709	0.726	0.564	0.603	0.669	0.495
Vertebral	Avg	0.837	0.707	0.836	0.819	0.836	0.837	0.778	0.774	0.817	0.651
	STD	0.007	0.178	0.012	0.017	0.007	0.005	0.056	0.052	0.015	0.170
	Best	0.843	0.838	0.848	0.848	0.843	0.838	0.866	0.862	0.833	0.866
	Worst	0.828	0.338	0.808	0.799	0.82	0.828	0.732	0.676	0.794	0.627
Parkinson	Avg	0.841	0.704	0.836	0.841	0.802	0.802	0.750	0.842	0.876	0.750
	STD	0.027	0.224	0.040	0.041	0.022	0.046	0.000	0.016	0.013	0.199
	Best	0.882	0.890	0.898	0.859	0.828	0.867	0.750	0.852	0.905	0.849
	Worst	0.789	0.203	0.773	0.804	0.773	0.695	0.750	0.800	0.863	0.623
Hepatitis	Avg	0.915	0.773	0.914	0.888	0.866	0.867	0.860	0.791	0.854	0.758
	STD	0.018	0.275	0.017	0.015	0.016	0.014	0.014	0.063	0.015	0.155
	Best	0.963	0.947	0.947	0.95	0.894	0.894	0.874	0.867	0.875	0.865
	Worst	0.873	0.221	0.884	0.863	0.842	0.842	0.842	0.695	0.828	0.669

Le tableau 5.5 présent les résultats statistiques : moyenne, meilleure, pire et écart type de la l'exactitude de la classification (accuracy). Les résultats de BAT-SDE ont surpassé d'autres approches dans Breast cancer, Blood, Liver, Vertebral and Hepatitis avec une exactitude moyenne de 0,961, 0,765, 0,751, 0,837, 0,915. En outre, Bat-SDE était classé deuxième dans les ensembles de données : Diabetes et Parkinson avec une exactitude moyenne de 0,782, 0,841 respectivement. De plus, on peut également voir que le BAT-SDE a un écart type plus petit qui indique que le BAT-SDE est stable.

TABLE 5.6 – Résultats de MSE

		BAT-SDE	BAT	PSO	CS	MFO	MVO	GWO	WOA	HACPSO	BP
B.cancer	Avg	0.032	0.053	0.032	0.040	0.033	0.032	0.048	0.047	0.040	0.049
	STD	0.001	0.020	0.001	0.005	0.001	0.002	0.007	0.004	0.001	0.015
	Best	0.030	0.037	0.030	0.039	0.310	0.030	0.042	0.043	0.038	0.030
	Worst	0.033	0.121	0.326	0.041	0.036	0.032	0.056	0.058	0.041	0.050
Blood	Avg	0.169	0193	0.178	0.175	0.176	0.170	0.178	0.180	0169	0.174
	STD	0.005	0.028	0.003	0.005	0.005	0.008	0.007	0.004	0.003	0.009
	Best	0.160	0.177	0.182	0.177	0.174	0.155	0.174	0.174	0.162	0.172
	Worst	0.174	0.261	0.172	0.182	0.177	0.181	0.181	0.187	0.175	0.175
Diabetes	Avg	0.151	0.193	0.155	0.175	0.171	0.147	0.166	0.186	0.163	0.179
	STD	0.002	0.028	0.004	0.002	0.005	0.001	0.005	0.013	0.002	0.066
	Best	0.149	0.158	0.157	0.160	0.165	0.149	0.166	0.168	0.160	0.168
	Worst	0.153	0.261	0.151	0.171	0.175	0.145	0.176	0.213	0.166	0.180
Liver	Avg	0.176	0.224	0.191	0.210	0.193	0.186	0.225	0.220	0.212	0.210
	STD	0.004	0.027	0.003	0.002	0.002	0.004	0.004	0.007	0.002	0.003
	Best	0.170	0.196	0.185	0.208	0.189	0.179	0.218	0.210	0.208	0.190
	Worst	0.180	0.245	0.195	0.211	0.194	0.194	0.233	0.233	0.215	0.220
Vertebral	Avg	0.130	0.154	0.134	0.143	0.136	0.133	0.162	0.163	0.146	0.168
	STD	0.006	0.032	0.002	0.003	0.002	0.002	0.008	0.018	0.002	0.0015
	Best	0.120	0.136	0.132	0.141	0.135	0.130	0.153	0.137	0.142	0.160
	Worst	0.138	0.182	0.135	0.146	0.137	0.134	0.166	0.202	0.147	0.172
Parkinson	Avg	0.134	0.164	0.141	0.128	0.158	0.147	0.123	0.165	0.119	0.158
	STD	0.014	0.114	0.023	0.006	0.019	0.020	0.006	0.030	0.005	0.018
	Best	0.119	0.090	0.099	0.102	0.135	0.125	0.117	0.127	0.112	0.137
	Worst	0.157	0.137	0.182	0.139	0.203	0.197	0.137	0.228	0.130	0.203
Hepatitis	Avg	0.092	0.097	0.089	0.098	0.119	0.116	0.117	0.143	0.090	0.168
	STD	0.016	0.058	0.016	0.003	0.011	0.010	0.006	0.020	0.003	0.015
	Best	0.070	0.054	0.056	0.093	0.100	0.102	0.106	0.109	0.084	0.159
	Worst	0.119	0.202	0.117	0.102	0.138	0.130	0.125	0.176	0.094	0.173

Le tableau 5.6 montre la moyenne, la meilleure et la pire MSE avec l'écart type, obtenue pour chaque algorithme. En conséquence, on peut noter que le BAT-SDE surpasse les autres techniques dans quatre ensembles de données : Breast cancer, Blood, Liver et Vertebral avec une MSE moyenne de 0,032, 0,169, 0,176, 0,130, respectivement. De plus, on peut également remarquer que le BAT-SDE a une faible valeur d'écart type pour tous les jeux de données, ce qui prouve l'efficacité et la robustesse de cet algorithme.

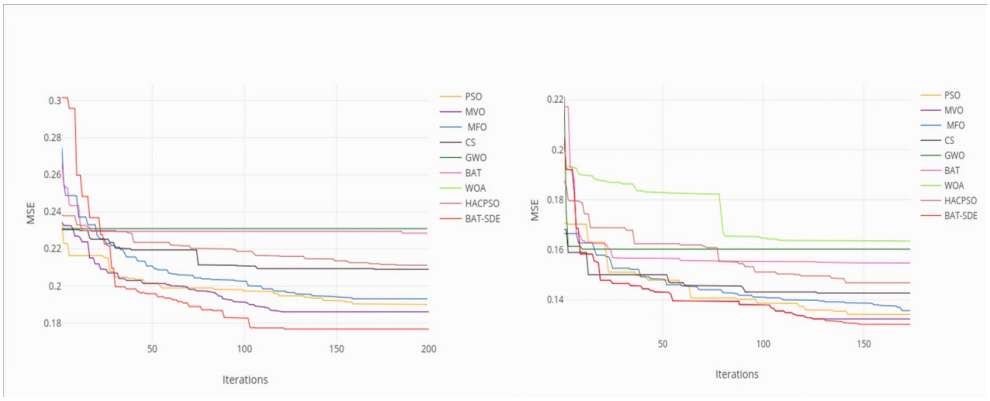

FIGURE 5.4 – Courbe de convergence basé sur le MSE pour Hepatitis et Vertebral, respectivement

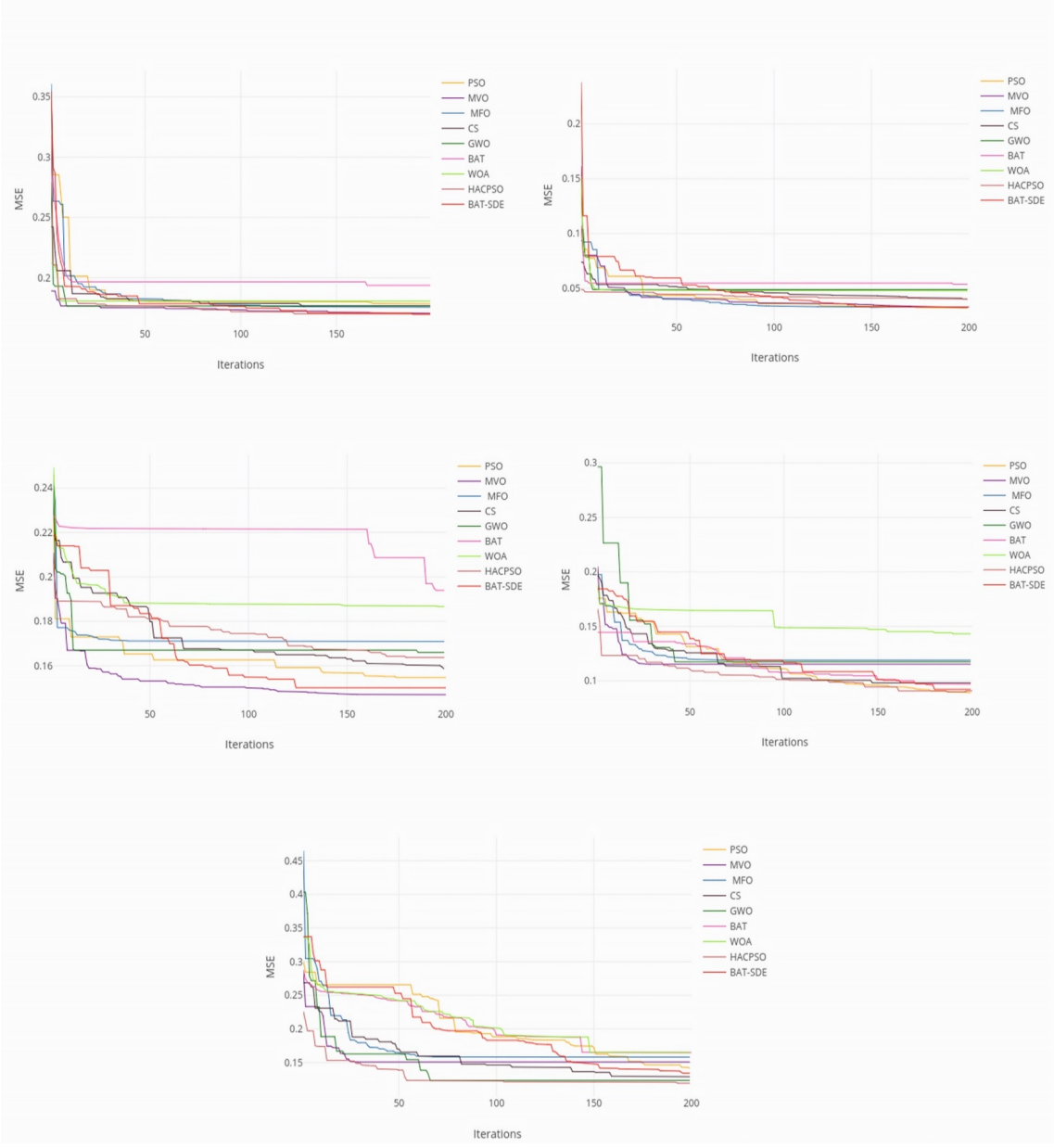

FIGURE 5.5 – Courbe de convergence basé sur le MSE pour blood, breast cancer, diabetes, hepatitis

Les figures 5.4 et 5.5 montrent les courbes de convergence de tous les algorithmes d'entrainement basés sur les valeurs moyennes de MSE. Les courbes de convergence montrent que BAT-SDE a la valeur la plus basse de MSE pour quatre ensembles de données : Breast cancer, Blood, Liver et Vertebral. De plus, le BAT-SDE a la vitesse de convergence la plus rapide dans les ensembles de données Liver, Vertebral, Blood et European. Pour l'ensemble de données Diabete, BAT-SDE fournit des performances très proches par rapport à l'algorithme MVO. Ces résultats montrent que le BAT-SDE a une convergence plus rapide et une meilleure optimisation par rapport aux autres algorithmes métaheuristiques.

Classification de l'ensemble de données européen

L'ensemble de données européen, extrait du répertoire de données Kaggle (2016), contient les transactions par carte de crédit effectuées par les détenteurs de cartes européens en septembre 2013. Il a été collecté et analysé dans le cadre d'une collaboration de recherche entre Worldline et le groupe d'apprentissage automatique de l'Université Libre de Bruxelles pour l'exploration des données et la détection des fraudes (Groupe d'apprentissage automatique, 2018). L'ensemble des données est très déséquilibré ; la classe positive (fraudes) représente 0,172 % de toutes les transactions avec 492 fraudes sur 284 807 transactions (STORN et PRICE, 1997b).
La méthode combinée SMOTE + ENN a été utilisée pour résoudre ce problème en tant que technique de prétraitement utilisée pour traiter des ensembles de données déséquilibrées et l'élimination récursive des caractéristiques avec validation croisée comme méthode de sélection d'attributs. Il est important de noter que pour l'ensemble des données européen, la taille de la population et le nombre maximum de générations sont fixés à 50 et 50 respectivement.
Toutes les techniques de prétraitement et d'évaluation utilisées avec cet ensemble de données sont définies comme suit :

— Éliminations récursives des caractéristiques avec validation croisée : une élimination récursive des caractéristiques avec réglage automatique du nombre de caractéristiques sélectionnées avec validation croisée (GUYON et al., 2002).

Ensemble de données déséquilibré

Dans de nombreuses applications d'apprentissage supervisé, il existe une différence entre les probabilités antérieures des différentes classes. Cette situation est connue sous le nom de problème du déséquilibre entre les classes.
De nombreux groupes de recherche ont découvert qu'un ensemble de données déséquilibré pouvait être l'un des problèmes fondamentaux de l'apprentissage automatique (MURPHEY et al., 2007 ; BREST et al., 2006 ; ZHOU et LIU, 2005) et il est commun dans de nombreux problèmes réels des télécommunications, du web, de l'écologie, du monde de la finance, la médecine, la biologie, etc. Pendant le processus d'apprentissage dans les algorithmes d'apprentissage automatique, si le rapport entre les classes minoritaires et les classes majoritaires est significativement différent, le ML tend à être dominé par les classes majoritaires. En conséquence, le classificateur tend à favoriser la classe majoritaire (appelée classe "négative") et l'exactitude de classification de la classe minoritaire (appelée classe "positive") peut être faible par rapport à l'exactitude de classification de la classe majoritaire. Une étude de Murphey et al. (MURPHEY et al., 2007 ; YANG et WU, 2006) a montré que le réseau neuronal a propagation avant a des difficultés à apprendre à partir des ensembles de données déséquilibrés en raison des instances d'entraînement écrasantes de la classe majoritaire. Le réseau a tendance à ignorer la classe minoritaire et la traite comme du bruit (SEXTON et al., 1998).

Prétraitement dans les ensembles de données déséquilibrés

De nombreuses solutions ont été proposées pour résoudre ce problème. En général, elles peuvent être classées en deux catégories : l'approche au niveau des données et l'approche algorithmique. L'approche au niveau des données vise à modifier les instances de formation pour produire une distribution de classe plus ou moins équilibrée, c'est-à-dire rééquilibrer la distribution de classe avant la classification. L'approche au niveau de l'algorithme vise à modifier le classificateur en ajustant les algorithmes pour reconnaître les classes minoritaires (MURPHEY et al., 2007). Pour l'ensemble de données européen, des techniques de rééchantillonnage, ont été utilisées et elles sont définies comme suit :

— **Technique de sur-échantillonnage des minorités synthétiques (Synthetic Minority Oversampling Technique) (SMOTE)**
L'approche SMOTE est une technique qui peut générer des individus artificiels dans la classe minoritaire. Pour chaque individu de la classe minoritaire, on calcule ses k plus proches voisins de la même classe. Ensuite, un certain nombre de voisins sont sélectionnés (les voisins des k plus proches voisins sont choisis au hasard). Des individus artificiels sont ensuite répartis de manière aléatoire le long de la ligne entre l'individu de la classe minoritaire et ses voisins sélectionnés (CHAWLA et al., 2002).

— **Propriétés asymptotiques des règles du plus proche voisin à l'aide de données éditées (ENN)**
ENN tend à supprimer tout exemple qui est mal classé par ses k plus proches voisins, c'est-à-dire (dont la classe diffère de la majorité de ses KNN) (HE et GARCIA, 2009).

— **SMOTE + ENN**
Tout d'abord, la technique SMOTE est appliquée à l'ensemble de données d'origine et ensuite, la méthode ENN est appliqué (HE et GARCIA, 2009).

Évaluation dans des domaines déséquilibrés

Les mesures de performances les plus utilisées avec les jeux de données déséquilibrés sont Le coefficient de corrélation de Matthews (MCC) (Équation 4.7) et la G-Moyenne (Équation 4.8).

TABLE 5.7 – Comparaison des algorithmes en terme de F-mesure, MCC, GMEAN, ENTROPIE et MSE

DataSets/Algorithmes	BAT-DE	BAT	PSO	CS	MFO	MVO	GWO	BP
PRÉCISION	0.94	0.93	0.90	0.84	0.82	0.86	0.86	0.79
RAPPEL	0.94	0.92	0.90	0.82	0.79	0.83	0.85	0.77
F-mesure	0.94	0.92	0.90	0.92	0.78	0.82	0.84	0.77
ENTROPIE	0.08	0.09	0.13	0.21	0.23	0.18	0.18	0.26
MCC	0.87	0.84	0.79	0.65	0.60	0.68	0.70	0.56
G-MOYENNE	0.94	0.92	0.90	0.82	0.87	0.85	0.84	0.79
MSE	0.07	0.12	0.28	0.29	0.31	0.24	0.30	0.34

Sur la base des résultats obtenus, le tableau 5.7 indique que le BAT-SDE est plus performant que les autres approches dans toutes les méthodes d'évaluation. Une différence importante peut être observée en termes de F-mesure, MCC et MSE. Alors que l'algorithme de rétropropagation a le moins F-mesure, G-MOYENNE, MCC et le plus grand MSE, on peut également noter que l'approche proposée converge avec un MSE plus petit de 0,06 parmi les autres algorithmes.

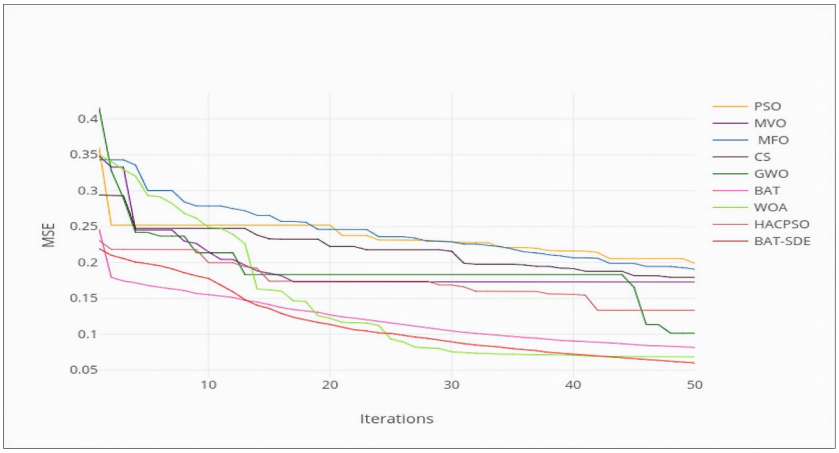

FIGURE 5.6 – Courbe de convergence basée sur le MSE pour Européen dataset

Les résultats obtenus par BAT-SDE sur la base de l'exactitude moyenne et du MSE montrent que cet algorithme peut trouver le meilleur ensemble de poids et de biais et empêcher une convergence prématurée vers l'optimum local.
En outre, le test statistique de Friedman a été utilisé pour évaluer les performances de BAT-SDE par rapport aux autres algorithmes de formation dans chaque exécution indépendante et confirmer la signification des résultats. Cette statistique a été réalisée en classant les différentes techniques (BAT-SDE, PSO, MVO, CS, BAT, MFO, WOA, HACPSO et BP) en fonction des valeurs d'exactitude moyennes pour chaque ensemble de données.

TABLE 5.8 – Classements moyens des algorithmes (le test de Friedman)

Algorithmes	Classement
BAT-SDE	1.35
BAT	9.14
PSO	3.28
CS	4.5
MVO	3.42
MFO	4.5
GWO	8.64
WOA	5.78
HACPSO	6.21
BP	8.14

Le tableau 5.8 montre les classements moyens obtenus par chaque technique d'optimisation dans le test de Friedman (le plus bas est le meilleur). L'étude comparative montre que l'algorithme d'entrainement proposé surpasse les autres algorithmes.

5.3 Conclusion de l'approche

Une nouvelle approche de formation basée sur l'optimisation des chauves-souris avec une évolution différentielle auto-adaptative pour former le réseau de neurones à propagation avant (FFNN) a été proposée. La méthode de formation a pris en compte les capacités de BAT-SDE en termes d'exploration et d'exploitation élevées pour localiser les valeurs optimales pour les poids et les biais de FFNN.

L'approche a été comparée et évaluée à l'aide de cinq ensembles de données biomédicales standard et d'un grand ensemble de données de détection de fraude. De plus, les méthodes de prétraitement SMOTE + ENN ont été utilisées pour traiter le problème fondamental du déséquilibre de classe. Comme le jeu de données européen est très bruyant, la sélection d'attributs a également été utilisée avec l'élimination récursive des caractéristiques et la validation croisée.
La comparaison entre l'algorithme proposé et PSO, CS, MFO, GWO, MVO, WOA, HACPSO et BP montre la supériorité de l'algorithme proposé avec une grande exactitude et une petite MSE dans la plupart des ensembles de données par rapport à autres algorithmes de formation.
De plus, la faible valeur de l'écart type montre que l'algorithme proposé est robuste et stable.
Les résultats obtenus par BAT-SDE sur la base de l'exactitude moyenne et du MSE prouvent que cet algorithme peut trouver le meilleur ensemble de poids et de biais et empêcher une convergence prématurée vers des optima locaux. Les résultats ont été confirmés en utilisant le test de Friedman et comparés avec PSO, CS, MFO, GWO, MVO, WOA, HACPSO et BP le BAT-SDE a le rang le plus élevé parmi toutes les autres méthodes de formation.
Enfin, sur la base de ces expériences, on peut conclure que BAT-SDE a fourni de bons résultats et peut-être considérée comme une alternative aux autres méthodes de formation pour les petits et grands ensembles de données.

Chapitre 6

Sélection automatique des neurones cachés et des poids dans les réseaux de neurones pour la classification des données à l'aide de l'optimisation hybride des essaims de particules et l'optimisation multi-verse basée sur le vol de Lévy

6.1 Introduction

Plusieurs règles sont proposées dans la littérature pour définir le nombre de neurones cachés en fonction des propriétés des données telles que le nombre de classes et le nombre de variables (MIRJALILI, MIRJALILI et LEWIS, 2014b). Comme ce nombre dépend de la nature et de la complexité des données, aucune règle n'a été retenue jusqu'à présent comme étant la meilleure pour définir ce nombre (FARIS, MIRJALILI et ALJARAH, 2019a). Certains chercheurs ont proposé l'utilisation des algorithmes métaheuristiques pour optimiser la structure et la connexion des poids du réseau de neurones simultanément. Par exemple, dans (YU, XI et WANG, 2007), un PSO améliorée a été proposée pour l'optimisation simultanée de la structure et des poids d'un réseau neurones à trois couches. Chaque particule a une structure hiérarchique composée de dimensions à plusieurs niveaux. L'approche a été comparée aux autres algorithmes de classification basée sur huit ensembles de données. Les résultats obtenus ont montré que cet algorithme surpassait les autres algorithmes. Dans (ZHAO et QIAN, 2011) les auteurs ont proposés l'optimisation de la structure et de la connexion des poids du réseau de neurones simultanément en utilisant un algorithme PSO coopératif binaire-réel. La structure est optimisée à l'aide du PSO binaire, tandis que l'optimisation du poids est attribuée au PSO réel.
L'optimisation multi-verse proposée par Mirjalili en 2015 (MIRJALILI, MIRJALILI et HATAMLOU, 2016) est influencée par les idées de trous blancs, de trous noirs et de trous de ver en multivers. La fonction de fitness est indiquée par le taux d'inflation pour chaque agent de recherche. Chaque univers de l'agent de recherche représente une solution candidate et chaque objet de l'univers représente une variable dans la solution candidate (MIRJALILI, MIRJALILI et HATAMLOU, 2016). Les trous noirs ont un taux d'inflation plus faible et les trous blancs ont un taux d'inflation plus élevé. Toute la matière dans l'univers se déplacera aléatoirement à travers le trou de ver pour atteindre l'emplacement optimal de l'univers.
Plusieurs études utilisent l'algorithme MVO dans la littérature. Faris et al. (FARIS, ALJARAH

et MIRJALILI, 2016a) ont résolu le problème de l'entrainement du réseau neurones à propagation avant en utilisant l'algorithme MVO. Dans (SINGH, MEHTA et PRASHAR, 2016), l'algorithme MVO a été proposé pour résoudre le problème de l'ordonnancement économique de la charge (ELD). Faris et al. (FARIS et al., 2018) ont proposé le MVO pour la sélection d'attributs et l'optimisation des paramètres du SVM. Bien que MVO soit plus performant que les autres algorithmes, il n'y a pas d'échange d'informations entre ses particules, il n'a donc pas éliminé les problèmes de vitesse de convergence lente, de faible exactitude et de chute facile dans l'optimum local. Le vol de Lévy est une sorte de marche aléatoire, suivant les règles de la multiplicité des puissances. De temps en temps, des pas importants aident l'algorithme à effectuer une recherche globale. Le vol de Lévy permet de trouver un meilleur équilibre entre l'exploration et l'exploitation des algorithmes, qui a l'avantage d'éviter les optima locaux. Pour cette raison nous avons utilisé dans cette approche une version améliorée de MVO basée sur le vol de Lévy, appelé LMVO (JIA et al., 2019), qui vise à améliorer l'exactitude et la vitesse de convergence de MVO traditionnel.
L'optimisation des essaims de particules est une méthode inspirée par le déplacement d'un groupe d'oiseaux ou un groupe de poissons (KENNEDY et EBERHART, 1995a). Dans le PSO, l'individu est un poisson ou un oiseau qui a une position et une vitesse dans l'espace de recherche. Les particules essaient de suivre leurs meilleures positions locales avant de rechercher la meilleure position globale (KENNEDY et EBERHART, 1995b). Cette approche présente un nouvel algorithme basé sur l'optimisation des essaims de particules (PSO) avec l'optimisation Multi-Verse (MVO) basé sur le vol de Lévy, appelé PLMVO.
Pour évaluer la qualité de l'algorithme proposé, nous avons utilisé trois séries expérimentales. Dans la première, l'algorithme PLMVO est comparé aux algorithmes MVO et PSO pour résoudre un ensemble de 15 fonctions de référence afin de trouver la solution globale. Dans la deuxième expérience, l'algorithme proposé est utilisé pour optimiser le nombre de neurones cachés et les poids de connexion simultanément dans le réseau de neurones à propagation avant (FFNN). Neuf ensembles de données ont été résolus par l'entraîneur proposé. De plus, l'application de l'entraîneur a été étudiée dans le domaine biomédical. Les performances du PLMVO ont été comparées à celles de cinq algorithmes métaheuristiques d'entraînement bien connus dans la littérature : PSO (MENDES et al., 2002b), MFO (FARIS, ALJARAH et MIRJALILI, 2017), MVO (FARIS, ALJARAH et MIRJALILI, 2016a), WOA (ALJARAH, FARIS et MIRJALILI, 2018b), HACPSO (KHAN et al., 2019). Dans la troisième expérience, le PLMVO-MLP proposé est utilisé pour prédire les fichiers exécutables malveillants de Linux. Les performances du PLMVO-MLP ont été comparées à celles de deux travaux précédents qui utilisaient des réseaux de neurones pour prédire les fichiers de logiciels malveillants.

Optimisation des essaims de particules (PSO)

En 1995, Russell Eberhart et James Kennedy ont inventé l'optimisation des essaims de particules, une technique d'optimisation stochastique basée sur la population inspirée par les oiseaux qui affluent autour des sources de nourriture. Comme les autres algorithmes de calcul évolutifs. Dans PSO, chaque individu est un oiseau dans l'espace de recherche. Nous l'appelons une particule. Toutes les particules ont des valeurs de fitness qui sont évaluées par la fonction de fitness (fonction objectif à optimiser) et volent dans l'espace avec une vitesse qui est ajustée dynamiquement en fonction de sa propre expérience de vol (KENNEDY et EBERHART, 1995a).

$$V_{i,j}^{t+1} = V_{i,j}^{t} w + C_1 R_1 (Pbest^t - X^t) + C_2 R_2 (Gbest^t - X^t) \tag{6.1}$$

$$X^{t+1} = X^t + V^{t+1} - i = 1, 2...NP) And (j = 1, 2...NG) \tag{6.2}$$

Où, *Pbest* et *Gbest* désignent la meilleure position de particule et la meilleure position de groupe et w est le poids d'inertie, $w = w^{max} - (\frac{(w^{max}-w^{min})*iteration}{maxiteration})$, C_1,C_2 deux constantes positives r_1,r_2 sont des nombres aléatoires dans l'intervalle de [0, 1], $V_{i,j}^{t+1}$ est la vitesse de j_{th} membre de la particule i_{th} au nombre d'itérations (t) et $(t+1)$. Les nouvelles valeurs de position X^{t+1} sont obtenues en ajoutant les mises à jour de la vitesse déterminées par la formule donnée dans l'équation (6.1).
Les étapes du PSO sont données dans l'algorithme 9.

Algorithm 9 L'algorithme PSO

Initialiser la population
Répéter
for I = 1 to la taille de la population **do**
  Calculez la valeur de la fonction objective.
  if la valeur de fitness est meilleure que la meilleure valeur (*Pbest*) **then**
    Définir la valeur actuelle comme la nouvelle *Pbest*.
    Choisissez la particule avec la meilleure valeur de fitness de toutes les particules comme *Gbest*.
    Calculer une nouvelle vitesse selon l'équation (6.1).
    Mettre à jour la position de la particule conformément à l'équation (6.2)
  end if
end for

Optimiseur multi-vers (MVO)

Optimiseur multi-vers (MIRJALILI, MIRJALILI et HATAMLOU, 2016) est un algorithme basé sur la population proposé par (MIRJALILI, MIRJALILI et HATAMLOU, 2016) en 2015. MVO s'inspire des interactions entre univers via trois concepts : les trous noirs, les trous blancs et les trous de ver dans les théories de la multi-verse et du big bang.
Dans cet algorithme, les modèles de ces trois concepts sont développés pour effectuer l'exploration et l'exploitation et la recherche locale. La fonction de fitness pour chaque agent de recherche est indiquée par le taux d'inflation, et chaque objet et chaque univers dans l'agent de recherche représentent une solution candidate et une variable dans la solution candidate. Dans cet algorithme, les grands univers ont tendance à envoyer des objets à des univers plus petits. Un grand univers est défini sur la base du taux d'inflation dans la théorie multi-verse. Les règles suivantes sont appliquées aux univers du MVO :

- Si le taux d'inflation est plus élevé, la probabilité d'avoir un trou blanc est plus élevée.
- Si le taux d'inflation est plus élevé, la probabilité d'avoir des trous noirs est plus faible.
- Les univers ayant un taux d'inflation plus élevé envoient les objets à travers des trous blancs.
- Les univers ayant un taux d'inflation plus faible ont tendance à recevoir plus d'objets à travers les trous noirs.
- Les objets de tous les univers peuvent être remplacés par les objets de l'univers avec le taux d'inflation le plus élevé.

Il y a deux coefficients principaux dans MVO qui doivent être déterminés en premier pour mettre à jour les solutions. Le taux de distance de déplacement (TDR) et la probabilité d'existence d'un trou de ver (WEP). Ces coefficients déterminent dans quelle mesure les solutions sont modifiées pendant l'optimisation. Les formules adaptatives pour les deux

paramètres sont définies comme suit :

$$WEP = min + t \times (\frac{max - min}{T}) \tag{6.3}$$

$$TDR = 1 - \frac{t^{\frac{1}{p}}}{T^{\frac{1}{p}}} \tag{6.4}$$

Où t est l'itération actuelle, T représente le nombre maximum d'itérations, min indique le minimum (0,2 dans ce document), et max indique le maximum (1 dans ce document).
Où p est la précision d'exploitation (6 dans ce document).
La position des solutions MVO peut être mise à jour à l'aide des équations suivantes :

$$x_i^j = \begin{cases} \begin{cases} x_j + TDR + ((ub_j - lb_j)) * r_4 + lb_j, if\ r_3 < 0.5 \\ x_j - TDR + ((ub_j - lb_j)) * r_4 + lb_j, if\ r_3 \geq 0.5 \end{cases} & if\ r_2 < WEP \\ x_i^j, if\ r_2 \geq WEP \end{cases} \tag{6.5}$$

Où, x_j indique la j_{th} variable dans le meilleur univers, lb_i indique la limite inférieure de la j_{th} variable, ub_i montre la limite supérieure de la j_{th} variable, r_2 ,r_3 , r_4 sont des nombres aléatoires dans l'intervalle $[0, 1]$, TDP/WEP sont des coefficients modifiés pour chaque solution et x_i^j indique le j_{th} paramètre dans l'univers. Les étapes générales de l'algorithme MVO sont présentées dans l'Algorithme 10.
Dans l'algorithme MVO, le processus d'optimisation commence par la génération d'un ensemble de solutions aléatoires et le calcul de leurs objectifs correspondants (MIRJALILI, MIRJALILI et HATAMLOU, 2016).

Algorithm 10 L'algorithme MVO

SU= Univers stockés (Stored universes)
Normaliser le taux d'inflation (fitness)
for chaque objet indexé par i **do**
  $Black-hole-index=i$;
  for chaque objet indexé par j **do**
    r_1=random[0,1]
    if $r_1 < NI(U_i)$ **then**
      $White-hole-index$=$RouletteWheelSelection(-NI)$
      $U(Black-hole-index,j)$=$SU(white-hole-index,j)$
    end if
  end for
end for
for chaque objet indexé par i **do**
  for chaque objet indexé par j **do**
    r_2=random[0,1]
    if $r_2 < Wormhole-existance-probability$ **then**
      r_3=random[0,1];
      r_4=random[0,1];
      if $r_3 < 0.5$ **then**
        $U(i,j)$=$Best-universe(j)+Travelling-distance-rate*((ub_j,lb_j))*r_4+lb_j$
      else
        $U(i,j)$=$Best-universe(j)-Travelling-distance-rate*((ub_j,lb_j))*r_4+lb_j$
      end if
    end if
  end for
end for

Vol de Lévy

Le vol de Lévy introduit par le mathématicien Paul Levy en 1937. Le vol de Lévy est une sorte de marche aléatoire qui se caractérise par une grande taille de pas et un petit nombre de pas (JIA et al., 2019). Le vol de Lévy a une grande longueur de pas et un grand avantage dans l'exploration de l'espace de recherche inconnu et à grande échelle par rapport aux autres types de marche aléatoire. La définition du vol de Lévy est dérivée de la théorie du chaos qui est utile dans la simulation et la mesure stochastique (JIA et al., 2019). Le vol de Lévy peut être mathématiquement défini comme :

$$Levy(\lambda) = 0.01 \times \frac{\mu \times \sigma}{|\nu|^{\frac{1}{\beta}}} \tag{6.6}$$

Où μ et ν sont tirés de la distribution normale, $\lambda = \beta + 1$.

$$\mu \sim N(0,\sigma^2), \nu \sim N(0,\sigma_\nu^2) \tag{6.7}$$

Avec

$$\sigma = \left(\frac{\Gamma(1+\beta)\times sin\frac{\pi\beta}{2}}{\Gamma(\frac{1+\beta}{2})\times\beta\times 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}}, \sigma_\nu = 1 \tag{6.8}$$

La longueur de l'étape s peut être représentée par

$$s = \frac{\mu}{|\nu|^{\frac{1}{\beta}}} \tag{6.9}$$

Où β est une constante. $\beta = 1.5$

Optimisation multi-verse basée sur le vol de Lévy (LMVO)

L'algorithme d'optimisation multi-verse peut résoudre l'optimisation dimensionnelle d'un seul mode (JIA et al., 2019). Mais, la solution obtenue par MVO n'est pas très satisfaisante lorsqu'il s'agit de problèmes d'optimisation multimodale de grandes dimensions (JIA et al., 2019).
Dans l'algorithme MVO, si le trou noir et le trou blanc sont générés autour de la solution optimale actuelle, c'est probablement de faire tomber l'algorithme dans un optimum local (JIA et al., 2019). À ces fins, l'algorithme sera optimisé dans l'univers le plus large et sortira de l'optimisation locale si le trou noir / blanc est reformé à distance par le vol de Lévy (JIA et al., 2019). Le trou de ver aide l'optimisation multi-verse (MVO) pour développer l'espace de recherche sous l'action du vol de Lévy et le trou blanc / trou noir explore l'espace de recherche à travers MVO (JIA et al., 2019).

$$x_i^j = \begin{cases} \begin{cases} x_j + TDR + ((ub_j - lb_j)) * Levy(\lambda) + lb_j, if\ r_3 < 0.5 \\ x_j - TDR + ((ub_j - lb_j)) * Levy(\lambda) + lb_j, if\ r_3 \geq 0.5 \end{cases} & if\ r_2 < WEP \\ x_i^j, if\ r_2 \geq WEP \end{cases} \tag{6.10}$$

Tous les paramètres sont les mêmes que l'algorithme traditionnel sauf $Levy(\lambda)$. En théorie, le MVO avec le vol de Lévy peut trouver une meilleure solution que l'algorithme MVO traditionnel.

Hybride PSO-LMVO (PLMVO)

Le PSO-LMVO est une combinaison séquentielle de PSO et de LMVO (PLMVO) (YANG, 2011). L'algorithme fusionne la meilleure force de PSO en exploitation et de LMVO en exploration vers la solution optimale lorsque la valeur universelle de LMVO remplace le *Pbest* de PSO (YANG, 2011 ; SAGARIKA et JYOTHSNA, 2015 ; KARTHIKEYAN et DHAL, 2015).
(*Remarque :* cette hybridation est de type relais de haut niveau (JOURDAN, 2003)).
Dans cette approche, nous proposons un nouvel algorithme d'apprentissage basé sur cet algorithme pour la première fois dans la section suivante. L'équation peut s'écrire comme suit :

$$V_{i,j}^{t+1} = V_{i,j}^t W + C_1 R_1 (Universes^t - X^t) + C_2 R_2 (Gbest^t - X^t) \tag{6.11}$$

Les étapes du PLMVO peuvent être démontrées comme suit :

1. Étape 1 : initialiser les valeurs de LMVO
2. Étape 2 : évaluer le taux d'inflation de l'univers (fonction de fitness)
3. Étape 3 : mettre à jour la position des univers
4. Étape 4 : si le critère de convergence est atteint ; obtenir les résultats
5. Étape 5 : si le critère de convergence n'est pas atteint ; continuez le processus à partir de l'étape 2-5.
6. Étape 6 : utiliser les solutions optimales de LMVO comme des limites à l'algorithme PSO

7. Étape 7 : initialiser les valeurs de PSO
8. Étape 8 : évaluer la fonction de fitness de chaque particule
9. Étape 9 : déterminer *Gbest* à partir de la valeur de *Pbest*
10. Étape 10 : mise à jour des valeurs de vitesse et de position de chaque particule lorsque la valeur de l'univers de LMVO remplace la valeur *Pbest* de PSO.
11. Étape 11 : vérifier la solution si elle est faisable ou non
12. Étape 12 : les étapes 8 à 12 ont été répétées jusqu'à ce que le nombre maximum d'itérations soit atteint.

PLMVO pour la formation du MLP

Cette section présente l'approche proposée basée sur le PLMVO pour optimiser la structure et le réseau de neurones à propagation avant nommé PLMVO. Deux points importants sont pris en considération : la fonction de fitness et la représentation des solutions de PLMVO qui impliquent la représentation de la population et l'encodage des individus. Dans notre travail, nous avons utilisé la même représentation de la solution que celle utilisée dans (FARIS, MIRJALILI et ALJARAH, 2019b).
L'encodage des individus : Chaque individu est constitué de deux parties principales : une partie de la structure de contrôle et la partie des poids. Ce schéma d'encodage est illustré dans la figure 6.1.

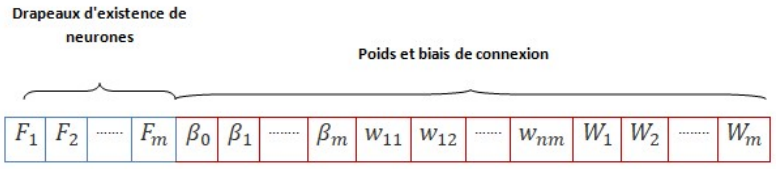

FIGURE 6.1 – Le schéma de codage utilisé pour représenter PLMVO pour la formation du MLP

La première partie est un vecteur de drapeau binaire dans lequel chaque drapeau est responsable de l'existence de son neurone correspondant. La longueur maximale de ce vecteur est égale au nombre maximal possible de neurones dans la couche cachée (FARIS, MIRJALILI et ALJARAH, 2019b). Par conséquent, le neurone existe si un drapeau Fi donné est activé ; sinon, il est supprimé. Pour illustrer cette idée, trois exemples de vecteurs de contrôle aléatoire et leurs réseaux respectifs sont donnés dans la figure 6.2. Dans ce travail, l'algorithme PLMVO a été appliqué pour optimiser les poids et le nombre de neurones avec une seule couche cachée, la deuxième partie des solutions représente les paramètres d'apprentissage (poids et biais) qui ont été formé de trois parties : les poids de connexion entre la couche d'entrée et la couche cachée, les poids entre la couche cachée et la couche de sortie, et les poids de biais (FARIS, ALJARAH et MIRJALILI, 2016b ; FARIS, MIRJALILI et ALJARAH, 2019b).

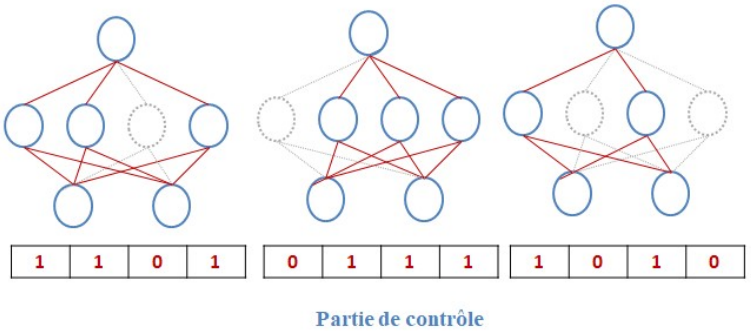

FIGURE 6.2 – Le schéma de codage utilisé pour représenter PLMVO pour la formation du MLP

La longueur de chaque vecteur de solution est donnée par l'équation (6.12), où N est le nombre d'entités d'entrée et m est le nombre maximum de neurones dans la couche cachée (FARIS, MIRJALILI et ALJARAH, 2019b). Dans cette approche, m est égal à $2N+1$ (FARIS, ALJARAH et MIRJALILI, 2016b).
Selon l'équation (6.12), pour représenter les drapeaux de contrôle correspondant aux neurones cachés nous avons besoin de m bits, les poids cachés entre la couche d'entrée et la couche cachée représentent par $N \times m$ bits, pour représenter les poids entre la couche cachée et la couche de sortie nous avons utilisé m bits, les termes de polarisation des neurones dans la couche cachée et la couche de sortie sont représentés par $m+1$ bits (FARIS, MIRJALILI et ALJARAH, 2019b).
Encodage de la population : indiqué dans l'Eq. (6.14), la population est définie comme une matrice R de taille $N \times L$, où N est le nombre d'individus dans la population (FARIS, MIRJALILI et ALJARAH, 2019b). La population à un nombre fixe d'individus où chacun d'eux forme une solution candidate (FARIS, MIRJALILI et ALJARAH, 2019b).
Mappage : chaque solution candidate est mappée à un réseau dans ce processus (FARIS, MIRJALILI et ALJARAH, 2019b).
La solution est divisée en ses parties principales : la partie poids et biais de connexion et la partie contrôle. La partie contrôle est mappée en représentation binaire pour former les drapeaux chargés de déterminer les neurones actifs (FARIS, MIRJALILI et ALJARAH, 2019b). Ce mappage a été fait comme indiqué dans l'équation (6.15), où E_i est l'ième élément de l'individu et $1 \leq i \leq m$ (m est le nombre maximum de neurones cachés autorisés).

$$L = (N \times m) + (3 \times m) + 1 \tag{6.12}$$

Les solutions PLMVO sont implémentées sous forme de vecteurs de nombres réels lorsque chaque vecteur appartient à l'intervalle $[-1, 1]$ (FARIS, ALJARAH et MIRJALILI, 2016b). L'erreur quadratique moyenne (MSE) a été utilisée pour mesurer la valeur de fitness des solutions PLMVO (FARIS, ALJARAH et MIRJALILI, 2016b). La MSE a été calculée sur la base de la différence entre les valeurs estimées et réelles du réseau de neurones à l'aide des ensembles de données d'apprentissage, comme indiqué dans l'équation (6.13), où n est le nombre d'échantillons dans l'ensemble de données d'apprentissage y et $\hat{y}$ sont respectivement les valeurs réelles et prédites :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2 \tag{6.13}$$

$$P = \begin{bmatrix} F_{11} \cdots F_{m1} & \beta_{01} \cdots \beta_{m1} & \omega_{111} \cdots \omega_{nm1} \\ F_{12} \cdots F_{m2} & \beta_{02} \cdots \beta_{m2} & \omega_{112} \cdots \omega_{nm2} \\ \vdots \cdots \vdots & \vdots \cdots \vdots & \vdots \cdots \vdots \\ F_{1N} \cdots F_{mN} & \beta_{0N} \cdots \beta_{mN} & \omega_{11N} \cdots \omega_{nmN} \end{bmatrix} \tag{6.14}$$

$$F_i = \begin{cases} 0 & E_i < 0 \\ 1 & E_i > 0 \end{cases} \tag{6.15}$$

La figure 6.3 représente les étapes de l'approche PLMVO-MLP.

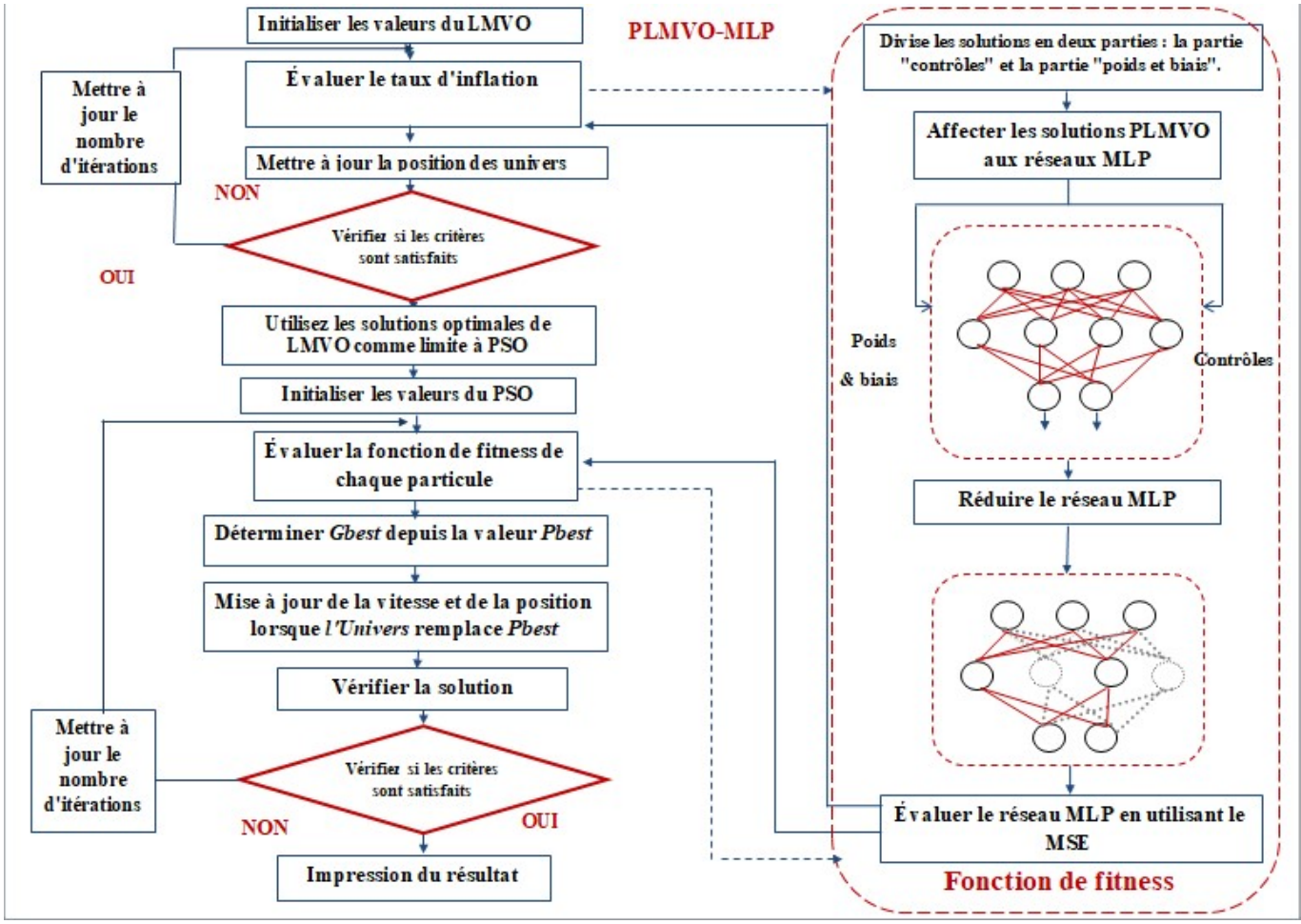

FIGURE 6.3 – Les étapes générales de l'approche PLMVO-MLP

Expérimentation et résultats

Dans cette section, nous utilisons trois expériences pour évaluer la qualité de l'algorithme proposé. Le premier utilise des fonctions de test et nous comparons les résultats aux algorithmes standards PSO et MVO. La deuxième présente l'évaluation de l'algorithme PLMVO proposé pour la formation des réseaux MLP sur neuf ensembles de données, qui ont été sélectionnés à partir des dépôts d'ensembles de données (UCI). Le tableau 6.1 montre la classification de ces ensembles de données en termes de nombre d'attributs, nombre classes, nombre d'instances dans l'ensemble d'apprentissage et du test. La comparaison du PLMVO a été effectuée avec cinq approches utilisées pour former le réseau neurones à propagation avant dans la littérature : PSO (MENDES et al., 2002b), MFO (FARIS, ALJARAH et MIRJALILI,

2017), MVO (FARIS, ALJARAH et MIRJALILI, 2016a), WOA (ALJARAH, FARIS et MIRJALILI, 2018b), De plus, l'algorithme proposé a été comparé à la rétropropagation avec moment et le taux d'apprentissage adaptatif, le PLMVO-MLP proposé est utilisé pour prédire les fichiers exécutables malveillants de Linux. Les performances du PLMVO-MLP ont été comparées à celles de deux travaux antérieurs qui utilisaient des réseaux de neurones pour prédire les fichiers Linux exécutables malveillants.

TABLE 6.1 – Résumé des ensembles de données de classification

Datasets	Attributs	Ensemble d'apprentissage	Ensemble de test
Blood	4	493	255
Breast cancer	8	461	238
Diabetes	8	506	262
Vertebral	6	204	106
Liver	6	79	41
Parkinson	22	128	67
Hepatitis	10	102	53
Heart (Statlog)	13	179	91
BreastEW	31	178973	93986

Dispositif expérimental

Le formateur proposé et d'autres algorithmes ont été mis en œuvre avec le langage Python et un ordinateur personnel équipé d'un processeur Intel(R) Core(TM) 1,60 GHz 2,30 GHz, d'un système d'exploitation Windows 10 64 bits et de 4 Go (RAM). Les métaheuristiques sont sensibles à la valeur de leurs paramètres, ce qui nécessite une initialisation soigneuse. Par conséquent, les paramètres de contrôle recommandés dans la littérature ont été utilisés (MENDES et al., 2002b ; FARIS, ALJARAH et MIRJALILI, 2016b ; ALJARAH, FARIS et MIRJALILI, 2018b) et résumés dans le tableau 5.1. Tous les ensembles de données ont été divisés en 66% pour l'apprentissage et 34% pour le test (FARIS, MIRJALILI et ALJARAH, 2019b). De plus, toutes les caractéristiques ont été mises en correspondance avec l'intervalle [0, 1] pour éliminer l'effet des caractéristiques qui ont des échelles différentes (FARIS, ALJARAH et MIRJALILI, 2016b). La normalisation min-max est appliquée pour effectuer une transformation linéaire sur les données originales (KIRANYAZ et al., 2009b), où v' est la valeur normalisée de v dans l'intervalle $[min_A, max_A]$ comme indiqué dans (6.16).

$$v' = \frac{v_i - min_A}{max_A - min_A} \tag{6.16}$$

Série d'expériences 1 : fonctions de test

Quinze fonctions de référence ont été sélectionnées dans le tableau 6.2 pour évaluer la performance de l'algorithme PLMVO proposé (IBRAHIM et al., 2019). Pour vérifier les résultats, PLMVO a été comparé à PSO (MENDES et al., 2002b) et MVO (MIRJALILI, MIRJALILI et HATAMLOU, 2016). Tous les algorithmes ont été testés 30 fois avec la valeur de dimension D=30 pour chaque fonction de test. Le nombre de générations a été fixé à 1000 et les performances de l'algorithme PLMVO ont été évaluées en fonction de la meilleure, la pire, la moyenne et l'écart-type de la fonction objective.

TABLE 6.2 – Les fonctions d'optimisation

ID	Équation	Lower	Upper	Dim	Type
F1	$f(x) = \sum_{i=1}^{n} x_i^2$	-100	100	30	Unimodal
F2	$f(x) = \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	-10	10	30	Unimodal
F3	$f(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_i)^2$	-100	100	30	Unimodal
F4	$f(x) = max_i \left[\lvert x_i \rvert, 1in \right]$	-100	100	30	Unimodal
F5	$f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	-30	30	30	Unimodal
F6	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	-100	100	30	Unimodal
F7	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1]$	-1.28	1.28	30	Unimodal
F8	$f(x) = \sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i \rvert})$	-500	500	30	Multimodal
F9	$f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10cos(2\pi x_i) + 10 \right]$	-5.12	5.12	30	Multimodal
F10	$f(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e$	-32	32	30	Multimodal
F11	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	-600	600	30	Multimodal
F12	$\frac{\pi}{n}\left[10sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \left[1 + 10sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right] + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -ax_iak(-x_i - a)^m, \quad x_i < -a \end{cases}$	-50	50	30	Multimodal
F13	$f(x) = 0.1\left[sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + sin^2(2\pi x_n)\right]\right] + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	-50	50	30	Multimodal
F14	$f(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}) - 1$	-65.536	65.536	2	Multimodal
F15	$f(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	-5	5	4	Multimodal

TABLE 6.3 – Les résultats statistiques des algorithmes PLMVO et MVO avec 15 fonctions de test

F.No	PLMVO				MVO			
	Avg	STD	Best	Worst	Avg	STD	Best	Worst
F1	0.125	0.039	0.063	0.249	0.192	0.052	0.099	0.310
F2	0.208	0.042	0.131	0.289	0.302	0.072	0.199	0.444
F3	19.494	8.127	8.673	48.637	19.008	8.376	7.577	48.430
F4	0.656	0.269	0.303	1.486	0.579	0.210	0.309	1.0946
F5	169.209	546.494	9.783	3097.027	435.494	791.939	27.920	2648.715
F6	0.117	0.030	0.0451	0.1770	0.180	0.054	0.090	0.293
F7	0.012	0.004	0.002	0.022	0.013	0.006	0.003	0.024
F8	-8063.907	613.740	-9338.537	-6425.713	-4945.161	1550.403	-3002.279	-6883.158
F9	86.200	30.477	38.809	180.299	107.478	28.311	47.845	162.282
F10	0.711	0.707	0.078	2.605	0.722	0.646	0.080	1.966
F11	0.011	0.012	1.58E-10	0.049	0.414	0.081	0.258	0.530
F12	1.342	0.716	0.226	3.316	1.706	1.001	0.2104	3.857
F13	0.004	0.009	3.967E-11	0.043	0.020	0.011	0.006	0.046
F14	0.998	0.000	0.998	0.998	0.998	0.000	0.998	0.999
F15	0.002	0.005	0.0003	0.0203	0.003	0.007	0.0003	0.0203

TABLE 6.4 – Les résultats statistiques des algorithmes PLMVO et PSO avec 15 fonctions de test

F.No	PLMVO				PSO			
	Avg	STD	Best	Worst	Avg	STD	Best	Worst
F1	0.125	0.039	0.063	0.249	0.192	0.000	9.75E-10	4.652E-07
F2	0.208	0.042	0.131	0.289	5.333	6.700	1.34E-05	20.00
F3	19.494	8.127	8.673	48.637	14.641	6.261	6.253	32.793
F4	0.656	0.269	0.303	1.486	0.593	0.125	0.362	0.831
F5	169.209	546.494	9.783	3097.027	291.711	430.043	27.110	1671.129
F6	0.117	0.030	0.0451	0.1770	0.000	0.000	1.56E-07	4.46E-07
F7	0.012	0.004	0.002	0.022	3.447	3.976	0.024	16.148
F8	-8063.907	613.740	-9338.537	-6425.713	-7631.743	528.781	-8606.851	-6830.447
F9	86.200	30.477	38.809	180.299	95.358	27.714	49.911	186.758
F10	0.711	0.707	0.078	2.605	0.000	0.000	2.23E-05	0.0006
F11	0.011	0.012	1.58E-10	0.049	0.318	0.076	0.1504	0.532
F12	1.342	0.716	0.226	3.316	0.031	0.047	3.55E-11	0.103
F13	0.004	0.009	3.967E-11	0.043	0.032	0.012	0.008	0.058
F14	0.998	0.000	0.998	0.998	2.119	1.683	0.998	6.903
F15	0.002	0.005	0.0003	0.0203	0.003	0.007	0.0003	0.020

Les tableaux 6.3 et 6.4 indiquent que l'algorithme PLMVO a surpassé les deux autres algorithmes. Plus précisément, l'algorithme PLMVO a les valeurs les plus faibles dans la mesure moyenne en F2, F5, F7, F8, F9, F11, F13, et F15. En ce qui concerne le meilleur, l'algorithme proposé a dépassé les autres algorithmes dans 7/15 fonctions : F4, F5, F7, F8, F9, F11, et F13. En termes de pire, le PLMVO a obtenu de meilleurs résultats que le PSO et le MVO dans 4/15 fonctions : F2, F7, F11, et F13. En outre, on peut également constater que le PLMVO a une petite valeur d'écart-type qui prouve l'efficacité de cet algorithme. D'après

les résultats, il est clair que l'hybridation entre le PSO et le LMVO donne de meilleurs résultats par rapport aux PSO et LMVO. Ceci est dû à la combinaison de PSO en exploitation et de LMVO en exploration, qui a crée un bon équilibre entre l'exploitation et l'exploration, et permet d'éviter les minima locaux.

Série d'expériences 2 : l'entrainement du réseau de neurones à propagation avant

Dans cette section, nous présentons l'évaluation de l'algorithme proposé pour la formation des réseaux MLP sur neuf ensembles de données bien connus. Chaque ensemble de données a été divisé en deux parties pour évaluer le MLP, 66% pour l'apprentissage, et 34% pour le test. Tous les algorithmes ont été testés dix fois pour chaque ensemble de données et les performances du MLP ont été évaluées en fonction des valeurs meilleures, pires, moyennes et l'écart-type de l'exactitude de la classification et de la MSE. La taille de la population et le nombre maximum de générations ont été fixés à 50 et 200, respectivement.

Le tableau 6.5 présente les résultats statistiques : moyenne, meilleure, pire et écart type de l'exactitude de la classification. Les résultats du PLMVO ont surpassé les autres approches pour Breast cancer, Blood, Liver, Vertebral, Parkinson, Hepatitis, Diabetes, et Heart avec une exactitude moyenne de 0,963, 0,769, 0,739, 0,839, 0,921, 0,975, 0,790 et 0,757. En outre, le PLMVO a donné les mêmes résultats que le MVO dans l'ensemble de données BreastEW avec une exactitude moyenne de 1,00. En outre, on peut également constater que le PLMVO a un écart type plus petit qui indique que le PLMVO est robuste et stable.
En termes de meilleure exactitude, PLMVO-MLP a donné de meilleurs résultats dans 3/9 ensembles de données : Parkinson, Hepatitis, BreastEW avec une meilleure exactitude de 0,927, 0,989 et 0,867, respectivement. Pour BreastEW et Breast cancer, PLMVO-MLP offre les mêmes meilleures performances que MVO et HACPSO. Pour la pire exactitude de classification, notre algorithme a surpassé les autres algorithmes pour Breast cancer, Parkinson, et Hepatitis avec des valeurs de 0,958, 0,882 et 0,898, respectivement.
(FARIS, ALJARAH et MIRJALILI, 2016b).

TABLE 6.5 – Résultats de l'exactitude de la classification

		PLMVO	PSO	MFO	MVO	WOA	HACPSO	BP
B.cancer	Avg	0.963	0.959	0.958	0.958	0.956	0.959	0.744
	STD	0.001	0.003	0.002	0.002	0.006	0.004	0.254
	Best	0.965	0.960	0.963	0.960	0.963	0.965	0.945
	Worst	0.958	0.953	0.954	0.954	0.947	0.952	0.680
	CPU time (s)	3605.50	2114.57	2120.06	2333.28	34195.09	3933.65	1335.83
Blood	Avg	0.769	0.762	0.763	0.764	0.762	0.760	0.744
	STD	0.009	0.002	0.005	0.009	0.004	0.003	0.254
	Best	0.788	0.765	0.765	0.784	0.774	0.760	0.945
	Worst	0.760	0.760	0.760	0.760	0.758	0.758	0.680
	CPU time (s)	3331.40	2524.05	2173.41	2147.15	18643.37	4911.2	1234.44
Diabetes	Avg	0.790	0.780	0.724	0.788	0.720	0.768	0.619
	STD	0.004	0.011	0.008	0.006	0.028	0.006	0.080
	Best	0.795	0.796	0.735	0.802	0.750	0.774	0.690
	Worst	0.738	0.758	0.709	0.782	0.657	0.764	0.601
	CPU time (s)	3398.15	2369.64	1992.49	2096.46	81951.03	3954.74	1259.18
Liver	Avg	0.739	0.722	0.722	0.737	0.665	0.679	0.519
	STD	0.006	0.017	0.010	0.008	0.030	0.020	0.055
	Best	0.746	0.748	0.744	0.732	0.700	0.709	0.586
	Worst	0.726	0.700	0.709	0.709	0.603	0.669	0.495
	CPU time (s)	1923.27	1141.52	928.99	1000.76	7723.80	2497.16	712.66
Vertebral	Avg	0.839	0.836	0.836	0.836	0.774	0.817	0.651
	STD	0.004	0.012	0.007	0.005	0.052	0.015	0.170
	Best	0.843	0.848	0.843	0.838	0.862	0.833	0.866
	Worst	0.823	0.808	0.820	0.828	0.676	0.794	0.627
	CPU time (s)	1870.43	1015.52	1118.60	1019.84	9623.48	2640.55	693.08
Parkinson	Avg	0.921	0.836	0.802	0.890	0.824	0.876	0.750
	STD	0.013	0.040	0.022	0.046	0.016	0.013	0.199
	Best	0.927	0.898	0.828	0.906	0.852	0.905	0.849
	Worst	0.882	0.773	0.773	0.835	0.800	0.863	0.623
	CPU time (s)	2002.64	1488.58	1033.22	1042.92	9228.47	2640.55	742.07
Hepatitis	Avg	0.975	0.914	0.866	0.867	0.791	0.854	0.758
	STD	0.013	0.017	0.016	0.014	0.063	0.015	0.155
	Best	0.989	0.947	0.894	0.894	0.867	0.875	0.865
	Worst	0.947	0.884	0.842	0.842	0.695	0.828	0.669
	CPU time (s)	1729.04	1124.42	1054.07	1030.06	9128.25	1218.35	640.69
Heart	Avg	0.757	0.704	0.733	0.741	0.703	0.738	0.651
	STD	0.063	0.019	0.044	0.039	0.064	0.020	0.020
	Best	0.867	0.722	0.816	0.827	0.822	0.761	0.704
	Worst	0.660	0.665	0.700	0.700	0.577	0.705	0.577
	CPU time (s)	2089.19	1115.34	1076.17	1199.61	35288.82	2353.68	774.14
BreastEW	Avg	1.000	0.966	0.993	1.00	0.973	0.994	0.865
	STD	0.000	0.014	0.006	0.000	0.026	0.006	0.080
	Best	1.000	0.994	1.000	1.000	0.994	1.000	0.892
	Worst	1.000	0.955	0.978	1.000	0.907	0.981	0.884
	CPU time (s)	5609.38	2114.57	2858.62	3089.03	20421.69	3503.72	2078.53

TABLE 6.6 – Résultats de MSE

		PLMVO	PSO	MFO	MVO	WOA	HACPSO	BP
B.cancer	Avg	0.030	0.038	0.033	0.032	0.047	0.040	0.049
	STD	0.001	0.003	0.001	0.002	0.004	0.001	0.015
	Best	0.029	0.044	0.310	0.030	0.043	0.038	0.030
	Worst	0.0326	0.032	0.036	0.032	0.058	0.041	0.050
Blood	Avg	0.165	0.178	0.172	0.170	0.180	0.169	0.174
	STD	0.007	0.003	0.005	0.008	0.004	0.003	0.009
	Best	0.160	0.182	0.174	0.170	0.174	0.162	0.172
	Worst	0.174	0.182	0.177	0.181	0.187	0.175	0.175
Diabetes	Avg	0.149	0.155	0.171	0.149	0.186	0.163	0.179
	STD	0.001	0.004	0.005	0.001	0.013	0.002	0.066
	Best	0.147	0.151	0.165	0.145	0.168	0.160	0.168
	Worst	0.151	0.157	0.175	0.151	0.213	0.166	0.180
Liver	Avg	0.186	0.191	0.193	0.188	0.220	0.212	0.210
	STD	0.001	0.003	0.002	0.004	0.007	0.002	0.003
	Best	0.183	0.185	0.189	0.179	0.210	0.208	0.190
	Worst	0.188	0.195	0.194	0.194	0.233	0.215	0.220
Vertebral	Avg	0.125	0.130	0.128	0.127	0.163	0.146	0.168
	STD	0.002	0.002	0.002	0.001	0.018	0.002	0.015
	Best	0.124	0.125	0.125	0.125	0.137	0.142	0.160
	Worst	0.130	0.130	0.132	0.129	0.202	0.147	0.172
Parkinson	Avg	0.078	0.141	0.094	0.085	0.165	0.119	0.158
	STD	0.014	0.023	0.005	0.003	0.030	0.005	0.018
	Best	0.080	0.099	0.101	0.0873	0.127	0.112	0.137
	Worst	0.089	0.182	0.084	0.081	0.228	0.130	0.203
Hepatitis	Avg	0.036	0.097	0.075	0.042	0.129	0.106	0.168
	STD	0.010	0.032	0.011	0.008	0.022	0.004	0.015
	Best	0.018	0.034	0.100	0.027	0.098	0.100	0.159
	Worst	0.056	0.157	0.109	0.051	0.173	0.114	0.173
Heart	Avg	0.169	0.198	0.178	0.178	0.203	0.191	0.205
	STD	0.034	0.009	0.022	0.021	0.030	0.005	0.050
	Best	0.113	0.189	0.133	0.152	0.133	0.182	0.194
	Worst	0.194	0.208	0.196	0.198	0.243	0.199	0.243
BreastEW	Avg	0.001	0.040	0.017	0.005	0.041	0.017	0.060
	STD	0.000	0.009	0.005	0.001	0.024	0.004	0.015
	Best	0.0008	0.054	0.008	0.004	0.010	0.009	0.040
	Worst	0.0017	0.022	0.023	0.005	0.070	0.024	0.075

Le tableau 6.6 montre la moyenne, la meilleure et la pire MSE avec l'écart type, obtenue pour chaque algorithme. D'après le résultat, on peut noter que le PLMVO a surpassé les autres techniques dans : Breast cancer, Blood, Liver, Vertebral, Parkinson, Heart ,Hepatitis et BreastEW avec une MSE moyenne de 0,030, 0, 165, 0,186, 0,126, 0,078, 0,169, 0,036 et 0,001 respectivement. En outre, on peut également remarquer que PLMVO a une petite valeur d'écart type pour tous les jeux de données, ce qui prouve la robustesse et l'efficacité de cet algorithme. En termes de meilleur MSE, il a donné de meilleurs résultats pour 4/9 ensembles de données : Blood, Liver, Heart, et BreastEW avec des valeurs de 0,174, 0,184, 0,194 et 0,0017, respectivement, et il a donné les mêmes résultats que MVO pour l'ensemble

de données Diabetes. Pour le pire MSE, le PLMVO-MLP à mieux performé que les autres techniques pour Breast cancer, Blood, Vertebral, Parkinson, Heart et BreastEw. En outre, il est classé le deuxième pour les autres.

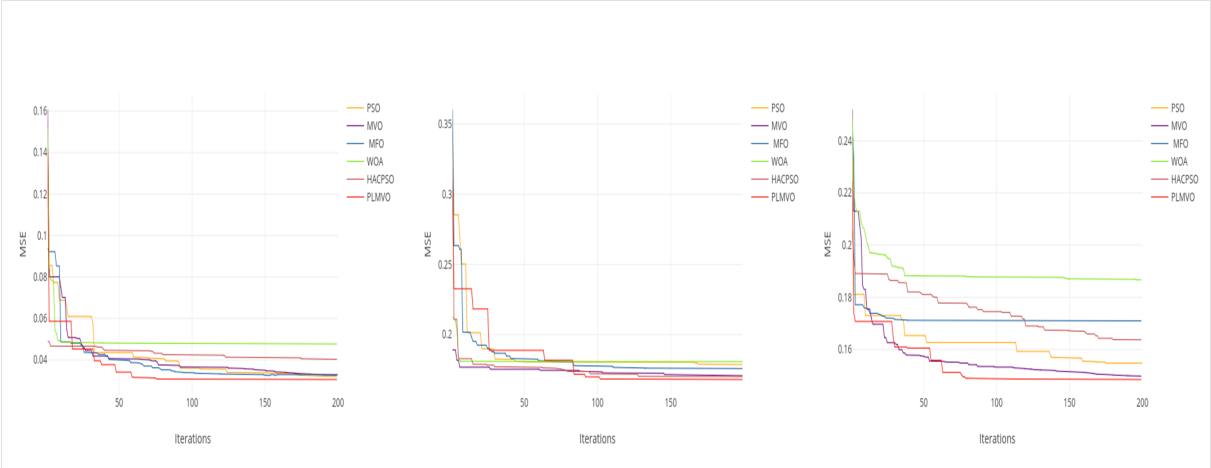

FIGURE 6.4 – Courbe de convergence basée sur le MSE pour Breast cancer, Blood et Diabete datasets, respectivement

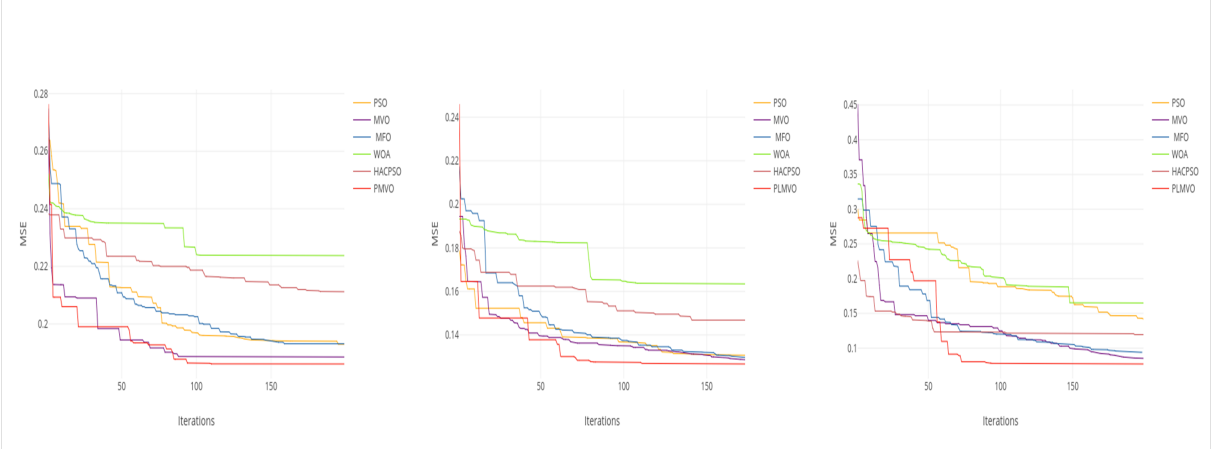

FIGURE 6.5 – Courbe de convergence basée sur le MSE pour Liver, Vertebral et Parkinson datasets, respectivement

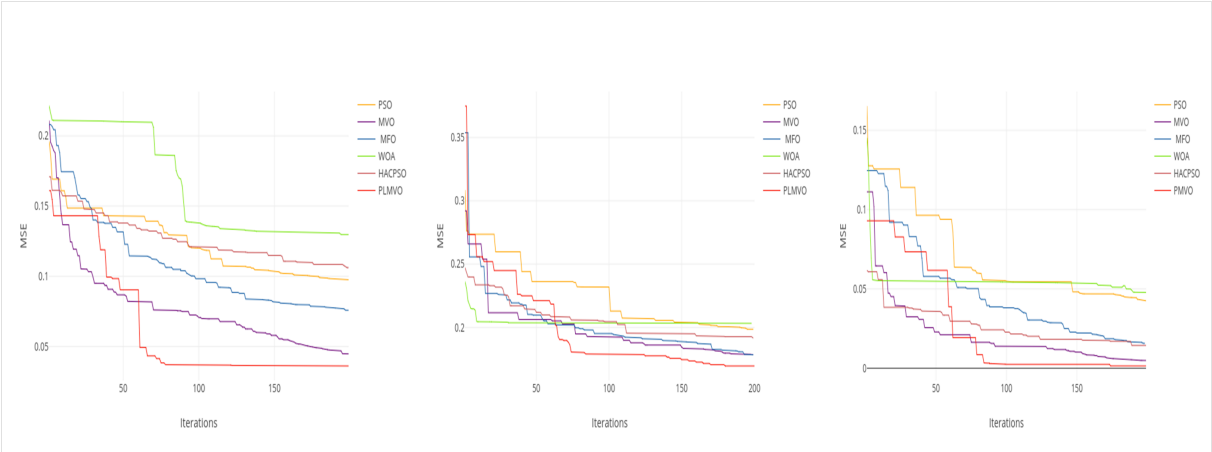

FIGURE 6.6 – Courbe de convergence basée sur le MSE pour Hepatitis, Heart et BrestEW datasets, respectivement

Les figures 6.4, 6.5, 6.6 montrent les courbes de convergence de tous les algorithmes métaheuristiques basés sur les valeurs moyennes de MSE. Les courbes de convergence montrent que le PLMVO a la valeur la plus basse de MSE pour huit ensembles de données : Breast cancer, Blood, Liver, Vertebral, Parkinson, Hepatitis, Heart et BreastEW.
Les chiffres montrent que le PLMVO a la vitesse de convergence la plus rapide dans : Breast cancer, Diabetes, Liver, Vertebral and Hepatitis.

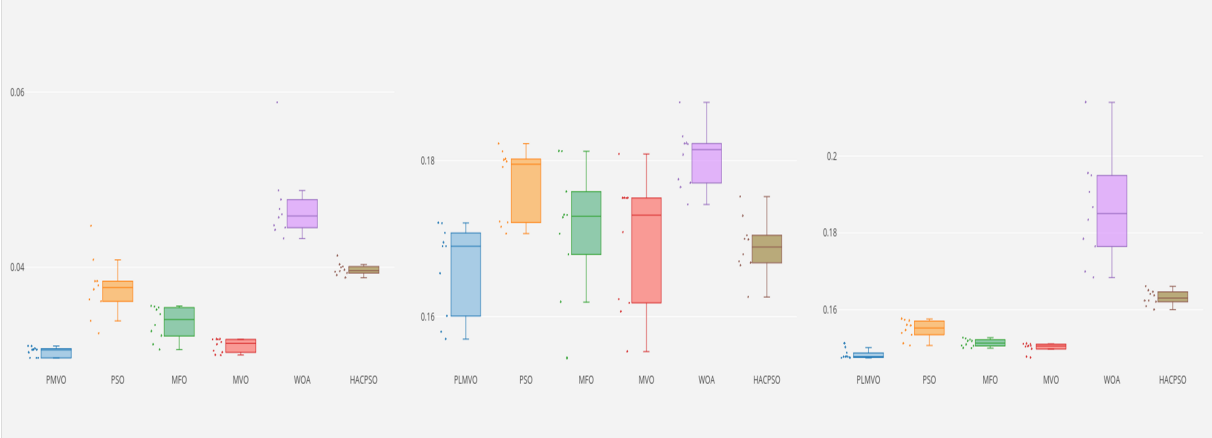

FIGURE 6.7 – Boxplot basé sur le MSE pour Breast cancer, Blood et Diabetes datasets, respectivement

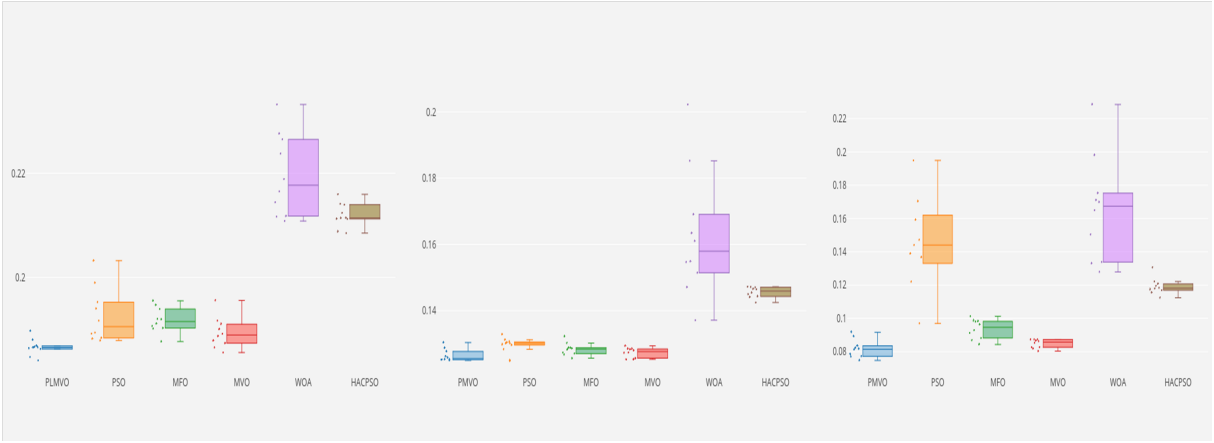

FIGURE 6.8 – Boxplot basé sur le MSE pour Liver, Vertebral et Parkinson datasets, respectivement

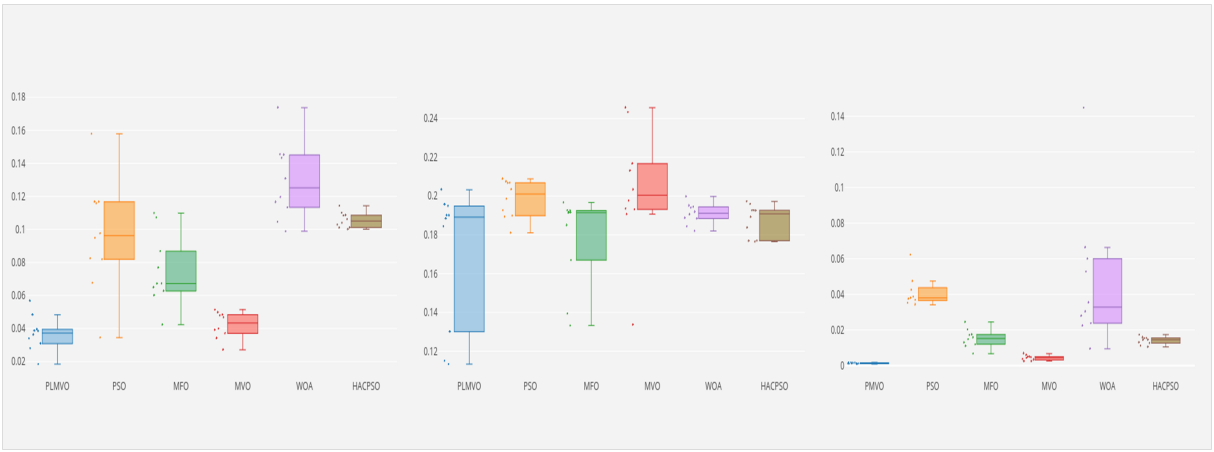

FIGURE 6.9 – Boxplot basé sur le MSE pour Hepatitis, Heart et BrestEW datasets, respectivement

Les figures 6.7, 6.8, 6.9 illustrent le diagramme en boîte (boxplot) relative à 10 exécutions de PLMVO, MVO, PSO, MFO, WOA, HACPSO, et BP. Les diagrammes en boîte sont utilisés pour analyser la variabilité des optimiseurs de MLP pour obtenir les valeurs MSE dans toutes les exécutions. Dans ce graphe, la boîte fait référence à l'intervalle interquartile, les moustaches sont la MSE la plus éloignée, la barre dans la boîte est la médiane et les valeurs aberrantes sont représentées par un petit cercle. Les diagrammes en boîte montrent et justifient la meilleure optimisation de PLMVO dans la formation de MLP.
Les résultats obtenus par PLMVO sur la base de l'exactitude moyenne et du MSE prouvent que cet algorithme peut trouver les valeurs optimales des structures et des poids de connexion et éviter une convergence prématurée vers des optima locaux. En outre, le test statistique de Friedman a été utilisé pour évaluer les performances de PLMVO par rapport aux autres algorithmes d'entraînement dans chaque exécution indépendante et confirmer la signification des résultats. Cette statistique a été réalisée en classant différentes techniques (PLMVO, PSO, MFO, MVO, WOA, HACPSO et BP) en fonction des valeurs de l'exactitude moyennes pour chaque ensemble de données.

TABLE 6.7 – Classement moyen des algorithmes en fonction de l'exactitude moyenne (test de Friedman)

Algorithmes	Classement
PLMVO	1.10
PSO	3.72
MFO	4.11
MVO	2.38
WOA	5.61
HACPSO	4.05
BP	7.00

TABLE 6.8 – Classement moyen des algorithmes en fonction de de la meilleure exactitude (test de Friedman)

Algorithmes	Classement
PLMVO	2.40
PSO	3.49
MFO	3.55
MVO	3.49
WOA	4.09
HACPSO	4.64
BP	5.84

TABLE 6.9 – Classement moyen des algorithmes en fonction de de la plus mauvaise exactitude (test de Friedman)

Algorithmes	Classement
PLMVO	1.04
PSO	4.47
MFO	4.15
MVO	2.18
WOA	5.67
HACPSO	4.20
BP	5.78

Les tableau 6.7, 6.8 et 6.9 montrent les classements obtenus par chaque technique d'optimisation dans le test de Friedman (le plus bas est le meilleur). L'étude comparative montre que l'algorithme d'entrainement proposé a dépassé les autres algorithmes. D'après les résultats, nous pouvons voir que le schéma de codage hybride utilisé dans cette approche pour optimiser les valeurs optimales des structures et des poids de connexion facilite le processus d'optimisation pour toutes les métaheuristiques car la méthode permet aux algorithmes métaheuristiques d'atteindre une faible MSE avec une exactitude de classification élevée.
Les algorithmes d'intelligence en essaim tels que le PSO bénéficient d'un bon taux de convergence et d'une bonne exploitation mais peuvent être bloqués dans un optimum local (KENNEDY et EBERHART, 1995b). La combinaison de PSO et de LMVO a donné de meilleurs résultats pour la plupart des ensembles de données par rapport aux autres algorithmes car le PLMVO peut créer un bon compromis entre l'exploration et l'exploitation. Après les expériences, il est clair que le PLMVO surpasse d'autres algorithmes bien considérés dans la littérature.

L'hybridation entre PSO et LMVO a un impact majeur sur la recherche globale.
Dans certaines situations, la PSO peut être piégée dans un optimum local, ce qui entraîne une précision et une exactitude inférieures (KENNEDY et EBERHART, 1995b). En outre, la combinaison de PSO en exploitation avec LMVO en exploration donne de meilleurs résultats grâce à l'exploration supérieure de l'algorithme LMVO ; les trous noirs et blancs permettent aux objets de se déplacer entre les univers pour faciliter l'exploration et empêcher les optima locaux. Les changements dans les univers brusques sont utiles pour faire stagner l'optimum local (MIRJALILI, MIRJALILI et HATAMLOU, 2016). De plus, si le trou noir / trou blanc est modifié par le vol de Lévy, l'algorithme sera optimisé dans le grand univers et sortira de l'optimum local. La trajectoire du vol Lévy permet de parvenir à un meilleur équilibre entre l'exploitation et l'exploitation dans le MVO (JIA et al., 2019). On peut conclure que le LMVO peut contribuer à l'exploration et le PSO à l'exploitation. Une bonne exploitation implique une vitesse de convergence et une bonne exploration empêche la stagnation des optima locaux.

Série d'expériences 3 : classification et détection des logiciels malveillants

Un logiciel malveillant peut être décrit comme tout type de code malveillant susceptible d'endommager un ordinateur ou un réseau. Chaque année, le volume de logiciels malveillants augmente plus rapidement et constitue une menace sérieuse pour la sécurité mondiale. La détection des logiciels malveillants est devenue par la suite un sujet crucial de la sécurité informatique. Avec la variété croissante des menaces, les solutions de protection actuelles ne sont plus en mesure de faire leur travail correctement. Les programmes antivirus classiques utilisent des signatures statiques pour détecter les logiciels malveillants. Malgré l'utilisation étendue de cet outil, les logiciels malveillants ne peuvent être identifiés qu'après que le dommage a déjà été causé par l'exécutable malveillant et à condition que le logiciel malveillant soit correctement enregistré. Cette technique atteint ses limites face à des logiciels malveillants évolutifs et polymorphes. Ces dernières années, des algorithmes d'apprentissage automatique sont utilisés pour effectuer une analyse efficace des logiciels malveillants. Schultz et ses collaborateurs (SCHULTZ et al., 2000) ont été les premiers à appliquer des modèles d'exploration de données et des algorithmes d'apprentissage automatique pour détecter les programmes malveillants sur la base de leurs codes binaires respectifs. En particulier, ils ont appliqué des classificateurs multiples à trois approches d'extraction de fonctions : les fonctions de chaîne, les en-têtes de programme et les fonctionnalités de séquence d'octets. Perdisci et coll. (PERDISCI, LANZI et LEE, 2008) ont suggéré leur première méthode basée sur l'extraction de caractéristiques de l'exécutable portable (PE) et la classification à l'aide d'algorithmes d'apprentissage machine, par exemple, J48, MLP, et Naïve Bayes. Un système de classification des logiciels malveillants développé avec un réseau de neurones à deux couches cachées a été proposé par Saxe et Berlin (SAXE et BERLIN, 2015). Ils utilisent des caractéristiques statiques, notamment l'entropie, l'importation PE, les métadonnées et l'histogramme 2D de chaîne. Les deux couches cachées étaient composées de 1024 unités linéaires rectifiées paramétriques (PReLu), tandis que leur couche de sortie était un neurone sigmoïde qui classait l'instance comme bénigne ou malveillante. En menant une expérience sur un ensemble de données de 400 000 échantillons, ils ont obtenu un taux de détection de 95 avec un taux de faux positifs de 0.1. Les universitaires et l'industrie antivirus se sont concentrés sur le développement des applications et des méthodes utilisées pour analyser et détecter les logiciels malveillants de Windows. Cela est probablement dû à la popularité et à la part de marché du système d'exploitation Windows (PANDIT, 2019). La croissance rapide de l'Internet des objets fait évoluer la détection des malwares vers des systèmes basés sur Linux. Comme la plupart de ces périphériques embarqués sont capables d'interagir les uns avec les autres en fonction du système d'exploitation Linux. Asmitha et coll. (ASMITHA

et VINOD, 2014) ont proposé une nouvelle approche utilisant des algorithmes d'apprentissage automatique : Naïve bayes, J48, Adaboost, IBK et les forêts aléatoires pour identifier les fichiers exécutables linux malveillants. Les auteurs ont établi la meilleure combinaison d'échantillons bénins et malveillants pour créer un modèle de classification capable de classer les logiciels bénins et malveillants. L'expérimentation montre des résultats prometteurs avec une exactitude de classification de 97%.
Dans notre approche, nous avons utilisé les données comportementales de Cozzi et al. (COZZI et al., 2018). extraite lors de l'exécution d'échantillons dans un pipeline d'analyse, et nous testons la capacité de notre algorithme PLMVO-MLP à prédire si un exécutable Linux est malveillant ou non. Les principales contributions de cette approche sont :

1. Démontrer que notre réseau de neurones PLMVO-MLP peut prédire les fichiers Linux exécutables malveillants en utilisant les données d'activité de la machine avec la meilleure exactitude, Fmesure, précision et, rappel.
2. Démontrer que la sélection d'attributs à l'aide de l'algorithme PSO peut améliorer les résultats.
3. Démontrer que l'exécutable Linux peut être prédit comme étant malveillant ou non avec un haut niveau d'exactitude en utilisant l'algorithme PLMVO-MLP.

Ensemble de données :

Dans notre approche, nous avons utilisé les ensembles de données de Cozzi et al. (COZZI et al., 2018) le fichier csv est créé par (PANDIT, 2019) contient des valeurs importantes issues de l'analyse des échantillons de Linux.
Shivam et al (PANDIT, 2019) ont utilisé une analyse ELF multi-architecture en ligne appelée Padawan (PADAWAN, 2018). La plate-forme avait déjà des rapports d'analyse de plusieurs échantillons Linux. Ces rapports étaient disponibles au format JSON. La bibliothèque python a été utilisée pour la conversion de JSON en csv, et plusieurs fichiers csv ont été créés. L'ensemble de données pour les logiciels malveillants de Linux a été créé. L'ensemble de données final contient 58 attributs et 10548 échantillons. Pour l'apprentissage automatique, des échantillons malveillants et bénins sont nécessaires. Afin d'obtenir des échantillons bénins, les auteurs ont utilisé la valeur vt.positives. (Si vt.positives = 1, cette étiquette est bénigne). L'ensemble de données final contenait 2513 échantillons, 1978 malveillants et 535 bénins, comme indiqué dans le tableau 6.10.

TABLE 6.10 – L'ensemble de données utilisé pour la prédiction des logiciels malveillants pour Linux

	N ° d'échantillons
Bénigne	535
Malicieux	1978
Totale	2513

L'approche proposée pour détecter les malwares Linux

Dans cette section, nous représentons notre approche pour détecter les logiciels malveillants de Linux. Les principales contributions de cette approche sont :

— La sélection du meilleur ensemble d'attributs à partir de l'ensemble de données d'origine en utilisant l'algorithme PSO.
— L'utilisation de l'algorithme que nous proposons, PLMVO-MLP, pour prédire et détecter les logiciels malveillants pour Linux.

La sélection d'attributs (FS) couramment utilisé pour réduire les problèmes liés à un jeu de données à grande dimension (IBRAHIM et al., 2019) . Cette tâche permet d'extraire les informations les plus représentatives à partir de données de grande taille, ce qui réduit l'effort de calcul dans d'autres tâches telles que la classification (IBRAHIM et al., 2019). L'utilisation de la sélection d'attributs présente quatre avantages réels :

1. Améliorer la précision : améliorer les performances des algorithmes d'apprentissage automatique si les bons attributs sont choisis (IBRAHIM et al., 2019).
2. Réduit le temps d'apprentissage : rendre les algorithmes d'apprentissage automatique capables de s'entraîner plus rapidement (IBRAHIM et al., 2019).
3. Réduire la complexité de l'algorithme : réduit la complexité du modèle d'apprentissage automatique et facilite son interprétation (IBRAHIM et al., 2019).
4. Réduit le sur-ajustement : les décisions sont plus robustes car moins de données redondantes impliquent moins de bruit (IBRAHIM et al., 2019).

Ces dernières années, un grand nombre de méthodes d'optimisation basées sur les métaheuristiques ont été proposées pour la sélection des caractéristiques (attributs). Dans ce travail, l'ensemble de données contient 57 caractéristiques donc, pour améliorer la précision et l'exactitude de notre modèle et pour rendre la classification plus rapide, nous avons utilisé l'optimisation de l'essaim de particules (PSO) (KENNEDY et EBERHART, 1995b) pour sélectionner le meilleur ensemble de caractéristiques en fonction de la fonction objective qui représente la valeur de l'exactitude de MLP.
Il y a deux clés importantes qui doivent être abordées dans la mise en œuvre de PSO pour la sélection d'attributs : le schéma de codage et la fonction de fitness (fonction objectif).
Ces clés sont décrites comme suit :

— Schéma d'encodage : les individus PSO sont représentés sous forme de vecteurs de nombres réels (KENNEDY et EBERHART, 1995b). Dans chaque vecteur, le nombre d'éléments est égal au nombre d'attributs dans l'ensemble de données (FARIS et al., 2018). Chaque élément est un nombre aléatoire généré dans l'intervalle [0, 1]. Les éléments représentant des attributs sont arrondis : si l'élément est supérieur ou égal à 0,5, la valeur doit être arrondie à 1 et l'attribut est choisi, sinon la valeur est arrondie à 0 et l'attribut n'est pas choisi (FARIS et al., 2018) comme indique la figure 6.10.

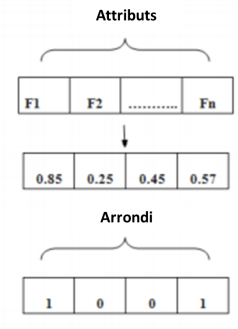

FIGURE 6.10 – Schéma de codage des individus PSO pour la sélection d'attributs (FARIS, MIRJALILI et ALJARAH, 2019b)

— Fonction de fitness : la fonction objective était l'exactitude du modèle MLP. Sur la base de la matrice de confusion présentée dans le tableau 6.11, l'exactitude de la classification est calculée comme indiquée dans l'équation 6.17.

$$Exactitude(Accuracy) = \frac{TP + TN}{TP + FN + FP + TN} \tag{6.17}$$

TABLE 6.11 – La matrice de confusion

		Classe réelle	Classe réelle
		Positive	Négative
Classe prédite	Positive	Vrai Positif (TP)	Faux Positif (FP)
Classe prédite	Négative	Faux Négative (FN)	Vrai Négative (TN)

Expériences et résultats

Dans cette section, nous représentons l'évaluation de l'algorithme proposé PLMVO-MLP pour détecter les logiciels malveillants de Linux. L'algorithme MLP a été appliqué avec une seule couche cachée. Le PLMVO a été utilisé pour optimiser les poids et le nombre de neurones dans la couche cachée selon l'approche présentée dans la figure 6.3. Les paramètres initiaux de l'utilisation de PSO pour la sélection d'attributs et de PLMVO pour la prédiction des logiciels malveillants de Linux sont présentés dans le tableau 5.1 (FARIS et al., 2018). Pour la sélection d'attributs, la taille de la population et le nombre de générations ont été fixés à 20 et 50, respectivement. Dans chaque vecteur, le nombre d'éléments est égal au nombre d'attributs dans l'ensemble de données (FARIS et al., 2018). Pour le PLMVO, la taille de la population et le nombre maximum d'itérations ont été fixés à 50 et 200, respectivement. La performance du PLMVO-MLP a été évaluée selon l'exactitude moyenne, F-mesure, précision, et rappel pour 10 exécutions. Pour vérifier les résultats, le PLMVO-MLP a été comparé par rapport aux deux approches utilisées pour détecter les logiciels malveillants dans la littérature : RNN (RHODE, BURNAP et JONES, 2018), et MLP (RAD, NEJAD et SHAHPASAND, 2018). L'ensemble de données a été divisé en plusieurs parties, 66% pour l'apprentissage et 34% pour le test.

TABLE 6.12 – Comparaison des résultats des algorithmes d'apprentissage automatique sans la sélection d'attributs

		PLMVO-MLP	MLP	RNN
Avant la sélection d'attributs	Exactitude	0.967	0.896	0.911
	F-mesure	0.970	0.934	0.946
	Précision	0.970	0.907	0.935
	Rappel	0.970	0.963	0.957

Le tableau 6.12 présente les résultats statistiques : l'exactitude, F-mesure, précision et rappel de la classification de l'ensemble de données original avant la sélection d'attributs. Sur la base du tableau ci-dessus, on peut constater que le PLMVO-MLP a surpassé les autres approches dans toutes les mesures avec une exactitude moyenne de 0,958, une F-mesure moyenne de 0,96, une précision et un rappel de 0,96 qui indiquent que sur 1000 fichiers, 960 d'entre eux sont correctement classés comme malveillants. En outre, on peut également noter que l'algorithme RNN (RHODE, BURNAP et JONES, 2018) a mieux performé par rapport à MLP (RAD, NEJAD et SHAHPASAND, 2018) avec une exactitude moyenne de 0,911.

TABLE 6.13 – Comparaison des résultats des algorithmes d'apprentissage automatique avec la sélection d'attributs

		PLMVO-MLP	MLP	RNN
Après la sélection d'attributs	Exactitude	1.0	0.928	0.949
	F-mesure	1.00	0.957	0.973
	Précision	1.00	0.978	0.965
	Rappel	1.00	0.936	0.981

Le tableau 6.13 présente les résultats de la classification après la sélection d'attributs en utilisant l'algorithme PSO. Après la sélection d'attributs, 11 caractéristiques ont été utilisées pour la création des modèles d'apprentissage automatique. D'après les résultats, on peut noter que la sélection d'attributs améliore les résultats de la classification. On peut également noter que le PLMVO-MLP a surpassé les autres techniques et a donné les meilleurs résultats avec une très haute exactitude, une meilleur F-mesure , une meilleur précision et un meilleur rappel de 1,00 qui indiquent que tous les fichiers ont été correctement classés. Sur la base des résultats ci-dessus, on peut observer que le modèle dépasse les travaux précédents qui utilisaient des réseaux de neurones, comme le RNN utilisé par Rhode et al. (SCHULTZ et al., 2000) qui n'a pas obtenu une très grande exactitude, cela est peut-être dû au petit ensemble de données que nous avons utilisé et à l'absence de données chronologiques séquentielles.

6.2 Conclusion de l'approche

Dans cette approche, nous avons proposé une nouvelle approche basée sur l'optimisation de l'essaim de particules, l'optimisation multi-verse basée sur le vol de Lévy pour optimiser simultanément la structure, les poids de connexion et les biais du réseau de neurones à propagation avant (FFNN). De nombreux chercheurs ont étudié le problème de la recherche de valeurs optimales pour les poids de connexion et les biais. Cependant, la littérature contient peu de recherches sur l'optimisation de la structure et les poids / biais simultanément. Dans notre étude, nous avons utilisé le même schéma de codage hybride proposé par (FARIS, MIRJALILI et ALJARAH, 2019b) pour représenter les solutions de PLMVO pour la formation de MLP qui comprend le nombre de nœuds cachés, les poids de connexion et les biais. La méthode de formation a pris en compte les capacités du PLMVO en termes d'exploration et d'exploitation élevées pour localiser les valeurs optimales pour la structure, les poids et les biais de FFNN. L'approche est proposée afin de minimiser l'erreur de formation et d'augmenter l'exactitude de la classification. L'algorithme est comparé et évalué à l'aide de quinze fonctions de référence, neuf ensembles de données biomédicales. Les résultats indiquent que le schéma de codage hybride utilisé pour optimiser le nombre de nœuds cachés, les poids de connexion et les biais facilitent le processus d'optimisation des métaheuristiques. L'approche a permis aux algorithmes d'obtenir une très grande exactitude de classification sur tous les ensembles de données. La comparaison entre l'algorithme proposé et les autres algorithmes : PSO, MFO, MVO, WOA, HACPSO et BP montre la supériorité de l'algorithme PLMVO avec une haute exactitude, une petit MSE et une convergence rapide dans la plupart des ensembles de données par rapport aux autres algorithmes de formation. De plus, la faible valeur de l'écart type a prouvé que le PLMVO peut atteindre les mêmes résultats lors de différentes exécutions, ce qui confirme que notre entraineur est robuste et stable. Pour confirmer les résultats, nous avons utilisé le PLMVO-MLP pour détecter les logiciels malveillants de Linux. Le PLMVO-MLP a été comparé à deux approches utilisées pour détecter les logiciels malveillants dans la littérature : RNN (RHODE, BURNAP et JONES, 2018) et MLP (RAD, NEJAD et SHAHPASAND, 2018). Sur la base des résultats, il peut noter

que le modèle dépasse les travaux précédents avec une très grande exactitude, une meilleure F-mesure, une meilleure précision et un meilleur rappel de 1,00. Enfin, à partir de l'expérience, nous pouvons conclure que PLMVO peut donner de bons résultats et peut être une alternative aux autres méthodes de formation. Dans les travaux futurs, nous nous concentrons sur la façon d'étendre ce travail pour explorer des moyens plus efficaces de résoudre des problèmes complexes et l'application de cet algorithme aux données IoT. De plus, améliorez l'algorithme proposé pour résoudre le problème d'optimisation du Big Data.

Chapitre 7

L'hybridation entre l'optimiseur de loup gris et l'optimiseur de multi-vers (MVGWO) pour les problèmes d'optimisation globale à grande dimension

Ces dernières années, plusieurs techniques d'optimisation métaheuristiques ont été développées et utilisées pour résoudre des problèmes complexes en raison de leur flexibilité, simplicité, leur forte capacité et de leur capacité à éviter les optima locaux. Il s'agit notamment de l'algorithme génétique (GA) (HOLLAND, 1992), l'optimisation de l'essaim de particules (PSO) (KENNEDY et EBERHART, 1995b), la colonie d'abeilles artificielle (ABC) (KARABOGA et BASTURK, 2008), l'algorithme de recherche gravitationnelle (GSA) (RASHEDI, NEZAMABADI-POUR et SARYAZDI, 2009), l'évolution différentielle (DE) (STORN et PRICE, 1997b), l'optimisation multi-verse (MVO) (MIRJALILI, MIRJALILI et HATAMLOU, 2016), l'algorithme d'optimisation de la baleine (WOA) (MIRJALILI et LEWIS, 2016) et de l'optimiseur du loup gris (GWO) (MIRJALILI, MIRJALILI et LEWIS, 2014a).
L'objectif principal de ces algorithmes inspirés de la nature est de trouver la meilleure solution et les meilleures performances de convergence. Pour ce faire, l'exploration et l'exploitation doivent être équipées par ces algorithmes métaheuristiques pour s'assurer que l'optimum global est trouvé. L'exploitation est la recherche des voisins de la région prometteuse tandis que l'exploration est la recherche d'une zone inexplorée de la région réalisable. L'efficacité de l'algorithme métaheuristique dépend fortement de la façon dont ces deux comportements de recherche sont équilibrés pour trouver la solution optimale globale dans l'espace de recherche.
Les problèmes d'optimisation sont devenus plus complexes au cours des deux dernières décennies. Ce qui nécessite la meilleure méthode d'optimisation pour être résolu. Les algorithmes d'optimisation existants, tels que les métaheuristiques, sont très puissants pour résoudre de nombreux tests et problèmes d'optimisation de la vie réelle, mais le théorème "no free lunch" (NFL) indique qu'il n'existe pas de méthode d'optimisation supérieure pour résoudre tous les types de problèmes d'optimisation (FARIS, ALJARAH et MIRJALILI, 2016a). L'optimiseur de loup gris (GWO) (MIRJALILI, MIRJALILI et LEWIS, 2014a) est une nouvelle métaheuristique développée et inspirée par le mécanisme de la chasse et la hiérarchie de leadership des loups gris ; alpha, bêta, delta et oméga. Les performances de cet algorithme sont évaluées et comparées sur vingt- neuf fonctions de test. La comparaison avec l'optimisation de l'essaim de particules (PSO) (KENNEDY et EBERHART, 1995b), l'algorithme de recherche gravitationnelle (GSA) (RASHEDI, NEZAMABADI-POUR et SARYAZDI, 2009), l'évolution différentielle (DE) (STORN et PRICE, 1997b), la programmation évolutive (EP) (BÄCK,

RUDOLPH et SCHWEFEL, 1993) et la stratégie d'évolution (ES) (BEYER, 2001) démontre que l'algorithme GWO peut fournir des résultats compétitifs par rapport à ces algorithmes métaheuristiques bien connus, en termes de vitesse de convergence et de précision de la solution. En raison de ses avantages, l'algorithme GWO a été appliqué dans de nombreux domaines tels que la sélection d'attributs (EMARY et al., 2015), les problèmes d'ordonnancement de type "flow shop" (KOMAKI et KAYVANFAR, 2015), la prévision de séries chronologiques (STÜTZLE et HOOS, 2000), etc. L'algorithme GWO présente certains inconvénients dans le compromis entre l'exploitation et l'exploration (MITTAL, SINGH et SOHI, 2016 ; YUE, ZHANG et XIAO, 2020) et a également des problèmes pour résoudre des problèmes d'optimisation sans contrainte et des systèmes d'équations non linéaires (TAWHID et IBRAHIM, 2020 ; YUE, ZHANG et XIAO, 2020) .
Plusieurs recherches ont été développées pour améliorer la capacité de l'algorithme GWO en termes de performance de convergence, telles que le GWO binaire (EMARY, ZAWBAA et HASSANIEN, 2016), le GWO parallélisé (PAN, DAO, CHU et al., 2015), le GWO modifié (MITTAL, SINGH et SOHI, 2016), l'hybride DE avec GWO (TAWHID et IBRAHIM, 2020), l'intégration de DE avec GWO (ZHU et al., 2015), l'hybridation entre l'essaim de particules avec l'optimiseur de loup gris (SINGH et SINGH, 2017b), l'hybridation entre l'optimiseur de baleine avec l'optimiseur de loup gris (YUE, ZHANG et XIAO, 2020), l'optimiseur hybride de loup gris avec l'algorithme sinus-cosinus (SINGH et SINGH, 2017a) et l'optimiseur hybride de loup gris avec l'algorithme des feux d'artifice (BARRAZA et al., 2018).
La métaheuristique simple présente souvent certains inconvénients lors de la recherche de la valeur optimale dans un espace de grandes dimensions, comme la faible capacité de généralisation, la faible précision et l'incapacité à éviter les optima locaux. L'algorithme hybride tire parti de deux algorithmes d'optimisation, combine leurs avantages et compense leurs défauts pour améliorer les performances globales lors de la résolution de problèmes d'optimisations complexes (YUE, ZHANG et XIAO, 2020).
Pour surmonter les limites de GWO mentionnées ci-dessus, et pour améliorer ses performances dans les problèmes d'optimisation à grande dimension, un nouvel algorithme hybride basé sur l'optimiseur de loup gris (GWO) et l'optimisation multi-verse (MVO) appelé MVGWO est proposée. MVGWO est une combinaison de l'optimiseur de Loup Gris utilisé pour l'exploitation et l'optimisation multi-verse utilisé pour l'exploration. Deux séries d'expériences ont été utilisées pour évaluer la qualité de l'algorithme MVGWO proposé. Dans la première expérience, l'algorithme MVGWO proposé est comparé aux algorithmes GWO et MVO pour résoudre un ensemble de vingt-deux fonctions de référence afin de trouver la solution globale en utilisant différents types et dimensions. Dans la deuxième expérience, l'algorithme MVGWO est utilisé pour la sélection d'attributs et l'optimisation des paramètres du SVM simultanément afin d'améliorer la précision du SVM. Quinze jeux de données ont été résolus par le MVGWO-SVM proposé. Le MVGWO-SVM est appliqué à l'aide de deux architectures système (HUANG et WANG, 2006a ; LIN et al., 2008 ; FARIS et al., 2018). De plus, les performances de MVGWO ont été comparées à celles de quatre algorithmes métaheuristiques bien connus : GWO (MIRJALILI, MIRJALILI et LEWIS, 2014a), MVO (MIRJALILI, MIRJALILI et HATAMLOU, 2016), WOA (MIRJALILI et LEWIS, 2016) et BAT (YANG, 2010).
Les principales contributions de cette approche sont :

- Un nouvel algorithme appelé MVGWO est proposé basé sur GWO et MVO.
- Un nouvel optimiseur de loup gris amélioré avec un poids d'inertie adaptatif a été proposé.
- Vingt-deux fonctions de test ont été testées et évaluées par l'algorithme proposé en utilisant différents types et dimensions.
- Le MVGWO est utilisé pour la sélection d'attributs et l'optimisation des paramètres du SVM.

7.0.1 État de l'art sur l'optimisation des loups gris

Différentes modifications et hybridations ont été proposées par la recherche pour améliorer l'algorithme GWO. Le tableau 7.1 et le tableau 7.2 résument respectivement les modifications et les hybridations de l'algorithme GWO.

TABLE 7.1 – Les modifications de l'algorithme GWO

Modifications du GWO	REF	Objectif	Conclusion
GWO chaotique (Chaotic GWO)	(KOHLI et ARORA, 2018)	Différentes méthodes chaotiques ont été ajoutées afin d'augmenter la vitesse de convergence de GWO	Les expériences ont montré que le CGWO proposé surpasse les autres algorithmes et améliore le GWO standard
GWO modifié (Modified GWO)	(RASHID, ABBAS et TUREL, 2019)	MGWO a été proposé pour optimiser les paramètres du réseau de neurones récurrent (RNN) et MGWO-RNN a été utilisé pour classer les performances des étudiants	Les résultats ont montré que MGWO améliore les résultats du RNN et peut trouver la meilleure solution par rapport aux autres algorithmes
GWO binaire (Binary GWO)	(EMARY, ZAWBAA et HASSANIEN, 2016)	Le BGWO a été proposé pour résoudre le problème de l'engagement unitaire à grande échelle	L'algorithme BGWO a été comparé à une variété d'algorithmes binaires et au GWO standard. Les résultats indiquent que l'algorithme BGWO est plus performant que les autres algorithmes.
GWO de powell (Powell GWO)	(ZHANG et ZHOU, 2015)	Le GWO basé sur l'optimisation locale de powell a été utilisé pour résoudre l'optimisation complexe	Neuf jeux de données utilisés pour le clustering et sept fonctions de référence ont été utilisés pour tester PGWO. Les résultats obtenus montrent que PGWO a obtenu de bons résultats par rapport aux algorithmes les plus récents
GWO intelligent (Intelligent GWO)	(SAXENA et al., 2018)	IGWO a été proposé pour résoudre différents problèmes dans les entreprises qui vendent de l'énergie sur le marché de l'énergie	Vingt-deux fonctions de test ont été utilisés pour tester IGWO. IGWO a été comparé à GWO oppositionnel, PSO et GWO. Les résultats montrent que l'IGWO est plus performant que les autres algorithmes

TABLE 7.2 – Les hybridations de l'algorithme GWO

Hybridizations du GWO	REF	Objectif	Conclusion
Algorithme du feu d'artifice (FWA)	(BARRAZA et al., 2018)	L'optimiseur de loup gris et l'algorithme Firework ont été hybridés afin d'améliorer la faiblesse des deux algorithmes	Vingt-deux fonctions de référence ont été utilisées pour tester le FWA-GWO et cet algorithme a été comparé à GWO et FWA. Les résultats ont montré la supériorité de FWA-GWO
Algorithme du Libellule (Dragonfly (DA))	(SHILAJA et ARUNPRASATH, 2019)	L'optimiseur de loup gris et Dragonfly ont été hybridés afin de résoudre les problèmes des systèmes d'énergie renouvelable	Le système de bus IEEE 30 a été utilisé pour tester l'algorithme proposé. Les résultats ont montré que cet algorithme était plus rapide
Algorithme du Pollinisation des fleurs (Flower pollination (FPA))	(PAN, DAO, CHU et al., 2017)	Pour résoudre les problèmes du monde réel, l'optimiseur de loup gris et la pollinisation des fleurs ont été hybridés	Les performances de cet algorithme ont été vérifiées à l'aide de six fonctions de référence et comparées à PSO, FPA et GWO. Les résultats ont montré la supériorité de cet algorithme
Algorithme sinus-cosinus (Sine Cosine algorithm (SCA))	(SINGH et SINGH, 2017a)	Pour améliorer l'optimiseur de loup gris, l'hybridation entre l'algorithme d'optimisation de loup gris et le sinus cosinus (SCA) a été proposée	Le GWO-SCA a été comparé aux SCA, GWO, WOA, PSO et ALO. Les résultats ont montré que GWO-SCA a réussi à résoudre des problèmes du monde réel et des fonctions de test
Algorithme d'optimisation de la baleine (Whale optimization algorithm (WOA))	(MOHAMMED et RASHID, 2020)	L'optimiseur de loup gris a été hybridé avec l'algorithme d'optimisation de baleine pour l'optimisation numérique globale et la résolution de la conception des appareils sous pression	Vingt-trois fonctions de test, 25 fonctions CEC 2005 et 10 fonctions CFC 2019 ont été utilisés pour vérifier les résultats de WOAGWOA. Les résultats ont montré que WOAGWO peut obtenir une solution optimale et peut obtenir de meilleurs résultats que WOA et FDO

7.0.2 Optimiseur de loup gris (GWO)

L'optimiseur de loup gris (GWO) (MIRJALILI, MIRJALILI et LEWIS, 2014a) est un algorithme métaheuristique d'intelligence en essaim introduit et inspiré par la hiérarchie de leadership et le mécanisme de chasse des loups gris. Comme tous les autres algorithmes d'intelligence en essaim, le GWO est considéré comme un algorithme très utile pour trouver une solution optimale globale en raison de sa simplicité et de sa vitesse de convergence la plus rapide (MIRJALILI, MIRJALILI et LEWIS, 2014a).
Dans l'algorithme GWO, les loups gris sont divisés en quatre catégories : Alpha (α), Beta (β), delta (δ) et omega (ω). Alpha (α) est la première meilleure solution, Beta (β) est la deuxième, ensuite la troisième est delta (δ), la solution restante est nommée omega (ω).

— L'encerclement de la proie :
Les loups gris essaient d'encercler la proie pendant la chasse. La position des loups gris est modifiée par les équations suivantes :

$$\overrightarrow{D} = \left| \overrightarrow{C} \cdot \overrightarrow{X_p}(t) - \overrightarrow{X}(t) \right| \tag{7.1}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X_p}(t) - \overrightarrow{A} \cdot \overrightarrow{D} \tag{7.2}$$

Où t désigne l'itération courante, X_p est le vecteur de position de la proie et $\overrightarrow{X}$ indique le vecteur de position d'un loup gris, $\overrightarrow{A}$ et $\overrightarrow{C}$ sont des vecteurs de coefficients. $\overrightarrow{A}$ et $\overrightarrow{C}$ peuvent être calculés comme suit :

$$\overrightarrow{A} = 2 \cdot \overrightarrow{a} \cdot \overrightarrow{r_1} \cdot \overrightarrow{a} \tag{7.3}$$

$$\overrightarrow{C} = 2 \cdot \overrightarrow{r_2}$$

(7.4)

Où r_1, r_2 sont des vecteurs aléatoires dans [0,1] et $\overrightarrow{a}$ peut être diminué linéairement de 2 à 0 au cours des itérations.

— Le processus de la chasse :
Les loups gris peuvent facilement encercler la proie avec la capacité de reconnaître son emplacement. L'ensemble du processus de chasse est généralement dirigé par l'alpha (α). Cependant, dans un espace de recherche complexe, il est impossible d'obtenir l'emplacement de la proie au début. Par conséquent, GWO considère que les trois premières meilleures solutions, alpha, bêta et delta, ont plus d'informations sur l'emplacement de la proie. Ensuite, les autres loups mettent à jour leurs positions en fonction de ces trois positions (MIRJALILI, MIRJALILI et LEWIS, 2014a).
Les équations mathématiques de cette phase sont les suivantes :

$$\overrightarrow{D_\alpha} = \left| \overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \overrightarrow{X} \right|, \overrightarrow{D_\beta} = \left| \overrightarrow{C_2} \cdot \overrightarrow{X_\beta} - \overrightarrow{X} \right|, \overrightarrow{D_\delta} = \left| \overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \overrightarrow{X} \right| \tag{7.5}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot (\overrightarrow{D_\alpha}), \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot (\overrightarrow{D_\beta}), \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot (\overrightarrow{D_\delta}) \tag{7.6}$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{7.7}$$

Où $\overrightarrow{X_\alpha}$, $\overrightarrow{X_\beta}$, $\overrightarrow{X_\delta}$ sont la position de α, β et δ respectivement ; $\overrightarrow{C_1}$, $\overrightarrow{C_2}$ et $\overrightarrow{C_3}$ sont définis aléatoirement ; $\overrightarrow{X}$ est la position de la solution courante ; $\overrightarrow{A_1}$, $\overrightarrow{A_2}$, $\overrightarrow{A_3}$ représentent des

vecteurs aléatoires.

— L'attaque de la proie (exploitation) :
Un loup gris peut chasser en attaquant sa proie lorsqu'elle cesse de bouger. Ce mécanisme se fait en diminuant la valeur de a. A est une valeur aléatoire dans $[-a2, a2]$. Notez que la valeur de $\overrightarrow{A}$ est également diminuée de la valeur a. La prochaine position du loup peut être n'importe quelle position entre la position de la proie et sa position actuelle. Lorsque les valeurs aléatoires de $\overrightarrow{A}$ sont dans l'intervalle $[-1, 1]$. L'attaque de la proie peut être faite si la valeur de $|A| < 1$.
Le GWO souffre de la stagnation des optima locaux et les chercheurs essaient de découvrir divers mécanismes pour résoudre ce problème (MIRJALILI, MIRJALILI et LEWIS, 2014a).

— Recherche de la proie (exploration) :
L'exploration se produit si $|A| > 1$ ou $|A| < -1$ et $|C| < 1$. Si la valeur de $|A|$ est dans l'intervalle $[-1, 1]$ les loups sont forcés de s'écarter de la proie.
Si $|A| > 1$, le loup essaie de trouver de meilleures proies (MIRJALILI, MIRJALILI et LEWIS, 2014a).

7.0.3 L'algorithme MVGWO proposé

L'algorithme GWO a une bonne capacité d'exploitation mais peut s'empiler dans l'optimum local, et dans certains cas, il converge prématurément. D'un autre côté, le MVO a une forte capacité d'exploration. Pour obtenir une meilleure optimisation globale, l'algorithme MVGWO est proposé pour fusionner l'algorithme GWO en exploitation et l'algorithme MVO en exploration.
Dans cette section, nous proposons un nouvel algorithme hybride basé sur l'optimiseur de loup gris (GWO) et l'optimiseur multi-vers (MVO) qui est appelé MVGWO afin d'améliorer la performance du GWO en phase d'exploration pour obtenir de meilleurs résultats.
Premièrement, un coefficient adaptatif est proposé pour équilibrer l'exploitation et l'exploration. Deuxièmement, un poids d'inertie est proposé pour améliorer la précision et la vitesse de convergence de l'algorithme GWO et pour éviter qu'il tombe dans l'optimum local.

Coefficient d'équilibre adaptatif

Inspiré par (YUE, ZHANG et XIAO, 2020), dans cette approche, un coefficient adaptatif (p) est proposé pour équilibrer l'exploration et l'exploitation.
Le coefficient sera mis à jour pour ajuster la stratégie de recherche lorsque la position actuelle est proche de la solution optimale, comme le montre l'équation (7.8).

$$p = 0.9 \times (1 - cos(\frac{\pi}{2} \cdot \frac{t}{Max_{iter}})) \tag{7.8}$$

Où Max_{iter} représente le nombre maximal d'itérations, t est l'itération actuelle et p indique le coefficient d'équilibre adaptatif.
Pour la comparaison avec le coefficient d'équilibre adaptatif, nous avons utilisé une valeur aléatoire $\in [0, 1]$.
Si $r < p$, l'algorithme GWO est exécuté dans l'itération suivante ; sinon, le GWO-MVO est utilisé.
Après une itération de l'algorithme GWO-MVO, dans certains cas, l'algorithme MVGWO peut passer à l'itération suivante de GWO-MVO sans autre exploitation, lui permettant de quitter l'espace optimal local actuel et de sauter la solution optimale globale. Pour éviter ces situations, l'algorithme MVGWO exécute l'algorithme GWO au moins T itérations avant de passer à l'algorithme GWO-MVO suivant.

T est fixé à 10 dans cette approche, la variable K est utilisée pour compter le nombre d'itérations de GWO après la dernière itération de GWO-MVO.
Au début de MVGWO, K est initialisé. K augmente de un après chaque itération de GWO. GWO-MVO sera utilisé pour effectuer l'itération si $K > T$ et $r > p$. K sera mis à 0 à la fin de ces itérations.

L'hybridation entre l'optimiseur de loup gris et l'optimiseur de multi-vers (GWO-MVO)

Dans cet algorithme, la position, la précision de convergence et la vitesse des loups gris (Alpha (α), Beta (β), Delta (δ)) ont été améliorées en utilisant la mise à jour de position de l'algorithme MVO en utilisant l'Equation (6.10) afin d'étendre la convergence et d'atteindre un meilleur équilibre entre l'exploitation et l'exploration.
La différence entre MVGWO et GWO se trouve dans l'équation (7.5). Les autres équations de GWO sont les mêmes. Le mécanisme d'apprentissage de l'optimiseur de loup gris est amélioré à l'aide de l'équation suivante :

$$\begin{cases} \overrightarrow{D_\alpha} = \left| \overrightarrow{X_\alpha} + TDR + ((ub - lb)) * rand + lb \right|, \\ \overrightarrow{D_\beta} = \left| \overrightarrow{X_\beta} + TDR + ((ub - lb)) * rand + lb \right|, \\ \overrightarrow{D_\delta} = \left| \overrightarrow{X_\delta} + TDR + ((ub - lb)) * rand + lb \right| \end{cases} \tag{7.9}$$

(*Remarque :* cette hybridation est de type hybridation relais de bas niveau (JOURDAN, 2003))

7.0.4 La complexité de MVGWO

La complexité temporelle de l'algorithme MVGWO est affectée par le nombre maximal d'itérations t, la taille de la population n et les dimensions de la solution d. O(MVGWO)=O(initialisation de la population)+O(calcul de la valeur de fitness de la population entière)+O(mise à jour de la position de la population).
La complexité temporelle de l'initialisation de la population est $O(n \times d)$, la complexité temporelle du calcul de la valeur de fitness de la population entière est $O(t \times n \times d)$. La complexité temporelle de la solution de mise à jour est égale à : O = O(mise à jour de la position de GWO)+(mise à jour de la position de GWO-MVO) équivalent à $O(10/11 \times t \times n \times d + 1/11 \times t \times n \times d) = O(t \times n \times d)$.
En conséquence, la complexité temporelle de l'algorithme MVGWO est $O(n \times d + t \times n \times d + t \times n \times d) = O((2t + 1) \times n \times d)$. En comparant, la complexité temporelle du GWO qui est $O((2t + 1) \times n \times d)$ avec la complexité temporelle du MVGWO. Le MVGWO a la même complexité temporelle que le GWO.

7.0.5 Expériences et résultats

Nous utilisons deux expériences pour évaluer la performance de l'algorithme proposé (MVGWO). Dans la première expérience, nous utilisons vingt-deux fonctions de référence (benchmarks) bien connues pour tester l'optimisation globale et nous comparons les résultats à : MVO (FARIS, ALJARAH et MIRJALILI, 2016a) et GWO (MIRJALILI, MIRJALILI et LEWIS, 2014a). Dans la deuxième expérience, nous proposons notre algorithme pour la sélection des caractéristiques et l'optimisation des paramètres des SVM en utilisant quinze jeux de données sélectionnés à partir du dépôt d'apprentissage machine de l'université de Californie à Irvine (UCI). La comparaison de notre algorithme (MVGWO) a été effectuée avec quatre algorithmes métaheuristiques : MVO (FARIS, ALJARAH et MIRJALILI, 2016a),

GWO (MIRJALILI, MIRJALILI et HATAMLOU, 2016), WOA (MIRJALILI et LEWIS, 2016) et BAT (YANG, 2010).

7.0.6 Configuration de l'expérience

Dans ce travail, les expériences ont été menées sur un ordinateur personnel avec Intel ® core ™ CPU (1,60) GHz, 64 bits, système d'exploitation Windows 10 et 4 Go (RAM), l'algorithme proposé et les autres algorithmes ont été mis en œuvre avec l'environnement MATLAB R2019a, et la bibliothèque LIBSVM est utilisée pour le classificateur SVM.
Les métaheuristiques sont très sensibles à leurs paramètres qui nécessitent une initialisation minutieuse. Ainsi, les paramètres initiaux recommandés dans la littérature de MVO (FARIS, ALJARAH et MIRJALILI, 2016a), GWO (MIRJALILI, MIRJALILI et HATAMLOU, 2016), WOA (MIRJALILI et LEWIS, 2016) et BAT (YANG, 2010) ont été utilisés et listés dans le tableau 5.1. En outre, toutes les caractéristiques dans tous les ensembles de données ont été mappées à $[0, 1]$ pour éliminer l'influence de la caractéristique qui a une échelle différente.

7.0.7 Série d'expériences 1 : l'optimisation globale

Pour évaluer la performance de l'algorithme MVGWO, vingt-deux fonctions de test ont été sélectionnées dans les Figures (7.1), (7.2) et (7.3). L'algorithme proposé MVGWO a été comparé à MVO et GWO. Le nombre d'itérations a été fixé à 1000 et tous les algorithmes ont été testés 30 fois. La performance de MVGWO a été évaluée en fonction du meilleur, la moyenne, le pire et de l'écart type de la fonction de fitness (la fonction objective).

Résultats et discussions

Dans cette section, les performances et l'efficacité de notre algorithme dans la résolution des problèmes d'optimisation ont été vérifiées à l'aide de vingt-deux fonctions de test. Ces fonctions de test ont été répertoriées dans les figures (7.1), (7.2) et (7.3) et peuvent être divisées en fonctions unimodales et multimodales (FAN et al., 2020).

Function	Dim	Range	$f_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30/100/500/1000	$[-100,100]$	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert - \prod_{i=1}^{n} \lvert x_i \rvert$	30/100/500/1000	$[-10,10]$	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30/100/500/1000	$[-100,100]$	0
$F_4(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	30/100/500/1000	$[-100,100]$	0
$F_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right]$	30/100/500/1000	$[-30,30]$	0
$F_6(x) = \sum_{i=1}^{n} \left(\lvert x_i + 0.5 \rvert \right)^2$	30/100/500/1000	$[-100,100]$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1]$	30/100/500/1000	$[-1.28,1.28]$	0

FIGURE 7.1 – Description des fonctions de test unimodales (FAN et al., 2020)

Function	Dim	Range	$f_{\min}$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30/100/500/1000	[−500,500]	-418.9829 ×*Dim*
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30/100/500/1000	[−5.12,5.12]	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30/100/500/1000	[−32,32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30/100/500/1000	[−600,600]	0
$F_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_i) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(x_i,10,100,4)$, $y_i = 1+\frac{x_i+1}{4}$, $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, x_i > a \\ 0, -a < x_i < a \\ k(-x_i-a)^m, x_i < a \end{cases}$	30/100/500/1000	[−50,50]	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_i) + \sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right] + (x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	30/100/500/1000	[−50,50]	0

FIGURE 7.2 – Description des fonctions de test multimodales (FAN et al., 2020)

Function	Dim	Range	$f_{\min}$
$F_{14}(x) = \left(\frac{1}{500} + \sum_{i=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	[−65,65]	1
$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	[−5,5]	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5,5]	−1.0316
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)\cos x_1 + 10$	2	[−5,5]	0.398
$F_{18}(x) = \left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right] \times \left[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	[−2,2]	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j-p_{ij})^2\right)$	3	[1, 3]	−3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2\right)$	6	[0,1]	−3.32
$F_{21}(x) = -\sum_{i=1}^{5}\left[(X-a_i)(X-a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.1532
$F_{22}(x) = -\sum_{i=1}^{7}\left[(X-a_i)(X-a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.4028
$F_{23}(x) = -\sum_{i=1}^{10}\left[(X-a_i)(X-a_i)^T + c_i\right]^{-1}$	4	[0,10]	−10.5363

FIGURE 7.3 – Description des fonctions de test multimodales à dimension fixe (FAN et al., 2020)

Les fonctions de test unimodales (F1-F7) ont une seule solution (un optimum global),

elles conviennent donc pour évaluer la capacité d'exploration de l'algorithme proposé. Cependant, les fonctions de référence multimodales (F8-F23) ont plusieurs optimums locaux et un optimum global, elles sont donc utiles pour évaluer la capacité d'exploitation de notre algorithme. Parmi les fonctions multimodales (F14-F23) on trouve les fonctions fixes de faible dimension, qui seront appliquées pour tester la capacité d'équilibrer l'exploitation et l'exploration des algorithmes.

Paramètres initiaux

Le nombre maximum d'itérations T et les individus N de l'algorithme sont fixés respectivement à 1000 et 30. Les poids d'inertie initiaux dans MVGWO sont fixés à $w_{max} = 0,9$ et $w_{min} = 0,2$. La valeur moyenne (AVG) et l'écart type (STD) des résultats sur différentes dimensions obtenus en exécutant chaque algorithme 30 fois, et sont indiqués dans les tableaux (7.3), (7.4), (7.5), (7.6), (7.7), (7.8), (7.9), (7.10), et (7.11).

Analyse statistique

TABLE 7.3 – Résultats des fonctions de test unimodales à 30 dimensions (F1-F7)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F1	0.0/0.0	2.4781E-70/6.8439E-70	0.1841/0.0716
F2	1.0E-32/0.0	5.9784E-41/1.0325E-40	0.2932/0.0738
F3	0.0/0.0	3.4677E-19/1.3935E-18	20.7219/7.2441
F4	1.3895E-314/0.0	2.4862E-17/4.8193E-17	0.6451/0.2453
F5	27.544/0.3991	26.5444/0.8508	149.0924/248.4989
F6	2.1486/0.2297	0.3301/0.2797	0.1855/0.0481
F7	2.1523E-05/2.3303E-05	4.725E-04/2.4135E-04	0.0145/0.0055

Le tableau (7.3) montre les résultats de la comparaison entre MVGWO, MVO et GWO dans les fonctions unimodales de 30 dimensions. Ces fonctions de référence sont faciles à analyser la capacité d'exploitation des algorithmes d'optimisation puisqu'elles ne contiennent qu'une seule valeur optimale globale. Nous observons que MVGWO obtient de meilleurs résultats que GWO sur les fonctions unimodales, sauf pour F2 et F5. En particulier, MVGWO peut atteindre la valeur optimale théorique sur les fonctions F1, F3, F4 et F7, mais GWO ne peut pas, ce qui souligne que MVGWO a une meilleure précision et stabilité de recherche.

Les fonctions de test multimodaux comprennent de nombreuses optimisations locales et globales, ils sont utiles pour évaluer la capacité d'exploration des algorithmes d'optimisation. Les résultats des tests des fonctions multimodales à 30 dimensions sont énumérés dans le tableau (7.4).

TABLE 7.4 – Résultats des fonctions de test multimodales à 30 dimensions (F8-F13)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F8	-3.0801E+03/400.9843	-6.3058E+03/566.9973	-7.7073E+03/776.2942
F9	0.0/0.0	0.4164/1.2302	110.7968/29.2215
F10	4.0856E-15/1.0840E-15	1.2967E-14/3.3118E-15	0.7882/0.8124
F11	0.0/0.0	0.0015/0.0040	0.4367/0.0981
F12	0.1591/0.0236	0.0255/0.0155	1.0587/0.8646
F13	1.3782/0.1671	0.2609/0.1745	0.0390/0.0198

À partir du tableau (7.4), on peut noter que la performance de l'algorithme MVGWO proposé est meilleure que les autres algorithmes dans F9, F10 et F11. Cela montre que l'algorithme hybride peut améliorer la capacité de l'algorithme GWO dans la résolution des problèmes multimodaux.
Pour tester la performance de MVGWO dans le traitement des problèmes d'optimisation à plus grande dimension. Les fonctions évolutives (F1-F3) sont étendues à 100, 500 et 1000 dimensions avec des paramètres inchangés. Les algorithmes sont ensuite réexécutés indépendamment 30 fois dans MATLAB 2019a. Les résultats expérimentaux sont présentés dans les tableaux (7.5), (7.6), (7.7), (7.8), (7.9), et (7.10).

TABLE 7.5 – Résultats des fonctions de test unimodales à 100 dimensions (F1-F7)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F1	0.0/0.0	2.0722E-34/3.5627E-34	22.4033/3.8315
F2	1.1840E-314/0.0	7.4046E-21/3.8423E-21	2.7976E+18/1.3085E+19
F3	0.0/0.0	0.1029/0.2777	3.1607E+04/3.7046E+03
F4	4.2935E-293/0.0	2.4782E-04/6.6822E-04	41.9082/8.0965
F5	98.0970/0.4150	97.2646/0.8976	1.1480E+03/570.8081
F6	16.7699/0.4561	7.533/0.7490	23.0017/2.9315
F7	3.0802E-05/3.1034E-05	0.0015/6.8114E-04	0.2395/0.0452

TABLE 7.6 – Résultats des fonctions de test multimodales à 100 dimensions (F8-F13)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F8	-5.4243E+03/683.2958	-1.7007E+04/1.5321E+03	-2.440E+04/1.2312E+03
F9	0.0/0.0	0.9318/2.664	622.3446/93.704
F10	4.4409E-15/0.0	6.8863E-14/5.6547E-15	5.6960/5.6617
F11	0.0/0.0	7.1547E-04/0.0028	1.2035/0.0238
F12	0.5737/0.0421	0.1998/0.0636	9.1719/2.4152
F13	8.8630/0.1673	5.4718/0.3361	82.54/32.7562

TABLE 7.7 – Résultats des fonctions de test unimodales de 500 dimensions (F1-F7)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F1	0.0/0.0	2.4519E-14/1.0484E-14	1.5845E+04/1.1162E+03
F2	4.9286E-298/0.0	5.3588E-09/1.2259E-09	1.6308E+202/Inf
F3	0.0/0.0	8.8359E+04/9.5536E+04	1.4264E+06/9.7324E+04
F4	9.4303E-262/0.0	52.66590/6.9409	91.6941/1.5146
F5	498.3681/0.1831	497.4259/0.3758	4.4948E+06/6.6081E+05
F6	113.9463/0.9456	88.2003/2.0312	1.6085E+04/936.3921
F7	2.2499E-05/1.9279E-05	0.0068/0.0018	51.3149/7.6690

TABLE 7.8 – Résultats des fonctions de test multimodales de 500 dimensions (F8-F13)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F8	-1.1956E+04/1.2465E+03	-6.555E+04/4.1359E+03	-9.3055E+04/3.6829E+03
F9	0.0/0.0	2.6270/4.4834	5.5422E+03/202.3015
F10	4.4409E-15/0.0	7.037E-09/2.0274E-09	20.7208/0.0646
F11	0.0/0.0	0.0025/0.0075	145.8283/13.1024
F12	0.9939/0.0155	0.6892/0.0264	2.8988E+05/2.1615E+05
F13	49.4510/0.0674	45.1045/0.6016	3.4288E+06/9.2258E+05

Comme on peut le voir dans les tableaux (7.5), (7.6), (7.7),et (7.8), les résultats de GWO et MVO augmentent avec la dimension croissante de toutes les fonctions et seul MVGWO peut toujours atteindre la solution optimale globale. Nous pouvons également noter que l'algorithme proposé surpasse les autres algorithmes dans F1, F2, F3, F4, et F6, on peut donc conclure que l'algorithme MVGWO proposé à la supériorité dans la capacité d'exploitation et l'efficacité dans la recherche de l'optimum global pour les fonctions unimodales.

TABLE 7.9 – Résultats des fonctions de test unimodales de 1000 dimensions (F1-F7)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F1	0.0/0.0	5.617E-10/2.3221E-10	2.3885E+05/1.6136E+04
F2	4.7262E-294/0.0	3.9858E-05/9.3187E-05	1.3942E+258/inf
F3	0.0/0.0	5.6516E+05/1.4034E+05	5.6111E+06/4.2299E+05
F4	6.0707E-249/0.0	70.9899/4.8047	97.0079/0.7748
F5	998.4177/0.1159	996.9766/0.0983	3.4068E+08/2.8758E+07
F6	238.2598/0.9990	203.0186/2.4183	2.3965E+04/1.3729E+04
F7	3.8702E-05/3.1765E-05	0.0126/0.0035	4.7109E+03/522.441

TABLE 7.10 – Résultats des fonctions de test multimodales de 1000 dimensions (F8-F13)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F8	-16676E+04/1.7657E+03	-10590E+05/1.7046E+04	-1.4605E+05/5.8559E+03
F9	0.0/0.0	7.3880/6.8919	1.3226E+04/250.4819
F10	4.3225E-15/6.4863E-16	7.0916E-07/1.1344E-07	20.92/0.0318
F11	403225E-15/6.4863E-16	7.0916E-07/1.1344E-17	20.9250/0.0318
F12	1.0791/0.009	0.8356/0.0216	3.6895E+08/6.5997E+07
F13	99.5018/0.0842	94.8339/0.5289	1.0567E+09/1.5256E+08

À partir du tableau (7.9) et (7.9), nous pouvons voir que l'effet d'optimisation des algorithmes GWO et MVO a été légèrement réduit dans les fonctions de grandes dimensions lorsque la dimension atteint 1000, MVGWO obtient toujours des valeurs optimales théoriques sur les fonctions F9, F10, et F11.

En résumé, nous pouvons conclure que l'algorithme MVGWO peut rechercher des solutions plus difficiles sur des problèmes de grandes dimensions dans la plupart des cas.

TABLE 7.11 – Résultats des fonctions de test multimodales à dimension fixe (F14-F23)

F.No	MVGWO	GWO	MVO
	AVG/STD	AVG/STD	AVG/STD
F14	3.8719/3.9532	3.2182/3.7199	0.998/4.2330
F15	0.0011/0.0037	0.0024/0.0061	0.0039/0.0075
F16	-1.0316/6.8251E-06	-1.0316/3.5060E-09	-1.0316/5.8742E-08
F17	3.00/3.3236E-06	3.7000/14.7885	3.00/6.9230E-07
F18	-38587/0.0025	-38618/0.0026	-38628/1.2168E-07
F19	-3.1730/0.0792	-3.2475/0.0803	-3.2584/0.0605
F20	-6.0970/1.1197	-9.1394/2.0617	-7.5438/2.9225
F21	-6.7263/1.2988	-10.2268/0.9630	-8.3060/2.8577
F22	-6.5264/1.3243	-10.0853/1.7518	-9.3821/2.3816

Parmi les vingt-deux fonctions (F14-F23) sont des fonctions multimodales fixes par rapport aux fonctions multimodales à dimension variable F8-F13, ces problèmes de référence ont moins d'optima locaux en raison de leurs faibles dimensions, et pourraient donc être utilisés pour tester l'équilibre entre les capacités d'exploration et d'exploitation des algorithmes.
D'après le tableau (7.11), nous pouvons voir que ces algorithmes sont très proches de l'optimum théorique dans les problèmes F16-F19. Dans F15, MVGWO montre une meilleure performance par rapport aux autres algorithmes. Par conséquent, nous pouvons conclure que notre algorithme MVGWO peut équilibrer entre l'exploration et l'exploitation.

Analyse de convergence

Figure (7.4) et (7.5) illustrent le comportement de convergence de MVGWO et les autres algorithmes sur les fonctions à 30 dimensions F1-F13. L'axe x et l'axe y représentent respectivement le numéro d'itération de la population et la valeur moyenne des 30 résultats de calcul.
Nous observons que l'algorithme MVGWO proposé commence à converger rapidement dans les générations précédentes de ces 13 fonctions et se maintient plus rapidement que les autres algorithmes dans les générations suivantes. De plus, MVGWO obtient l'optimum global sur les fonctions unimodales F1-F4 et multimodales F9, F10 et F11.

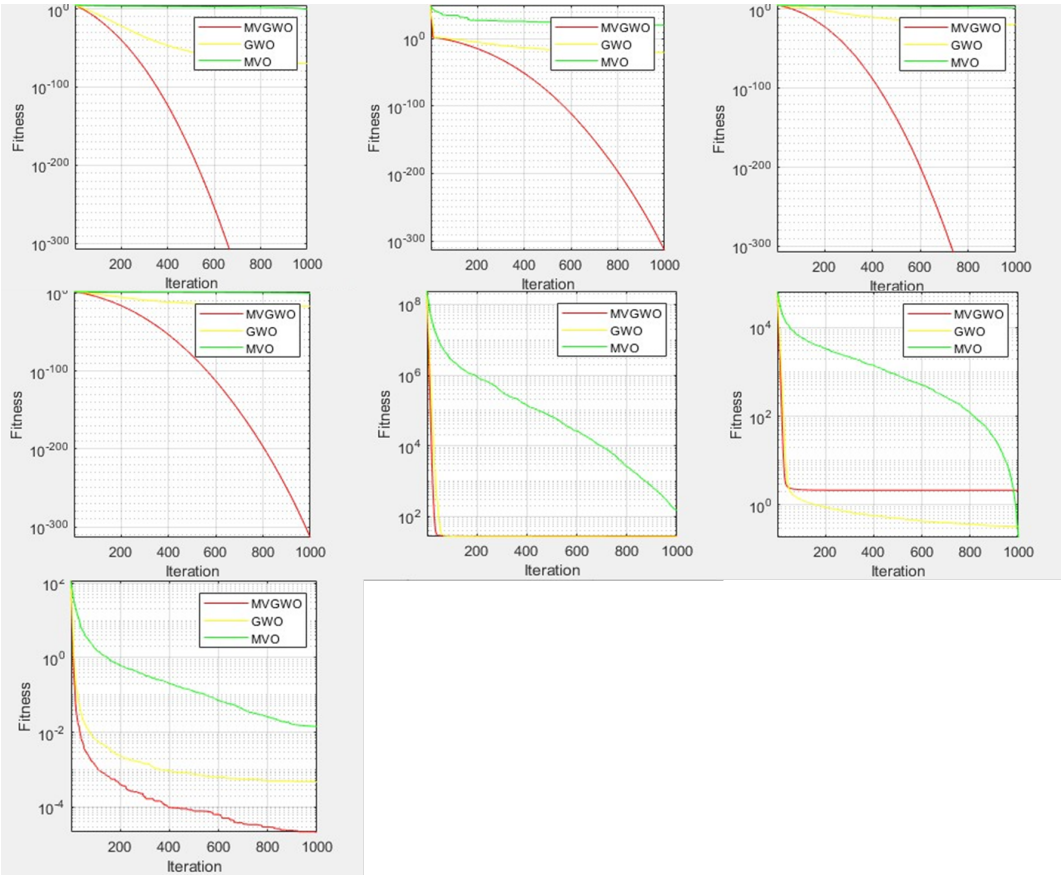

FIGURE 7.4 – Résultats des fonctions de test unimodales à 30 dimensions (F1-F7)

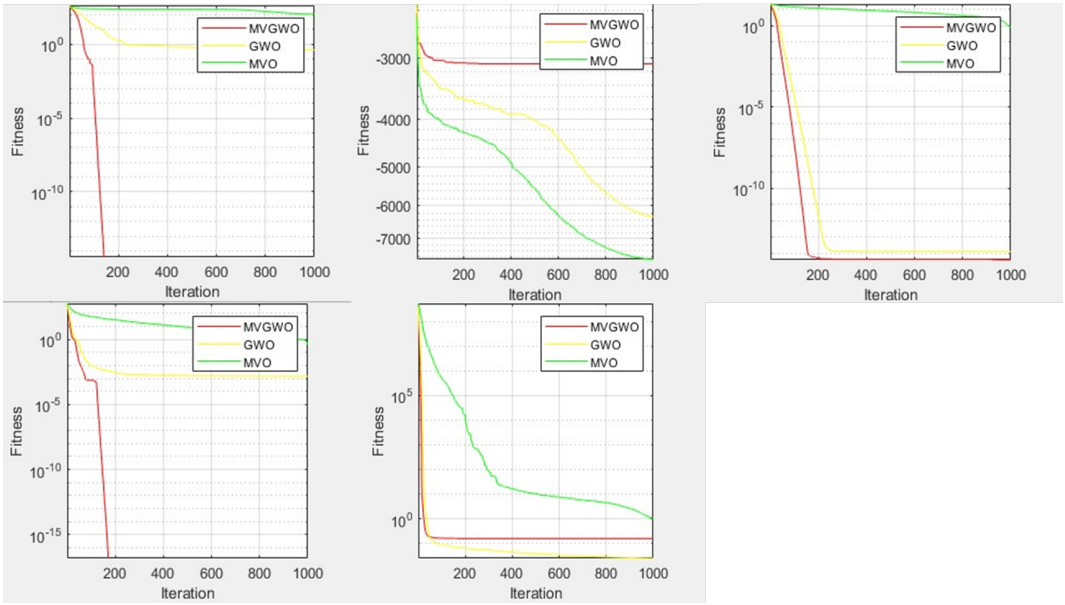

FIGURE 7.5 – Résultats des fonctions de test multimodales à 30 dimensions (F8-F13)

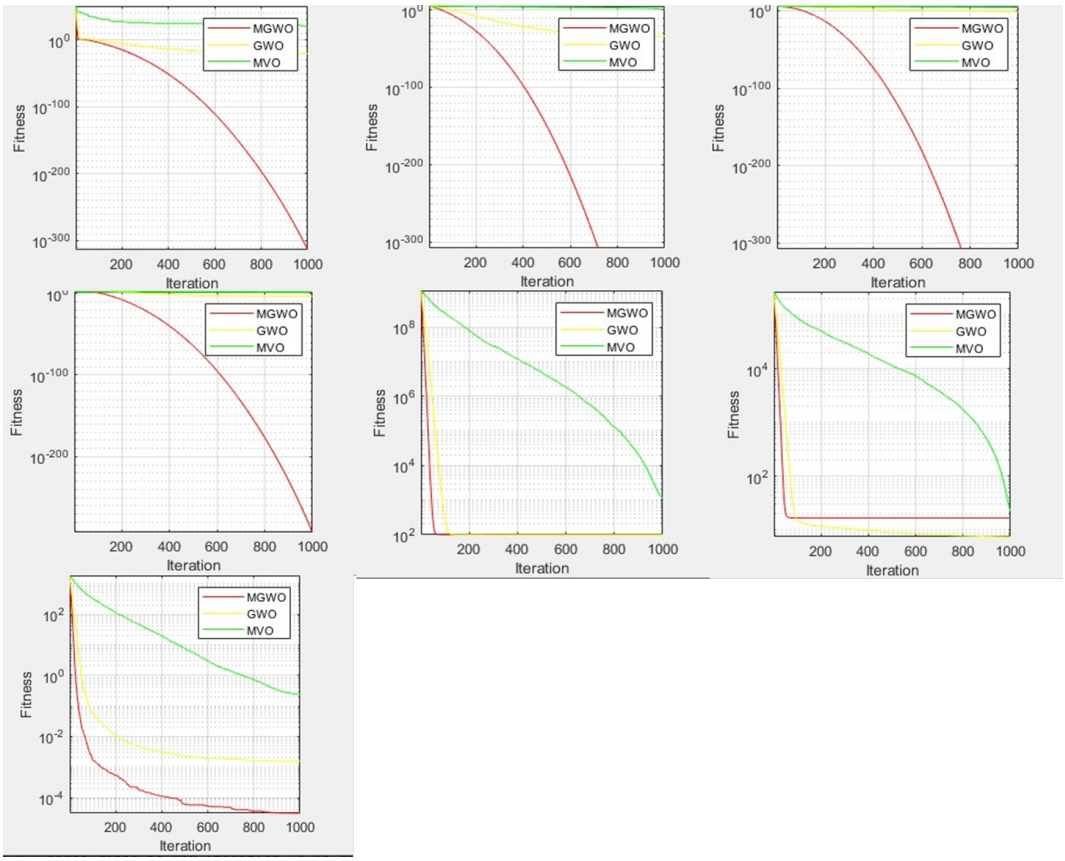

FIGURE 7.6 – Résultats des fonctions de test unimodales à 100 dimensions (F1-F7)

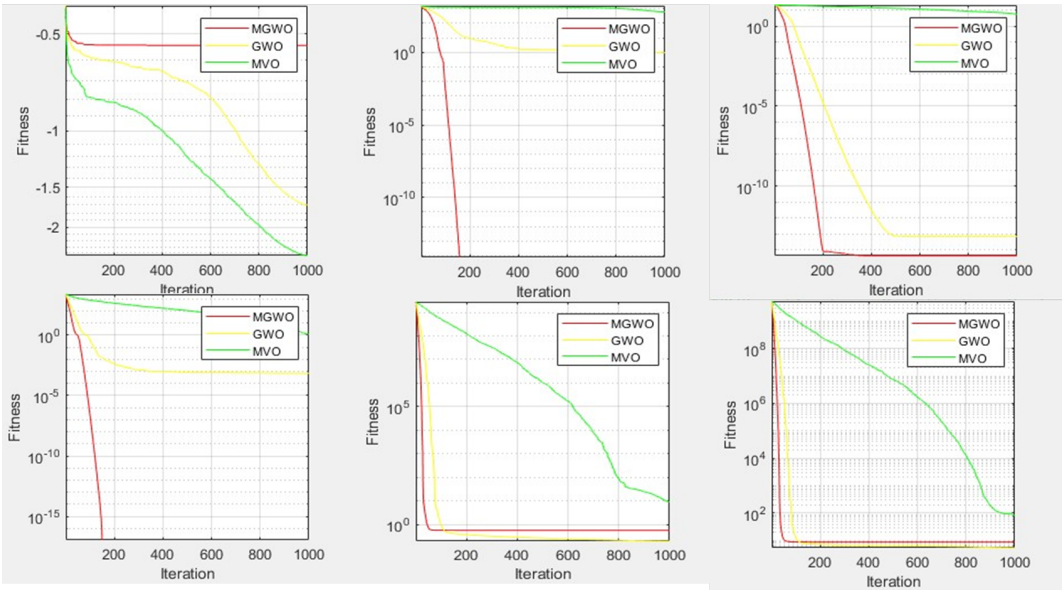

FIGURE 7.7 – Résultats des fonctions de test multimodales à 100 dimensions (F8-F13)

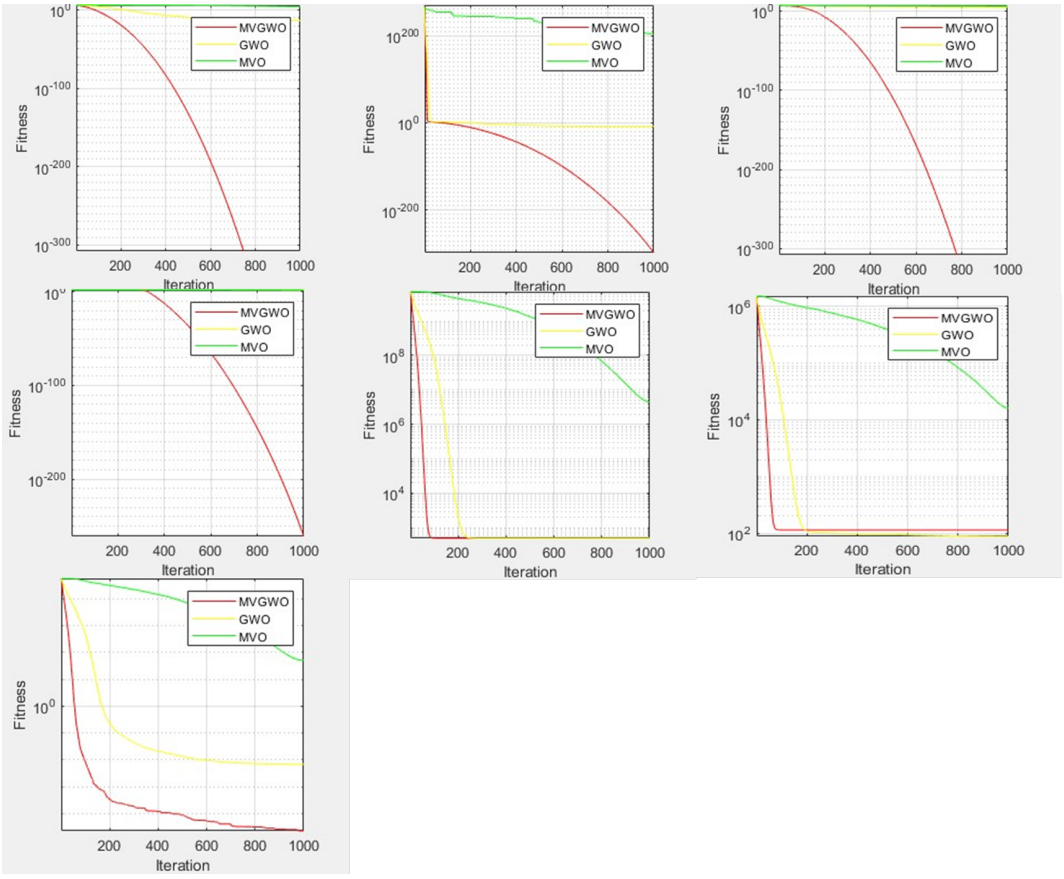

FIGURE 7.8 – Résultats des fonctions de test unimodales à 500 dimensions (F1-F7)

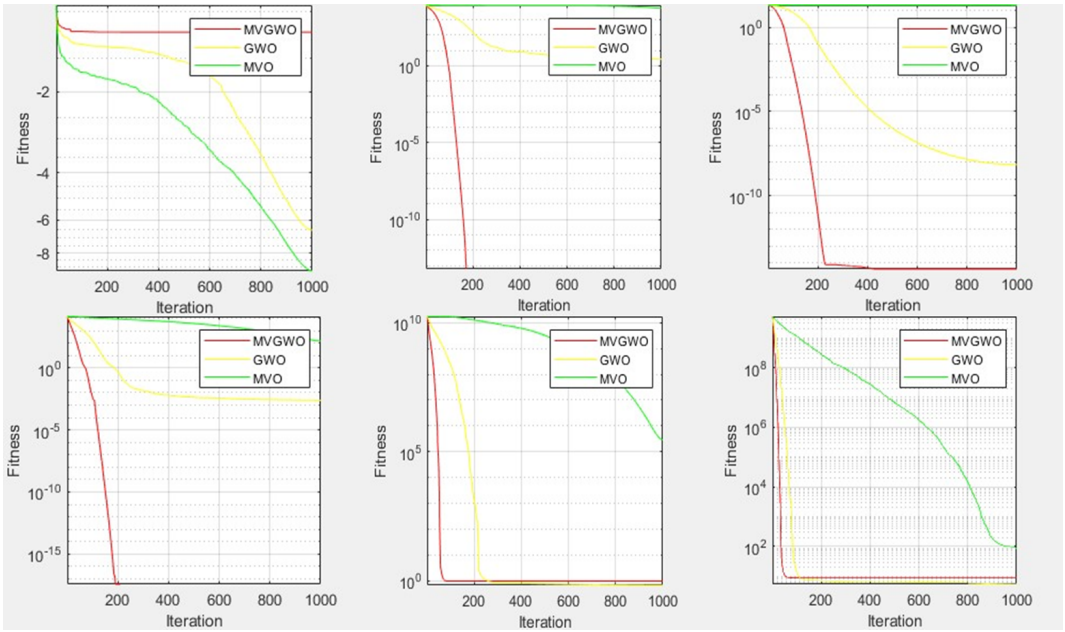

FIGURE 7.9 – Résultats des fonctions de test multimodales à 500 dimensions (F8-F13)

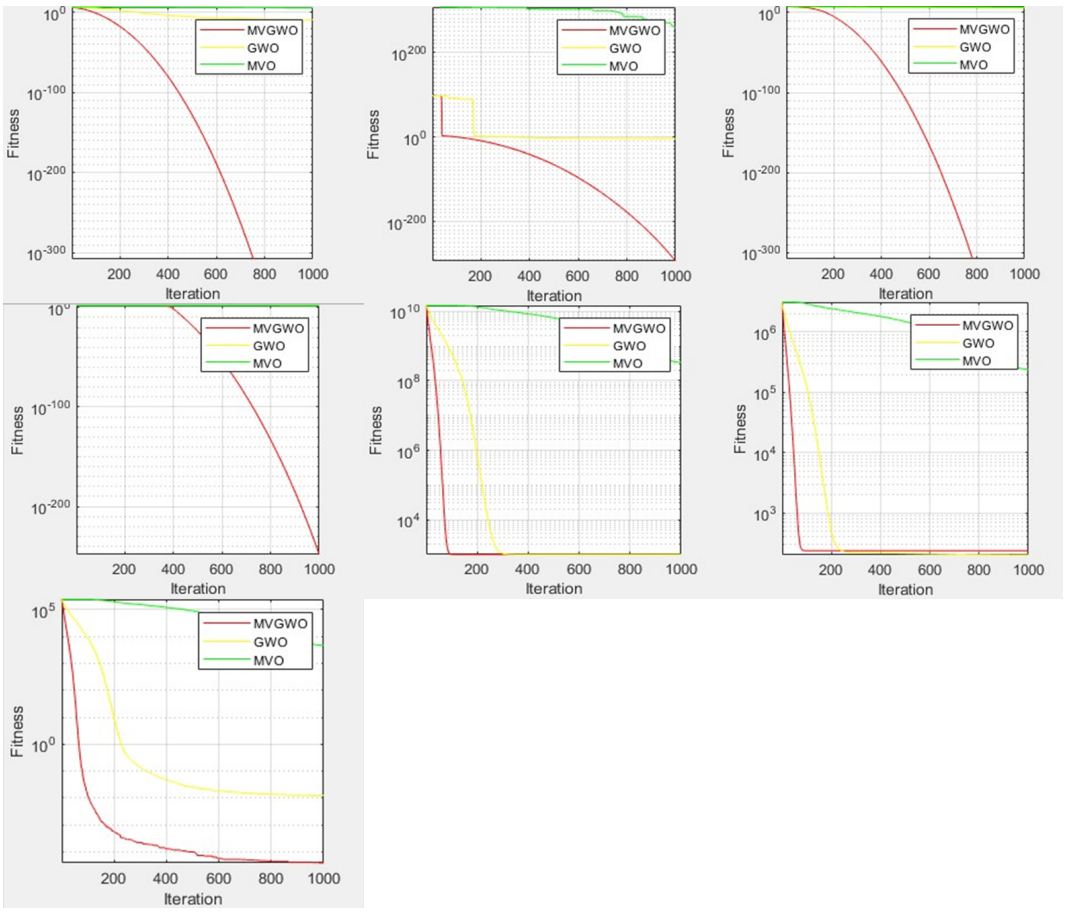

FIGURE 7.10 – Résultats des fonctions de test unimodales à 1000 dimensions (F1-F7)

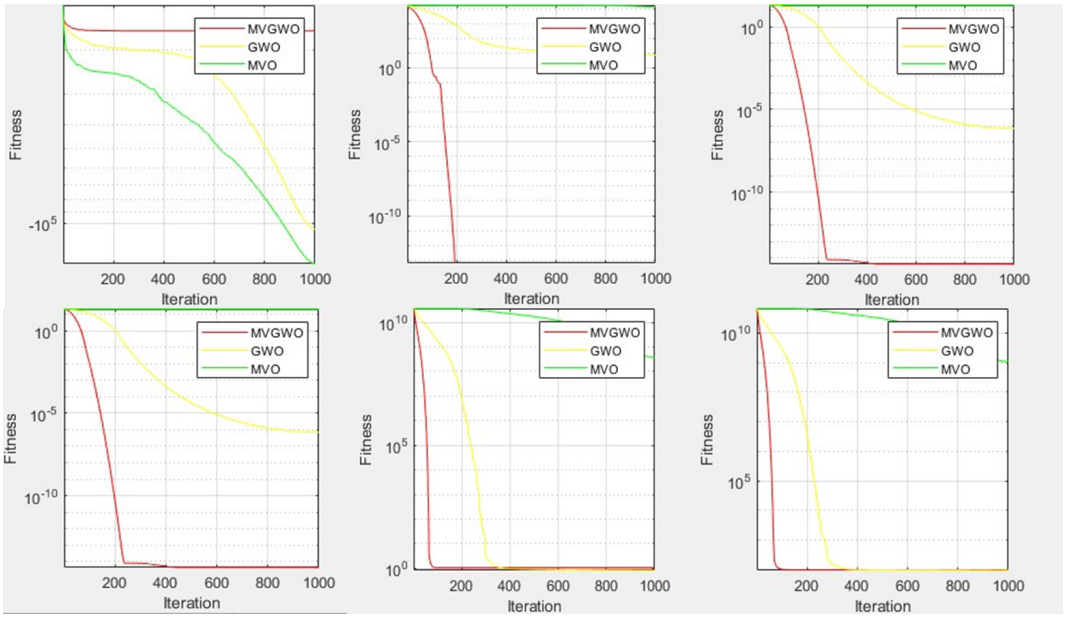

FIGURE 7.11 – Résultats des fonctions de test multimodales à 1000 dimensions (F8-F13)

Les graphiques de convergence des trois algorithmes sur les fonctions de grandes dimensions sont présentés dans les figures (7.6), (7.7), (7.8), (7.9), (7.10) et (7.11).
Bien que la dimension ait augmenté, MVGWO peut converger rapidement dès le début des itérations sur toutes les fonctions, et son taux de convergence est toujours plus élevé que GWO et MVO. Comme on peut le voir sur les figures (7.8) et (7.9), MVGWO converge rapidement vers la valeur optimale globale sur les fonctions de 500 dimensions : F1-F4, F7, F9, F10, et F11. Comparé à GWO et MVO, MVGWO est fortement compétitif dans la résolution des problèmes de grandes dimensions.

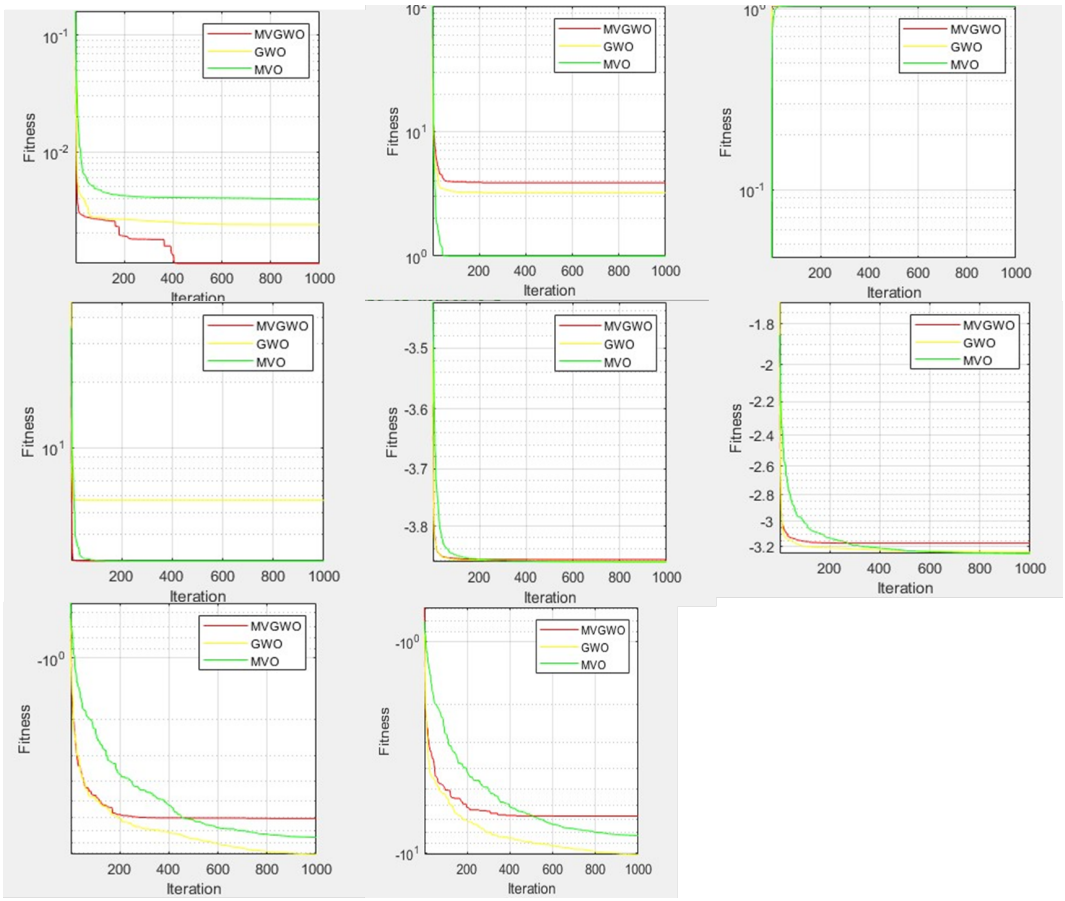

FIGURE 7.12 – Résultats des fonctions de référence multimodales à dimension fixe (F14-F22)

La figure (7.12) montre les courbes de convergence de MVGWO, GWO, et MVO sur les fonctions multimodales à dimension fixe. De toute évidence, la tendance de convergence de ces derniers est très proche. Cependant, MVGWO converge vers l'optimum global dans les fonctions F15, F18 et F19. Par conséquent, nous concluons que, bien qu'il existe de nombreux optima locaux dans les problèmes de fonctions multimodales fixes, le MVGWO a une bonne vitesse de convergence et une bonne précision.

7.0.8 Série d'expériences 2 : MVGWO pour la sélection d'attributs et l'optimisation des paramètres de SVM

Les machines à vecteurs de support (SVM) (CORTES et VAPNIK, 1995) sont une famille d'algorithmes d'apprentissage automatique qui résolvent les problèmes de classification, de régression et de détection d'anomalies (CORTES et VAPNIK, 1995). Les SVM ont été utilisés dans une variété d'applications, notamment la bio-informatique (ZHOU et TUCK, 2007), l'énergie renouvelable (BOUZERDOUM, MELLIT et PAVAN, 2013), et la chimio-informatique (VATSA, SINGH et NOORE, 2005).
Les SVM ont été développés pour la première fois par Vladimir Valpinik en 1990 (CORTES et VAPNIK, 1995), l'idée principale des SVM est de projeter les données d'apprentissage dans une dimension plus grande, de sorte que les données non linéairement séparables dans l'espace des caractéristiques d'entrée deviennent linéairement séparables dans l'espace des caractéristiques de haute dimension par la construction d'un hyperplan optimal avec des catégories séparées des données d'apprentissage (FARIS et al., 2018). Dans la littérature, SVM

prouve une excellente classification avec une grande précision de prédiction dans un large éventail de problèmes de classification et de régression (FARIS et al., 2018). Les SVM ont des paramètres de noyau et de pénalité qui ont un impact majeur sur leurs performances. Le paramètre de pénalité C contrôle le compromis entre la maximisation d'une marge de classification et la minimisation de l'erreur d'apprentissage. Le paramètre du noyau définit la transformation non linéaire de l'espace d'entrée vers un espace à haute dimension. Les chercheurs ont utilisé et mis en œuvre diverses fonctions noyau dans la littérature (polynomiale, sigmoïdes ou linéaires). Le noyau commun et fortement recommandé est la fonction de base radiale (RBF). Le noyau RBF aide le SVM à obtenir une prédiction précise et une efficacité fiable, comme l'indiquent plusieurs études (CHAO et HORNG, 2015 ; HUANG et WANG, 2006b ; YU, LIONG et BABOVIC, 2004). De plus, le RBF a moins de paramètres à optimiser et peut effectuer une meilleure analyse des données de plus grandes dimensions.
Deux problèmes importants doivent être abordés afin d'obtenir l'avantage du SVM et d'atteindre la meilleure capacité de généralisation avec une précision de prédiction élevée. Le premier est l'optimisation du paramètre du noyau du SVM et de ses paramètres de pénalité. Le second est la sélection du meilleur sous-ensemble d'attributs à inclure dans la phase d'apprentissage (FARIS et al., 2018 ; THARWAT et HASSANIEN, 2018). L'un des algorithmes classiques utilisés pour régler les paramètres est la recherche de grille. Il est utilisé pour optimiser les paramètres du SVM. Dans cet algorithme, les paramètres varient dans l'espace de recherche avec une taille de pas fixe. Cependant, cet algorithme est complexe, prend du temps et ne convient que pour optimiser quelques paramètres (STAELIN, 2003 ; ZHANG, CHEN et HE, 2010). Dans la littérature, de nombreuses études utilisent des algorithmes métaheuristiques pour optimiser les paramètres de SVM en raison de leur grande efficacité à générer des solutions acceptables, notamment lorsque l'espace de recherche est extrêmement grand et lorsque le problème est complexe (WANG, ZHAO et DEB, 2015). La sélection de caractéristiques (attributs) est une technique de prétraitement qui consiste à choisir un petit sous-ensemble ou une liste de N variables (caractéristiques) à partir d'un ensemble de données de M caractéristiques $(N < M)$.
La sélection de caractéristiques (FS) permet de :

- Éliminer les caractéristiques non pertinentes et redondantes dans un ensemble de données (FARIS et al., 2018).
- Diminuer le temps de formation (FARIS et al., 2018).
- Réduire la complexité du modèle de classification (FARIS et al., 2018).
- Augmenter les performances de classification du modèle (FARIS et al., 2018).

Il existe deux approches classiques pour effectuer la sélection des caractéristiques dans la classification : le filtre et l'enveloppe (wrapper).
L'approche filtrante évalue les caractéristiques d'un ensemble de données sans utiliser un algorithme d'apprentissage, tandis que l'approche enveloppante utilise un algorithme de classification spécifique pour évaluer la qualité du sous-ensemble de caractéristiques sélectionné.
De nombreux algorithmes métaheurtiques ont été proposés pour la sélection des caractéristiques et l'optimisation simultanée des paramètres de SVM.

7.0.9 Machine à vecteur de support (SVM)

La machine à vecteurs de support (SVM) est un modèle mathématique proposé par Vapinik (CORTES et VAPNIK, 1995). Une étude comparative a montré que le SVM est l'un des meilleurs classificateurs d'apprentissage automatique et peut-être utilisé à la fois pour la régression et la classification. L'objectif principal des SVM est d'orienter les hyperplans pour séparer les différentes classes. Lorsque les données sont linéairement séparables, l'algorithme SVM atteint une grande précision de classification. Cependant, l'efficacité des SVM

est affectée par des données non linéairement séparables. Les fonctions noyau sont utilisées pour résoudre ce problème. Par conséquent, les fonctions de noyau sont utilisées pour mapper les caractéristiques actuelles dans un nouvel espace de dimension supérieure où les données peuvent être séparées linéairement. L'utilisation des SVM présente deux défis principaux : la sélection des fonctions noyau appropriées et le réglage de leurs paramètres. Sur le plan informatique, la recherche du plan optimal est considérée comme un problème d'optimisation pour aider la fonction noyau à rechercher des décisions linéaires à travers une transformation non linéaire (CORTES et VAPNIK, 1995 ; WANG, 2005).
La figure (7.13) illustre un exemple d'un jeu de données de classe binaire séparé par des hyperplans optimaux de SVM.

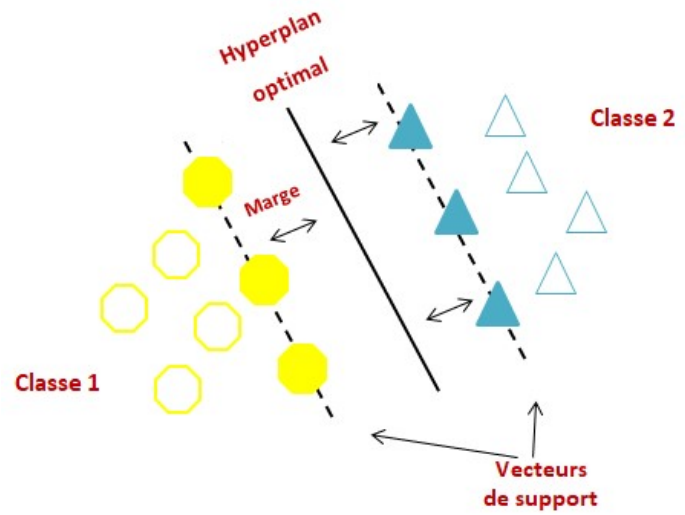

FIGURE 7.13 – Hyperplan optimal dans une machine à vecteurs de support

Il existe différents types de fonction noyau, les plus utilisés dans la littérature étant le noyau hyperbolique tangent, le noyau polynomial et le noyau RBF, comme le montrent les équations (7.10), (7.11), et (7.12) respectivement.

$$K_h(x_i, x_j) = tan\ h(c_1(x_i, x_j) + c_2) \tag{7.10}$$

$$K_p(x_i, x_j) = \left\langle x_i, x_j + 1 \right\rangle^d \tag{7.11}$$

$$K_{rbf}(x_i, x_j) = exp(-\gamma \left\| x_j - x_i \right\|^2),\ where\ \gamma > 0 \tag{7.12}$$

— L'un des problèmes majeurs des SVM est la solution de la fonction noyau et les valeurs de leurs paramètres qui ont un impact significatif sur la précision et la performance du modèle SVM (WANG, 2005).

Il existe de nombreuses approches dans la littérature pour optimiser les paramètres des SVM. Parmi elles, les algorithmes métaheuristiques qui ont prouvé leur efficacité et leur robustesse pour trouver des solutions optimales aux problèmes d'optimisation. Par conséquent, dans cette thèse, nous avons introduit un nouvel algorithme nommé MVGWO pour optimiser les paramètres du SVM pour la première fois dans la littérature.

7.0.10 Modèle MVGWO-SVM proposé

Trois points importants sont pris en considération concernant l'implémentation proposée de MVGWO pour la sélection d'attributs et l'optimisation des paramètres de SVM : la fonction de fitness, le schéma d'encodage des individus, et l'architecture du système (FARIS et al., 2018). Ces points sont décrits comme suit :

Schéma d'encodage

Les individus sont encodés sous la forme d'un vecteur de N éléments de nombres réels, où N est le nombre d'attributs originales dans un ensemble de données, plus deux éléments pour représenter les paramètres du SVM : le coût (cost) (C) et le gamma (γ). Chaque élément du vecteur est un nombre aléatoire dans $[0, 1]$. Par conséquent, ces éléments sont arrondis et peuvent prendre la valeur 1 ou 0 ; indiquant si l'attribut est sélectionné ou non comme le montre la figure (7.14).
Pour les paramètres du SVM C et γ ; C est mappé à l'intervalle $[0, 35000]$ et γ est mappé à $[0, 32]$ (FARIS et al., 2018). Dans notre expérience, les valeurs de C et γ sont transformées linéairement selon l'équation (7.13) :

$$B = \frac{A - min_A}{max_A - min_A}(max_B - min_B) + min_B \tag{7.13}$$

Évaluation de fitness (fonction objectif)

Pour évaluer les solutions générées, nous utilisons l'exactitude de la classification qui est calculée à l'aide de la matrice de confusion présentée dans le tableau (6.11). La matrice de confusion est la principale source d'évaluation des modèles de classification (FARIS et al., 2018).

$$Exactitude = \frac{TP + TN}{TP + FN + FP + TN} \tag{7.14}$$

Architecture du système

Afin d'effectuer la sélection des caractéristiques et l'optimisation des paramètres du SVM en utilisant MVGWO. Nous avons utilisé deux architectures de système. Le premier système mis en œuvre et utilisé dans (HUANG et WANG, 2006a ; LIN et al., 2008) est nommé architecture (1). Le second est une amélioration du premier système proposé dans (FARIS et al., 2018) nommé architecture (2).
Le flux de travail de ces architectures est donné ci-dessous.

— Normalisation des données : nous avons mis en correspondance toutes les caractéristiques de tous les ensembles de données à la même échelle pour qu'elles soient dans l'intervalle $[0, 1]$ afin d'éviter le mauvais effet de certaines caractéristiques qui ont des valeurs de biais différentes sur le processus d'apprentissage de l'algorithme SVM. Cette étape du processus a été appliquée en utilisant l'équation (7.15).

$$B = \frac{A - min_A}{max_A - min_A} \tag{7.15}$$

— Décodage des individus MVGWO : dans cette étape, les vecteurs sont divisés en deux parties : la première partie du vecteur est pour les paramètres du SVM tandis que la seconde est pour les caractéristiques sélectionnées comme représenté dans la figure (7.14).

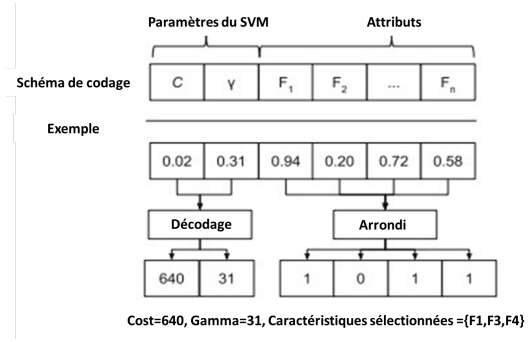

FIGURE 7.14 – Schéma de codage des individus MVGWO pour l'optimisation de SVM et la sélection d'attributs (FARIS, MIRJALILI et ALJARAH, 2019b)

— Sous-ensemble de caractéristiques sélectionnées : après que les individus ont été décodés en un vecteur binaire, les caractéristiques sont choisies dans l'ensemble de données d'apprentissage.
— Évaluation du fitness : chaque solution générée par l'algorithme MVGWO qui représente les caractéristiques sélectionnées et les paramètres SVM est évaluée à l'aide de la fonction fitness qui représente l'exactitude de la classification.
— Critère d'arrêt : l'ensemble du processus s'arrête si une condition d'arrêt est satisfait. Dans ce travail, la condition de terminaison est le nombre maximum d'itérations.

La différence entre ces deux architectures réside dans la méthode d'apprentissage et de test et dans l'évaluation du fitness.
Dans l'architecture (1), l'ensemble des données d'apprentissage est déployé pour créer le modèle SVM, puis la valeur objective renvoyée par la fonction objective est l'évaluation du SVM entraîné en utilisant l'ensemble de données de test. Par conséquent, dans l'architecture (2), l'ensemble de données d'apprentissage est divisé en parties plus petites afin d'effectuer une validation croisée K-blocs, de sorte que nous exécutons l'algorithme SVM K fois et les résultats moyens sont retournés. Dans cette architecture, l'ensemble de données de test n'est pas utilisé dans l'itération de la métaheuristique, il est utilisé pour évaluer la solution finale générée (les meilleurs paramètres du SVM et les attributs finaux sélectionnés).

7.0.11 Expériences et résultats

Dans cette section, nous présentons l'évaluation de l'algorithme proposé (MVGWO) pour la sélection d'attributs et l'optimisation des paramètres de SVM. Quinze jeux de données (tableau (7.12)) ont été utilisés pour évaluer les cinq algorithmes et pour chaque jeu de données, les expériences ont été répétées dix fois pour obtenir des résultats statistiquement significatifs. Le nombre d'itérations et la taille de la population ont été fixés à 50 et 20 respectivement. En général, l'interaction entre l'algorithme de classification et la sélection des caractéristiques rend la sélection des caractéristiques basée sur la méthode d'enveloppe (wrapper) très puissante. Mais il est coûteux en calcul, et il peut sur-ajuster. Par conséquent, il a été observé par l'expérience qu'un petit nombre de générations peut converger vers une solution et réduire le temps de calcul de l'algorithme métaheuristique (FARIS et al., 2018).
Comme indiqué précédemment, la différence entre les architectures (1) et (2) réside dans la méthodologie d'apprentissage et de test utilisée dans chacune d'elles. Dans l'architecture (1), nous avons utilisé 10 validations croisées ; 9 blocs sont utilisés pour l'apprentissage du SVM et un pour le test, tandis que la valeur de fitness est renvoyée par la fonction de fitness basée sur le 10 ème bloc de test. Dans l'architecture (2), nous avons utilisé 10 blocs pour la

validation croisée externe et 3 blocs pour la validation interne (FARIS et al., 2018).
Pour obtenir des résultats statistiquement significatifs, les expériences sont répétées dix fois.

TABLE 7.12 – Les ensembles de données de classification

Datasets	Attributs	Ensemble d'apprentissage	Ensemble de test
Blood	4	493	255
Breast cancer	8	461	238
Diabetes	8	506	262
Heart	13	179	91
Hepatitis	10	102	53
Liver	6	79	41
Parkinson	22	128	67
Lymphography	18	98	50
Vote	16	198	102
Wine	13	118	60
Zoo	16	67	34
Sonar	60	138	70
Exactly	13	660	340
Inosophere	34	232	119
BreastEW	31	178973	93986

Résultats et discussions

La comparaison de MVGWO a été effectuée avec quatre algorithmes métaheuristiques : MVO, GWO, WOA et BAT. Les performances des cinq algorithmes métaheuristiques ont été évaluées en fonction du meilleur, la moyenne, le pire et de l'écart type de l'exactitude de la classification et de la moyenne, et l'écart type des caractéristiques sélectionnées. Le tableau(7.13) et le tableau(7.14) montrent les résultats obtenus avec l'architecture (1) et l'architecture (2), respectivement.

TABLE 7.13 – Résultats de l'architecture 1

Algorithm/	MVGWO+SVM		MVO+SVM		GWO+SVM		WOA+SVM		BAT+SVM	
Datasets	Exactitude	N° de AS	Exactitude	N° de AS	Exactitude	N° de AS	Exactitude	N° de AS	Exactitude	N° de AS
BreastCancer	98.56±1.17	4.2±0.918	98.14±1.512	4.3±1.159	98.13±1.66	4.3±1.159	97.712±2.149	4±1.414	98.142±1.910	4.1±1.728
Blood	98.61±1.09	2.8±0.788	97.21±1.68	3.4±0.699	98.61±1.59	3.1±0.994	98.613±1.093	2.8±1.1353	97.05±2.70	3.4±1.264
Diabetes	82.414±3.69	4.3±1.059	82.289±2.155	3.7±1.3375	82.67±4.66	4.2±1.135	79.431±3.372	4.4±0.966	81.64±3.156	6.5±1.779
Heart	90.74±5.58	4.9±1.370	88.148±5.99	4.6±1.837	89.62±7.36	5±1.333	81.85±7.7	4.1±1.663	86.29±6.544	5.7±4.164
Hepatitis	93.66±5.119	5.119±6.4	91.52±6.589	8.5±2.273	90.09±7.625	10.5±2.46	83.1905±12.344	10.7±2.008	83.90±7.89	7.2±5.391
Liver	81.487±5.13	3.8±1.31	82.94±4.804	3.5±0.971	79.714±6.085	3.6±1.075	78.873±5.51	3.9±0.994	81.18±7.084	4.5±1.354
Parkinson	100±0.0	11.2±1.47	100±0.0	11.5±2.415	99.5±1.5811	11.2±1.8135	99.473±1.664	11.8±2.485	99.5±1.58	11.5±4.00
Lymphography	97.28±4.709	8.7±2.907	95.952±3.486	7.9±1.852	95.23±5.661	9.7±1.418	91.142±66.02	8.9±1.523	93.90±4.937	9.9±5.83
Vote	99±1.610	7.6±1.89	99±1.610	6.6±2.065	94±6.62	4.3±1.567	98.333±2.37	7.1±1.3703	98±2.3308	7.9±4.173
Wine	100±0.0	6.3±1.251	100±0	7±1.633	98±2.33	7.6±1.577	100±0.0	7.2±1.619	100±0.0	6.3±1.418
Zoo	99±3.16	7.7±1.70	99.09±2.007	27.4±2.065	100±0.0	8.1±1.663	100±0.0	7.7±1.494	99±3.162	7.7±2.311
Sonar	99.52±1.5	24.9±4.09	98±6.324	7.8±1.135	97.59±2.53	28.1±3.164	93.738±4.55	28.2±2.44	97.57±2.561	27.3±10.93
Exactly	97.8±6.957	8.3±0.948	98.57±2.020	16.8±2.740	88.3±12.69	6.8±1.475	80.1±1.967	7.4±1.711	97.5±5.038	9.1±2.378
Ionosphere	99.42±1.2	16±3.80	99.473±0.847	15.4±2.716	98.015±2.324	15.9±2.601	97.166±3.253	18.3±3.365	98.57±2.42	15.6±6.749
BreastEW	99.12±1.49	14.9±2.80	96.112±3.642	15.7±2.359	98.94±1.49	15.5±2.646	99.116±1.503	15.1±2.469	98.77±2.346	14.3±4.347

TABLE 7.14 – Résultats de l'architecture 2

Algorithme/	MVGWO+SVM		MVO+SVM		GWO+SVM		WOA+SVM		BAT+SVM	
Datasets	Exactitude	N° de AS	Exactitude	N° de AS	Exactitude	N° de AS	Exactitude	N° de AS	Exactitude	N° de AS
BreastCancer	95.56±1.978	5.2±0.994	94.99±2.45	5.3±0.674	95.7±2.875	5.2±0.918	5.277±2.528	7.8±0.632	95.41±2.01	4.9±0.875
Blood	97.91±1.598	2.8±0.918	98.61±1.09	3.3±0.948	97.566±2.618	2.7±0.823	79.566±2.618	3.6±0.516	97.743±1.17	3.1±1.370
Diabetes	75.921±3.785	4.8±1.316	75.91±5.59	4.9±1.286	76.309±4.32	5.1±2.13	76.56±4.61	8±0	75.647±4.86	3.4±1.349
Heart	80.370±7.62	4.3±0.948	80.3704±8.19	5±2.05	81.48±8.55	4.9±1.96	78.148±8.633	5.6±4.056	75.92±8.597	4.8±1.398
Hepatitis	65.57±10.416	7.1±3.071	61.857±11.556	9.4±1.646	61.19±6.33	7.5±1.9	65.428±12.013	10.7±5.86	66.952±12.271	9.5±2.990
Liver	64.59±11.63	3±1.490	66.932±6.58	3±1.247	66.42±8.04	4±1.247	70.092±8.895	5.6±1.264	62.92±9.185	3.6±1.075
Parkinson	96.44±3.40	40.1±3.247	90.78±6.57	12.3±2.86	94.78±4.96	9±1.054	92.789±6.62	13±5.316	93.868±4.059	12.1±1.792
Lymphography	82.47±6.43	8.8±1.873	73.809±14.78	8.9±2.131	75.04±14.03	7.6±2.366	77.571±10.05	14.4±5.815	93.868±4.059	9.5±2.460
Vote	94.33±4.172	4.1±1.912	93±4.288	7.2±2.699	94±6.62	4.3±1.567	92.66±8.72	7.4±4.550	93.33±5.665	6.9±1.912
Wine	97.156±4.08	7.3±1.251	97.67±4.083	8.2±1.619	97.74±3.91	8.1±1.595	99.77±5.36	9.5±3.407	97.74±3.916	8.0±2.0
Zoo	93±8.23	6.3±1.251	90.18±7.93	7.7±1.94	92.09±7.86	7.2±2.394	92±6.324	11.2±4.366	89±9.944	8.3±1.567
Sonar	85.09±7.72	17.2±2.25	84.54±6.9	27.8±3.119	86.59±8.351	22.7±4.37	85.119±3.432	31.2±3.43	83.59±5.83	29.6±2.633
Exactly	100±0.0	6.9±0.31	100±0.0	7.4±0.843	91.1±14.46	7.1±2.333	97.1±5.68	9.3±2.263	79.4±17.01	7.5±1.649
Ionosphere	93.45±4.46	11.7±1.337	92.03±4.79	18.7±3.625	92.58±4.3	4.38±13.6	92.9±4.6	21.9±7.4	93.142±5.25	16.3±2.162
BreastEW	97.54±2.36	14.2±2.573	96.112±3.642	15.7±2.359	96.309±2.40	13.4±2.63	97.183±2.518	25.3±6.945	96.121±2.733	16.7±2.869

Les résultats de l'architecture (1) montrent que MVGWO a obtenu les taux d'exactitude moyens les plus élevés par rapport à MVO, GWO, WOA et BAT dans 10 jeux de données sur 15, dépassant un taux de précision de 99 % dans cinq d'entre eux. De plus, MVGWO présente des valeurs d'écart type plus faibles pour la plupart des jeux de données. On peut également remarquer que les quatre optimiseurs ont des résultats très proches en ce qui concerne le nombre de caractéristiques sélectionnées.

TABLE 7.15 – Meilleurs résultats obtenus sur la base de l'architecture 1

Datasets	MVGWO+SVM	MVO+SVM	GWO+SVM	BAT+SVM	WOA+SVM
BreastCancer					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	3	3	3	2	2
Cost(C)	10634.8524	19528.547	0.01	34990.6633	2191.842
γ	4.1416	0.1627	7.9359	31.9542	4.1416
Blood					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	2	2	2	1	1
Cost(C)	135828.124	20491.198	26581.7513	35000	7528.2267
γ	0.13408	27.799	25.7082	32	30.1525
Diabetes					
Meilleure exactitude	88.3117	90.909	83.1169	85.7143	86.842
No. d'attributs sélectionnés	5	2	2	4	7
Cost(C)	26531.5558	4425.6434	18386.297	27841.0099	33226.2976
γ	0.05752	9.1906	10.0843	0.0001	24.3417
Heart					
Meilleure exactitude	96.296	96.2963	96.2963	96.2963	100
No. d'attributs sélectionnés	2	3	3	5	6
Cost(C)	7038.7962	13045.43	27389.7794	35000	21185.4602
γ	8.6774	32	14.2662	0.0001	0.0001
Hepatitis					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	4	11	7	9	4
Cost(C)	27572.2471	4832.0676	11928.6103	8345.7252	2659.458
γ	16.6689	0.0001	1.4293	31.6455	1.3209
Liver					
Meilleure exactitude	88.2353	88.2353	88.2353	91.1765	94.1176
No. d'attributs sélectionnés	3	3	4	4	3
Cost(C)	18299.355	23546.2523	25641.8595	34149.645	6519.7337
γ	5.7563	8.8822	3.2262	0.023717	10.9441
Parkinson					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	8	10	6	8	8
Cost(C)	22681.9024	7916.9982	11888.601	35000	29747.2304
γ	11.0146	1.4787	20.4663	31.9367	20.2004
Lymphography					
Meilleure exactitude	100	100	100	100	100
NNo. d'attributs sélectionnés	4	9	8	10	4
Cost(C)	6213.8803	7395.9228	17539.8848	35000	2303.3763
γ	32	15.5389	0.0001	0.0001	22.083
Vote					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	6	7	6	6	1
Cost(C)	23343.948	19106.5268	5059.3321	43.8848	0.01
γ	32	27.4206	7.477	31.8956	19.5221
Wine					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	4	4	4	5	4
Cost(C)	27281.1682	30827.594	27844.9409	35000	9096.3939
γ	16.8916	23.0442	22.9366	0.002473	22.3304
Zoo					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	6	6	4	5	4
Cost(C)	8424.6553	21250.6541	34801.9705	592.625	12155.075
γ	12.4382	10.8859	9.3777	32	23.287
Sonar					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	19	24	24	26	24
Cost(C)	5404.9692	18835.3194	27637.7038	31904.46	25765.4137
γ	3.0578	4.6774	0.6325	3.0741	1.0302
Exactly					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	7	7	7	7	6
Cost(C)	26063.4256	15578.1379	32668.9288	35000	17081.8763
γ	10.1892	15.0837	11.1187	32	15.8769
Inosophere					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	13	12	14	17	6
Cost(C)	17943.1875	14807.6794	34508.7709	34936.5561	19369.766
γ	4.2251	7.1903	4.5114	1.7072	0.33787

Les meilleurs résultats et paramètres obtenus sur la base de l'architecture (1) sont listés dans le tableau (7.15). On peut voir que tous les algorithmes ont atteint un taux d'exactitude de 100% dans 12 ensembles de données sur 15.
En examinant les résultats de l'architecture (2) dans le tableau (7.14), on peut observer que les taux d'exactitude obtenus par tous les optimiseurs sont inférieurs à ceux de l'architecture (1). Cette diminution est due au schéma d'apprentissage / test incorporé dans cette architecture. Comme il a été mentionné précédemment, la partie teste de cette architecture n'a pas été représentée pendant le processus d'optimisation.
Par conséquent, les résultats devraient être faibles mais plus crédibles. D'après les résultats obtenus par l'architecture (2) dans le tableau (7.14). L'algorithme MVGWO montre une exactitude moyenne plus élevée que les autres optimiseurs dans 7 jeux de données et il est classé deuxième dans 5 jeux de données. De plus, MVGWO montre une plus grande robustesse dans la plupart des jeux de données, en particulier pour Breast cancer, Blood, Diabetes, Heart, Parkinson, Lymphography, vote, exactly, and BreastEW.

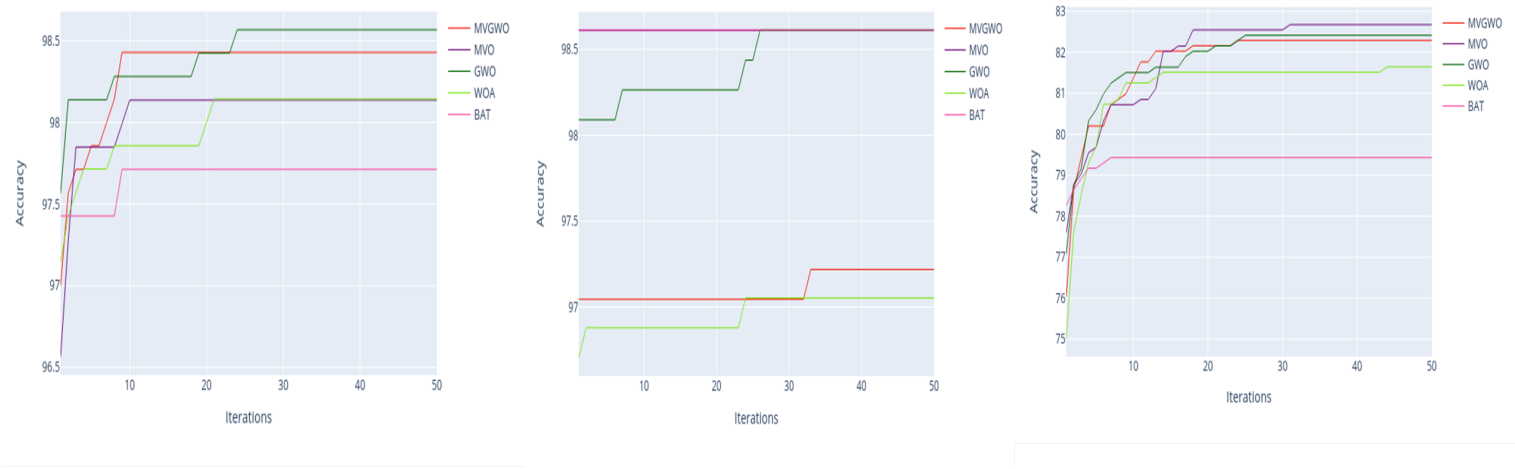

FIGURE 7.15 – Courbe de convergence basé sur l'exactitude pour BreastCancer, Blood, et Diabetes respectivement

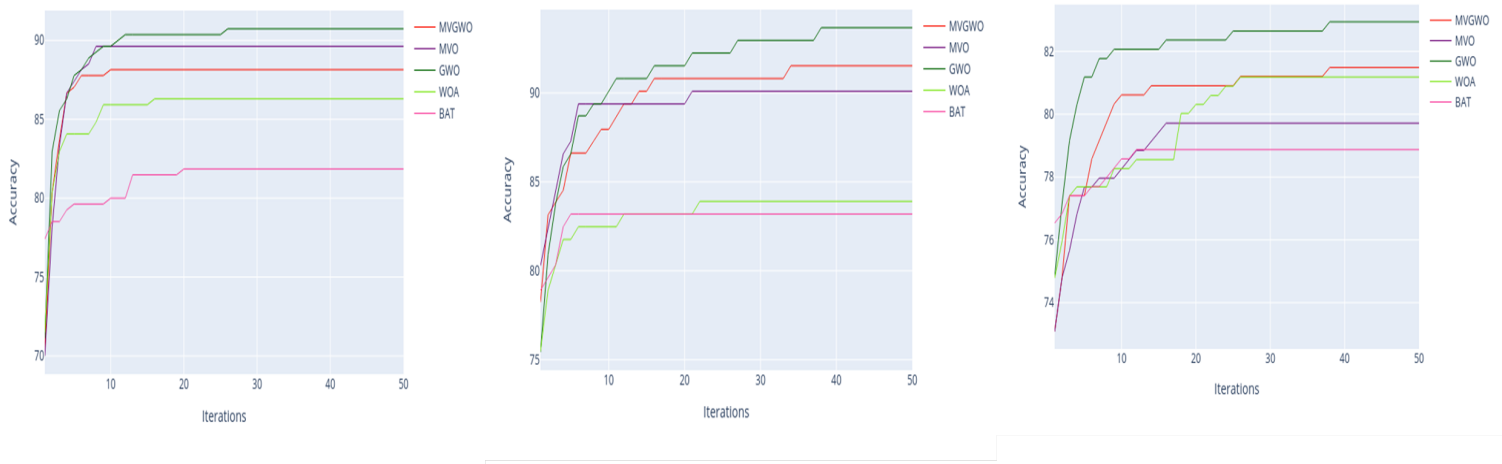

FIGURE 7.16 – Courbe de convergence basé sur l'exactitude pour Heart, Hepatitis, et Liver, respectivement

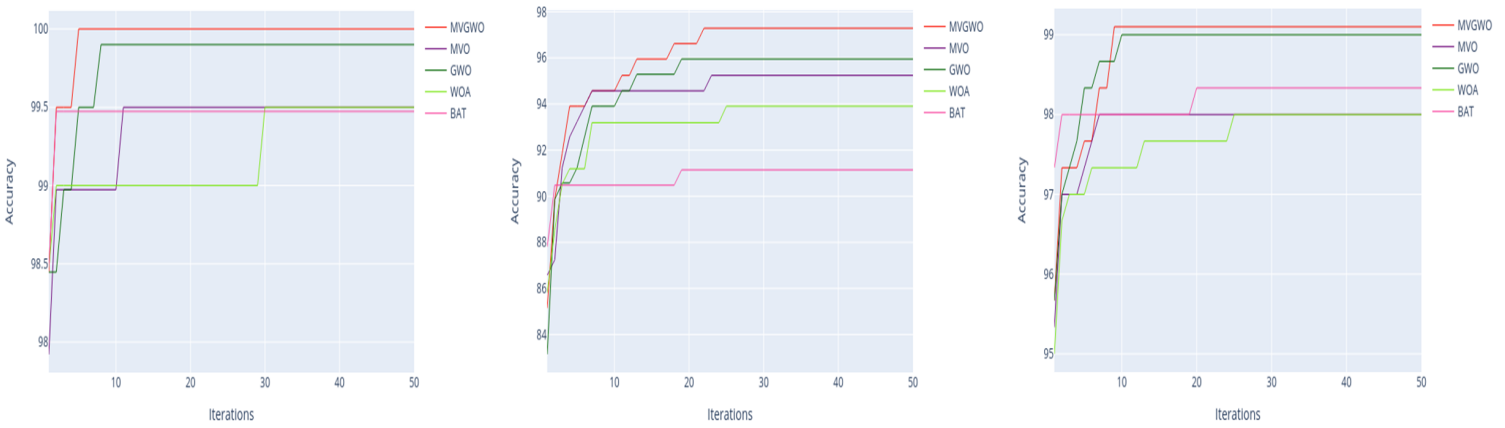

FIGURE 7.17 – Courbe de convergence basé sur l'exactitude pour Parkinson, Lymphography, et Vote, respectivement

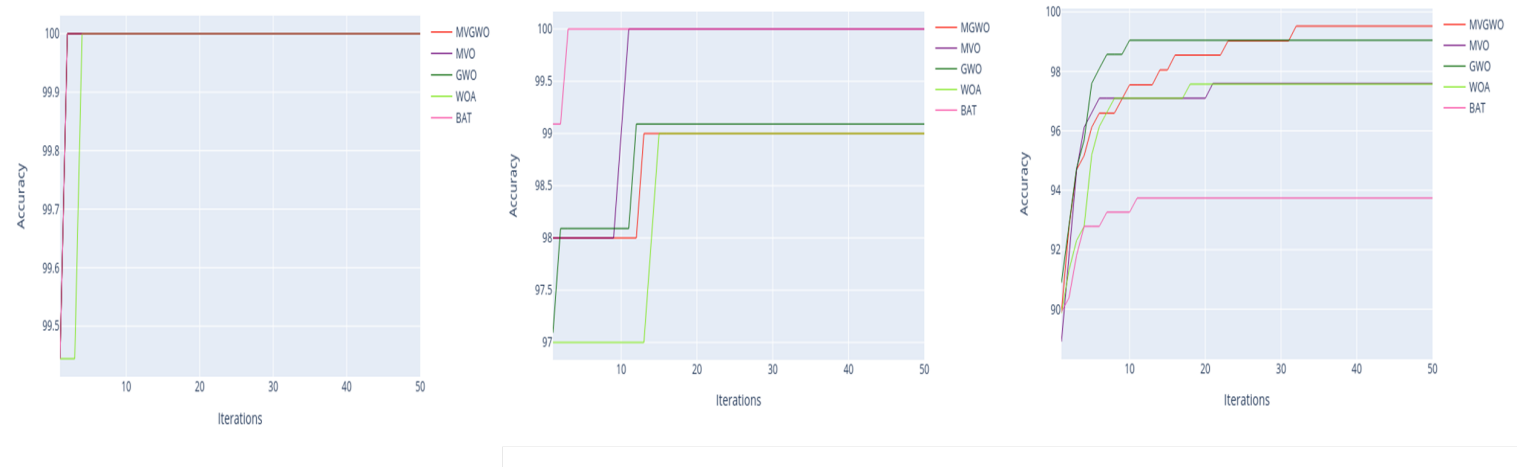

FIGURE 7.18 – Courbe de convergence basé sur l'exactitude pour Wine, Zoo, et Sonar, respectivement

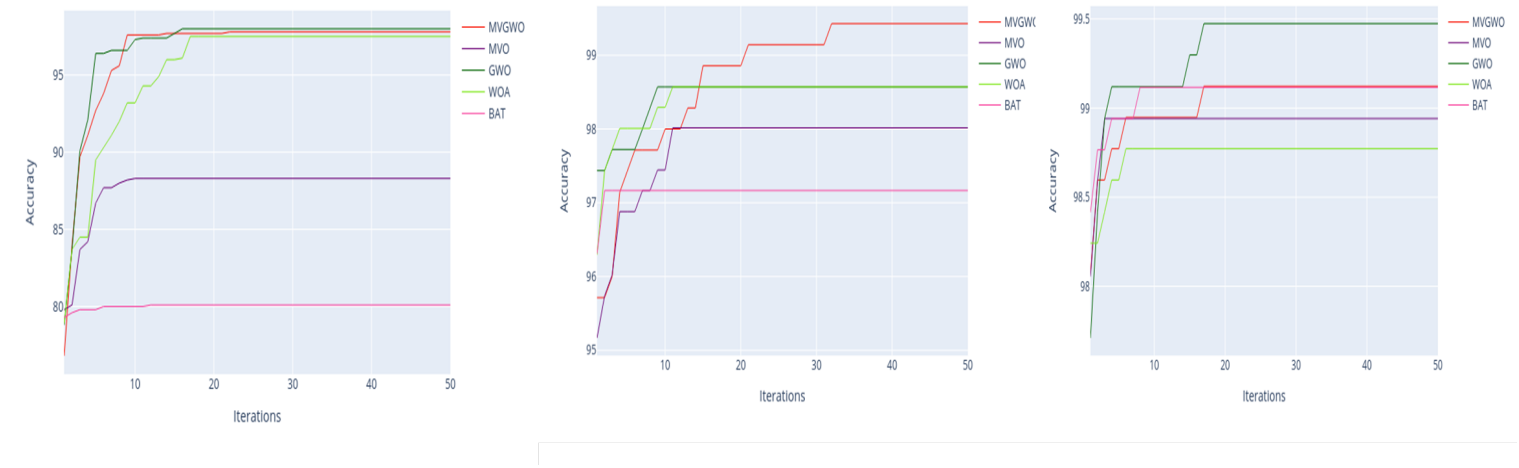

FIGURE 7.19 – Courbe de convergence basé sur l'exactitude pour Exactly, Inosophere, et BreastEW, respectivement

Les figures (7.15), (7.16), (7.17), (7.18) et (57.19) montrent les courbes de convergence de tous les optimiseurs basés sur l'architecture (2). Dans ces figures, MVGWO montre une vitesse de convergence plus élevée dans la plupart des jeux de données.

TABLE 7.16 – Meilleurs résultats obtenus sur la base de l'architecture 2

Datasets	MVGWO+SVM	MVO+SVM	GWO+SVM	BAT+SVM	WOA+SVM
BreastCancer					
Meilleure exactitude	98.5710	98.571	100	98.571	100
No. d'attributs sélectionnés	5	5	6	2	8
Cost(C)	35000	3572.08	13082.418	34511.67	1303.78
γ	0.001	22.3438	0.0126	1.231	0.0381
Blood					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	2	3	1	3	3
Cost(C)	8990.477	18158.72	7470.902	34949.82	81.299
γ	13.2306	0.065	1.7649	31.9641	17.0019
Diabetes					
Meilleure exactitude	82.894	84.4156	81.8182	83.116	81.818
No. d'attributs sélectionnés	5	8	4	3	8
Cost(C)	30906.389	20155.5298	14794.88	34946.08	2515.502
γ	0.2346	0.0001	0.0059	0.01385	0.016
Heart					
Meilleure exactitude	88.888	92.592	92.592	88.888	92.59
No. d'attributs sélectionnés	3	7	3	5	4
Cost(C)	22681.9024	7916.9982	11888.601	35000	29747.2304
γ	2.0629	0.0001	0.2568	0.0118	25.894
Hepatitis					
Meilleure exactitude	85.714	78.571	73.333	78.571	80
No. d'attributs sélectionnés	7	10	8	7	19
Cost(C)	18259.609	32576.59	16821.697	1.1755	508.396
γ	8.7467	13.381	29.822	1.7722	0.0001
Liver					
Meilleure exactitude	80	77.142	73.529	82.352	82.8571
No. d'attributs sélectionnés	2	5	5	4	6
Cost(C)	25545.019	32705.51	2636.208	28642.78	188.411
γ	0.2764	0.0511	0.2054	0.0568	0.1415
Parkinson					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	5	14	8	12	9
Cost(C)	1899.665	10829.48	20207.238	34495.89	34420.544
γ	21.0349	2.931	213.525	15.9891	9.933
Lymphography					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	4	9	8	10	4
Cost(C)	6213.8803	7395.9228	17539.8848	35000	2303.3763
γ	32	15.5389	0.0001	0.0001	22.083
Vote					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	6	7	6	6	1
Cost(C)	23343.948	19106.5268	5059.3321	43.8848	0.01
γ	32	27.4206	7.477	31.8956	19.5221
Wine					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	4	4	4	5	4
Cost(C)	27281.1682	30827.594	27844.9409	35000	9096.3939
γ	16.8916	23.0442	22.9366	0.002473	22.3304
Zoo					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	6	6	4	5	4
Cost(C)	8424.6553	21250.6541	34801.9705	592.625	12155.075
γ	12.4382	10.8859	9.3777	32	23.287
Sonar					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	19	24	24	26	24
Cost(C)	5404.9692	18835.3194	27637.7038	31904.46	25765.4137
γ	3.0578	4.6774	0.6325	3.0741	1.0302
Exactly					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	7	7	7	7	6
Cost(C)	26063.4256	15578.1379	32668.9288	35000	17081.8763
γ	10.1892	15.0837	11.1187	32	15.8769
Inosophere					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	13	12	14	17	6
Cost(C)	17943.1875	14807.6794	34508.7709	34936.5561	19369.766
γ	4.2251	7.1903	4.5114	1.7072	0.33787
BreastEW					
Meilleure exactitude	100	100	100	100	100
No. d'attributs sélectionnés	10	12	11	12	5
Cost(C)	35000	30415.7459	4015.307	35000	1630.0844
γ	6.7689	20.2696	0.057051	0.0001	6.1259

Les meilleurs résultats et paramètres obtenus sur la base de l'architecture (2) sont présentés dans le tableau (7.16). On peut voir que tous les algorithmes ont atteint un taux de précision de 100 dans 10 jeux de données sur 15.

Pour vérifier la signification des différences entre les résultats de MVGWO et ceux des autres optimiseurs, le test statistique non paramétrique de Wilcoxon est effectué.

TABLE 7.17 – Valeurs p du test de Wilcoxon des résultats de la classification MVGWO par rapport aux autres algorithmes ($p >= 0,05$).

	GWO	MVO	BAT	WOA
BreastCancer	0.090	0.4698	0.0172	0.4515
Blood	0.207	0.4777	0.5828	0.2142
Diabetes	1.697E-04	0.4777	1.693E-04	1.707E-04
Heart	0.006	1.000	0.0107	0.049
Hepatitis	0.001	0.054	0.049	0.0256
Liver	0.760	0.5703	0.3066	0.056
Parkinson	0.034	0.0173	0.054	0.2266
Lymphography	2.696E-04	2.218E-04	0.0015	1.639E-04
Vote	0.030	0.2520	0.6734	0.4347
Wine	0.368	0.7891	0.3681	0.3681
Zoo	6.654E-04	0.056	0.3699	0.6807
Sonar	0.025	0.056	0.0015	0.054
Exactly	0.056	0.3681	2.297E-04	0.0350
Inosophere	0.818	0.3827	0.7764	0.5057
BreastEW	0.030	0.2997	0.7578	0.7578

Le test est effectué sur la base des résultats du MVGWO par rapport à chacun des autres optimiseurs à un niveau de signification de 5%. Dans le tableau (7.17), les valeurs p obtenues par le test sont listées. Toutes les valeurs p du tableau inférieures à 0,05 signifient que l'hypothèse nulle est rejetée (indiquant une différence significative) à un niveau de signification de 5%. D'après les résultats, on peut voir que MVGWO est significativement meilleur que GWO dans 10 ensembles de données sur 15. Alors que MVGWO est significativement meilleur que BAT, WOA et MVO dans 8, 7 et 5 ensembles de données, respectivement.

Comparaison avec la recherche de grille (sans sélection d'attributs)

Dans cette expérience, nous comparons MVGWO avec la recherche de grille pour optimiser les paramètres de SVM. Pour rendre la comparaison équitable, MVGWO est appliqué uniquement pour l'optimisation des paramètres sans la partie de sélection des caractéristiques, car la recherche de grille n'a pas cette capacité. Les deux techniques ont été appliquées avec une validation croisée 10-fois. La recherche de grille est utilisée comme décrite dans (FARIS et al., 2018 ; HUANG et WANG, 2006a).

TABLE 7.18 – Comparaison entre MVGWO et la recherche de grille pour l'optimisation des paramètres du SVM

	Arch 1		Arch 2	
	MVGWO	Grid	MVGWO	Grid
BreastCancer	97.56±1.664	94.90±0.020	95.70±2.781	96.60±0.050
Blood	98.09±0.9878	95.00±0.041	97.39±2.048	93.30±0.054
Diabetes	80.98±4.1438	77.33±0.089	75.78±4.692	76.10±0.031
Heart	81.11±7.2934	68.70±0.226	76.29±6.343	74.00±0.091
Hepatitis	77.47±8.6971	82.90±0.097	59.23±8.732	55.60±0.022
Liver	80.32±4.7133	68.30±0.014	71.26±7.756	67.60±0.009
Parkinson	97.42±3.6817	85.00±0.214	94.94±4.084	85.60±0.059
Lymphography	92.57±6.6219	87.00±0.019	83.04±6.667	80.30±0.077
Vote	96.33±2.4595	93.70±0.081	94.00±6.614	93.30±0.018
Wine	99.44±6.9921	96.60±0.074	97.18±3.953	95.00±0.018
Zoo	96.00±6.9921	95.00±0.134	94.00±6.992	93.30±0.054
Sonar	92.73±4.782	67.60±0.299	88.40±8.043	81.20±0.085
Exactly	100.0±0.000	68.80±0.008	100.0±0.0	69.00±0.001
Inosophere	96.57±3.7616	88.80±0.132	95.14±4.269	87.30±0.055
BreastEW	98.59±1.811	97.90±0.041	98.06±1.744	97.10±0.017

Le tableau (7.18) montre les résultats de la comparaison basée sur les architectures (1) et (2) susmentionnées. Les résultats d'architecture (1) montrent que MVGWO est nettement meilleur que la recherche de grille dans 14 ensembles de données. Le jeu de données Hepatitis était le seul qui montre de meilleures performances pour la recherche de grille.
En vérifiant les exactitudes moyennes de classification obtenues par architecture (2). On peut voir que les taux de précision obtenus par MVGWO sont plus élevés que ceux de la recherche de grille dans 13 jeux de données. Les seules exceptions sont pour Breast cancer et Diabetes.

7.0.12 Conclusion de l'approche

Dans cette approche, un algorithme hybride basé sur l'optimiseur loup gris avec l'optimiseur multi-vers nommé MVGWO est proposé pour résoudre des problèmes d'optimisation à grande dimension. Une version améliorée de GWO basée sur un poids d'inertie adaptatif est proposée pour améliorer la capacité de recherche globale de cet algorithme et pour maintenir la diversité des solutions. L'algorithme MVGWO combine la bonne capacité d'exploitation de GWO avec la forte capacité d'exploration de l'algorithme MVO.
Afin d'équilibrer l'exploitation et l'exploration, un coefficient d'équilibre adaptatif est utilisé dans cet algorithme. La probabilité d'exploitation ou d'exploration est contrôlée par le coefficient d'équilibre. En changeant le coefficient d'équilibre, l'algorithme MVGWO peut éviter la valeur optimale locale autant que possible et peut avoir une vitesse de convergence rapide.
Pour vérifier la supériorité des performances de l'algorithme MVGWO, vingt-deux fonctions de test de différents types et dimensions sont utilisées pour tester le MVGWO proposé et les résultats démontrent clairement que les performances du MVGWO proposé sont supérieures aux algorithmes GWO et MVO. Il est évident que MVGWO présente une précision de solution et une vitesse de convergence élevée et convient à la résolution de problèmes d'optimisation de grandes dimensions. Pour confirmer la supériorité de notre algorithme, nous l'avons appliqué pour la sélection des attributs et l'optimisation des paramètres de

SVM simultanément. Deux architectures système ont été implémentées pour l'approche proposée. L'approche développée est évaluée et comparée avec quatre algorithmes métaheuristiques bien considérés (MVO, GWO, BAT et WOA) et à la recherche de grille, l'expérience montre que MVGWO a été en mesure d'optimiser SVM en obtenant la plus grande précision par rapport aux autres optimiseurs basés sur les deux architectures étudiées. Pour les travaux futurs, nous essaierons d'appliquer MVGWO à différents problèmes pratiques tels que le traitement d'images.

7.1 Un réseau de neurones récurrent optimisé par l'hybridation entre l'optimiseur de loup gris et l'optimiseur de multi-vers (MVGWO) pour la sécurité IoT

Les réseaux de neurones sont formés avec des algorithmes d'apprentissage à rétropropagation, qui sont généralement lents et nécessitent donc des taux d'apprentissage et une dynamique plus élevés pour atteindre une convergence plus rapide. Ces approches ne sont performantes que si l'apprentissage incrémental est nécessaire. Cependant, ils sont encore trop lents pour les applications «réelles». Néanmoins, le modèle levengerg-Marquardt est toujours utilisé pour les réseaux de petite et moyenne taille (RASHID, ABBAS et TUREL, 2019). Le manque de mémoire disponible est ce qui empêche l'utilisation d'algorithmes plus rapides. La rétropropagation est un algorithme déterministe qui s'attaque aux problèmes linéaires et non linéaires. Un autre problème associé aux algorithmes de rétropropagation est la sélection d'un taux d'apprentissage approprié, qui est une question compliquée. Pour un réseau linéaire, un taux d'apprentissage trop rapide entraînerait un apprentissage instable ; à l'inverse, un taux d'apprentissage trop lent entraîne un temps d'apprentissage excessivement long (RASHID, ABBAS et TUREL, 2019).
Le problème est plus complexe pour les réseaux multicouches non linéaires, car il est difficile de trouver une méthode simple pour sélectionner un taux d'apprentissage. La surface d'erreur des réseaux non linéaires est également plus difficile que celle des réseaux linéaires (RASHID, ABBAS et TUREL, 2019 ; HAGAN et al., 1996).

D'autre part, l'utilisation de réseaux de neurones avec des fonctions de transfert non linéaires présenterait plusieurs solutions minimales locales dans la surface d'erreur. Il est donc possible qu'une solution dans un réseau soit "coincée" dans une solution locale. Cela peut se produire en fonction des conditions initiales de départ. Il est à noter qu'avoir une solution dans les minima locaux pourrait être une solution satisfaisante si la solution est proche du minimum global. Sinon, la solution est incorrecte (RASHID, ABBAS et TUREL, 2019). En outre, l'algorithme d'apprentissage par rétropropagation ne produit pas des connexions de poids parfaites pour la solution optimale. Dans ce cas, le réseau doit être réinitialisé à plusieurs reprises pour garantir l'obtention de la meilleure solution (RASHID, ABBAS et TUREL, 2019 ; SIKDER, UDDIN et HALDER, 2016 ; BARADWAJ et PAL, 2012).
En revanche, il existe des algorithmes inspirés de la nature, qui sont dérivés du comportement naturel des animaux. Ces algorithmes sont stochastiques. L'élément essentiel qui est importé dans ces algorithmes est le caractère aléatoire. Cela signifie que les algorithmes utilisent des solutions initiales aléatoires qui sont ensuite améliorées grâce à une séquence d'itérations qui évitent les optima locaux élevés (RASHID, ABBAS et TUREL, 2019).
De plus, un réseau de neurones multicouches est subtil lorsqu'il s'agit de décider de la sélection des neurones cachés. Il existe un problème de sous-ajustement qui peut survenir lorsqu'un petit nombre de neurones cachés sont utilisés ; de plus, un sur-ajustement peut survenir lorsque trop de neurones cachés sont utilisés. Une alternative à un réseau de neurones multicouches est le réseau de neurones récurrent (RNN). Un RNN utilise moins

de neurones cachés car il a une couche de contexte pour préserver les réseaux de neurones cachés précédents. Par conséquent, le réseau est plus stable et peut gérer avec succès des modèles temporels (RASHID, ABBAS et TUREL, 2019; RASHID, 2009). Ce travail présente, une optimisation d'un réseau de neurones récurrent (RNN) par notre algorithme hybride MVGWO basé sur l'optimiseur de loup gris (GWO) et l'optimisation multi-verse (MVO) (déjà détaillait dans les sections précédentes).
La sécurité et les menaces augmentent considérablement en raison de l'utilisation accrue des applications de l'internet des objets dans tous les domaines. En raison de la nature déséquilibrée des données de sécurité IoT, la conception de la détection d'anomalies basée sur un modèle dans le réseau IoT représente un défi pour les modèles d'apprentissage automatique, car la plupart de ces modèles supposent un nombre égal d'échantillons pour chaque classe.

Environ 2,79% des profils de réseau IoT sont des types d'anomalies qui imposent un déséquilibre sévère, là où il y a trois échantillons dans les types d'anomalies pour des centaines d'échantillons dans la classe normale majoritaire. Il en résulte des performances prédictives médiocres pour l'identification du type d'anomalie, ce qui est essentiellement un problème car le type d'anomalie est plus sensible que le type d'activité normal. Ce travail propose un modèle basé sur les réseaux de neurones récurrents optimisé par l'algorithme MVGWO nommé (M-RNNMVGWO) en utilisant la technique de sur-échantillonnage des minorités synthétiques (SMOTE) pour la prédiction des anomalies dans le réseau IoT. Les approches proposées sont simulées avec des données DS2OS et les performances sont comparées à d'autres approches d'apprentissage automatique. Les paramètres d'évaluation tels que la sensibilité, la précision, la spécificité, et la F-mesure sont utilisés pour confirmer la supériorité de notre approche proposée.

7.1.1 Réseau de neurones récurrents (RNN)

Le perceptron multicouche transmet les données des couches inférieures aux couches supérieures, tandis que les réseaux de neurones récurrents (RNN) sont considérés comme des réseaux de neurones à flux de données bidirectionnels (RASHID, ABBAS et TUREL, 2019). Le flux de données se propage des phases de traitement précédentes vers les phases antérieures. Dans ce travail de recherche, on utilise le concept de réseau de neurones récurrent simple, qui a été proposé pour la première fois par Jeff Elman (RASHID, ABBAS et TUREL, 2019; ELMAN, 1990).
Le modèle de la figure 7.20 utilise un réseau à trois couches. Au niveau de la couche cachée, la sortie de chaque neurone caché au moment $(t-1)$ est sauvegardée dans les neurones de contexte puis, au moment (t), est alimentée conjointement avec l'entrée initiale de la couche cachée (RASHID, ABBAS et TUREL, 2019). Ainsi, des copies des valeurs précédentes des neurones cachés sont continuellement conservées dans les neurones de contexte, en raison de la propagation à travers les connexions récurrentes depuis le temps $(t-1)$, avant qu'une règle de mise à jour des paramètres soit appliquée au temps (t). Par conséquent, le modèle de réseau conserve et acquiert un ensemble d'états résumant les entrées précédentes (RASHID, ABBAS et TUREL, 2019).
Dans cette approche proposée, un modèle RNN est développé en utilisant MVGWO pour optimiser les valeurs des biais et des poids du modèle. Dans un premier temps, le modèle de réseau de neurones est formé en utilisant un ensemble de données d'apprentissage, et ses poids et biais sont optimisés en utilisant un réseau récurrent modifié avec MVGWO. Dans la deuxième étape, pour évaluer le modèle formé, le modèle conçu est testé avec un ensemble de données de test.

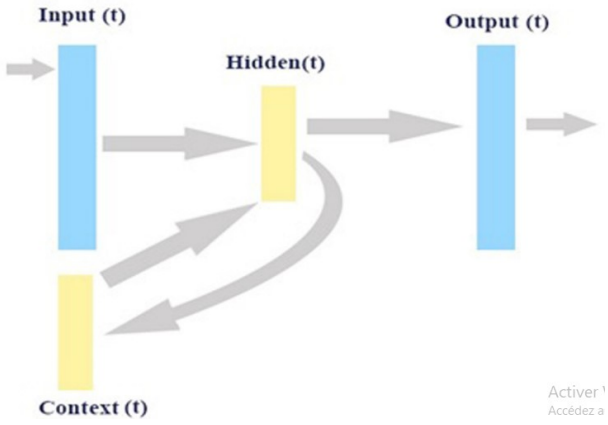

FIGURE 7.20 – Un modèle simple de RNN (RASHID, ABBAS et TUREL, 2019)

7.1.2 Un réseau de neurones récurrent modifié

Le modèle de réseau de neurones développé consiste à utiliser le concept de RNN sur un perceptron multicouche avec deux couches cachées et deux couches de contexte (un contexte pour chaque couche cachée). La structure du modèle est la suivante : 13, 10-10, 10-10, 1 ; 13 neurones dans la couche d'entrée, 10 neurones dans la première couche cachée avec 10 neurones pour la première couche de contexte, 10 neurones dans la deuxième couche cachée avec 10 neurones pour la deuxième couche de contexte, et 1 neurone dans la couche de sortie. Les neurones des premières et deuxièmes couches de contexte sont des copies des neurones de l'instant précédent des premières et deuxièmes couches cachées, respectivement (voir les équations ci-dessous) (RASHID, ABBAS et TUREL, 2019).

$$C_l^1(t) = h_j^1(t-1) \tag{7.16}$$

$C_l^1(t)$ représente le i_{me} neurone dans la première couche de contexte au temps t, ou il est égal à $h_j^1(t-1)$ qui représente le j_{me} neurone dans la première couche cachée à l'instant précédent (RASHID, ABBAS et TUREL, 2019).

$$C_m^2(t) = h_g^2(t-1) \tag{7.17}$$

$C_m^2(t)$ représente le i_{me} neurone dans la deuxième couche de contexte au temps t ou il est égal à $h_g^2(t-1)$ qui représente le j_{me} neurone de la deuxième couche cachée à l'instant précédent.
La propagation avant vers la première couche cachée peut être représentée comme suit :

$$h_j^1(t) = f(\sum_{i}^{\mathbb{I}} v_{ij}^1 x_i(t)) + f(\sum_{l}^{Con^1} u_{lj}^1 C_l^1(t)) \tag{7.18}$$

$$f(net) = \frac{1}{1+e^{-net}} \tag{7.19}$$

Où $f(net)$ représente une fonction d'activation dans laquelle les fonctions Sigmoïde et Softamax sont utilisées à des fins expérimentales dans chaque neurone caché au niveau des couches cachées.
v_{ij}^1 et u_{lj}^1, indiquent les connexions de poids concernant la première couche cachée $h_j^1(t)$ et la couche d'entrée $x_i(t)$, et la première couche cachée $h_j^1(t)$ et la première couche de contexte

$C_l^1(t)$, respectivement.
La propagation avant vers la deuxième couche cachée peut être représentée comme suit :

$$h_g^2(t) = f(\sum_{j}^{\mathbb{H}1} v_{jg}^2 h_i^1(t)) + f(\sum_{m}^{Con^2} u_{mg}^2 C_m^2(t)) \tag{7.20}$$

Où v_{jg}^2 et u_{mg}^2 indiquent les connexions de poids entre la deuxième couche cachée $h_g^2(t)$ et la première couche $h_i^1(t)$, et entre la deuxième couche cachée $h_g^2(t)$ et la deuxième couche de contexte $C_m^2(t)$, respectivement.
La propagation avant vers la couche de sortie peut être écrite comme suit :

$$O_k(t) = f\sum_{g}^{H2} w_{gk} h_g^2(t) \tag{7.21}$$

Où w_{gk} représente le poids de la connexion entre la couche de sortie $O_k(t)$ et la deuxième couche cachée $h_g^2(t)$.
De même, la fonction objective ici pour l'entraînement du modèle est la plus petite erreur quadratique moyenne (MSE) pour obtenir la classification la plus élevée, où MSE représente la variance entre la sortie prédite et la sortie cible. Le MSE est calculé comme suit :

$$MSE_p = \frac{1}{n}\sum_{k=1}^{n}(O_k(t) - d_k(t))^2 \tag{7.22}$$

Où n représente le nombre de neurones de sortie et $d_k(t)$ et O_j^k désignent les sorties souhaitées et réelles du K^{ieme} neurone.
La MSE totale de tous les échantillons peut être exprimée comme suit :

$$Total_{MSE} = \frac{1}{n}\sum_{p=1}^{S} MSE_p \tag{7.23}$$

Où p représente un modèle d'échantillon et S représente le nombre de modèles d'apprentissage. Notez que l'entrée du MVGWO est le MSE et que la sortie est les poids et les biais.

7.1.3 Le M-RNNMVGWO

Dans la phase d'apprentissage, le M-RNNMVGWO a deux parties : le RNN modifié et le MVGWO. Le MVGWO définit initialement ses variables, ses poids et ses biais au RNN modifié sous la forme de vecteur. Le premier échantillon est ensuite envoyé au RNN modifié, puis une copie de la sortie de la première couche cachée au temps t est conservée dans la première couche de contexte. Ensuite, à l'instant $(t+1)$, le réseau est réinjecté dans la première couche cachée.
Simultanément, une copie de la sortie de la deuxième couche cachée au temps (t) est conservée dans la deuxième couche de contexte. Puis, au temps $t+1$, le réseau est réinjecté dans la deuxième couche cachée. Ce modèle de réseau neuronal récurrent préserve et apprend un ensemble d'états résumant les entrées précédentes. Ce processus se poursuit de manière itérative pour alimenter tous les autres échantillons d'apprentissage au RNN modifié en utilisant les mêmes poids et biais initialisés (RASHID, ABBAS et TUREL, 2019). Après avoir calculé Le MSE totale ($Total_{MSE}$) sur les échantillons d'apprentissage, le MVGWO reçoit le MSE totale.
Le MVGWO évalue le MSE totale ($Total_{MSE}$). Ensuite, après la modernisation de la valeur de fitness et de la position de chacun des meilleurs loups, le vecteur des poids et biais, qui désigne les positions des agents de recherche, est ajustée de manière itérative. Une fois que

les pois et les biais sont mis à jour par MVGWO, ils sont transmis au RNN modifié.
En conclusion, les échantillons d'apprentissage et les poids et les biais mis à jour sont utilisés pour entraîner le RNN modifié et pour archiver un nouveau MSE total. La procédure d'apprentissage est constante jusqu'à ce que la condition d'arrêt soit satisfaite. Pour finir, les poids et les biais optimisés sont utilisés pour tester le M-RNNMVGWO en utilisant le jeu de données de test sans utiliser le MVGWO.

7.1.4 Calcul de la complexité des poids

Dans tous les modèles, l'utilisateur peut spécifier les couches cachées, les couches de contextes et les neurones de chaque couche. L'exercice de base consiste à choisir le plus petit nombre possible des paramètres pour trouver le meilleur arrangement possible en fonction des exigences. Cependant, dans la pratique, cela n'est pas facile car il faut faire plusieurs essais en utilisant diverses structures et en évaluant leurs résultats pour déterminer la structure de modèle la mieux adaptée à la tâche. Sur la base de nos essais, une ou deux couches cachées peuvent être suffisantes. L'équation suivante définit le calcul des poids de connexion pour M-RNNMVGWO :

$$d = (i+1) * h1 + (h1+1) * h2 + (h2+1) * o + c1 * h1 + c2 * h2 \tag{7.24}$$

Où d dénote la dimension du problème, $i, h1, h2, c1, c2$, et o représentent les neurones de la couche d'entrée, les neurones de la première couche cachée, les neurones de la deuxième couche cachée, les neurones de la premiere couche de contexte, les neurones de la deuxième couche de contexte et les neurones de la sortie, respectivement. Les couches d'entrées et les couches cachées ont toutes les deux un biais ; un neurone est donc ajouté à chacune d'elles (RASHID, ABBAS et TUREL, 2019).

7.1.5 La prédiction d'anomalies dans le réseau IoT

Un réseau de neurones récurrent modifié avec l'algorithme MVGWO est redéveloppé pour la prédiction des anomalies dans le réseau IoT.
Le travail proposé comporte deux phases : (a) l'obtention d'un corpus équilibré de profils IoT à partir de données originales déséquilibrées (PAHL et AUBET, 2018) en utilisant SMOTE et (b) la conception d'un modèle basé sur notre algorithme proposé M-RNNMVGWO pour la prédiction d'anomalies dans le réseau IoT (DASH et al., 2020). Dans ce travail, nous avons utilisé le jeu de données de sécurité IoT de Kaggle [1]. (PAHL M-O, 2018) pour l'évaluation du modèle. Cet ensemble de données est produit dans un environnement virtuel par le biais d'un système d'orchestration d'espace intelligent distribué (DS2OS) qui est composé d'informations de communication entre divers nœuds IoT dans la couche d'application. Il y a un total de 357 952 échantillons, chacun ayant un nombre de 13 attributs. Cet ensemble de données couvre huit types d'anomalies : "anomalous (dataProbing)" (dP), "anomalous (DoSattack)" (DoS), "anomalous (malitiousControl)" (mC), "anomalous (malitiousOperation)" (MO), "anomalous (scan)" (scan), "anomalous (spying)" (spying), et "anomalous (wrongSetUp)" (wSU).Les détails du pourcentage des distributions des anomalies "anomalous (dataProbing)", "anomalous (DoSattack)", "anomalous (malitiousControl)", "anomalous (malitiousOperation)", "anomalous (scan)", "anomalous (spying)", et "anomalous (wrongSetUp)" ayant les distributions 03. 41% 57,70% 08,87% 08,03% 15,44% 05,31% et 01,21% respectivement (tableau 7.19).

1. https ://www.kaggle.com/francoisxa/ds2ostraffictraces

TABLE 7.19 – L'ensemble de données utilisé pour la prédiction d'anomalies dans le réseau IoT

N ° d'échantillons	N ° d'attributs	N ° de classes
357 952	13	8

Lors de la mise en œuvre d'un algorithme d'apprentissage automatique, la structure des données, en particulier l'équilibre entre les nombres d'observations pour chaque cible potentielle, a une influence significative sur les performances du modèle de prédiction (DASH et al., 2020; LEMAÎTRE, NOGUEIRA et ARIDAS, 2017). En raison de la nature déséquilibrée des données d'anomalies IoT. Nous avons utilisé SMOTE (CHAWLA et al., 2002) afin d'obtenir un équilibre entre les classes sous-représentées et la classe majoritaire (DASH et al., 2020).

7.1.6 Configuration Expérimentale et analyse des résultats

La méthode proposée a été mise en œuvre sur un système doté d'un processeur Intel(R) Core(TM) i5, 1,60 GHz 2,30 GHz, de 4,00 Go de RAM et d'un système d'exploitation 64 bits avec Windows 10.
Diverses mesures de performance telles que la sensibilité, la précision, la spécificité, et la F-mesure, ont été calculés et comparés (tableau 7.20) pour étudier l'efficacité de la méthode proposée.

Résultats et discussion

TABLE 7.20 – Analyse des performances avec SMOTE et sans SMOTE

Algorithmes	Sensibilité	Précision	Spécificité	F-mesure
RF	0.87	0.99	0.25	0.93
NB	0.97	1.00	1.00	0.98
MLP	0.99	1.00	1.00	0.99
RNN	0.98	1.00	1.00	0.99
M-RNNMVGWO	1.00	0.99	0.22	0.99
RF-SMOTE	0.99	1.00	1.00	0.99
NB-SMOTE	0.97	1.00	1.00	0.98
MLP-SMOTE	0.99	0.99	0.23	0.99
RNN-SMOTE	0.98	1.00	1.00	0.99
M-RNNMVGWO-SMOTE	1.00	1.00	1.00	1.00

Les comparaisons de tous ces modèles sur la base de diverses mesures de performance sont présentées dans le tableau 7.20. En observant de près les mesures de performance, on constate que l'approche proposée est meilleure en termes de sensibilité, précision, spécificité, et F-mesure, ce qui signifie l'efficacité de l'identification des activités anormales et la capacité à gérer le déséquilibre des données IoT par rapport aux autres modèles. La comparaison entre RNN et M-RNNMVGWO montre que MVGWO a amélioré le RNN avec une grande précision. on peut conclure que MVGWO a fourni de bons résultats et peut-être considérée comme une alternative aux autres méthodes de formation de réseaux de neurones récurrents (RNN).

7.1.7 Conclusion de l'approche

Le développement d'un système IoT approprié avec une tolérance zéro pour les erreurs de classification a toujours été un défi pour les chercheurs de tous niveaux. L'apprentissage automatique est l'une des meilleures approches pour résoudre un tel problème dans l'IoT. Ce travail propose une méthode basée sur les réseaux de neurones récurrents optimisés par l'algorithme MVGWO (M-RNNMVGWO) en utilisant la technique SMOTE pour la prédiction des anomalies dans les données du réseau IoT (DS2OS) (DASH et al., 2020). Les résultats de la simulation montrent que l'approche proposée a réussi à gérer la nature déséquilibrée des données et s'est avérée efficace pour identifier les types d'anomalies et l'activité normale. Cette méthode proposée s'avère supérieure aux autres méthodes en termes de diverses mesures d'évaluation, à savoir la sensibilité, la précision, la spécificité et la F-mesure. Dans les scénarios IoT du monde réel, la plupart des données sont dans un format non structuré et la conception d'une méthode basée sur l'apprentissage automatique pour de telles données est toujours un défi de recherche.

7.2 Conclusion

Dans ce chapitre, des nouveaux algorithmes d'optimisation, nommés BAT-SDE, PLMVO et MVGWO ont été présentés. Le premier algorithme est basé sur l'optimisation des chauves-souris avec une évolution différentielle auto-adaptative pour l'optimisation globale et pour former les réseaux de neurones à propagation avant (FFNN). À partir de l'analyse et de l'expérimentation fait, nous constatons que BAT-SDE surpasse les autres algorithmes pour tous les problèmes testés. Le deuxième algorithme est basé sur l'optimisation de l'essaim de particules, l'optimisation multi-verse basée sur le vol de Lévy pour l'optimisation globale et pour optimiser simultanément la structure, les poids de connexion et les biais du réseau de neurones à propagation avant (FFNN). L'algorithme fusionne la meilleure force de PSO en exploitation et de LMVO en exploration. Les résultats montrent la supériorité de l'algorithme PLMVO avec une haute exactitude, et une convergence rapide. De plus, la faible valeur de l'écart type a prouvé que notre algorithme est robuste et stable. Le troisième algorithme est basé sur l'optimiseur loup gris avec l'optimiseur multi-vers nommé MVGWO. Cet algorithme est proposé pour résoudre des problèmes d'optimisation à grande dimension. L'algorithme MVGWO combine la bonne capacité d'exploitation de GWO avec la forte capacité d'exploration de l'algorithme MVO. MVGWO est utilisé pour l'optimisation globale, la sélection d'attributs et l'optimisation les paramètres des SVM, et pour optimiser les réseaux de neurones récurrents. Les résultats montrent que MVGWO présente une précision de solution et une vitesse de convergence élevée et convient à la résolution de problèmes d'optimisation de grandes dimensions, et que les performances de GWO et MVO ont été améliorées de manière significative.

Chapitre 8

Conclusion générale

Les travaux de recherche présentés dans cette thèse concernent le développement de nouvelles méthodes d'optimisation globale qui s'appuient sur l'hybridation des algorithmes métaheuristiques et le datamining. Nous avons focalisé nos recherches sur l'hybridation entre les algorithmes évolutionnaires et les algorithmes de l'intelligence des essaims destinés à résoudre des problèmes d'optimisation globale et à résoudre les problèmes liés à l'extraction des connaissances. Dans cette partie, nous allons présenter les contributions issues de ces travaux. Ainsi, nous discuterons des perspectives d'amélioration qui pourrait faire suite à ce travail.

Dans notre première contribution, l'algorithme de chauve-souris a été hybridé avec un algorithme d'évolution différentielle auto-adaptatif (SDE) nommé (BAT-SDE) où la solution est modifiée en utilisant la meilleure stratégie de DE. Cette hybridation a pour but d'améliorer la convergence prématurée de l'algorithme de chauve-souris pour les problèmes de grandes dimensions. Notant que l'algorithme d'évolution différentielle auto-adaptative a été amélioré, le mécanisme permettant d'ajuster les paramètres de contrôle à chaque itération a été modifié. L'algorithme proposé est comparé à l'évolution différentielle et à l'algorithme des chauves-souris pour résoudre un ensemble de cinq fonctions de tests afin de trouver la solution globale.
Une nouvelle approche de formation basée sur BAT-SDE pour former le réseau de neurones à propagation avant (FFNN) a été proposée. La méthode de formation a pris en compte les capacités de BAT-SDE en termes d'exploration et d'exploitation élevées pour localiser les valeurs optimales pour les poids et les biais de FFNN. L'approche a été comparée et évaluée à l'aide de sept ensembles de données biomédicales standard et d'un grand ensemble de données de détection de fraude. Cet ensemble de données est très déséquilibré ; la méthode SMOTE + ENN a été utilisée pour résoudre ce problème en tant que technique de prétraitement utilisée pour traiter des ensembles de données déséquilibrées. L'ensemble de données étant très bruyant ; l'élimination récursive des caractéristiques avec validation croisée est utilisée comme méthode de sélection d'attributs.
La comparaison entre l'algorithme proposé et huit algorithmes utilisés pour former les réseaux de neurones à propagation avant dans la littérature montre la supériorité de l'algorithme proposé avec une grande exactitude et une petite erreur quadratique moyenne (MSE) dans la plupart des ensembles de données par rapport aux autres algorithmes de formation. De plus, la faible valeur de l'écart type montre que l'algorithme proposé est robuste et stable. Les résultats obtenus par BAT-SDE sur la base de l'exactitude moyenne et du MSE prouvent que cet algorithme peut trouver le meilleur ensemble de poids et de biais et empêcher une convergence prématurée vers des optima locaux. Les résultats ont été confirmés en utilisant le test de Friedman, le BAT-SDE a le rang le plus élevé parmi toutes les autres méthodes de formation. Enfin, sur la base de ces expériences, on peut conclure que le BAT-SDE a fourni de bons résultats et peut-être considérée comme une alternative à d'autres méthodes de formation pour les petits et grands ensembles de données.

Dans notre deuxième contribution, nous avons proposé une hybridation séquentielle de l'optimisation de l'essaim de particules (PSO) et de l'optimisation multi-verse basée sur le vol de Lévy (LMVO). L'algorithme fusionne la meilleure force de PSO en exploitation et de LMVO en exploration. L'algorithme proposé est utilisé pour optimiser simultanément la structure, les poids de connexion et les biais du réseau de neurones à propagation avant (FFNN). De nombreux chercheurs ont étudié le problème de la recherche de valeurs optimales pour les poids de connexion et les biais. Cependant, la littérature contient peu de recherches sur l'optimisation de la structure et les poids / biais simultanément. Dans notre étude, nous avons utilisé un schéma de codage hybride pour représenter les solutions de PLMVO pour la formation de MLP qui comprend le nombre de nœuds cachés, les poids de connexion et les biais. La méthode de formation a pris en compte les capacités du PLMVO en termes d'exploration et d'exploitation élevées pour localiser les valeurs optimales pour la structure, les poids et les biais de FFNN. L'approche est proposée afin de minimiser l'erreur de formation et d'augmenter l'exactitude de la classification. L'algorithme est comparé et évalué à l'aide de quinze fonctions de tests, neuf ensembles de données biomédicales. Les résultats indiquent que le schéma de codage hybride utilisé pour optimiser le nombre de nœuds cachés, les poids de connexion et les biais facilite le processus d'optimisation des métaheuristiques. L'approche a permis aux algorithmes d'obtenir une très grande exactitude de classification sur tous les ensembles de données. La comparaison entre l'algorithme proposé et les autres algorithmes montre la supériorité de l'algorithme PLMVO avec une haute exactitude, une petit MSE et une convergence rapide dans la plupart des ensembles de données par rapport aux autres algorithmes de formation.
De plus, la faible valeur de l'écart type a prouvé que le PLMVO peut atteindre les mêmes résultats lors de différentes exécutions, ce qui confirme que notre entraineur est robuste et stable. Pour confirmer les résultats, nous avons utilisé le PLMVO-MLP pour détecter les logiciels malveillants de Linux. Le PLMVO-MLP a été comparé à deux approches utilisées pour détecter les logiciels malveillants dans la littérature. Sur la base des résultats, il peut noter que le modèle dépasse les travaux précédents avec une très grande exactitude, une meilleure F-mesure, une meilleure précision et un meilleur rappel. Enfin, à partir de l'expérience, nous pouvons conclure que PLMVO peut donner de bons résultats et peut être une alternative aux autres méthodes de formation.
Dans la troisième contribution, un algorithme hybride basé sur l'optimiseur loup gris avec l'optimiseur multi-vers nommé MVGWO est proposé pour résoudre des problèmes d'optimisation à grande dimension. Une version améliorée de GWO basée sur un poids d'inertie adaptatif est proposée pour améliorer la capacité de recherche globale de cet algorithme et pour maintenir la diversité des solutions. Afin d'équilibrer l'exploitation et l'exploration, un coefficient d'équilibre adaptatif est utilisé dans cet algorithme. La probabilité d'exploitation ou d'exploration est contrôlée par le coefficient d'équilibre. En changeant le coefficient d'équilibre, l'algorithme MVGWO peut avoir une vitesse de convergence rapide et peut éviter la valeur optimale locale autant que possible.
Pour vérifier la supériorité des performances de l'algorithme MVGWO, vingt-deux fonctions de tests de différents types et dimensions sont utilisées pour tester le MVGWO proposé et les résultats démontrent clairement que les performances du MVGWO proposé sont supérieures aux algorithmes GWO et MVO. Il est évident que MVGWO présente une précision de solution et une vitesse de convergence élevée et convient à la résolution de problèmes d'optimisation de grandes dimensions. Pour confirmer la supériorité de notre algorithme, nous l'avons appliqué pour la sélection d'attributs et l'optimisation des paramètres de SVM simultanément. L'approche développée est évaluée et comparée avec quatre algorithmes métaheuristiques bien considérés (MVO, GWO, BAT et WOA) et à la recherche de grille, l'expérience montre que MVGWO a été en mesure d'optimiser SVM en obtenant la plus grande précision par rapport aux autres algorithmes.

Dans notre quatrième contribution, nous avons proposé une méthode basée sur les réseaux de neurones récurrents optimisés par l'algorithme MVGWO (M-RNNMVGWO) en utilisant la technique SMOTE pour la prédiction des anomalies dans les données du réseau IoT. Les résultats de la simulation montrent que l'approche proposée a réussi à gérer la nature déséquilibrée des données et s'est avérée efficace pour identifier les types d'anomalies et l'activité normale. Cette méthode proposée s'avère supérieure aux autres méthodes en termes de diverses mesures d'évaluation, à savoir la sensibilité, la précision, la spécificité et la F-mesure.

Perspectives

Certes les méthodes présentées dans cette thèse ont donné des résultats pertinents mais la recherche continue !
Dans les travaux futurs, nous nous concentrons sur la façon d'étendre ces approches pour explorer et résoudre des problèmes d'optimisation complexes comme les problèmes d'optimisation du big data et l'application de ces algorithmes aux données IoT. De plus, améliorez les algorithmes proposés pour résoudre des problèmes d'optimisation multiobjectifs.
Nous proposons ainsi d'hybrider d'autres algorithmes métaheuristiques pour résoudre des problèmes réels liés à l'extraction de connaissances.

Bibliographie

ABBATTISTA, Fabio, Nicola ABBATTISTA et Laura CAPONETTI (1995). « An evolutionary and cooperative agents model for optimization ». In : *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*. T. 2. IEEE, p. 668-671.

AGGARWAL, Charu C et al. (2018). « Neural networks and deep learning ». In : *Springer* 10, p. 978-3.

AGHDAM, Mehdi Hosseinzadeh, Nasser GHASEM-AGHAEE et Mohammad Ehsan BASIRI (2009). « Text feature selection using ant colony optimization ». In : *Expert systems with applications* 36.3, p. 6843-6853.

AGRAWAL, Rakesh, Ramakrishnan SRIKANT et al. (1994). « Fast algorithms for mining association rules ». In : *Proc. 20th int. conf. very large data bases, VLDB*. T. 1215. Citeseer, p. 487-499.

AL-ANI, Ahmed, Akram ALSUKKER et Rami N KHUSHABA (2013). « Feature subset selection using differential evolution and a wheel based search strategy ». In : *Swarm and Evolutionary Computation* 9, p. 15-26.

ALDENDERFER, Mark S et Roger K BLASHFIELD (1984). *Cluster analysis. Newberry Park*.

ALJARAH, Ibrahim, Hossam FARIS et Seyedali MIRJALILI (2018a). « Optimizing connection weights in neural networks using the whale optimization algorithm ». In : *Soft Computing* 22.1, p. 1-15.

— (2018b). « Optimizing connection weights in neural networks using the whale optimization algorithm ». In : *Soft Computing* 22.1, p. 1-15.

ALJARAH, Ibrahim et al. (2018). « Training radial basis function networks using biogeography-based optimizer ». In : *Neural Computing and Applications* 29.7, p. 529-553.

AMINE, Abdelmalek et al. (2011). « A hybrid approach based on self-organizing neural networks and the k-nearest neighbors method to study molecular similarity ». In : *International Journal of Chemoinformatics and Chemical Engineering (IJCCE)* 1.1, p. 75-95.

AREL, Itamar, Derek C ROSE et Thomas P KARNOWSKI (2010). « Deep machine learning-a new frontier in artificial intelligence research [research frontier] ». In : *IEEE computational intelligence magazine* 5.4, p. 13-18.

ASMITHA, KA et P VINOD (2014). « A machine learning approach for linux malware detection ». In : *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*. IEEE, p. 825-830.

AZIMI, Zahra Naji (2005). « Hybrid heuristics for examination timetabling problem ». In : *Applied Mathematics and Computation* 163.2, p. 705-733.

BÄCK, Thomas, Günter RUDOLPH et Hans-Paul SCHWEFEL (1993). « Evolutionary programming and evolution strategies : Similarities and differences ». In : *In Proceedings of the Second Annual Conference on Evolutionary Programming*. Citeseer.

BACKER, Eric (1995). *Computer-assisted reasoning in cluster analysis*. Prentice Hall International (UK) Ltd.

BARADWAJ, Brijesh Kumar et Saurabh PAL (2012). « Mining educational data to analyze students' performance ». In : *arXiv preprint arXiv :1201.3417*.

BARRAZA, Juan et al. (2018). « A new hybridization approach between the fireworks algorithm and grey wolf optimizer algorithm ». In : *Journal of Optimization* 2018.

BARTZ-BEIELSTEIN, Thomas (2006). *Experimental Research in Evolutionary Computation : The New Experimentalism (Natural Computing Series)*. Springer.

BENGIO, Yoshua (2009). *Learning deep architectures for AI*. Now Publishers Inc.

BENUWA, Ben Bright et al. (2016). « A review of deep machine learning ». In : *International Journal of Engineering Research in Africa* 24, p. 124-136.

BEYER, Hans-Georg (2001). *The theory of evolution strategies*. Springer Science et Business Media.

BLUM, Christian et al. (2011). « Hybrid metaheuristics in combinatorial optimization : A survey ». In : *Applied soft computing* 11.6, p. 4135-4151.

BONABEAU, Eric, Marco DORIGO et Guy THERAULAZ (1999). *Swarm intelligence from natural to artificial isystems*. Oxford University Press.

BORGELT, Christian (2003). « Efficient implementations of apriori and eclat ». In : *FIMI'03 : Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations*, p. 90.

BOSTANI, Hamid et Mansour SHEIKHAN (2017). « Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems ». In : *Soft computing* 21.9, p. 2307-2324.

BOUBEZOUL, Abderrahmane et Sébastien PARIS (2012). « Application of global optimization methods to model and feature selection ». In : *Pattern Recognition* 45.10, p. 3676-3686.

BOUGHORBEL, Sabri, Fethi JARRAY et Mohammed EL-ANBARI (2017). « Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric ». In : *PloS one* 12.6, e0177678.

BOUHLEL, Ines et al. (2007). « Screening of antimutagenicity via antioxidant activity in different extracts from the leaves of Acacia salicina from the center of Tunisia ». In : *Environmental Toxicology and Pharmacology* 23.1, p. 56-63.

BOUZERDOUM, Moufida, Adel MELLIT et A Massi PAVAN (2013). « A hybrid model (SARIMA–SVM) for short-term power forecasting of a small-scale grid-connected photovoltaic plant ». In : *Solar Energy* 98, p. 226-235.

BREST, Janez et al. (2006). « Self-adapting control parameters in differential evolution : A comparative study on numerical benchmark problems ». In : *IEEE transactions on evolutionary computation* 10.6, p. 646-657.

BUCHNER, Alex G et al. (1999). « An internet-enabled knowledge discovery process ». In : *9th International Database Conference on Heterogeneous and Internet Databases*. IDC.

CARRIZOSA, Emilio et Dolores Romero MORALES (2013). « Supervised classification and mathematical optimization ». In : *Computers & Operations Research* 40.1, p. 150-165.

CHAO, Chih-Feng et Ming-Huwi HORNG (2015). « The construction of support vector machine classifier using the firefly algorithm ». In : *Computational intelligence and neuroscience* 2015.

CHATTERJEE, Snehamoy et Ashis BHATTACHERJEE (2011). « Genetic algorithms for feature selection of image analysis-based quality monitoring model : An application to an iron mine ». In : *Engineering applications of artificial intelligence* 24.5, p. 786-795.

CHAWLA, Nitesh V et al. (2002). « SMOTE : synthetic minority over-sampling technique ». In : *Journal of artificial intelligence research* 16, p. 321-357.

CHEN, Bolun, Ling CHEN et Yixin CHEN (2013). « Efficient ant colony optimization for image feature selection ». In : *Signal processing* 93.6, p. 1566-1576.

CHEN, Huadong et al. (2007). « A hybrid of artificial fish swarm algorithm and particle swarm optimization for feedforward neural network training ». In : *International Conference on Intelligent Systems and Knowledge Engineering 2007*. Atlantis Press.

CHO, Ming-Yuan et Thi Thom HOANG (2017). « Feature selection and parameters optimization of SVM using particle swarm optimization for fault classification in power distribution systems ». In : *Computational intelligence and neuroscience* 2017.

CHUANG, Li-Yeh, Sheng-Wei TSAI et Cheng-Hong YANG (2011). « Improved binary particle swarm optimization using catfish effect for feature selection ». In : *Expert Systems with Applications* 38.10, p. 12699-12707.

CHUANG, Li-Yeh et al. (2008). « Improved binary PSO for feature selection using gene expression data ». In : *Computational Biology and Chemistry* 32.1, p. 29-38.

CIOS, Krzysztof J et al. (2007). « The knowledge discovery process ». In : *Data Mining*. Springer, p. 9-24.

COELLO, Carlos (jan. 2005). « An Introduction to Evolutionary Algorithms and Their Applications ». In : p. 425-442. ISBN : 978-3-540-28063-7. DOI : 10.1007/11533962_39.

CORNE, David, Clarisse DHAENENS et Laetitia JOURDAN (2012). « Synergies between operations research and data mining : The emerging use of multi-objective approaches ». In : *European Journal of Operational Research* 221.3, p. 469-479.

CORTES, Corinna et Vladimir VAPNIK (1995). « Support-vector networks ». In : *Machine learning* 20.3, p. 273-297.

COŞKUN, Musab et al. (2017). « An overview of popular deep learning methods ». In : *European Journal of Technique* 7.2, p. 165-176.

COZZI, Emanuele et al. (2018). « Understanding linux malware ». In : *2018 IEEE symposium on security and privacy (SP)*. IEEE, p. 161-175.

DA SILVA, Ivan Nunes et al. (2017). « Artificial neural network architectures and training processes ». In : *Artificial neural networks*. Springer, p. 21-28.

DAS, Nibaran et al. (2012). « A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application ». In : *Applied Soft Computing* 12.5, p. 1592-1606.

DAS, Swagatam et Ponnuthurai Nagaratnam SUGANTHAN (2010). « Differential evolution : A survey of the state-of-the-art ». In : *IEEE transactions on evolutionary computation* 15.1, p. 4-31.

DASH, Pandit Byomakesha et al. (2020). « Model based IoT security framework using multiclass adaptive boosting with SMOTE ». In : *Security and Privacy* 3.5, e112.

DENG, Li et Dong YU (2014). « Deep learning : methods and applications ». In : *Foundations and trends in signal processing* 7.3–4, p. 197-387.

DHAENENS, Clarisse (2016). *Metaheuristics for big data*. London, UK.

DORIGO, Marco (1992). « Optimization, learning and natural algorithms ». In : *Ph. D. Thesis, Politecnico di Milano*.

EL DOR, Abbas (2012). « Perfectionnement des algorithmes d'optimisation par essaim particulaire : applications en segmentation d'images et en électronique ». Thèse de doct. Université Paris-Est.

EL GHAZALI, Talbi (2009). *Metaheuristics : From Design to Implementation*. Wiley Sons, Incorporated.

ELBENANI, Bouazza, Jacques A FERLAND et Jonathan BELLEMARE (2012). « Genetic algorithm and large neighbourhood search to solve the cell formation problem ». In : *Expert Systems with Applications* 39.3, p. 2408-2414.

ELMAN, Jeffrey L (1990). « Finding structure in time ». In : *Cognitive science* 14.2, p. 179-211.

EMARY, Eid, Hossam M ZAWBAA et Aboul Ella HASSANIEN (2016). « Binary grey wolf optimization approaches for feature selection ». In : *Neurocomputing* 172, p. 371-381.

EMARY, Eid et al. (2015). « Feature subset selection approach by gray-wolf optimization ». In : *Afro-European conference for industrial advancement*. Springer, p. 1-13.

FALTINGS, Boi et Michael SCHUMACHER (2009). *L'intelligence artificielle par la pratique*. PPUR presses polytechniques.

FAN, Qian et al. (2020). « A new improved whale optimization algorithm with joint search mechanisms for high-dimensional global optimization problems ». In : *Engineering with Computers*, p. 1-28.

FARIS, Hossam, Ibrahim ALJARAH et Seyedali MIRJALILI (2016a). « Training feedforward neural networks using multi-verse optimizer for binary classification problems ». In : *Applied Intelligence* 45.2, p. 322-332.

— (2016b). « Training feedforward neural networks using multi-verse optimizer for binary classification problems ». In : *Applied Intelligence* 45.2, p. 322-332.

— (2017). « Evolving radial basis function networks using moth–flame optimizer ». In : *Handbook of neural computation*. Elsevier, p. 537-550.

— (2018). « Improved monarch butterfly optimization for unconstrained global search and neural network training ». In : *Applied Intelligence* 48.2, p. 445-464.

FARIS, Hossam, Seyedali MIRJALILI et Ibrahim ALJARAH (2019a). « Automatic selection of hidden neurons and weights in neural networks using grey wolf optimizer based on a hybrid encoding scheme ». In : *International Journal of Machine Learning and Cybernetics* 10.10, p. 2901-2920.

— (2019b). « Automatic selection of hidden neurons and weights in neural networks using grey wolf optimizer based on a hybrid encoding scheme ». In : *International Journal of Machine Learning and Cybernetics* 10.10, p. 2901-2920.

FARIS, Hossam et al. (2018). « A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture ». In : *Neural Computing and Applications* 30.8, p. 2355-2369.

FAYYAD, Usama, Gregory PIATETSKY-SHAPIRO et Padhraic SMYTH (1996). « From data mining to knowledge discovery in databases ». In : *AI magazine* 17.3, p. 37-37.

FAYYAD, Usama M et al. (1996). « Advances in knowledge discovery and data mining ». In : American Association for Artificial Intelligence.

FEDOROVICI, Lucian-Ovidiu et al. (2012). « Embedding gravitational search algorithms in convolutional neural networks for OCR applications ». In : *2012 7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, p. 125-130.

FISTER JR, Iztok, Dušan FISTER et Xin-She YANG (2013). « A hybrid bat algorithm ». In : *arXiv preprint arXiv :1303.6310*.

FOGEL, Lawrence J, Alvin J OWENS et Michael J WALSH (1966). « Artificial intelligence through simulated evolution ». In :

FOUILHOUX, Pierre (2015). « Optimisation combinatoire : Programmation linéaire et algorithmes ». In : *Université Pierre et Marie Curie*.

FREITAS, Alex A (2002a). *Data mining and knowledge discovery with evolutionary algorithms*. Springer Science et Business Media.

— (2002b). *Data mining and knowledge discovery with evolutionary algorithms*. Springer Science et Business Media.

FREITAS, Alex A. (2011). *Data mining and knowledge discovery with evolutionary algorithms*. Springer.

FRÉDÉERIC, Héeliodore et al. (2017). *Metaheuristics for intelligent electrical networks*. ISTE, Ltd.

GAMBARDELLA, Luca Maria, Roberto MONTEMANNI et Dennis WEYLAND (2012). « Coupling ant colony systems with strong local searches ». In : *European Journal of Operational Research* 220.3, p. 831-843.

GAREY, Michael R (1979). « A Guide to the Theory of NP-Completeness ». In : *Computers and intractability*.

GHAEMI, Manizheh et Mohammad-Reza FEIZI-DERAKHSHI (2016). « Feature selection using forest optimization algorithm ». In : *Pattern Recognition* 60, p. 121-129.

GOODFELLOW, Ian et al. (2016). *Deep learning*. T. 1. 2. MIT press Cambridge.

GRINSTEIN, Usama M Fayyad Georges G et Andreas WIERSE (2002). *Information visualization in data mining and knowledge discovery*. Morgan Kaufmann.

GU, Shenkai, Ran CHENG et Yaochu JIN (2018). « Feature selection for high-dimensional classification using a competitive swarm optimizer ». In : *Soft Computing* 22.3, p. 811-822.

GUENDOUZ, Mohamed, Abdelmalek AMINE et Reda Mohamed HAMOU (2017). « A discrete modified fireworks algorithm for community detection in complex networks ». In : *Applied Intelligence* 46.2, p. 373-385.

GUPTA, Tarun Kumar et Khalid RAZA (2019). « Optimization of ANN architecture : a review on nature-inspired techniques ». In : *Machine learning in bio-signal analysis and diagnostic imaging*, p. 159-182.

GUYON, Isabelle et al. (2002). « Gene selection for cancer classification using support vector machines ». In : *Machine learning* 46.1, p. 389-422.

HACHIMI, Hanaa (2013). « Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications ». Thèse de doct. INSA de Rouen.

HAFEZ, Ahmed Ibrahem et al. (2016). « Sine cosine optimization algorithm for feature selection ». In : *2016 international symposium on innovations in intelligent systems and applications (INISTA)*. IEEE, p. 1-5.

HAGAN, MT et al. (1996). *Neural network design vol. 20 : Pws Pub*.

HAMM, Lonnie, B Wade BRORSEN et Martin T HAGAN (2002). « Global optimization of neural network weights ». In : *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*. T. 2. IEEE, p. 1228-1233.

HAMOU, Reda Mohamed, Abdelmalek AMINE et Amine BOUDIA (2013). « A new metaheuristic based on social bees for detection and filtering of spam ». In : *International Journal of Applied Metaheuristic Computing (IJAMC)* 4.3, p. 15-33.

HAND, David J et Niall M ADAMS (2014). « Data mining ». In : *Wiley StatsRef : Statistics Reference Online*, p. 1-7.

HASSANIN, Mohamed F, Abdullah M SHOEB et Aboul Ella HASSANIEN (2016). « Grey wolf optimizer-based back-propagation neural network algorithm ». In : *2016 12th International Computer Engineering Conference (ICENCO)*. IEEE, p. 213-218.

HE, Haibo et Edwardo A GARCIA (2009). « Learning from imbalanced data ». In : *IEEE Transactions on knowledge and data engineering* 21.9, p. 1263-1284.

HE, Kaiming et al. (2016). « Deep residual learning for image recognition ». In : *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 770-778.

HEATON, Jeff (2015). « Artificial Intelligence for Humans, Volume 3 : Neural Networks and Deep Learning, 1.0 ». In : *Chesterfield, USA : Heaton Research Inc*.

HODGKIN, Alan L et Andrew F HUXLEY (1952). « A quantitative description of membrane current and its application to conduction and excitation in nerve ». In : *The Journal of physiology* 117.4, p. 500-544.

HOLLAND, John H (1962a). « Concerning efficient adaptive systems ». In : *Self-Organizing Systems* 230.

— (1962b). « Outline for a logical theory of adaptive systems ». In : *Journal of the ACM (JACM)* 9.3, p. 297-314.

— (1992). « Genetic algorithms ». In : *Scientific american* 267.1, p. 66-73.

HUANG, Cheng-Lung (2009). « ACO-based hybrid classification system with feature subset selection and model parameters optimization ». In : *Neurocomputing* 73.1-3, p. 438-448.

HUANG, Cheng-Lung et Chieh-Jen WANG (2006a). « A GA-based feature selection and parameters optimization for support vector machines ». In : *Expert Systems with applications* 31.2, p. 231-240.

— (2006b). « A GA-based feature selection and parameters optimizationfor support vector machines ». In : *Expert Systems with applications* 31.2, p. 231-240.

HUANG, Hu et al. (2012). « Ant colony optimization-based feature selection method for surface electromyography signals classification ». In : *Computers in biology and medicine* 42.1, p. 30-38.

IBRAHIM, Rehab Ali et al. (2019). « Improved salp swarm algorithm based on particle swarm optimization for feature selection ». In : *Journal of Ambient Intelligence and Humanized Computing* 10.8, p. 3155-3169.

JACQUIN, Sophie (2015). « Hybridation des métaheuristiques et de la programmation dynamique pour les problémes d'optimisation mono et multi-objectif : application à la production dénergie ». Thèse de doct. Lille 1.

JADDI, Najmeh Sadat, Salwani ABDULLAH et Abdul Razak HAMDAN (2015). « Optimization of neural network model using modified bat-inspired algorithm ». In : *Applied Soft Computing* 37, p. 71-86.

JAYARAMAN, Vijayashree et H Parveen SULTANA (2019). « Artificial gravitational cuckoo search algorithm along with particle bee optimized associative memory neural network for feature selection in heart disease classification ». In : *Journal of Ambient Intelligence and Humanized Computing*, p. 1-10.

JIA, Heming et al. (2019). « Multiverse optimization algorithm based on Lévy flight improvement for multithreshold color image segmentation ». In : *IEEE Access* 7, p. 32805-32844.

JOURDAN, Laetitia (2003). « Métaheuristiques pour l'extraction de connaissances : application à la génomique ». Thèse de doct. Université des Sciences et Technologie de Lille-Lille I.

KABIR, Md Monirul, Md SHAHJAHAN et Kazuyuki MURASE (2011). « A new local search based hybrid genetic algorithm for feature selection ». In : *Neurocomputing* 74.17, p. 2914-2928.

— (2012). « A new hybrid ant colony optimization algorithm for feature selection ». In : *Expert Systems with Applications* 39.3, p. 3747-3763.

KABLI, Fatima, Reda Mohamed HAMOU et Abdelmalek AMINE (2018). « Protein classification using n-gram technique and association rules ». In : *International Journal of Software Innovation (IJSI)* 6.2, p. 77-89.

KARABOGA, Dervis et Bahriye BASTURK (2008). « On the performance of artificial bee colony (ABC) algorithm ». In : *Applied soft computing* 8.1, p. 687-697.

KARASOZEN, Bulent, Alexander RUBINOV, Gerhard-Wilhelm WEBER et al. (2006). « Optimization in data mining ». In : *European Journal of Operational Research* 173.3, p. 701-704.

KARTHIKEYAN, K et PK DHAL (2015). « Transient stability enhancement by optimal location and tuning of STATCOM using PSO ». In : *Procedia Technology* 21, p. 345-351.

KENNEDY, James et Russell EBERHART (1995a). « Particle swarm optimization ». In : *Proceedings of ICNN'95-international conference on neural networks*. T. 4. IEEE, p. 1942-1948.

— (1995b). « Particle swarm optimization ». In : *Proceedings of ICNN'95-international conference on neural networks*. T. 4. IEEE, p. 1942-1948.

KENTER, Tom et al. (2017). « Neural networks for information retrieval ». In : *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, p. 1403-1406.

KHAN, Abdullah et al. (2019). « An alternative approach to neural network training based on hybrid bio meta-heuristic algorithm ». In : *Journal of Ambient Intelligence and Humanized Computing* 10.10, p. 3821-3830.

KIRA, Kenji et Larry A RENDELL (1992). « A practical approach to feature selection ». In : *Machine learning proceedings 1992*. Elsevier, p. 249-256.

KIRANYAZ, Serkan et al. (2009a). « Evolutionary artificial neural networks by multi-dimensional particle swarm optimization ». In : *Neural networks* 22.10, p. 1448-1462.

— (2009b). « Evolutionary artificial neural networks by multi-dimensional particle swarm optimization ». In : *Neural networks* 22.10, p. 1448-1462.

KOHAVI, Ron et George H JOHN (1997). « Wrappers for feature subset selection ». In : *Artificial intelligence* 97.1-2, p. 273-324.

KOHLI, Mehak et Sankalap ARORA (2018). « Chaotic grey wolf optimization algorithm for constrained optimization problems ». In : *Journal of computational design and engineering* 5.4, p. 458-472.

KOMAKI, GM et Vahid KAYVANFAR (2015). « Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time ». In : *Journal of Computational Science* 8, p. 109-120.

KOZA, John R et John R KOZA (1992). *Genetic programming : on the programming of computers by means of natural selection*. T. 1. MIT press.

KRUEGER, Martin (1994). « Méthode d'analyse d'algorithmes d'optimisation stochastiques à l'aide d'algorithmes génétiques ». Thèse de doct.

KUBAT, Miroslav (1999). « Neural networks : a comprehensive foundation by Simon Haykin, Macmillan, 1994, ISBN 0-02-352781-7. » In : *The Knowledge Engineering Review* 13.4, p. 409-412.

LEMAÎTRE, Guillaume, Fernando NOGUEIRA et Christos K ARIDAS (2017). « Imbalanced-learn : A python toolbox to tackle the curse of imbalanced datasets in machine learning ». In : *The Journal of Machine Learning Research* 18.1, p. 559-563.

LI, Jing et al. (2012). « Brief introduction of back propagation (BP) neural network algorithm and its improvement ». In : *Advances in computer science and information engineering*. Springer, p. 553-558.

LI, Shijin et al. (2011). « An effective feature selection method for hyperspectral image classification based on genetic algorithm and support vector machine ». In : *Knowledge-Based Systems* 24.1, p. 40-48.

LIN, Hai et Katsumi YAMASHITA (2002). « Hybrid simplex genetic algorithm for blind equalization using RBF networks ». In : *Mathematics and Computers in Simulation* 59.4, p. 293-304.

LIN, Shih-Wei et al. (2008). « Particle swarm optimization for parameter determination and feature selection of support vector machines ». In : *Expert systems with applications* 35.4, p. 1817-1824.

LIU, Huan et Hiroshi MOTODA (2007). *Computational methods of feature selection*. CRC Press.

MACQUEEN, James et al. (1967). « Some methods for classification and analysis of multivariate observations ». In : *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. T. 1. 14. Oakland, CA, USA, p. 281-297.

MAFARJA, Majdi et Seyedali MIRJALILI (2018). « Whale optimization approaches for wrapper feature selection ». In : *Applied Soft Computing* 62, p. 441-453.

MAFARJA, Majdi M et Seyedali MIRJALILI (2017). « Hybrid whale optimization algorithm with simulated annealing for feature selection ». In : *Neurocomputing* 260, p. 302-312.

MAIMON, Oded et Lior ROKACH (2005). « Data mining and knowledge discovery handbook ». In :

— (2007). *Soft computing for knowledge discovery and data mining*. Springer Science et Business Media.

MANTOVANI, Rafael G et al. (2015). « Effectiveness of random search in SVM hyper-parameter tuning ». In : *2015 International Joint Conference on Neural Networks (IJCNN)*. Ieee, p. 1-8.

MARTIN, R (1990). « Single-interval learning by simile within a simulated hebbian neural network ». In : *Computers & Mathematics with Applications* 20.4-6, p. 217-226.

MARTI, Rafael et Abdellah EL-FALLAHI (2004). « Multilayer neural networks : an experimental evaluation of on-line training methods ». In : *Computers and Operations Research* 31.9, p. 1491-1513.

MCCULLOCH, Warren S et Walter PITTS (1943). « A logical calculus of the ideas immanent in nervous activity ». In : *The bulletin of mathematical biophysics* 5.4, p. 115-133.

MEISEL, Stephan et Dirk MATTFELD (2010). « Synergies of operations research and data mining ». In : *European Journal of Operational Research* 206.1, p. 1-10.

MENDES, Rui et al. (2002a). « Particle swarms for feedforward neural network training ». In : *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*. T. 2. IEEE, p. 1895-1899.

— (2002b). « Particle swarms for feedforward neural network training ». In : *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*. T. 2. IEEE, p. 1895-1899.

MINAR, Matiur Rahman et Jibon NAHER (2018). « Recent advances in deep learning : An overview ». In : *arXiv preprint arXiv :1807.08169*.

MIRJALILI, Seyedali (2015). « How effective is the Grey Wolf optimizer in training multi-layer perceptrons ». In : *Applied Intelligence* 43.1, p. 150-161.

MIRJALILI, Seyedali et Andrew LEWIS (2016). « The whale optimization algorithm ». In : *Advances in engineering software* 95, p. 51-67.

MIRJALILI, Seyedali, Seyed Mohammad MIRJALILI et Abdolreza HATAMLOU (2016). « Multi-verse optimizer : a nature-inspired algorithm for global optimization ». In : *Neural Computing and Applications* 27.2, p. 495-513.

MIRJALILI, Seyedali, Seyed Mohammad MIRJALILI et Andrew LEWIS (2014a). « Grey wolf optimizer ». In : *Advances in engineering software* 69, p. 46-61.

— (2014b). « Let a biogeography-based optimizer train your multi-layer perceptron ». In : *Information Sciences* 269, p. 188-209.

MITRA, Mandar et BB CHAUDHURI (2000). « Information retrieval from documents : A survey ». In : *Information retrieval* 2.2, p. 141-163.

MITTAL, Nitin, Urvinder SINGH et Balwinder Singh SOHI (2016). « Modified grey wolf optimizer for global engineering optimization ». In : *Applied Computational Intelligence and Soft Computing* 2016.

MOHAMMED, Hardi et Tarik RASHID (2020). « A novel hybrid GWO with WOA for global numerical optimization and solving pressure vessel design ». In : *Neural Computing and Applications*, p. 1-18.

MOSLEHI, Fateme et Abdorrahman HAERI (2020). « A novel hybrid wrapper–filter approach based on genetic algorithm, particle swarm optimization for feature subset selection ». In : *Journal of Ambient Intelligence and Humanized Computing* 11.3, p. 1105-1127.

MURPHEY, Yi L et al. (2007). « OAHO : an effective algorithm for multi-class learning from imbalanced data ». In : *2007 International Joint Conference on Neural Networks*. IEEE, p. 406-411.

NANDY, Sudarshan, Partha Pratim SARKAR et Achintya DAS (2012). « Training a feed-forward neural network with artificial bee colony based backpropagation method ». In : *arXiv preprint arXiv :1209.2548*.

NARENDRA, Patrenahalli M. et Keinosuke FUKUNAGA (1977). « A branch and bound algorithm for feature subset selection ». In : *IEEE Computer Architecture Letters* 26.09, p. 917-922.

NAWI, Nazri Mohd, Muhammad Zubair REHMAN et Abdullah KHAN (2014a). « A new bat based back-propagation (BAT-BP) algorithm ». In : *Advances in Systems Science*. Springer, p. 395-404.

— (2014b). « A new bat based back-propagation (BAT-BP) algorithm ». In : *Advances in Systems Science*. Springer, p. 395-404.

NDIAYE, Samba et al. (2014). « Approche de sélection d'attributs pour la classification basée sur l'algorithme RFE-SVM ». In : *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées* 17.

OLAFSSON, Sigurdur (2006). « Introduction to operations research and data mining ». In : *Computers and operations research* 33.11, p. 3067-3069.

OLAFSSON, Sigurdur, Xiaonan LI et Shuning WU (2008). « Operations research and data mining ». In : *European Journal of Operational Research* 187.3, p. 1429-1448.

OLIVEIRA, Adriano LI et al. (2010). « GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation ». In : *information and Software Technology* 52.11, p. 1155-1166.

ÖZÇIFT, Akın et Arif GÜLTEN (2013). « Genetic algorithm wrapped Bayesian network feature selection applied to differential diagnosis of erythemato-squamous diseases ». In : *Digital Signal Processing* 23.1, p. 230-237.

PADAWAN (2018). *platform for multi-architecture ELF analysis*. URL : https://padawan.s3.eurecom.fr/.

PAHL, Marc-Oliver et François-Xavier AUBET (2018). « All eyes on you : Distributed Multi-Dimensional IoT microservice anomaly detection ». In : *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, p. 72-80.

PAHL M-O, Aubet F-X (2018). *DS2OS traffic traces*. URL : https://www.kaggle.com/francoisxa/ds2ostraffictraces.

PALMES, Paulito P, Taichi HAYASAKA et Shiro USUI (2005). « Mutation-based genetic neural network ». In : *IEEE Transactions on Neural Networks* 16.3, p. 587-600.

PAN, Jeng-Shyang, Thi-Kien DAO, Shu-Chuan CHU et al. (2017). « A novel hybrid GWO-FPA algorithm for optimization applications ». In : *International conference on smart vehicular technology, transportation, communication and applications*. Springer, p. 274-281.

PAN, Tien-Szu, Thi-Kien DAO, Shu-Chuan CHU et al. (2015). « A communication strategy for paralleling grey wolf optimizer ». In : *International Conference on Genetic and Evolutionary Computing*. Springer, p. 253-262.

PANDIT, Shivam (2019). *Early-Stage-Malware-Prediction-using-Deep-Learning*. URL : https://github.com/shivam7066/Early-Stage-Malware-Prediction-using-Deep-Learning.

PERDISCI, Roberto, Andrea LANZI et Wenke LEE (2008). « Classification of packed executables for accurate computer virus detection ». In : *Pattern recognition letters* 29.14, p. 1941-1946.

PINTO, Pedro, Thomas A. RUNKLER et João M. SOUSA (2005). « Wasp swarm optimization of logistic systems ». In : *Adaptive and Natural Computing Algorithms, Springer*, 264–267. DOI : 10.1007/3-211-27389-1_63.

RAD, BABAK BASHARI, MOHAMMAD KAZEM HASSAN NEJAD et MARYAM SHAHPASAND (2018). « Malware classification and detection using artificial neural network ». In : *Journal of Engineering Science and Technology* 13, p. 14-23.

RAIDL, Günther R (2006). « A unified view on hybrid metaheuristics ». In : *International workshop on hybrid metaheuristics*. Springer, p. 1-12.

RAMOS, Caio CO et al. (2011). « A novel algorithm for feature selection using harmony search and its application for non-technical losses detection ». In : *Computers & Electrical Engineering* 37.6, p. 886-894.

RASHEDI, Esmat, Hossein NEZAMABADI-POUR et Saeid SARYAZDI (2009). « GSA : a gravitational search algorithm ». In : *Information sciences* 179.13, p. 2232-2248.

RASHID, Tarik (2009). « A heterogeneous ensemble network using machine learning techniques ». In : *International Journal of Computer Science an Network Security* 9.8, p. 335-339.

RASHID, Tarik A, Dosti K ABBAS et Yalin K TUREL (2019). « A multi hidden recurrent neural network with a modified grey wolf optimizer ». In : *PloS one* 14.3, e0213237.

RECHENBERG, Ingo (1965). « Cybernetic solution path of an experimental problem ». In : *Royal Aircraft Establishment Library Translation 1122*.

— (1973). *Evolutions strategie : Optimierung technischer systeme nach prinzipien der biologischen evolution, frommann–holzboog*.

RHODE, Matilda, Pete BURNAP et Kevin JONES (2018). « Early-stage malware prediction using recurrent neural networks ». In : *computers & security* 77, p. 578-594.

ROTHLAUF, Franz (2011). *Optimization Methods*. Springer-Verlag Berlin, Heidelberg GmbH et Co. KG.

RUMELHART, David E, Geoffrey E HINTON et Ronald J WILLIAMS (1988). *Neurocomputing : Foundations of research*.

RUNKLER, Thomas A (2008). « Wasp swarm optimization of the c-means clustering model ». In : *International Journal of Intelligent Systems* 23.3, p. 269-285.

SAGARIKA, A et TR JYOTHSNA (2015). « Tuning of PSO algorithm for single machine and multi machine power system using STATCOM controller ». In : *International research journal of engineering and technology* 2.04, p. 175-182.

SAHA, Sujan Kumar, Sudeshna SARKAR et Pabitra MITRA (2009). « Feature selection techniques for maximum entropy based biomedical named entity recognition ». In : *Journal of biomedical informatics* 42.5, p. 905-911.

SAXE, Joshua et Konstantin BERLIN (2015). « Deep neural network based malware detection using two dimensional binary program features ». In : *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, p. 11-20.

SAXENA, Akash et al. (2018). « Intelligent Grey Wolf Optimizer–Development and application for strategic bidding in uniform price spot energy market ». In : *Applied Soft Computing* 69, p. 1-13.

SCHULTZ, Matthew G et al. (2000). « Data mining methods for detection of new malicious executables ». In : *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. IEEE, p. 38-49.

SEXTON, Randall S et al. (1998). « Global optimization for artificial neural networks : A tabu search application ». In : *European Journal of Operational Research* 106.2-3, p. 570-584.

SHAW, D et Witold KINSNER (1996). « Chaotic simulated annealing in multilayer feedforward networks ». In : *Proceedings of 1996 Canadian Conference on Electrical and Computer Engineering*. T. 1. IEEE, p. 265-269.

SHILAJA, C et T ARUNPRASATH (2019). « Internet of medical things-load optimization of power flow based on hybrid enhanced grey wolf optimization and dragonfly algorithm ». In : *Future Generation Computer Systems* 98, p. 319-330.

SIKDER, Md Fahim, Md Jamal UDDIN et Sajal HALDER (2016). « Predicting students yearly performance using neural network : A case study of BSMRSTU ». In : *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, p. 524-529.

SINDHU, R et al. (2019). « A hybrid SCA inspired BBO for feature selection problems ». In : *Mathematical Problems in Engineering* 2019.

SINGH, Harminder, Shivani MEHTA et Sushil PRASHAR (2016). « Economic load dispatch using multi verse optimization ». In : *International Journal of Engineering Research & Science (IJOER)* 6.2, p. 2395-6992.

SINGH, N et SB SINGH (2017a). « A novel hybrid GWO-SCA approach for optimization problems ». In : *Engineering Science and Technology, an International Journal* 20.6, p. 1586-1601.

SINGH, Narinder et SB SINGH (2017b). « Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance ». In : *Journal of Applied Mathematics* 2017.

STAELIN, Carl (2003). « Parameter selection for support vector machines ». In : *Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1* 1.

STORN, Rainer et Kenneth PRICE (1997a). « Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces ». In : *Journal of global optimization* 11.4, p. 341-359.

— (1997b). « Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces ». In : *Journal of global optimization* 11.4, p. 341-359.

STÜTZLE, Thomas et Holger H HOOS (2000). « MAX–MIN ant system ». In : *Future generation computer systems* 16.8, p. 889-914.

TAILLARD, Éric (1993). « Parallel iterative search methods for vehicle routing problems ». In : *Networks* 23.8, p. 661-673.

TALBI, E-G (2002). « A taxonomy of hybrid metaheuristics ». In : *Journal of heuristics* 8.5, p. 541-564.

TANESE, Reiko (1987). « Parallel genetic algorithm for a hypercube ». In : *Genetic algorithms and their applications : proceedings of the second International Conference on Genetic Algorithms : July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ : L. Erlhaum Associates, 1987.

TARKHANEH, Omid et Haifeng SHEN (2019). « Training of feedforward neural networks for data classification using hybrid particle swarm optimization, Mantegna Lévy flight and neighborhood search ». In : *Heliyon* 5.4, e01275.

TAWHID, Mohamed A et Abdelmonem M IBRAHIM (2020). « A hybridization of grey wolf optimizer and differential evolution for solving nonlinear systems ». In : *Evolving Systems* 11.1, p. 65-87.

TEH, Yee Whye et Geoffrey E HINTON (2001). « Rate-coded restricted Boltzmann machines for face recognition ». In : *Advances in neural information processing systems*, p. 908-914.

THARWAT, Alaa et Thomas GABEL (2019). « Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm ». In : *Neural Computing and Applications*, p. 1-14.

THARWAT, Alaa, Thomas GABEL et Aboul Ella HASSANIEN (2017). « Parameter optimization of support vector machine using dragonfly algorithm ». In : *International conference on advanced intelligent systems and informatics*. Springer, p. 309-319.

THARWAT, Alaa et Aboul Ella HASSANIEN (2018). « Chaotic antlion algorithm for parameter optimization of support vector machine ». In : *Applied Intelligence* 48.3, p. 670-686.

THARWAT, Alaa, Aboul Ella HASSANIEN et Basem E ELNAGHI (2017). « A BA-based algorithm for parameter optimization of support vector machine ». In : *Pattern Recognition Letters* 93, p. 13-22.

THARWAT, Alaa, Yasmine S. MOEMEN et Aboul Ella HASSANIEN (2017). « Classification of toxicity effects of biotransformed hepatic drugs using whale optimized support vector machines ». In : *Journal of Biomedical Informatics* 68, p. 132-149. ISSN : 1532-0464. DOI : https://doi.org/10.1016/j.jbi.2017.03.002. URL : https://www.sciencedirect.com/science/article/pii/S1532046417300515.

UNLER, Alper, Alper MURAT et Ratna Babu CHINNAM (2011). « mr2PSO : A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification ». In : *Information Sciences* 181.20, p. 4625-4641.

VAPNIK, Vladimir N (1999). « An overview of statistical learning theory ». In : *IEEE transactions on neural networks* 10.5, p. 988-999.

VATSA, Mayank, Richa SINGH et Afzel NOORE (2005). « Improving biometric recognition accuracy and robustness using DWT and SVM watermarking ». In : *IEICE Electronics Express* 2.12, p. 362-367.

WANG, Gai-Ge, Xinchao ZHAO et Suash DEB (2015). « A novel monarch butterfly optimization with greedy strategy and self-adaptive ». In : *2015 Second International Conference on Soft Computing and Machine Intelligence (ISCMI)*. IEEE, p. 45-50.

WANG, Lipo (2005). *Support vector machines : theory and applications*. T. 177. Springer Science & Business Media.

WITTEN, Ian H et al. (2005). « Practical machine learning tools and techniques ». In : *Morgan Kaufmann*, p. 578.

YAMANY, Waleed et al. (2015). « Moth-flame optimization for training multi-layer perceptrons ». In : *2015 11th International computer engineering Conference (ICENCO)*. IEEE, p. 267-272.

YANG, Qiang et Xindong WU (2006). « 10 challenging problems in data mining research ». In : *International Journal of Information Technology & Decision Making* 5.04, p. 597-604.

YANG, Xin-She (2010). « A new metaheuristic bat-inspired algorithm ». In : *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, p. 65-74.

YANG, XS (2011). « A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization.(NICSO). 284, 6574. doi : 10.1007 ». In :

YAO, Xin (1993). « A review of evolutionary artificial neural networks ». In : *International journal of intelligent systems* 8.4, p. 539-567.

YE, Yanbin et Chia-Chu CHIANG (2006). « A parallel apriori algorithm for frequent itemsets mining ». In : *Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06)*. IEEE, p. 87-94.

YU, Jianbo, Lifeng XI et Shijin WANG (2007). « An improved particle swarm optimization for evolving feedforward artificial neural networks ». In : *Neural Processing Letters* 26.3, p. 217-231.

YU, Xinying, Shie-Yui LIONG et Vladan BABOVIC (2004). « EC-SVM approach for real-time hydrologic forecasting ». In : *Journal of Hydroinformatics* 6.3, p. 209-223.

YUE, Zhihang, Sen ZHANG et Wendong XIAO (2020). « A novel hybrid algorithm based on grey wolf optimizer and fireworks algorithm ». In : *Sensors* 20.7, p. 2147.

YUVARAJ, N et P THANGARAJ (2019). « Machine learning based adaptive congestion window adjustment for Congestion Aware Routing in Cross Layer Approach Handling of Wireless Mesh Network ». In : *Cluster Computing* 22.4, p. 9929-9939.

ZAKI, Mohammed J (2001). « Parallel sequence mining on shared-memory machines ». In : *Journal of Parallel and Distributed Computing* 61.3, p. 401-426.

ZANCHETTIN, Cleber, Teresa B LUDERMIR et Leandro Maciel ALMEIDA (2011). « Hybrid training method for MLP : optimization of architecture and training ». In : *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41.4, p. 1097-1109.

ZHANG, Chengqi et Shichao ZHANG (2003). *Association rule mining : models and algorithms*. T. 2307. Springer.

ZHANG, Chong et al. (2018a). « A cost-sensitive deep belief network for imbalanced classification ». In : *IEEE transactions on neural networks and learning systems* 30.1, p. 109-122.

ZHANG, Guoqiang Peter (2000). « Neural networks for classification : a survey ». In : *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.4, p. 451-462.

ZHANG, Li et al. (2018b). « Feature selection using firefly optimization for classification and regression models ». In : *Decision Support Systems* 106, p. 64-85.

ZHANG, Sen et Yongquan ZHOU (2015). « Grey wolf optimizer based on Powell local optimization method for clustering analysis ». In : *Discrete Dynamics in Nature and Society* 2015.

ZHANG, XiaoLi, XueFeng CHEN et ZhengJia HE (2010). « An ACO-based algorithm for parameter optimization of support vector machines ». In : *Expert systems with applications* 37.9, p. 6618-6628.

ZHAO, Liang et Feng QIAN (2011). « Tuning the structure and parameters of a neural network using cooperative binary-real particle swarm optimization ». In : *Expert Systems with Applications* 38.5, p. 4972-4977.

ZHOU, Xin et David P TUCK (2007). « MSVM-RFE : extensions of SVM-RFE for multiclass gene selection on DNA microarray data ». In : *Bioinformatics* 23.9, p. 1106-1114.

ZHOU, Zhi-Hua et Xu-Ying LIU (2005). « Training cost-sensitive neural networks with methods addressing the class imbalance problem ». In : *IEEE Transactions on knowledge and data engineering* 18.1, p. 63-77.

ZHU, Aijun et al. (2015). « Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC ». In : *Journal of Systems Engineering and Electronics* 26.2, p. 317-328.

Liste des publications

Articles scientifiques

Automatic selection of hidden neurons and weights in neural networks for data classification using hybrid Particle swarm optimization, Multi-Verse Optimization based on Levy flight. Springer : Evolutionary Intelligence. DOI : 10.1007/s12065-021-00579-w.

A self-adaptive hybrid Bat Algorithm for training feedforward neural network. Rabab bousmaha, Reda Mohamed Hamou, Abdelmalek Amine. International Journal of Swarm Intelligence Research. (Volume 12, issue 4).

Optimizing connection weights in neural networks using hybrid metaheuristics algorithm. Rabab bousmaha, Reda Mohamed Hamou, Abdelmalek Amine. International journal information retrieval research (IJIRR) (Volume 12, issue 1).

Conférences et posters

A Comparison of Metaheuristics and BP for Training Feed forward Neural Networks. Poster presentation at the third edition of national Study Day on Computer Science Research (JERI' 2019) On April 23th, 2019 at Dr-Moulay Taher- University, Saida, Algeria.

Language - Independent in Twitter Sentiment Analysis at the Doctorial Symposium of 6th IFIP International Conference on Computational Intelligence and Its Applications (IFIP DS CIIA'2018) on May 8th, 2018 at USTO-MB University, Oran, Algeria.

Training feedforward neural networks using hybrid particle swarm optimization, Multi-Verse Optimization at the 1st Conference on Innovative Trends in Computer Science CITCS 2019, which will be held at 8 may 1945, Guelma university, Algeria