République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Seddik Ben Yahia de Jijel
Faculté des Sciences Exactes et Informatique
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme
Master de Recherche en Informatique

Option :

Réseaux et Sécurité

Thème

Détection d'intrusions via des réseaux de neurones optimisés par des métaheuristiques

Présenté par :
Cherfi Sarra

Encadré par :
Mr.Boulaiche Ammar

Année universitaire 2019/2020

Remerciement

*Je tiens tout d'abord à remercier **Dieu** le tout puissant et miséricordieux, qui m'ont donné la force et la patience d'accomplir ce modeste travail.*

Je tiens à saisir cette occasion et adresser mes profonds remerciements et mes profondes reconnaissances à toutes personnes qui m'ont aidé de près ou de loin dans la réalisation de ce mémoire.

*Je remercie **Mr Ammar Boulaiche** pour l'encadrement, l'aide et l'encouragement. Grâce à ses conseils et ses orientations j'ai pu terminer ce travail.*

*Je remercie vivement l'enseignant **Mr Ali Lemouari** pour ses précieux conseils et orientations ainsi que sa disponibilité et son soutien.*

Je joins ces remerciements également aux membres du jury pour leur attention et intérêts portés envers ce travail.

Je veut aussi adresser mes sincères remerciements à tous les enseignants de département de l'informatique qui ont contribué à ma formation.

Enfin, et surtout je remercie vivement toute ma famille notamment mes parents qui m'ont toujours encouragé dans la poursuite de mes études, ainsi que pour leur aide, leur compréhension et leur soutien sans oublier de remercier mes amis pour leurs soutiens et leurs bonnes humeurs pendant la préparation de ce mémoire.

Par crainte d'avoir oublié quelqu'un, que tous ceux et toutes celles dont je suis redevable se voie ici vivement remerciés.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

LISTE DES ABRÉVIATIONS

IDS Intrusions Detection Systems
CIA Confidentiality Integrity Availability
TCP Transmission Control Protocol
IP Internet Protocol
DoS Denial of Service
UDP User Datagram Protocol
ICMP Internet Control Message Protocol
FTP File Transfer Protocol
DNS Domain Name System
DMZ Demilitarized Zone
NIDS Network based Intrusion Detection System
HIDS Host based Intrusion Detection System
CAH Classification Ascendante Hiérarchique
MLP Multi Layer Perceptron
RNA Réseaux de Neurones Artificiels
SA Simulated Annealing
TS Tabu Search
KDD Knowledge Discovery and Data Mining
U2R User To Root
R2L Remote To Local

INTRODUCTION GÉNÉRALE

L'augmentation de la connectivité entre les divers réseaux informatiques et l'augmentation correspondante de la dépendance à l'égard des systèmes d'information en réseau ont conduit à une augmentation spectaculaire du besoin d'une sécurité robuste pour faire respecter les restrictions d'accès et empêcher toute intrusion sur les systèmes sécurisés effectuée par des utilisateurs externes ainsi que les utilisateurs internes.

La sécurité informatique est devenue donc un défi pour les administrateurs réseau ainsi que pour ses utilisateurs. Ces dernières années, plusieurs techniques de sécurité ont été proposées par les chercheurs de ce domaine pour garantir un niveau élevé de sécurité et fournir des nouvelles fonctionnalités qui ne pourraient pas être assurées par les méthodes classiques(pare-feu, anti-virus...etc). Les systèmes de détection d'intrusions sont l'une de ces techniques. Ils consistent à examiner le trafic réseau pour détecter toute violation de la politique de sécurité et avertir les responsables des réseaux via des alertes. En effet, malgré que les IDS ont rencontré un succès majeur dans le monde de cyber-sécurité, ils présentent quelques défauts comme le taux élevé des fausses alertes ainsi que le temps relativement élevé de détection.

Selon leur principe de fonctionnement, les IDS peuvent être classés en deux grandes catégories : ceux qui cherchent à détecter les malveillances (on parle alors de l'approche par signature ou par scénario) et ceux qui cherchent à détecter les anomalies (on parle alors de l'approche comportementale), où la première approche consiste à comparer le comportement d'utilisation du système avec des signatures d'attaques connues préalablement , donc elle ne peut pas détecter les nouvelles attaques, tandis que la deuxième approche consiste à connaitre ce qui est considéré comme trafic normal et détecter tout ce qui est déviant de ce comportement, donc elle a la capacité de détecter des nouvelles attaques sans avoir nécessité de faire une mise à jour.

Dans ce travail, nous allons réaliser un modèle de détection d'intrusions comportemental, qui a la capacité de détecter des nouvelles attaques en minimisant le taux des fausses alertes et maximisant le taux de réussite. Pour effectuer ces tâches, nous allons utiliser les réseaux de neurones artificiels comme technique d'apprentissage et classification combinés avec le recuit simulé puis avec la recherche tabou comme techniques d'optimisation.

Organisation du mémoire

Le présent manuscrit s'articule autour de quatre chapitres :

- ☞ Dans le premier chapitre nous présentons les notions de base sur la sécurité informatique incluant sa définition, ses objectifs et aussi les différentes attaques peuvent intervenir ainsi que les techniques de sécurisation des systèmes informatiques notamment les systèmes de détection d'intrusions.
- ☞ Le deuxième chapitre donne une introduction au domaine de classification de données et ses différentes méthodes y compris les réseaux de neurones qui seront détaillés dans la deuxième partie de ce chapitre en présentant leur historique, leur définition, leurs techniques d'apprentissage ainsi que leur architecture. La dernière section de ce chapitre est consacrée à une description détaillée du perceptron multicouche utilisé dans ce travail.
- ☞ Le troisième chapitre fait un rappel sur les méthodes d'optimisation combinatoire. Où nous allons exposer ses différentes méthodes avant de passer aux détails de l'algorithme de recuit simulé et la méthode de recherche tabou vu que ce sont les techniques que nous allons utiliser pour réaliser ce travail.
- ☞ Le quatrième chapitre est consacré à l'implémentation du recuit simulé et la recherche tabou dans les réseaux de neurones multi-couches, en faisant des tests sur le benchmark NSL-KDD qui sera présenté dans ce chapitre.
- ☞ Dans le dernier chapitre, nous montrons les résultats obtenus pour les différentes expérimentations effectuées que se soit en utilisant les méthodes d'optimisation (recuit simulé et recherche tabou) ou non.

CHAPITRE 1

SÉCURITÉ INFORMATIQUE ET SYSTÈME DE DÉTECTION D'INTRUSIONS

1.1 Introduction

En raison de plusieurs facteurs notamment l'ouverture des systèmes d'information sur Internet, l'évolution de la technologie et des moyens de communication ainsi que la transmission de données à travers les réseaux, des risques d'accès et de manipulation des données par des personnes non autorisées d'une façon accidentelle ou bien intentionnelle sont apparus. Donc la mise en place d'une politique de sécurité autour de ces systèmes est devenu une nécessité incontournable.

Le système de détection d'intrusion est l'une des techniques utilisées pour garantir un contrôle permanent des attaques ainsi que la détection de toute violation de cette politique, c'est-à-dire toute intrusion.

Dans ce premier chapitre nous introduisons les principales notions de base de la sécurité informatique y compris sa définition, ses objectifs, les problèmes et les attaques informatiques et aussi les mécanismes permettant d'améliorer la sécurité. Ensuite, nous présentons les systèmes de détection d'intrusions, leur définition, architecture, classification...etc, et nous terminons par les limites des systèmes de détection d'intrusions actuels.

1.2 Sécurité informatique

1.2.1 Définition

La sécurité informatique est définie par la protection assurée aux systèmes informatiques ainsi qu'aux données stockées, transférées ou manipulées. Cette protection doit réaliser trois principaux objectifs : l'intégrité, la disponibilité et la confidentialité des ressources du système informatique hôte.[1]

1.2.2 Objectifs de la sécurité

Dans la littérature on trouve plusieurs définitions pour les objectifs de la sécurité, mais les standards (The Federal Information Processing Standards , 2004) citent trois principaux objectifs appelés la triade CIA.

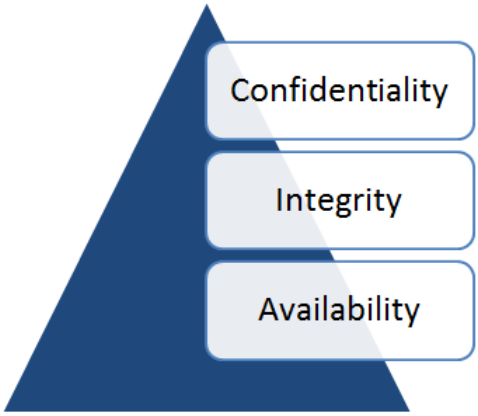


Figure 1.1 – Triade CIA

a) La confidentialité (confidentiality)

Permet d'assurer que les informations sauvegardées ou transmises sur le réseau ne soient pas dévoilées à des personnes, entités ou processus non autorisés, c'est-à-dire seules les personnes autorisées doivent pouvoir accéder aux données ou informations ainsi protégées.

b) L'intégrité (integrity)

Permet d'assurer que les données n'ont pas été altérées ou détruites de façon non autorisée, soit de manière accidentelle ou bien intentionnelle.

c) La disponibilité (availability)

Cet objectif vise à assurer l'accès aux ressources du système d'information conformément aux spécifications en terme de performances. Ceci implique que le temps d'attente et le temps de service sont tout les deux relativement raisonnables.

1.2.3 Soucis de la sécurité informatique

il existe trois problèmes qui affectent la sécurité informatique : les vulnérabilités, les menaces et les attaques.

a) Les vulnérabilités

Ce sont des failles ou des faiblesses dans la spécification, conception, implémentation ou bien configuration des systèmes informatiques dont l'exploitation peut créer une intrusion.

b) Les menaces

Une menace c'est la possibilité d'une violation d'une propriété de la sécurité en exploitant une ou plusieurs vulnérabilités d'une façon intentionnelle ou accidentelle.

c) Les attaques

Une attaque c'est une action malveillante qui tente d'exploiter une faiblesse dans le système et de violer un ou plusieurs besoins de sécurité.

1.2.4 Classification des attaques informatiques

Une attaque peuvent être classée selon son objectif, son point d'initiation ou la façon d'adresser la victime désirée.

a) Selon l'objectif d'attaque

On trouve deux types d'attaques principaux : passives et actives .

- **Les attaques passives :** ce type d'attaque ne provoque pas d'altération aux ressources du système ciblé ce qui le rend généralement indétectable (récupération du contenu d'un message ou bien l'observation du trafic).
- **Les attaques actives :** consistent à effectuer des modifications ou bien une destruction des ressources d'un système d'une manière non autorisée. Ce type d'attaque est plus dangereux que le premier et peut causer des dégâts (usurpation de l'identité, modification, replay, déni de service...etc).

b) Selon le point d'initiation

On distingue deux types d'attaques pour ce critère de classification : attaques de l'intérieur et attaques de l'extérieur.

- **Les attaques de l'intérieur :** provenant des utilisateurs légitimes d'un système lorsqu'ils se comportent de façon non autorisée.
- **Les attaques de l'extérieur :** venant de l'extérieur, souvent via Internet, en utilisant des techniques comme l'usurpation d'identité.

c) Selon la façon d'adresser la victime

Il existe deux façons pour adresser la victime soit d'une manière directe ou bien indirecte.

- **Les attaques directes :** dans ce type d'attaque, l'intrus adresse ses paquets directement à la victime sans passer par un intermédiaire.
- **Les attaques indirectes :** dans ce type d'attaque, l'adversaire envoie ses paquets vers une entité intermédiaire qui à son tour les retransmet vers la victime.

1.2.5 Buts des attaques

Il existe plusieurs objectifs pour les attaques informatiques :

- **Interruption :** vise la disponibilité des informations (DoS).
- **Interception :** vise la confidentialité des informations (sniffing, analyse de trafic,...etc).
- **Modification :** vise l'intégrité des informations.
- **Fabrication :** vise l'authenticité des Informations.

Les quatre objectifs sont illustrés dans la figure suivante :

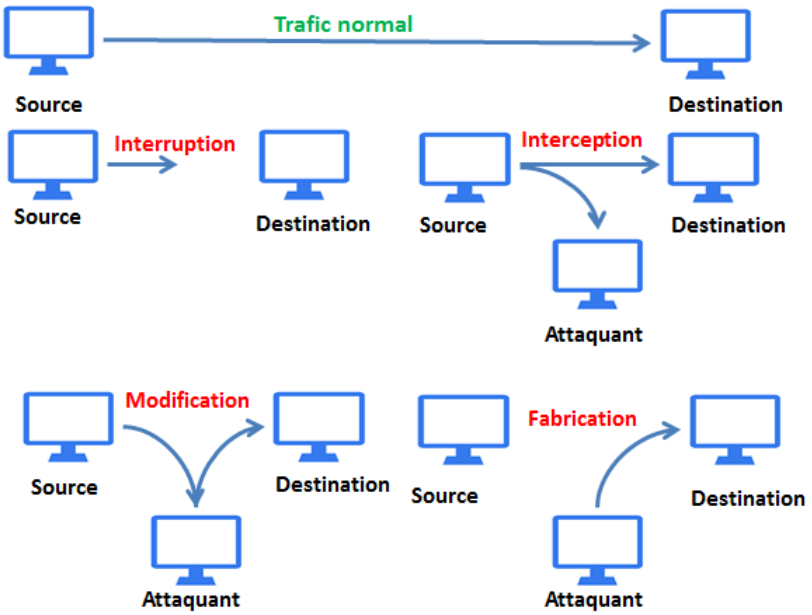


Figure 1.2 – Les objectifs des attaques informatiques
[2]

1.2.6 Motivations des attaques informatiques

Il existe plusieurs motivations d'un attaquant à vouloir exploiter une vulnérabilité et effectuer une attaque, parmi elles on peut citer les suivantes :[3]

- **motivations financières :** lorsqu'il s'agit de prendre possession des données pour rançonner l'entreprise ou le particulier.
- **motivations économique et concurrentielle :** on espionne ou on commet des actes de malveillances envers un concurrent dans le but d'acquérir un avantage commercial.
- **motivations politique ou idéologique :** comme semble l'indiquer la cyberattaque Not-Petya [1] ou le piratage de l'entreprise Ashley Madison (Mansfield-Devine, 2015) [2].

1.2.7 Exemples d'attaques informatiques

Il existe un nombre énorme d'attaques qui menacent les systèmes informatique à travers le monde entier, les plus connues aujourd'hui sont :

a) IP Spoofing

Le principe de l'attaque IP Spoofing est relativement ancien (aux alentours de 1985) alors que sa première application dans une vraie attaque ne remonte qu'à 1995. Kevin Mitnick, un célèbre hacker, l'utilise afin de s'infiltrer dans le réseau d'un expert en sécurité informatique, Tsutomo SHimomura. [4]

1. **NotPetya** est un logiciel malveillant de type wiper (il détruit les données), mais apparait sous la forme d'un rançongiciel (appelé aussi ransomware en anglais) en affichant sur l'écran de l'ordinateur infecté une demande de rançon. Son mécanisme de propagation permet de le classer comme ver informatique.
2. Les données de 32 millions de comptes sur Ashley Madison, le principal site de rencontres adultères aux Etats-Unis, ont été mises en ligne

Cette attaque consiste à usurper l'adresse IP d'une machine pour cacher la source d'attaque ou bien profiter d'une relation de confiance entre deux machines. Il existe des variantes car on peut spoofer aussi des adresses e-mail, des serveurs DNS ...etc.

b) Le dénis de service

Cette attaque consiste à envoyer des milliers des messages depuis des dizaines d'ordinateurs afin de saturer le système et donc le rendre indisponible. Ce type d'attaque est très facile à mettre en place mais très difficile à empêcher.
Un attaquant peut utiliser les DOS pour les raisons suivants :[4]

- obtenir le contrôle sur une machine cible ou sur un réseau. C'est le cas par exemple d'une attaque de type « SYN Flooding »qui est souvent utilisée de paire avec une tentative de spoofing.
- masquer les traces en détruisant les stations qui auraient pu contenir des traces d'un attaquant.
- se venger contre une personne, un administrateur ou bien encore une entreprise...etc.

Il existe plusieurs types d'attaques DOS comme il est montré dans la figure suivante :

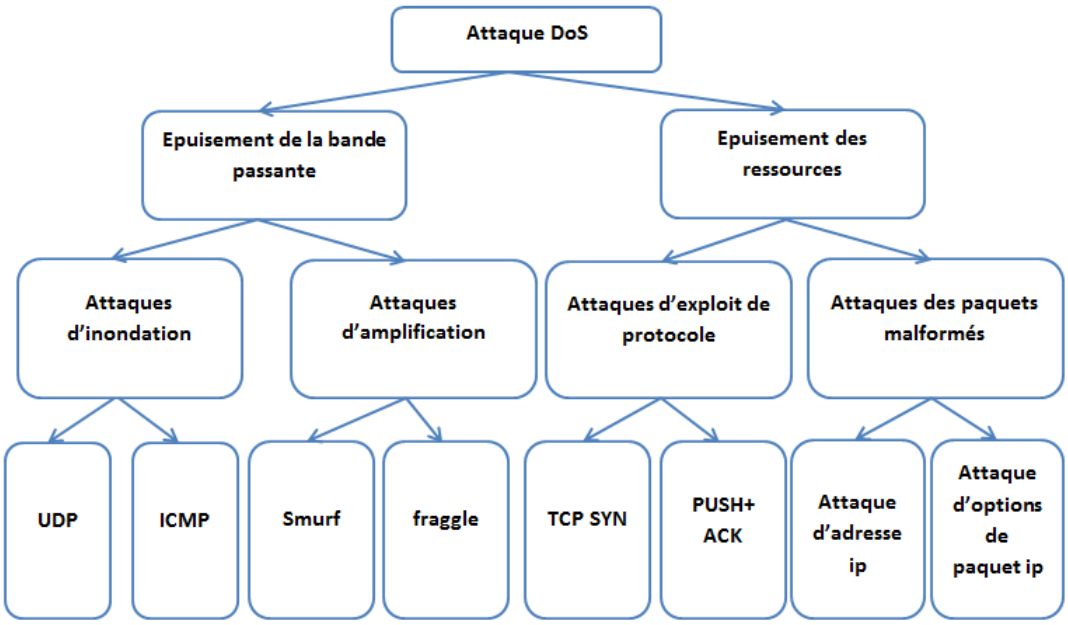


Figure 1.3 – Taxonomie des attaques dos
[5]

c) Probing (Sondage)

Le sondage est une attaque dans laquelle le pirate analyse une machine ou un réseau pour déterminer les faiblesses ou les vulnérabilités qui pourraient être exploitées plus tard afin de compromettre le système. Cette technique est couramment utilisée dans l'exploration de données, par exemple saint, portsweep, mscan, nmap...etc.[6]

Cette classe d'attaque est la plus étendue et qu'elle requit une expertise technique minime, donc il est très important de protéger le système de telles intrusions car elles sont à la base d'autres attaques comme R2L (Remote-to-Local), U2R (User-to-Root)...etc.

d) User to Root

Ces attaques sont des exploitations dans lesquelles le pirate démarre sur le système avec un compte d'utilisateur normal et tente d'abuser des vulnérabilités du système afin d'obtenir des privilèges de super utilisateur.[7]

En d'autre terme, l'objectif de cette attaque est d'obtenir les privilèges de l'administrateur système (Root) en allant d'un simple compte utilisateur (User) et cela en exploitant des failles dans le système comme le débordement de tampon, les erreurs de programmation..etc.

Il existe plusieurs attaques de ce type comme Eject,Ffbconfig, Fdformat, Load module, Perl, Xterm...etc.

e) Remote to Local

C'est une attaque dans laquelle l'attaquant exploite les vulnérabilités d'une machine distante comme les bugs des applications, les mauvaises configuration des systèmes d'exploitation et d'autres afin d'obtenir un accès illégal à celle-ci en exploitant les privilèges d'un utilisateur local.

Plusieurs attaques se trouvent dans cette catégorie parmi elles on cite : xlock, guest, xnsnoop, phf, Dict, warezmaster, spy, warezclient...etc.

1.2.8 Techniques et mécanismes de sécurisation

Pour réaliser les objectifs de sécurité cités dans la section 1.2.2, on doit prévoir un ensemble de mécanismes permettant de détecter toute attaque possible sur le système et même dans certains cas de prévenir ces attaques si cela est possible pour garantir un niveau élevé de protection du réseau et du système d'information. Ces mécanismes peuvent être implémentés à différents niveaux de l'architecture réseau en couche.

a) Le chiffrement

C'est un algorithme généralement basé sur des clefs pour transformer les données. Sa sécurité est dépendante du niveau de sécurité des clefs.[2]

Autrement dit, le chiffrement consiste à utiliser des algorithmes permettant de coder les données en une forme non intelligible afin de les protéger contre toute divulgation non autorisée. Cette technique permet d'assurer la confidentialité des données.

b) La signature numérique

Cette technique consiste à calculer une valeur à l'aide d'un algorithme de chiffrement, cette valeur sera ajoutée à une donnée d'une façon que tout récepteur de cette donnée puisse vérifier son origine.

La signature remplit deux fonctions juridiques principales [8] :

- L'identification de l'auteur et la manifestation de son consentement. La signature numérique est le pendant électronique à la signature manuscrite, mais la signature digitale est liée au document signé,elle n'est pas comparée à une signature témoin mais elle est vérifiée algorithmiquement alors elle est universellement vérifiable.
- Une signature numérique apporte la non-répudiation à l'origine, c'est-à-dire l'auteur d'une action ne peut dénier l'avoir effectué.

c) Le bourrage

Données ajoutées pour assurer la confidentialité, notamment au niveau du volume du trafic.[2]
Les mécanismes de bourrage servent à modifier les caractéristiques du trafic pour assurer différents niveaux de protection contre l'analyse du celui-ci.

d) Le contrôle d'accès

Ce mécanisme consiste à vérifier les droits d'accès aux données en laissant passer que les personnes autorisées et cela pour empêcher toute exploitation de vulnérabilités venant de l'extérieur.

e) La notarisation

Le mécanisme de notarisation consiste à reposer sur un tiers de confiance (notaire) qui détient les informations nécessaires pour assurer certains services de sécurité comme la non-répudiation.

f) Le pare-feu

Un pare-feu, ou coupe-feu ou encore firewall est un équipement ou des systèmes qui contrôlent le flux de trafic entre les différentes zones d'un réseau[9]. Donc, il assure un périmètre de protection entre le réseau interne à l'entreprise et le monde extérieur.
Voici une figure qui montre l'emplacement du pare-feu au seine d'une entreprise à fin de protéger le réseau local et les serveurs sensibles de l'entreprise (DMZ [3]).

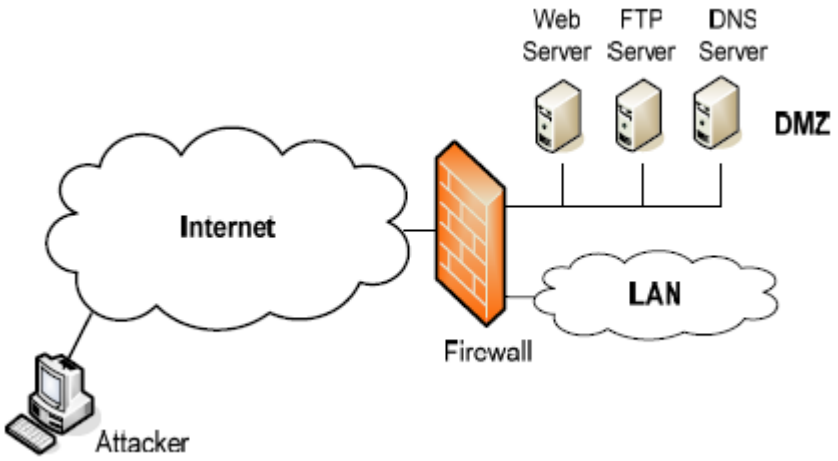


Figure 1.4 – Déploiement tri-hébergé d'un pare-feu de réseau d'entreprise
[10]

Un pare-feu peut assurer les tâches suivantes :

- bloquer l'accès à des services non autorisés.
- protéger en temps réel contre les menaces embarquées dans les applications.
- protéger contre les attaques de type DoS (Deni de service).
- intégrer des techniques de détection d'intrusions et envoyer des alertes afin de prévenir les équipes de surveillance technique.

3. **DMZ** :désigne (zone démilitarisée), est un sous-réseau séparé du réseau local et isolé de celui-ci et d'Internet (ou d'un autre réseau) par un pare-feu. Ce sous-réseau contient les serveurs sensibles de l'entreprise

- gérer les connexions sortantes à partir du réseau local.
- protéger le réseau interne des intrusions venant de l'extérieur.
- identifier et contrôler les applications partageant une même connexion.
- ...etc.

g) L'antivirus

C'est un logiciel permettant de préserver le système de tout type de maliciels (virus, vers, chevaux de troie...etc). Son principe de fonctionnement peut suivre l'une des trois approches :

- comparer la signature virale du virus aux codes vérifiés.
- utiliser les méta-heuristiques pour détecter les codes malveillants.
- utiliser le filtrage basé sur les règles.

h) La détection d'intrusions

La détection des intrusions est un mécanisme de cybersécurité courant dont la tâche est de détecter les activités malveillantes dans des environnements hôte et / ou réseau. La détection des activités malveillantes permettent de réagir en temps opportun, par exemple pour arrêter une attaque en cours. Vu l'importance de détection des intrusions, les milieux de la recherche et de l'industrie ont conçu et développé une variété de systèmes de détection d'intrusion (IDS).[11]

1.3 Systèmes de détection d'intrusions

1.3.1 Définitions

a) Intrusion

C'est toute utilisation d'un système informatique à des fins autres que celles prévues.[12] Autrement dit, c'est toute action malveillante qui vise l'un des objectifs de sécurité : La confidentialité, l'intégrité ou la disponibilité.

b) Détection d'intrusions

Consiste à analyser les informations collectées par les mécanismes d'audit de sécurité, à la recherche d'éventuelles attaques.[12]

c) Audit de sécurité

C'est un examen méthodique d'une organisation ou d'un site visant à identifier ses risques, ses vulnérabilités et les faiblesses de ses protections existantes ainsi qu'à statuer sur son niveau de sécurité et à recommander des solutions aux problèmes identifiés.[13]

d) Système de détection d'intrusions

Le système de détection d'intrusions inclure tous les systèmes logiciels et matériels permettant d'automatiser les processus de surveillance et d'analyse des évènements au sein d'un système informatique afin de détecter toute activité pouvant conduire à une défaillance de sécurité. Il peut être déployé sur une hôte, on parle alors de Host-Based Intrusion Detection System

(HIDS), ou bien sur un réseau, on parle alors de Network-Based Intrusion Detection System (NIDS).

1.3.2 Architecture d'un IDS

Plusieurs architectures ont été proposées pour décrire les différents éléments constituant un système de détection d'intrusions. L'architecture la plus simple est composée de trois modules :le capteur, l'analyseur et le manager. Cette architecture est montrée dans la figure suivante :

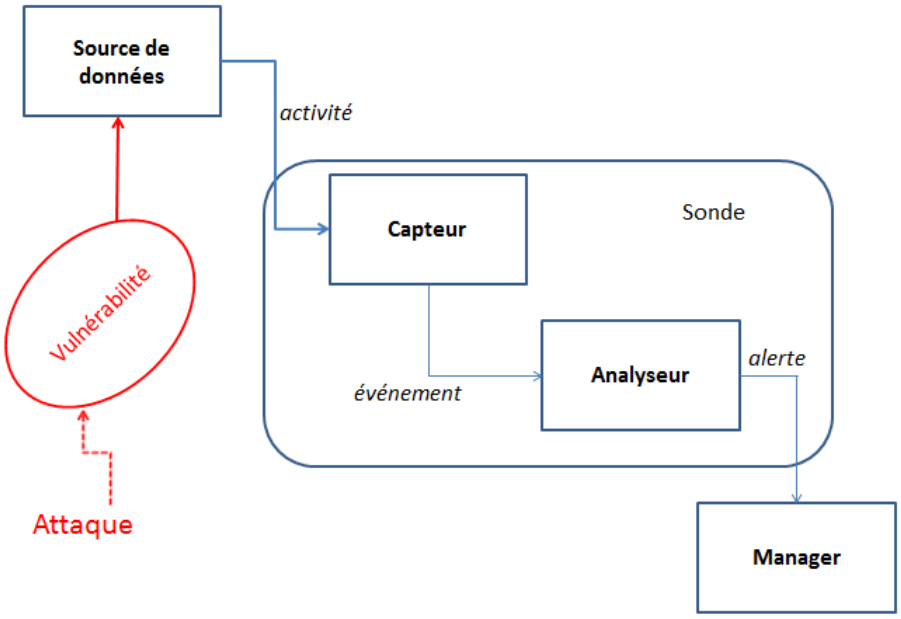


Figure 1.5 – Architecture classique d'un IDS
[14]

a) Le capteur

Chargé de collecter, filtrer et formater les informations brutes envoyées par la source de données concernant l'évolution de l'état du système. Le résultat de traitement est un message formaté appelé événement.

b) L'analyseur

Permet d'analyser les événements générés par le capteur en détectant toute activité malveillante qui peut se produire à partir d'un sous-ensemble de ces événements, et donc envoyer une alerte qui sera stockée dans les journaux du système ou bien utilisée pour lutter contre les attaques selon le type d'IDS.

c) Le manager

Permet de collecter et notifier les alertes envoyées par l'analyseur. Éventuellement, le manager est chargé de la réaction à adopter qui peut être :

- Isolement de l'attaque pour réduire les dégâts.
- Suppression d'attaque.

- Restauration du système dans un état sain.
- Identification du problème qui a engendré cette attaque.

1.3.3 Principe de fonctionnement d'un IDS

Le fonctionnement d'un IDS et le processus de détection d'intrusions sont illustrés dans la figure suivante :

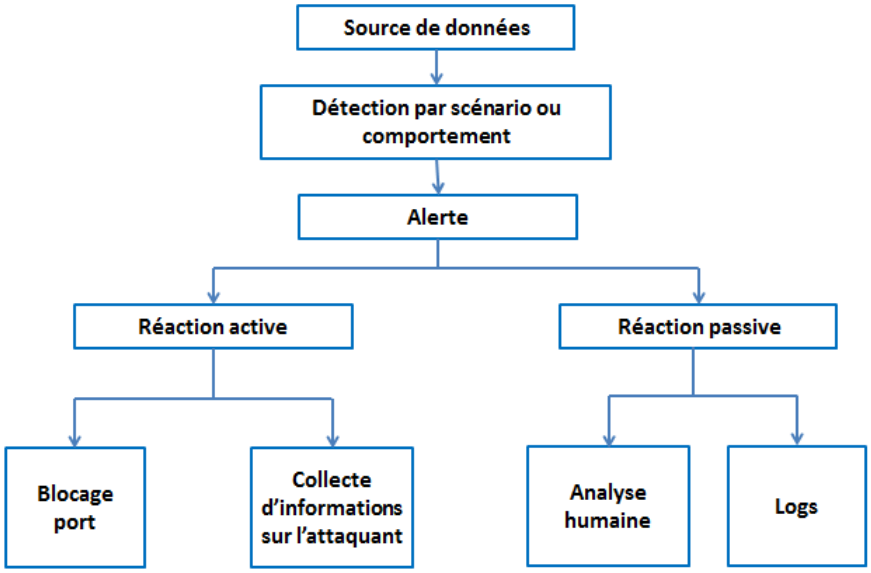


Figure 1.6 – Fonctionnement d'un IDS
[15]

1.3.4 Emplacement des IDS

Il existe plusieurs endroits stratégiques où il convient de placer un IDS pour atteindre le niveau de protection attendu selon la politique de sécurité choisie.

Le schéma suivant illustre un réseau local ainsi que les trois positions que peut y prendre un IDS :

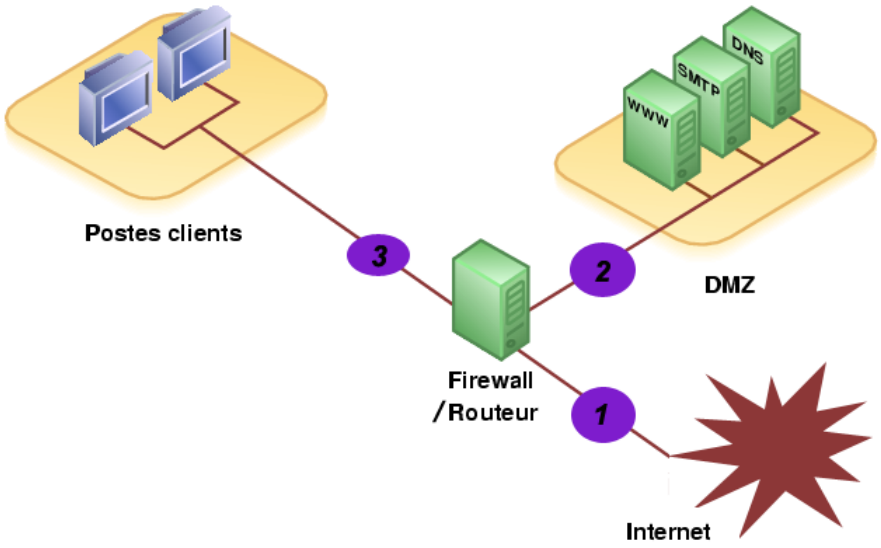


Figure 1.7 – Emplacement d'un IDS
[16]

Position 1 : Lorsque l'IDS prend cette position, son rôle sera de détecter l'ensemble des attaques frontales, provenant de l'extérieur, vers le pare-feu. Donc, plusieurs alertes seront remontées ce qui rendra les logs difficilement consultables.

Position 2 : Si l'IDS est placé sur la DMZ, il détectera les attaques qui n'ont pas été filtrées par le pare-feu et qui relèvent d'un certain niveau de compétence. Les logs seront ici plus clairs à consulter puisque les attaques bénines ne seront pas recensées.

Position 3 : L'IDS dans cette position a pour objectif de rendre compte des attaques internes, provenant du réseau local de l'entreprise. Il peut être judicieux d'en placer un à cet endroit étant donné le fait que 80% des attaques proviennent de l'intérieur. De plus, si des trojans ont contaminé le parc informatique (navigation peu méfiante sur internet) ils pourront être ici facilement identifiés pour être ensuite éradiqués.

1.3.5 Approches pour la détection d'intrusion

Il existe deux approches pour la détection d'intrusions. La première consiste à rechercher des signatures connues d'attaques tandis que la seconde consiste à définir un comportement normal du système et à rechercher ce qui ne rentre pas dans ce comportement.

Un système de détection d'intrusions par recherche de signatures connaît ce qui est mal, alors qu'un système de détection d'intrusions par analyse de comportement connaît ce qui est bien. On parle de détection de malveillances et de détection d'anomalies.[12]

a) Détection de malveillances

Cette approche est plutôt ancienne, remontant aux années 1990,et s'avère très efficace pour trouver des menaces connues. Elle consiste à rechercher des activités abusives par comparaison avec des descriptions abstraites de ce qui est considéré comme malveillant. Cette approche tente de mettre en forme des règles qui décrivent les usages non désirés, en s'appuyant sur des intrusions passées ou des faiblesses théoriques connues. En cas de détection d'une menace, une alerte est émise et le processus de remédiation est enclenché.
L'implémentation de cette approche peut être réalisée en utilisant plusieurs méthodes :

- **Systèmes experts :** Ils peuvent être utilisés pour coder les signatures de malveillance avec des règles d'implication si . . .alors. Les signatures décrivent un aspect d'une attaque ou d'une classe d'attaque. Il est possible d'ajouter des nouvelles règles pour les nouvelles attaques.
- **Analyse des transitions d'états :** On crée un modèle tel que le système au début ne soit pas compromis. L'intrus accède au système. Il exécute une série d'actions qui provoquent des transitions sur les états du modèle, qui peuvent être des états où on considère que le système soit compromis. Cette approche de haut niveau peut reconnaître des variations d'attaques qui passeraient inaperçues avec des approches de plus bas niveau.
- **Réseaux de neurones :** La flexibilité apportée par les réseaux neuronaux permet d'analyser des données même si elles sont incomplètes ou déformées. Ils peuvent de plus permettre une analyse non-linéaire de ces données. Leur rapidité permet l'analyse d'importants flux d'audit en temps réel. On peut utiliser les réseaux neuronaux pour filtrer et sélectionner les informations suspectes pour permettre une analyse détaillée par un système expert. On peut aussi les utiliser directement pour la détection de malveillances. Mais leur apprentissage est extrêmement délicat, et il est difficile de savoir quand un réseau est prêt pour l'utilisation. On peut également lui reprocher son côté boite noire (on ne peut pas interpréter les coefficients).
- **Raisonnement sur des modèles :** On essaye de modéliser les malveillances à un niveau élevé et intuitif d'abstraction en termes de séquences d'événements qui définissent l'intrusion. Cette technique peut être utile pour l'identification d'intrusions qui sont proches mais différentes. Elle permet aussi de cibler les données sur lesquelles une analyse approfondie doit être faite.
- **Algorithmes génétiques :** On définit chaque scénario d'attaque comme un ensemble pas forcément ordonné d'événements. Lorsqu'on veut tenir compte de tous les entremêlements possibles entre ces ensembles, l'explosion combinatoire qui en résulte interdit l'usage d'algorithmes de recherche traditionnels, et les algorithmes génétiques sont d'un grand secours.

On peut rapprocher les méthodes utilisées à cette approche à ceux qu'on peut les rencontrer au domaine des antivirus ou encore dans le domaine de la génomique où l'on recherche une séquence d'ADN dans un brin

La détection de malveillance a deux inconvénients principales :

- La difficulté de construction des bases de signatures.
- La non détection des attaques non connues.

b) Détection d'anomalies

Cette approche se base sur l'hypothèse que l'exploitation d'une faille du système nécessite une utilisation anormale de ce système, et donc un comportement inhabituel de l'utilisateur. Elle cherche donc à répondre à la question « *le comportement actuel de l'utilisateur ou du système est-il cohérent avec son comportement passé ?* ».

Il existe plusieurs méthodes pour la mise en œuvre de cette approche, parmi elles on peut citer :

- **Observation de seuils :** On fixe le comportement normal d'un utilisateur à certaine valeur (seuil), par exemple le nombre maximum de mots de passe erronés, mais Il est très difficile de caractériser un comportement intrusif en termes de seuils. En effet, on peut avoir

beaucoup de fausses alertes ou d'intrusions non détectées dans une population d'usagers non uniforme par exemple.

- **Profilage d'utilisateurs :** On crée et on maintiens des profils individuels du travail des utilisateurs, auxquels ils sont censés adhérer ensuite. Au fur et à mesure que l'usager change ses activités, son profil de travail attendu se met a jour. Certains systèmes tentent de concilier l'utilisation de profils à court terme et de profils à long terme. Il reste cependant difficile de profiler un utilisateur irrégulier ou très dynamique. De plus, un utilisateur peut arriver à habituer lentement le système à un comportement intrusif.
- **Profilage de programmes exécutables :** On observe l'utilisation des ressources du système par les programmes exécutables. Les virus, chevaux de Troie, vers, bombes logiques et autres programmes du même goût se voient démasqués en profilant la façon dont les objets du système comme les fichiers ou les imprimantes sont utilisés. Le profilage peut se faire par type d'exécutable.
- **Profilage adaptatif à base de règles :** Contrairement à la détection de malveillances à base des règles, là on n'a pas besoin des connaissances d'un expert car ces règles sont générées automatiquement lors de la phase d'apprentissage. Donc, l'efficacité de cette méthode nécessite la génération de beaucoup de règles ce qui engendre des problèmes de performance.
- **Réseaux de neurones :** Les réseaux neuronaux offrent une alternative à la maintenance d'un modèle de comportement normal d'un utilisateur. Ils peuvent offrir un modèle plus efficace et moins complexe que les moyennes et les déviations standards.

Cette approche a aussi beaucoup d'inconvénients comme :

- La difficulté à dire si les observations faites pour un utilisateur particulier correspondent à des activités que l'on voudrait prohiber.
- Pour un utilisateur au comportement erratique, toute activité est normale.
- Pas de prise en compte des tentatives de collusion entre utilisateurs.
- Choix délicat des différents paramètres du modèle statistique...etc.

c) Systèmes hybrides

Pour compenser les lacunes des méthodes précédentes, certains systèmes utilisent une combinaison de la détection d'anomalies et la détection de malveillances. Par exemple, un administrateur peut avoir un profil qui lui permet d'accéder à certains fichiers sensibles, mais on doit vérifier que les attaques connues ne soient pas utilisées contre ces fichiers. À l'inverse,l'utilisation des fichiers comportant le mot « nucléaire »ne caractérise aucune signature d'attaque, mais cela est possible si ce n'était pas dans les habitudes de l'utilisateur.

1.3.6 Efficacité des IDS

L'efficacité d'un système de détection d'intrusions est déterminée par les mesures suivantes : [17][18][19]

- **Exactitude :** Un système de détection d'intrusions n'est pas exact s'il déclare comme malicieux une activité légale. Ce critère correspond au faux positif.
- **Performance :** La performance de système de détection d'intrusions est le taux de traitement des événements. Si ce taux est faible, la détection en temps réel est donc impossible.

- **Perfection :** Un système de détection d'intrusions est imparfait s'il n'arrive pas à détecter une attaque.
- **Tolérance aux pannes :** Le système de détection d'intrusions doit lui-même résister aux attaques, en particulier dans le cas des attaques de déni de service. Ceci est important car plusieurs systèmes de détection d'intrusions s'exécutent sur des matériels ou logiciels connus vulnérables aux attaques.
- **Opportunité :** Un système de détection d'intrusions doit exécuter et propager son analyse d'une manière prompte pour permettre une réaction rapide dans le cas d'existence d'une attaque.

1.3.7 Limites des IDS

La plupart des systèmes de détection d'intrusions existants souffrent d'au moins deux des problèmes suivants :[20]

- Tout d'abord, les informations utilisées par le système de détection d'intrusions sont obtenues à partir d'un audit des chemins ou des paquets sur un réseau. Les données doivent parcourir un long chemin à partir de leur origine à l'IDS, donc elles peuvent potentiellement être détruites ou modifiées par un attaquant. En outre, le système de détection d'intrusions doit déduire le comportement du système à partir des données collectées, ce qui peut entraîner des interprétations erronées ou des événements manqués. Cela est appelé problème de fidélité.
- Deuxièmement, le système de détection d'intrusions utilise en permanence des ressources système supplémentaires, il surveille même lorsqu'il n'y a pas d'intrusions, car les composants du système de détection d'intrusions doivent fonctionner en permanence. C'est le problème de consommation des ressources.
- Troisièmement, parce que les composants du système de détection d'intrusions sont mis en œuvre comme programmes distincts, ils sont susceptibles d'être altérés. Un intrus peut potentiellement désactiver ou modifier les programmes exécutés sur un système, rendant la détection d'intrusions inutile ou peu fiable. C'est le problème de fiabilité.

1.4 Conclusion

Dans un monde où le progrès technologique avance à grande vitesse, où les gens, les entreprises, les organismes, les pays et même les objets sont de plus en plus connectés, les attaques informatiques sont de plus en plus fréquentes. La question de la cybersécurité se pose à tous les niveaux et tend à devenir un enjeu essentiel ces prochaines années.

Dans ce chapitre, nous avons abordé différentes notions concernant la sécurité informatique, où nous avons présenté les différents types d'attaques, leur classification, leurs objectifs et motivations et les techniques utilisées pour protéger le système contre ces attaques. Parmi ces mécanismes, nous avons détaillé les systèmes de détection d'intrusions, vu que c'est notre objectif dans ce mémoire, qui continuent d'évoluer pour répondre aux exigences et offre un éventail de fonctionnalités capables de satisfaire les besoins de tous les types d'utilisateurs.

Donc, nous avons détaillé l'architecture des systèmes de détection d'intrusions, leur principe de fonctionnement et les différentes approches pour la détection d'intrusions où on a divisé les IDS en deux grandes catégories , les IDS comportementaux et les IDS à base de signatures. Ces derniers consistent à détecter les attaques en se basant sur leurs signatures ce qui demande

une mise à jour périodique de la base des signatures et rend la détection des nouvelles attaques impossible. C'est pour cela qu'on a basé dans ce travail sur l'approche comportementale qui offre la possibilité de détecter les attaques inconnues en s'appuyant sur les réseaux de neurones et les méta-heuristiques qui seront détaillés dans les prochaines chapitres.

CHAPITRE 2

CLASSIFICATION ET RÉSEAUX DE NEURONES

2.1 Introduction

La classification de données est un problème délicat qui a apparu dans de nombreux domaines tels que l'analyse d'image, le diagnostic médical, l'apprentissage automatique...etc, et cela à cause de l'explosion de la quantité de données traitée par les systèmes informatiques lors de ces dernières années.

Dans le domaine de la sécurité informatique, la classification sert à classifier le trafic réseau ou les données du système surveillé en deux classes principale (normale/attaque,légal/illégal,...etc).En effet, les méthodes de classification sont implémentées dans les systèmes de détection d'intrusions pour les aider à prendre des décisions et générer des alertes en cas d'attaques avec le moins de fausses alarmes possibles. Pour ce faire, plusieurs méthodes de l'intelligence artificielle peuvent être utilisées, telle que les réseaux de neurones.

Dans ce chapitre, nous allons présenter la classification des données vu que c'est une phase principale dans le processus de génération de notre modèle de détection d'intrusions. Nous commençons tout d'abord par sa définition, ses catégories et les méthodes de classement les plus utilisées dans chacune. Ensuite, on focalise sur les réseaux de neurones en exposant leur historique, leur définition, leurs techniques d'apprentissage et leur architecture avant de passer aux détails de perceptron multicouches, vu que c'est l'outil de notre travail, et nous terminons par les avantages et les inconvénients des réseaux de neurones et leurs domaines d'applications.

2.2 Classification

2.2.1 Définition

La classification est l'opération statistique qui consiste à regrouper des objets(individus ou variables) en un nombre limité de classes de sort qu'on doit satisfaire deux conditions principales :[21]

- Ces classes ne sont pas prédéfinies par l'analyste mais découvertes au cours de l'opération.
- Une homogénéité interne et hétérogénéité externe, c'est-à-dire les classes de la classification regroupent les objets ayant des caractéristiques similaires et séparent les objets ayant des caractéristiques différentes.

2.2.2 Architecture typique d'une application basée sur la classification

La classification est une tâche très importante dans le data mining [1], mais le développement d'un outil de classification dans n'importe quel domaine doit être réalisé en appliquant d'autre phases d'extraction et de l'analyse d'informations qui sont montrées dans la figure suivante :

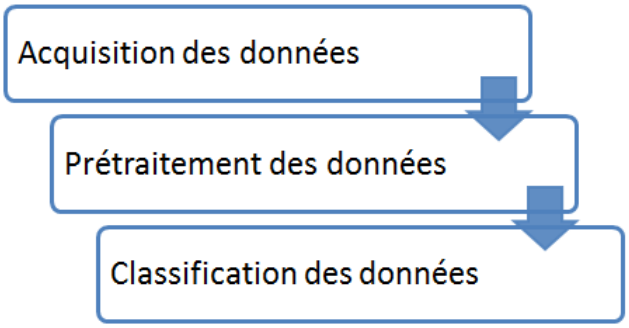


Figure 2.1 – Processus du data mining

- **Acquisition des données :** D'une manière générale, il s'agit de mettre en place l'ensemble d'instrumentation (capteurs, matériel d'acquisition, etc.) de façon à reproduire le phénomène observé le plus fidèlement possible. [22]. Dans notre cas, il s'agit de placer les sondes (IDS) pour écouter le trafic réseau.
- **Pré-traitement de données :** Cette phase correspond au filtrage des informations en ne conservant que ce qui est pertinent dans le contexte d'étude puisque ces données peuvent contenir plusieurs types d'anomalies (elles peuvent être omises à cause des erreurs de frappe ou à cause des erreurs dues au système lui-même, elles peuvent être incohérentes donc on doit les écarter ou les normaliser...etc). Parfois on est obligé à faire des transformations sur les données pour unifier leur poids.
- **Classification des données :** Dans cette étape, on doit choisir la bonne technique pour extraire les connaissances des données (les réseaux de neurones, les arbres de décision, les réseaux bayésiens...etc). Dans notre cas, la classification des connexions TCP/IP, on se base sur les réseaux de neurones.

2.2.3 Catégories de classification

Il existe un grand nombre des méthodes pour la résolution des problèmes de classification. Cependant, il est possible de les regrouper sous forme d'une hiérarchie de méthodes puisque certaines de ces approches partagent des caractéristiques communes, soit dans le type d'apprentissage utilisé (apprentissage supervisé ou non), ou bien dans la sortie réalisée (groupes disjoints ou classification floue).

La figure suivante montre une taxonomie des méthodes de classification dérivée de celle de Jain et Dubes (1988) :

1. Le **data mining** est un procédé d'exploration et d'analyse de grands volumes de données en vue d'une part de les rendre plus compréhensibles et d'autre part de découvrir des corrélations significatives

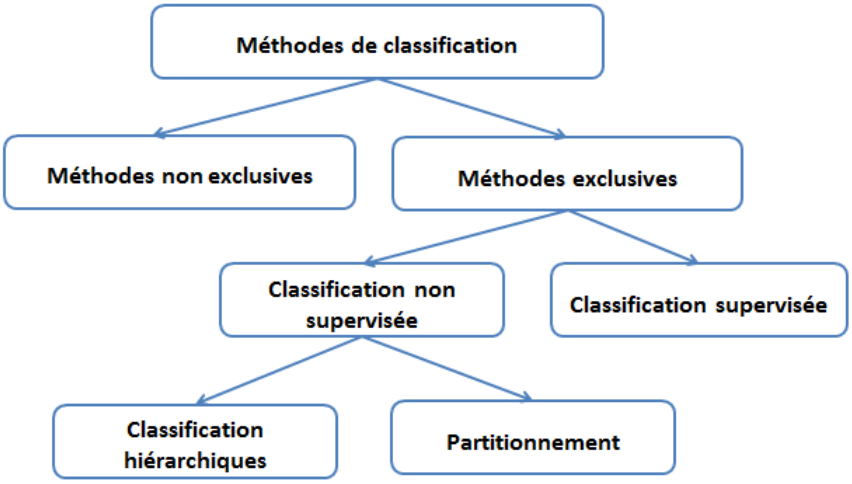


Figure 2.2 – Les méthodes de classification
[23]

a) Classification non exclusive

Une méthode est dite non exclusive si chaque objet est associé à une densité de probabilité qui indique pour chacune des classes la probabilité que l'objet considéré y appartienne (une classification floue).

b) Classification exclusive

Une méthode est dite exclusive si un objet ne peut être affecté qu'à une classe et une seule. Elle peut être divisée en deux autres classes :

I. **Classification non supervisée :** il s'agit d'extraire à partir d'une population des classes ou groupes d'individus présentant des caractéristiques communes. Le nombre et la définition des classes n'étant pas donnés a priori[24]. Le clustering regroupe un ensemble de techniques qui visent à regrouper les enregistrements d'une base de données en des groupes selon leur rapprochement les uns des autres en ne se basant sur aucune information antérieure.

il existe deux approches pour le clustering : le partitionnement et les méthodes hiérarchique.

- **Partitionnement :** consiste à construire plusieurs partitions puis les évaluer selon certains critères. Parmi les méthodes de partitionnement les plus connues on trouve **la méthode des k-moyennes(K-Means)**.
  Le principe de l'algorithme *k-means* est le suivant :
  - Choisir k objets formant ainsi k clusters.
  - (Ré)affecter chaque objet O au cluster Ci de centre Mi tel que distance(O,Mi) est minimale.
  - Recalculer Mi de chaque cluster.
  - Aller à la deuxième étape si on vient de faire une affectation.

Figure 2.3 – Exemple de partition obtenue en utilisant le K-means

- **Classification ascendante hiérarchique :** consiste à créer une décomposition hiérarchique des objets selon certains critères. Elle a pour objectif de construire une suite de partitions emboitées des données en n classes, n-1 classes,...,1 classe. Ces méthodes peuvent être vues comme la traduction algorithmique de l'adage « *qui se ressemble s'assemble* ».
  Le principe de l'algorithme CAH peut être exprimé comme suit :
  — A l'étape initiale, les n individus constituent des classes à eux seuls.
  — On calcule les distances deux à deux entre les individus, et les deux individus les plus proches sont réunis en une classe.
  — La distance entre cette nouvelle classe et les n-2 individus restants est ensuite calculée, et à nouveau les deux éléments les plus proches sont réunis.
  — Ce processus est réitère jusqu'à ce qu'il ne reste plus qu'une unique classe constituée de tous les individus.

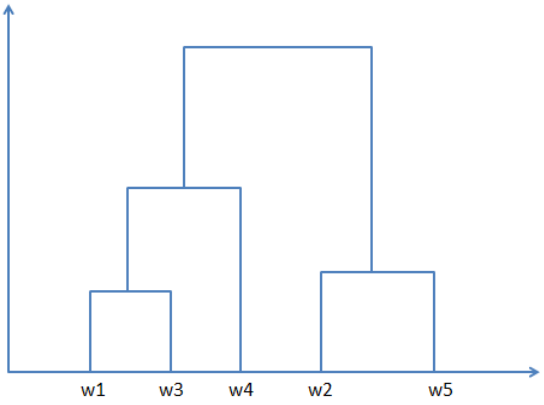


Figure 2.4 – Exemple de dendogramme (arbre hiérarchique)

II. **Classification supervisée :** elle consiste à inférer à partir d'un échantillon d'exemples classés une procédure de classification. Le problème est alors d'être capable d'associer à tout nouvel objet sa classe la plus adaptée, en se servant des exemples déjà étiquetés. [25]
Il existe plusieurs méthodes de classification supervisée, les plus connues sont : la méthode de k plus proche voisins et la méthode d'arbre de décision.

- **K plus proche voisins :** l'algorithme des k-plus proches voisins est un des algorithmes de classification les plus simples. Le seul outil dont on a besoin est une distance entre les éléments que l'on veut classifier.
  On considère que l'on dispose d'une base d'éléments dont on connait la classe, on parle de la base d'apprentissage, bien que cela soit de l'apprentissage simplifié. Dès que l'on reçoit un nouvel élément que l'on souhaite classifier, on calcule sa distance à tous les éléments de la base. Si cette base comporte 50 éléments, alors on calcule 50 distances et on obtient donc 50 nombres réels. Si k=5 par exemple, on cherche alors les 5 plus petits nombres parmi ces 50 nombres. Ces 5 nombres correspondent donc aux 5 éléments de la base qui sont les plus proches de l'élément que l'on souhaite classifier. On décide d'attribuer à l'élément à classifier la classe majoritaire parmi ces 5 éléments.
- **Arbres de décision :** les arbres de décision représentent l'une des techniques les plus connues et les plus utilisées en classification. Leur succès est notamment dû à leur aptitude à traiter des problèmes complexes de classification. En effet, ils offrent une représentation facile à comprendre et à interpréter, ainsi qu'une capacité à produire des règles logiques de classification.[26]
  Un arbre de décision est caractérisé par :
  — chaque nœud correspond à un test sur la valeur d'un ou plusieurs attributs.
  — chaque branche partant d'un nœud correspond à une ou plusieurs valeurs de ce test.
  — à chaque feuille, une valeur de l'attribut cible est associée.

  L'utilisation des arbres de décision dans les problèmes de classification se fait en deux étapes principales :
  — la construction d'un arbre de décision à partir d'une base d'apprentissage.
  — la classification ou l'inférence consistant à classer une nouvelle instance à partir de l'arbre de décision construit dans la première étape.

  Voici un exemple d'arbre de décision et la partition qu'il implique :

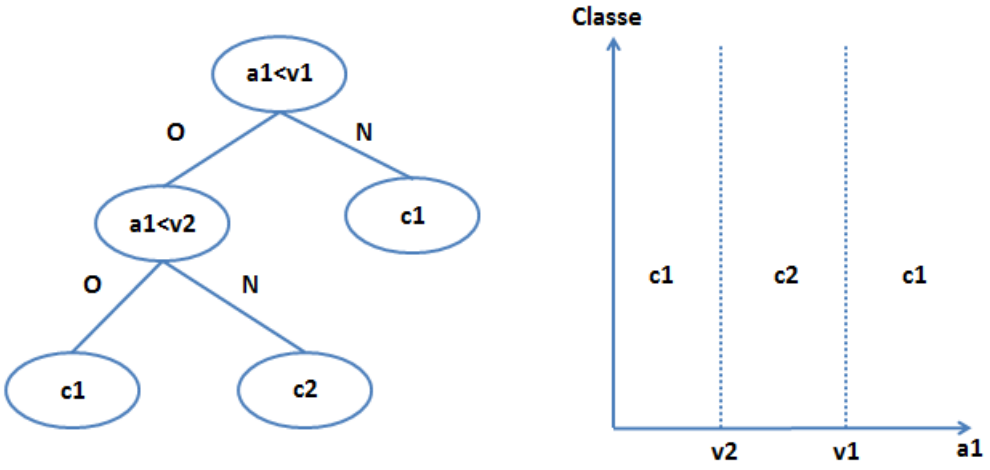


Figure 2.5 – Exemple d'une classification par arbre de décision
[27]

Il existe d'autres méthodes pour la classification supervisée, telle que la méthode de réseaux de neurones qui sert à résoudre les problèmes de classification avec les modèles non-linéaires. Cette approche sera abordée en détail dans le reste de ce chapitre.

2.3 Réseaux de neurones

Les réseaux de neurones artificiels sont inspirés de la méthode de travail du cerveau humain qui est totalement différente de celle d'un ordinateur. Le cerveau humain se base sur un système de traitement d'information parallèle et non linéaire, très compliqué, ce qui lui permet d'organiser ses composants pour traiter, d'une façon très performante et très rapide, des problèmes très compliqués tel que la reconnaissance des formes.[28]

2.3.1 Historique

Les premières tentatives de modélisation du cerveau sont anciennes et précèdent même l'informatique, voici quelques dates qui ont marqué l'histoire du domaine connexionniste :[29] [30] [31]

- En 1890, W. Jones a introduit le concept de mémoires associatives et proposa une loi de fonctionnement pour l'apprentissage dans les réseaux de neurones, connue plus tard sous le nom de «Loi de Hebb »
- En 1943, le neurologue Warren Sturgis McCulloch et le logicien Walter Pitts ont proposé un modèle simplifié de neurone biologique appelé neurone formel capable de représenter des fonctions booléennes simples.
- En 1949, D. Hebb a présenté dans son ouvrage « *The Organization of Behavior* »une règle d'apprentissage. De nombreux modèles de réseaux aujourd'hui s'inspirent encore de la règle de Hebb.
- En 1958, le premier succès est apparu quand F.Rosenblatt a présenté le premier modèle opérationnel nommé « Perceptron ».C'était le premier système artificiel qui pouvait apprendre par expérience, y compris lorsque son instructeur commettait des erreurs.
- En 1969, Les recherches sur les réseaux de neurones ont été pratiquement abandonnées lorsque M. Minsky et S. Papert ont publié leur livre « *Perceptrons* »(1969) et démontré les limites théoriques du perceptron mono-couche du point de vue performance.
- En 1982, Hopfield développe un modèle qui utilise des réseaux totalement connectés basés sur la règle de Hebb pour définir les notions d'attracteurs et de mémoire associative. En 1984 c'est la découverte des cartes de Kohonen avec un algorithme non supervisé basé sur l'auto-organisation et suivi une année plus tard par la machine de Boltzman (1985).
- En 1986, Rumelhart, Hinton et Williamsrreur ont introduit le perceptron multi-couches qui repose sur la rétro-propagation du gradient de l'erreur dans les systèmes à plusieurs couches.

A nos jours, l'utilisation des réseaux de neurones dans divers domaines ne cesse de croître. Les applications en sont multiples et variées.

2.3.2 Neurone artificiel et neurone biologique

a) Neurone biologique

Un neurone biologique comprend :[32]

— le corps cellulaire, qui fait la somme des influx qui lui parviennent. Si cette somme dépasse un certain seuil, il envoie lui-même un influx par l'intermédiaire de l'axone.

— l'axone, qui permet de transporter les influx en provenance du corps cellulaire vers d'autres cellules.

— les dendrites, qui sont les récepteurs principaux du neurone, captant les signaux qui lui parviennent.

— les synapses, qui permettent aux neurones de communiquer avec les autres via les axones et les dendrites.

La figure suivante représente un neurone biologique :

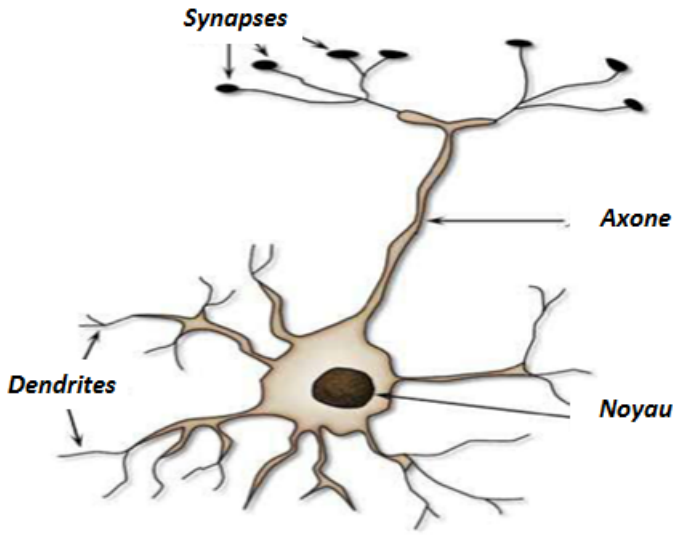


Figure 2.6 – Un neurone biologique
[33]

b) Neurone artificiel

Un neurone formel (ou simplement « neurone ») est une fonction algébrique non linéaire et bornée, dont la valeur dépend de paramètres appelés coefficients ou poids. Les variables de cette fonction sont habituellement appelées « entrées »du neurone, et la valeur de la fonction est appelée sa « sortie ». Un neurone est donc avant tout un opérateur mathématique, dont on peut calculer la valeur numérique par quelques lignes de programme informatique.[34]

Un réseau de neurones formel peut être représenté comme suit :

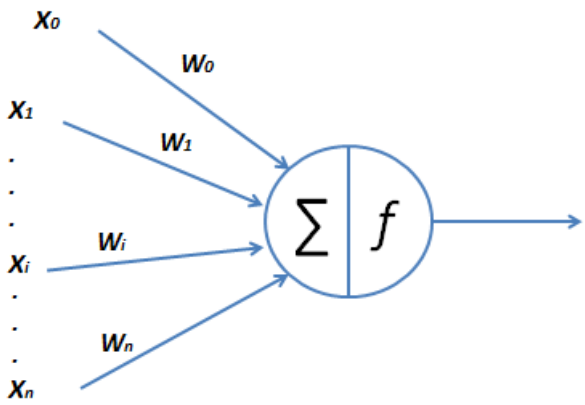


Figure 2.7 – Un neurone formel

La structure d'un neurone artificiel est en fait inspirée de la structure des neurones biologiques et la figure suivante montre la mise en correspondance entre ces deux types de neurones :

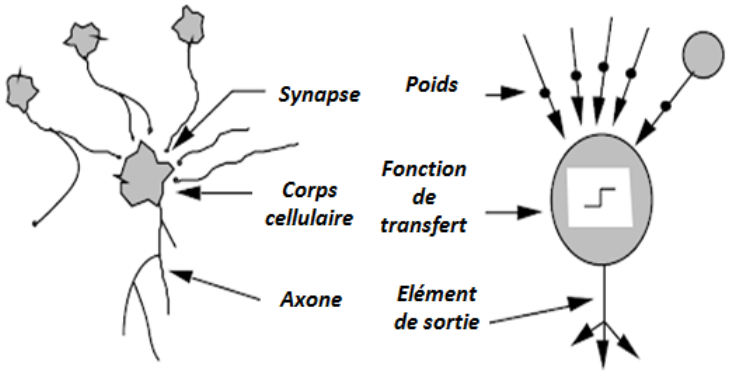


Figure 2.8 – Une correspondance entre neurone artificiel et neurone biologique [33]

2.3.3 Fonctionnement des réseaux de neurones

Un neurone est une unité de calcul élémentaire qui combine des entrées réelles $X_1, X_2, ...X_n$ en une sortie réelle S. Ces entrées proviennent soit d'autres éléments « processeurs »(neurones), soit de l'environnement et elles n'ont pas la même importance, donc à chaque entrée X_i est associé un poids (ou coefficient synaptique) W_i. En règle générale, l'activité en entrée est mesurée par la somme pondérée des entrées $\sum W_i X_i$. Puis on applique une fonction de transfert f (appelée fonction d'activation) au résultat afin d'obtenir la sortie.

Il existe de nombreuses formes possibles pour la fonction de transfert. Les plus courantes sont montrées ci-dessous :

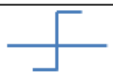
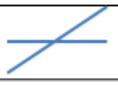
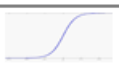
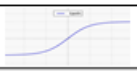
Nom de la fonction	*Relation d'entrée/sortie*	*Icône*
Seuil	a=0 si n<0 a=1 si n≥0	
Seuil symétrique	a=-1 si n<0 a=1 si n≥0	
Linéaire	a=n	
Linéaire saturée	a=0 si n<0 a=n si 0≤n≤1 a=1 si n>1	
Linéaire saturée symétrique	a=-1 si n<-1 a=n si -1≤n≤1 a=1 si n>1	
Linéaire positive	a=0 si n<0 a=n si n≥0	
Sigmoïde	$a=\frac{1}{1+e^{-n}}$	
Tangente hyperbolique	$a=\frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$	
compétitive	a=0 si n maximum a=1 autrement	C

Tableau 2.1 – Les fonctions de transfert les plus utilisées
[35]

2.3.4 Apprentissage des réseaux de neurones

Les réseaux de neurones ont la capacité d'apprendre à partir d'exemples de manière assez analogue à celle des humains basée sur leur expérience, cet apprentissage peut être de différents modes :[36]

- **L'apprentissage non supervisé :** le réseau doit détecter des points communs aux exemples présentés, par la modification des poids, afin de fournir la même sortie pour des entrées aux caractéristiques proches.
  L'apprentissage non supervisé est bien adapté à la modélisation des données complexes (images, sons, etc.), généralement des données symboliques, où l'on possède des règles moins précises qui gouvernent le comportement du système à modéliser par les réseaux de neurones.
- **L'apprentissage supervisé :** dans le cas de l'apprentissage supervisé, l'adaptation intervient directement lorsque le système compare la réponse qu'il a calculé avec la réponse attendue en fonction des entrées fournies. La performance de l'apprentissage est déterminée par l'intermédiaire d'un critère à optimiser.
- **Le mode hybride :** ce mode prend en fait les deux autres approches, puisqu'une partie des poids doit être déterminée par apprentissage supervisé et l'autre partie par apprentissage non-supervisé.

2.3.5 Architecture des réseaux de neurones

On distingue deux grands types d'architectures de réseaux de neurones : les réseaux de neurones non bouclés et les réseaux de neurones bouclés.

- **Les réseaux de neurones bouclés :**
  Dans ce type de réseau, les neurones sont connectés dans n'importe quel sens d'une façon cyclique, c'est-à-dire, lorsqu'on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ. Ce type de réseau peut être schématisé comme suit :

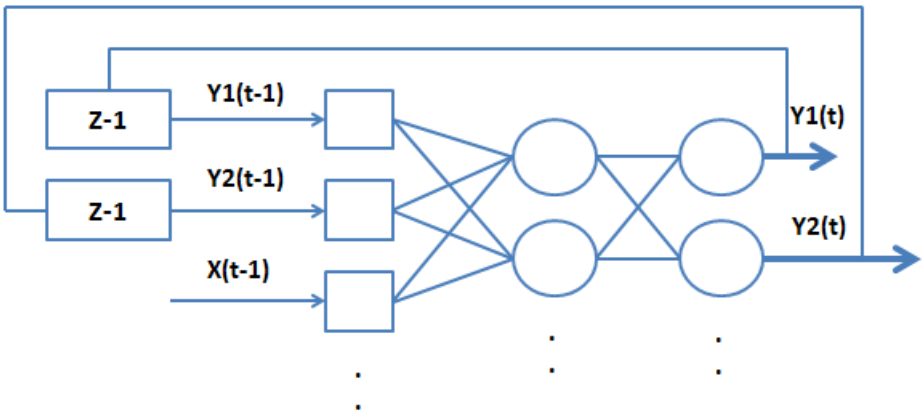


Figure 2.9 – Un réseau de neurone bouclé

- **Les réseaux de neurones non-bouclés :**
  Dans ce type de réseau, les connexions entre les neurones des différentes couches partent dans un seul sens. On a d'abord une couche d'entrée avec seulement des entrées du réseau, ensuite les couches cachées contenant des neurones cachés et en dernier la couche de sortie englobant les neurones visibles. Un tel réseau peut être représenté comme suit :

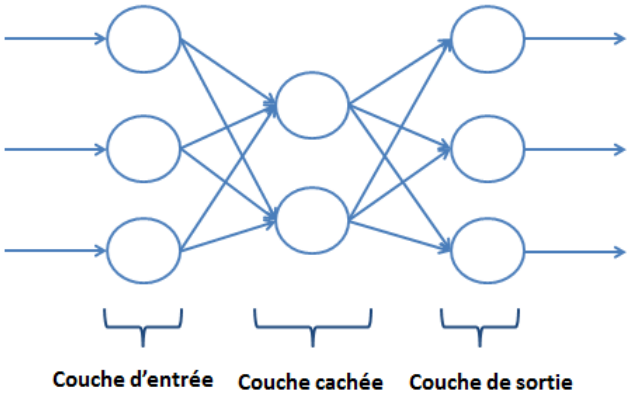


Figure 2.10 – Un réseau de neurone non-bouclé

2.3.6 Quelques modèles de réseaux de neurones

À nos jours, où les chercheurs n'ont de cesse que d'inventer de nouveaux modèles de réseaux de neurones plus adaptés à la résolution des problèmes particuliers, on trouve plusieurs modèles qui sont disponibles et largement utilisés, parmi eux on peut citer les suivants : [37]

- **Le perceptron simple :**
  C'est le modèle le plus simple qui comporte uniquement deux couches, une pour les entrées constituée de n neurones élémentaires dont la fonction d'activation est linéaire est une couche de sortie constituée d'un ou plusieurs neurones.

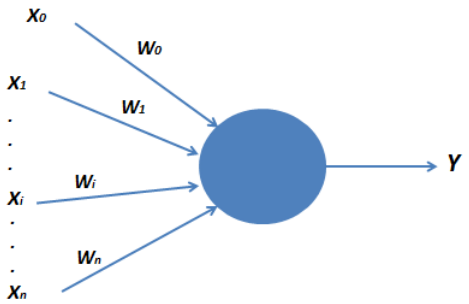


Figure 2.11 – Un perceptron simple(mono-couche)

- **Le perceptron multi-couches :**
  C'est le type le plus connu des réseaux de neurones. C'est une extension du perceptron mono-couche avec la présence d'une couche intermédiaire constituée d'une ou plusieurs couches cachées. Les connexions n'existent qu'entre les cellules d'une couche avec les cellules de la couche suivante. Et vu que ce modèle de réseaux de neurones est celui que nous allons utiliser pour mettre en œuvre notre modèle de détection d'intrusions, il sera détaillé dans la suite de ce chapitre.

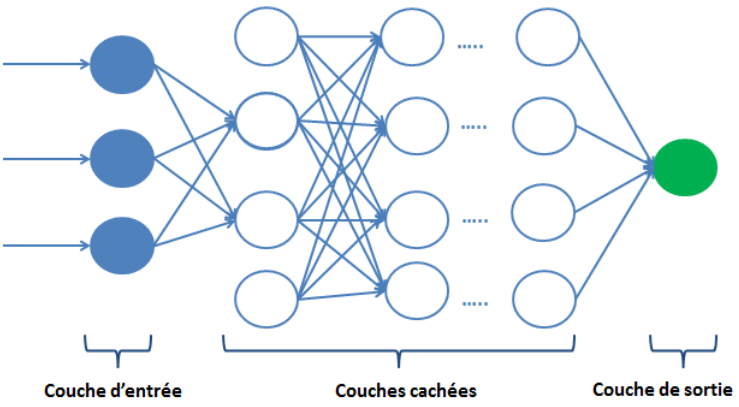


Figure 2.12 – Un perceptron multi-couches

- **Modèle de Kohonen :**
  Les cartes topologiques ou cartes auto organisatrices ont été introduites pour la première fois par T. Kohonen en 1981. Elles sont réalisées à partir d'un réseau à deux couches(une couche d'entrée et une couche de sortie).Notons que les neurones de la couche d'entrée sont entièrement connectés à la couche de sortie.Les neurones de la couche de sortie sont composés d'un certain nombre de neurones régulièrement répartis sur une carte.

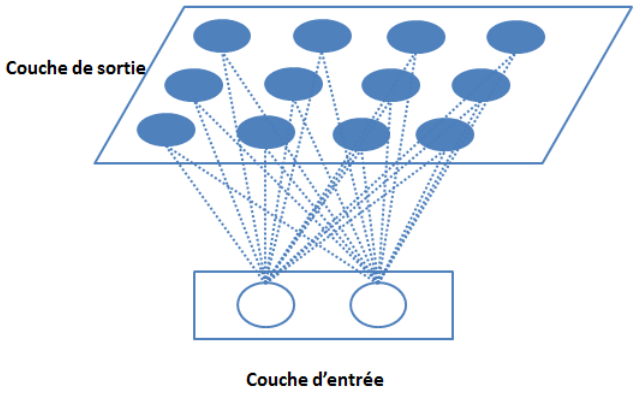


Figure 2.13 – Le modèle de Kohonen

- **Modèle de Hopfield :**
  Le modèle de Hopfield fut présenté en 1982. Ce modèle très simple est basé sur le principe des mémoires associatives. C'est d'ailleurs la raison pour laquelle ce type de réseau est dit associatif (par analogie avec le pointeur qui permet de récupérer le contenu d'une case mémoire). Le modèle de Hopfield utilise l'architecture des réseaux complètement connectés et récurrents. Les sorties sont en fonction des entrées et du dernier état pris par le réseau.

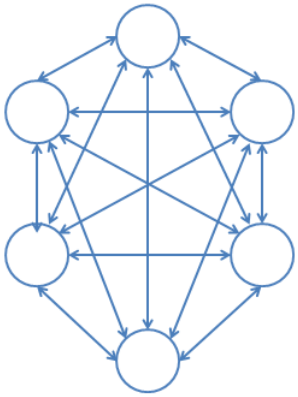


Figure 2.14 – Le modèle de Hopfield

2.3.7 Perceptron multi-couche

Le perceptron multicouche (MLP [2]) est un des réseaux de neurones les plus utilisés pour des problèmes d'approximation, de classification et de prédiction. Ce type de réseau est dans la famille générale des réseaux à propagation vers l'avant, c'est-à-dire qu'en mode normal d'utilisation, l'information se propage dans un sens unique, des entrées vers les sorties sans aucune rétroaction. Son apprentissage est de type supervisé par correction des erreurs. Dans ce cas uniquement, le signal d'erreur est rétro-propagé vers les entrées pour mettre à jour les poids des neurones. Il est habituellement constitué de deux ou trois couches de neurones totalement connectés.

a) Mise en oeuvre du réseau de neurones MLP

La mise en œuvre des réseaux de neurones comporte à la fois une partie conception, qui permet de choisir la meilleure architecture possible, et une partie de calcul numérique, pour réaliser l'apprentissage d'un réseau de neurones. En effet, un MLP peut avoir un nombre de couches et un nombre de neurones par couche quelconques, mais en vue de perfectionner le fonctionnement du MLP d'un part et réduire au maximum le temps de calcul d'autre part, on doit chercher une architecture optimale au point de vue nombre de couches , nombre de neurones par couche et nombre de sorties possibles. À partir d'une architecture de réseau de neurones donnée et des exemples disponibles (la base d'apprentissage), on détermine les poids optimaux, par l'algorithme de la rétro-propagation des erreurs, pour que la sortie du modèle s'approche le plus possible du fonctionnement désiré.[36]

b) Apprentissage des réseau MLP

L'apprentissage d'un réseau de neurones multicouches se fait généralement par l'algorithme de la rétro-propagation « back-propagation »qui est l'exemple d'apprentissage supervisé le plus utilisé. Cette technique de rétro-propagation du gradient permet de calculer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première. Le principe de cet algorithme peut être décrit en trois étapes principales : l'acheminement de l'information à travers le réseau, la rétro-propagation des sensibilités et calcul du gradient et enfin l'ajustement des paramètres par la règle du gradient approximé.[38]

- **Propagation avant :** dans cette étape, on calcule la valeur d'entrée pour chaque neurone *j* de la couche cachée ou bien de la couche de sortie. Cette valeur est égale à la somme pondérée des valeurs de sortie des neurones de la couche antérieure. Elle est calculée à partir de la formule suivante [39] :

$$\mathrm{VE_j} = \sum_{i \in pred(j)} W_{ij} VA_i$$

On calcule ensuite la valeur d'activation de chaque neurone *j* en utilisant une fonction de transfert f :

$$VA_\mathrm{j} = f(VE_\mathrm{j})$$

Il existe plusieurs fonctions d'activation, la plus utilisée dans le cas des réseaux de neurones multi-couches est la fonction sigmoïde, qui est utilisée dans ce travail. Donc la valeur d'activation est calculée comme suit :

2. Multi Layer Perceptron

$$VA_{\mathrm{j}} = f(VE_{\mathrm{j}}) = \frac{1}{1 + e^{-VE_{\mathrm{j}}}}$$

On répète cette opération jusqu'à le calcul des valeurs d'activation des neurones de la couche de sortie. La différence entre ces valeurs et les valeurs désirées représente l'erreur d'apprentissage appelée delta (Δ).

- **Rétro-propagation :** après le calcul de l'erreur d'apprentissage dans la phase de propagation avant, ce dernier va être distribué à travers les couches du réseau en allant des sorties vers les entrées (vers l'arrière) afin de pouvoir ajuster dans la phase suivante les poids du réseau.

  On calcule l'erreur (Δ) de chaque neurone s de la couche de sortie par la formule suivante [39] :

$$\Delta_{\mathrm{s}} = VA_{\mathrm{s}}(1 - VA_{\mathrm{s}})(\textit{val désirée de S} - VA_{\mathrm{s}})$$

  On calcule ensuite l'erreur de chaque neurone caché *j* en utilisant la formule suivante [39] :

$$\Delta_{\mathrm{j}} = VA_{\mathrm{j}}(1 - VA_{\mathrm{j}}) \sum_{k \in succ(j)} w_{jk}\Delta_k$$

- **Mise à jours des poids :** lorsque on distribue l'erreur d'apprentissage sur tous les neurones de la couche cachée, on passe à la mise à jours des poids et des biais à l'aide des formules suivantes [39] :

$$w_{ij} = w_{ij} + \alpha * VA_i * \Delta_j$$
$$b_j = b_j + \alpha * \Delta_j$$

  Où :α est le taux d'apprentissage choisi par l'utilisateur(entre 0 et 1)

On répète ce processus (propagation avant,rétro-propagation et mise à jours des poids) autant de fois que le nombre d'exemples de la base d'apprentissage, lorsque on termine on calcule la somme des erreurs au carré (SEC) tel que [39] :

$$SEC = \frac{1}{n} \sum_{i=1}^{n} (y_k^{des} - y_k)^2$$

Où : y_k^{des} est la valeur désirée dans l'exemple d'apprentissage
y_k est la valeur de sortie calculée par le MLP

Le problème consiste donc à minimiser la valeur de SEC en fonction de l'ensemble des valeurs de pondération des nœuds et des liaisons. Si la valeur du SEC n'est pas optimal, on répète la procédure avec d'autres valeurs initiales des poids (w_{ij}) et d'autres valeurs des biais (b_j).

La phase d'apprentissage d'un réseau de neurone peut donc être résumée par l'algorithme suivant :

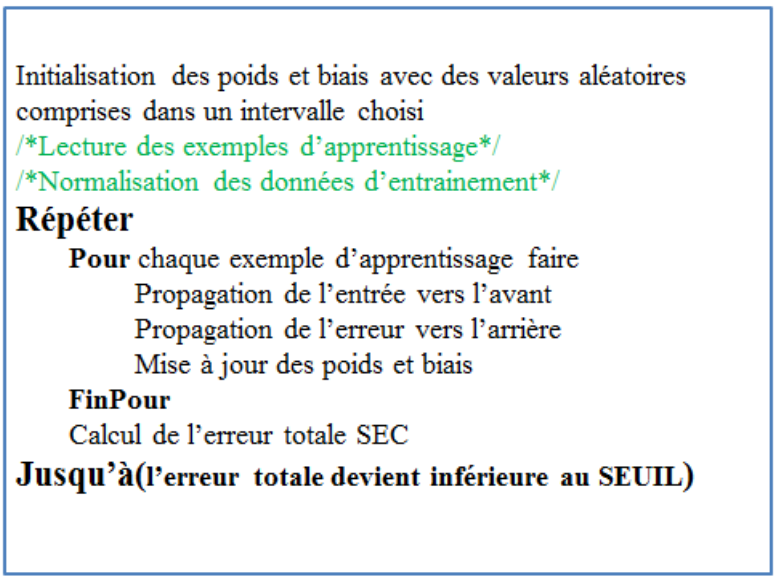


Figure 2.15 – L'algorithme de rétro-propagation de gradient

c) Validation :

La phase de validation vient après la phase d'apprentissage, elle utilise la matrice des poids optimaux et les vecteurs d'entrées qui correspondent aux exemples de la base de test. La validation dans un réseau de neurone correspond à l'opération de propagation d'états.

2.3.8 Avantages et limites des réseaux de neurones

a) Avantages

Les réseaux de neurones artificiels offrent plusieurs avantages, parmi eux on peut citer :[40] [41]

- **Réutilisabilité :** Un réseau de neurones n'est pas programmé pour une application mais pour une classe de problèmes : après une phase d'apprentissage adéquate, il peut traiter de nombreuses tâches.
- **Robustesse :** Les couches cachées du réseau de neurone forment une représentation abstraite des données (concepts), qui permettent de savoir catégoriser des données non traitées lors de l'apprentissage (non prévues).
- **Temps de réponse :** C'est l'un des avantages principaux du réseau de neurones : en effet une fois que le réseau a appris, il peut sortir quasi-instantanément la réponse. En fait, les opérations que fait un réseau de neurones sont très simples du point de vue informatique et peu gourmandes en CPU.
- **Parallélisme :** l'architecture d'un réseau de neurone permet le traitement parallèle et rapide des informations et aide à l'implantation des applications ayant notamment des contraintes temps-réel.

b) Limites

Bien que les réseaux de neurones soient capables d'effectuer beaucoup de tâches, ils souffrent néanmoins de certaines limites dont on peut citer : [40] [41]

- **Choix des attributs :** Avant de passer des exemples à un réseau de neurones, il faut trouver une structure permettant au réseau de bien apprendre les exemples(choix des valeurs initiales des poids du réseau, le nombre de neurones cachés nécessaires,réglage du pas d'apprentissage...etc).
- **Temps d'apprentissage :** Un réseau doit parfois apprendre les exemples plusieurs dizaines de milliers de fois. Si la base d'exemples est énorme, le temps d'apprentissage risque d'être démesuré et le réseau perd son pouvoir de généralisation(il reconnait les données de l'échantillon d'apprentissage, mais plus de nouvelles données similaires).
- **Exploitabilité :** Il existe une grande difficulté pour expliquer les résultats obtenus par le réseau de neurones, car ce dernier fonctionne comme une boite noire (on lui passe des entrées et il ressort un résultat) où les connaissances sont intelligibles pour l'utilisateur.
- **Sur apprentissage :** Un réseau de neurones peut donner une très grande précision face aux exemples d'entrainement, mais se comporte très mal avec les nouveaux exemples. cela représente un phénomène très connu en apprentissage qui est le sur-apprentissage ou l'apprentissage par cœur. Le sur apprentissage donne, généralement, des modèles à faible capacité de généralisation, et par conséquent la mesure de précision n'est pas suffisante pour qualifier les performances de ces modèles.

2.3.9 Domaines d'applications des réseaux de neurones

Depuis leur apparition, les réseaux de neurones ont été largement utilisés dans plusieurs domaines. On peut citer :[42]

- **Traitement d'image :** compression d'images, reconnaissance automatique de caractères et de signatures, reconnaissance de formes dans la lecture automatique des codes postaux, classification...etc.
- **Traitement du signal :** traitement de la parole, l'élimination du bruit et de l'écho, classification...etc.
- **La fouille de données :** extraction des connaissances et les problèmes d'optimisation et de classification.
- **La simulation :** simulation boîte noire, prévisions météorologiques et les estimations de probabilité de succès.
- **L'automatique :** l' identification/modélisation des systèmes non linéaires, la modélisation des procédés industriels et la Robotique.

Il existe d'autres domaines où on utilise différents types de réseaux de neurones selon le problème à résoudre. Voici un tableau qui représente la correspondance entre chaque domaine d'application et le type de réseau de neurones le plus approprié :

Domaine d'application	*Type de RNA*
Reconnaissance de formes	Mlp, Hopefied, Kohonen, PNN
Mémoires associatives	Hopefied, MLP récurrent, Kohonen
Optimisation	Hopefied, ART, CNN
Approximation de fonctions	MLP, RBF
Modélisation et control	MLP, MLP récurrent, FLN
Traitement d'images	CNN, Hopefied
Classification et clustering	MLP, Kohonen, RBF, ART, PNN

Tableau 2.2 – Correspondance type de RNA / Domaine d'application
[43]

2.4 Conclusion

Les réseaux de neurones artificiels sont considérés comme des approches très intéressantes dans le domaine de l'intelligence artificielle. Ils sont connus par leur puissance d'apprentissage et généralisation.

Dans ce deuxième chapitre, nous avons présenté dans la première partie une introduction au domaine de classification des données en détaillant l'architecture des applications basées sur cette notion ainsi que leur catégories avec les techniques utilisées dans chacune.

Dans la deuxième partie, nous avons introduit les définitions de base pour les réseaux de neurones vu que c'est la technique que nous allons utiliser pour réaliser ce travail, et cela en spécifiant leur principe de fonctionnement, leurs types d'apprentissage et leur architecture avec une brève présentation de quelques modèles de ces derniers, où nous avons concentré sur les réseaux de neurones multi-couches(MLP) qui sont les plus adaptés à la classification du trafic réseau, et enfin nous avons terminé par les avantages et les inconvénients de cette technique de classification supervisée ainsi que ses domaines d'application.

CHAPITRE 3

OPTIMISATION COMBINATOIRE

3.1 Introduction

Les problèmes d'optimisation occupent actuellement une place importante dans la communauté scientifique où il est nécessaire de trouver des solutions optimales pour des problèmes très compliqués et difficiles à résoudre. Parmi les méthodes les plus utilisées pour la résolution de tels problèmes, on trouve les métaheuristiques qui englobent un ensemble d'algorithmes capables de résoudre des problèmes assez compliqués en s'inspirant d'analogies avec la physique (recuit simulé), avec la biologie (algorithmes évolutionnaires)ou encore l'éthologie (colonies de fourmis, essaims particulaires).

Dans ce chapitre, nous avons présenter le concept d'optimisation combinatoire et ses différentes techniques, où nous avons concentrer sur les métaheuristiques vu que c'est la méthode d'optimisation que nous allons utiliser pour générer notre modèle de détection d'intrusions. Plus particulièrement, la méthode de recuit simulé et celle de recherche tabou qui sont utilisées dans ce travail conjointement avec le réseau de neurones multicouches pour optimiser les poids de ce dernier afin de perfectionner notre modèle de détection d'intrusions.

3.2 Optimisation Combinatoire

3.2.1 Définition

L'optimisation combinatoire recouvre toutes les méthodes qui permettent de déterminer l'optimum d'une fonction avec ou sans contraintes. En théorie, un problème d'optimisation combinatoire est défini par un ensemble d'instances. A chaque instance du problème est associé un ensemble discret de solutions S, un sous-ensemble X de S représentant les solutions admissibles (réalisables) et une fonction de coût f (ou fonction objectif) qui assigne à chaque solution $s \in$X le nombre réel (ou entier) $f(s)$. Résoudre un tel problème (plus précisément une telle instance du problème) consiste à trouver une solution $s^* \in$ X optimisant la valeur de la fonction de coût f. Une telle solution s^* s'appelle une solution optimale ou un optimum global.[44]

3.2.2 Classification des méthodes d'optimisation combinatoire

La résolution d'un problème d'optimisation combinatoire est réalisée à l'aide des méthodes illustrées dans la figure suivante :

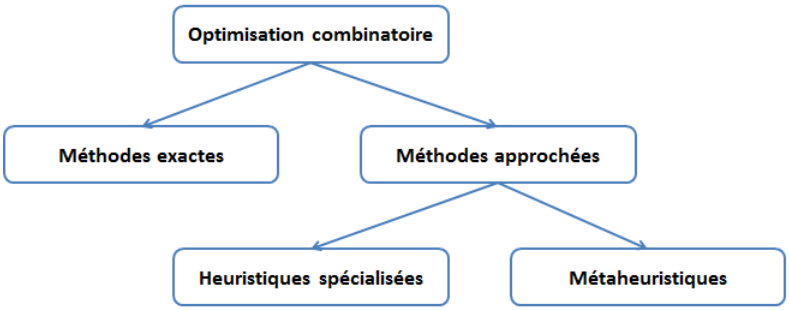


Figure 3.1 – Classification des méthodes d'optimisation
[45]

a) Méthodes exactes

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des solutions de l'espace de recherche. Pour améliorer l'énumération des solutions, une telle méthode dispose de techniques pour détecter le plus tôt possible les échecs (calculs de bornes) et d'heuristiques spécifiques pour orienter les différents choix. Parmi les méthodes exactes, on trouve la plupart des méthodes traditionnelles (développées depuis une trentaine d'années) telles les techniques de séparation et évaluation (Branch and Bound) ou la programmation dynamique. Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable.[21]

b) Méthodes approchées

Lorsque l'on dispose d'un temps de calcul limité ou lorsqu'on est confronté à des problèmes difficiles ou de taille importante, on peut avoir recours aux méthodes approchées, en se contentant de rechercher une solution de bonne qualité. Dans ce cas le choix est parfois possible entre une heuristique spécialisée et une métaheuristique :[46]

- **Heuristique :** Le mot *heuristique* vient du grec « eurisko »qui signifie « je trouve »d'où la célèbre Eureka d'Archimède. Une heuristique, ou méthode approximative, est un algorithme qui fournit rapidement(en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation difficile. Une méthode heuristique est généralement conçue pour un problème particulier, en s'appuyant sur sa structure propre.
- **Métaheuristique :** Le mot *métaheuristique* est dérivé de la composition de deux mots grecs : méta signifiant « au-delà »et heuristique. En effet, ces algorithmes se veulent des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changement profond dans l'algorithme employé.

3.3 Métaheuristiques

3.3.1 Définition

Contrairement aux méthodes exactes, les métaheuristiques permettent de s'attaquer aux instances problématiques de grande taille en fournissant des solutions satisfaisantes dans un délai raisonnable. Il n'y a aucune garantie de trouver des solutions globales optimales ou même des solutions limitées. Les métaheuristiques ont reçu de plus en plus de popularité au cours des 20 dernières années. Leur utilisation dans de nombreuses applications montre leur efficacité pour résoudre des problèmes vastes et complexes. Parmi ces domaines, on peut citer les suivants : [45]

- Conception technique, optimisation de la topologie et optimisation structurelle en électronique et VLSI, aérodynamique, dynamique des fluides, télécommunications, automobile, et la robotique.
- Apprentissage automatique et fouille de données en bioinformatique et biologie computationnelle, et les finances.
- Simulation et identification en chimie, physique et biologie, traitement du signal et de l'image...etc.
- Planification des problèmes de routage, planification des robots, problèmes d'ordonnancement et de production, logistique et transport, gestion de la chaîne d'approvisionnement...etc.

3.3.2 Concepts fondamentaux des métaheuristiques

Les métaheuristiques ne nécessitent pas une connaissance particulière sur les problèmes d'optimisation à résoudre. Il suffit d'associer une ou plusieurs variables à une ou plusieurs solutions (optimum).

Il existe deux points critiques pour toute métaheuristique :[47]

- **Diversification :** Le principe de diversification d'une méthode d'optimisation donnée correspond à sa capacité de parcourir aisément l'espace de recherche pour obtenir des solutions très différentes les unes des autres. Autrement dit, c'est une mécanisme pour une exploration assez large de l'espace de recherche.

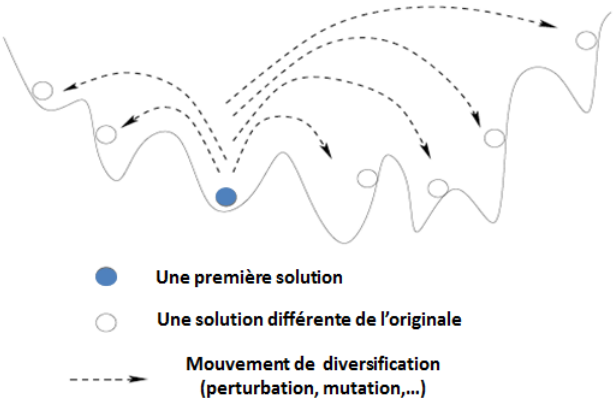


Figure 3.2 – Processus de diversification d'une solution

- **Intensification :** Autant le principe de diversification essaye de déplacer les solutions dans d'autres zones de l'espace de recherche, autant le processus d'intensification vise à forcer une solution donnée à tendre vers l'optimum local de la zone à laquelle elle est attachée. En effet, Elle permet une exploitation de l'information accumulée durant la recherche.

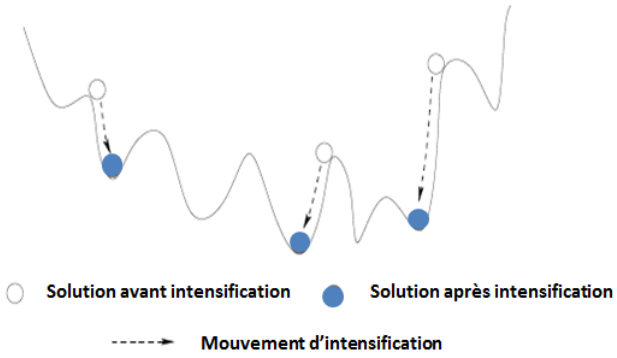


Figure 3.3 – Processus d'intensification de plusieurs solutions

3.3.3 Classification des métaheuristiques

Une manière de classifier les métaheuristiques est de distinguer celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois(méthodes de trajectoire).

Voici une figure qui montre cette classification avec des exemples typiques pour chaque catégorie :

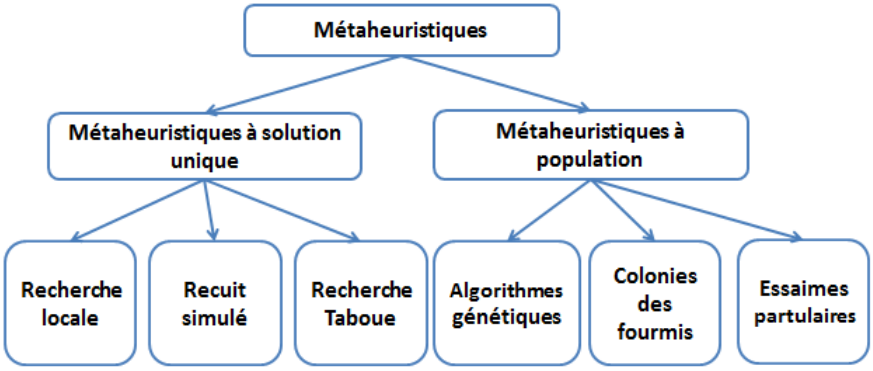


Figure 3.4 – Classification des métaheuristiques

a) Métaheuristiques à solution unique

Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage[48].

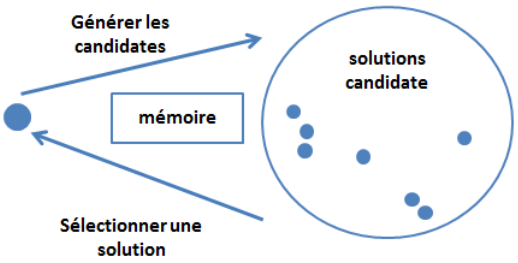


Figure 3.5 – Principe des métaheuristiques à solution unique

Il existe plusieurs méthodes pour cette classification, les plus connues sont :

- **La recherche locale :** La recherche locale itérative est un modèle de recherche locale qui améliore le principe de recherche locale multi-départ (« multiple start local search ») dans lequel des méthodes de descente sont lancées successivement sur des solutions initiales générées aléatoirement pour contrer l'aspect déterministe de la descente. Le principe est simple, une recherche par descente est appliquée sur une solution initiale pour générer une meilleure solution. On applique ensuite une nouvelle recherche par descente sur cette nouvelle solution après l'avoir « perturbée ». La solution obtenue est comparée avec la solution initiale pour savoir si elle la remplace ou non. Tout cela représente une itération de la recherche locale itérative.[47]
- **Le recuit simulé :** Le recuit simulé est une technique d'optimisation de type Monte-Carlo généralisé à laquelle on introduit un paramètre de température qui sera ajusté pendant la recherche. Elle s'inspire des méthodes de simulation de Metropolis (années 50) en mécanique statistique. L'analogie historique s'inspire du recuit des métaux en métallurgie : un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire. Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalente à un optimum global. Cette technique sera détaillée dans la suite de ce chapitre vu que c'est l'une des méthodes utilisées pour réaliser ce travail.[49]
- **La recherche tabou :** La recherche Tabou a été introduite par F. Glover et a montré sa performance sur de nombreux problèmes d'optimisation. Le principe de l'algorithme est le suivant : à chaque itération, le voisinage (complet ou sous-ensemble de voisinage) de la solution courante est examiné et la meilleure solution est sélectionnée. En appliquant ce principe, la méthode autorise de remonter vers des solutions qui semblent moins intéressantes mais qui ont peut être un meilleur voisinage. Cette méthode sera détaillée dans la suite de ce chapitre.[48]

b) Métaheuristique à population

Dans cette classe les métaheuristiques utilisent la notion de population : elles manipulent toutes un échantillonnage de la fonction objectif, via des processus communs. Autrement dit, les méthodes d'optimisation à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. L'intérêt de ces méthodes est d'utiliser la population comme facteur de diversité.

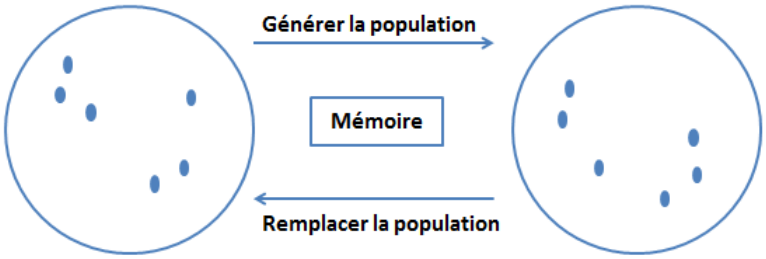


Figure 3.6 – Principe des métaheuristiques à population

Parmi les algorithmes inclus dans cette classification on peut citer les suivants :

- **Les algorithme génétiques :** Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : croisements, mutations, sélection, etc. Introduits par J.H. Holland au début des années 1970. Ils sont appliqués dans diverses domaines : l'économie, l'optimisation de fonctions, la finance, en théorie du contrôle optimal, théorie des jeux répétés et différentiels.
- **Les colonies des fourmis :** L'algorithme de colonies de fourmis a été à l'origine principalement utilisé pour produire des solutions quasi-optimales au problème du voyageur de commerce, puis, plus généralement, aux problèmes d'optimisation combinatoire. On observe, depuis ses débuts, que son emploi se généralise à plusieurs domaines, depuis l'optimisation continue jusqu'à la classification, ou encore le traitement d'image.[50]
- **Les algorithmes à essaim de particules :** Les algorithmes d'optimisation par essaim de particules (PSO) ont été introduit en 1995 par Kennedy et Eberhart comme une alternative aux algorithmes génétiques standards. Ces algorithmes sont inspirés des essaims d'insectes (ou des bancs de poissons ou des nuées d'oiseaux) et de leurs mouvements coordonnés. En effet, tout comme ces animaux se déplacent en groupe pour trouver de la nourriture ou éviter les prédateurs, les algorithmes à essaim de particules recherchent des solutions pour un problème d'optimisation. Les individus de l'algorithme sont appelés particules et la population est appelée essaim. [48]

3.4 Recuit simulé

3.4.1 Principe de base

La méthode de recuit simulé ou « simulated annealing »[51, 52] est un algorithme d'optimisation souvent utilisé lorsque le calcul de la solution optimale exacte demanderait un temps de calcul trop important.

Historiquement, cette technique tire son nom et son inspiration de pratiques issues de la thermodynamique et plus particulièrement, de la façon dont les métaux sont chauffés puis refroidis. Ce processus utilisé en métallurgie pour améliorer la qualité d'un solide cherche un état d'énergie minimale qui correspond à une structure stable du solide. En partant d'une haute température à laquelle le solide est devenu liquide, la phase de refroidissement conduit la matière liquide à

retrouver sa forme solide par une diminution progressive de la température. Chaque température est maintenue jusqu'à ce que la matière trouve un équilibre thermodynamique. Quand la température tend vers zéro, seules les transitions d'un état à un état d'énergie plus faible sont possibles.

Metropolis et al. [53] furent les premiers à implémenter, dès 1953, ce type de principe dans le calcul numérique. Ils utilisent une méthode stochastique pour générer une suite d'états successifs du système en partant d'un état initial donné. Tout nouvel état est obtenu en faisant subir un déplacement (une perturbation) aléatoire à un atome quelconque. Soit ΔE la différence d'énergie occasionnée par une telle perturbation. Le nouvel état est accepté si l'énergie du système diminue ($\Delta E \leq 0s$). Sinon, il est accepté avec une probabilité définie par : $p(\Delta E, T) = e^{(-\Delta E/(C_b * T))}$ où T est la température du système et C_b une constante physique connue sous le nom de *constante de Boltzmann*.

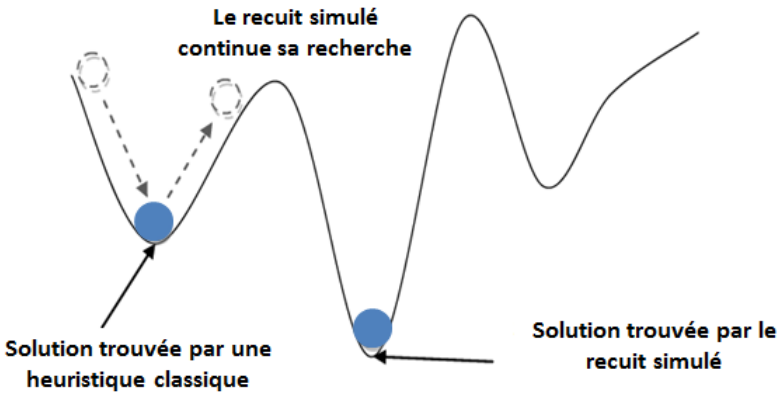


Figure 3.7 – Comparaison entre le recuit simulé et une heuristique classique

3.4.2 Algorithme

Le recuit simulé applique itérativement l'algorithme de Metropolis [1], pour engendrer une séquence de configurations qui tendent vers l'équilibre thermodynamique :

1. l'algorithme a été nommé d'après Nicholas Metropolis, qui avec Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller et Edward Teller rédigea l'article fondateur de 1953, « *Equations of State Calculations by Fast Computing Machine* »proposant l'algorithme pour le cas spécifique de la distribution de Boltzmann. Keith W. Hastings l'étendit au cas plus général en 1970.

engendrer une configuration (solution) initiale s0 de s: S←S0
Initialiser la température T
Répéter
    Voisinnage s' de s
    Calculer ΔE=f(s)-f(s') /* f fonction objective à maximiser
    Si ΔE<0 alors
        s←s'
    Sinon
        tirer u∈[0,1]
            si u< exp(-ΔE/T) alors s←s'
    Fsi
    Réduire la température T (T ←αT) (0.5<α<0.9999)
Jusqu'à(critère d'arrêt)
Retourner(S)

Figure 3.8 – Algorithme de recuit simulé

Une recherche de recuit simulé accepte n'importe quelles nouvelles solutions qui sont évaluées comme des solutions supérieures, mais elle accepte aussi les changements négatifs de qualité avec une probabilité qui dépend de la taille de diminution dans la qualité et la valeur courante de la température. La température commence à une haute valeur, idéalement assez haute que n'importe quelle solution inférieure aura presque une chance de 100 % d'être acceptée, et le processus suit son cours.[54]

3.4.3 Pseudo-code

La figure suivante donne un pseudo-code du recuit simulé :

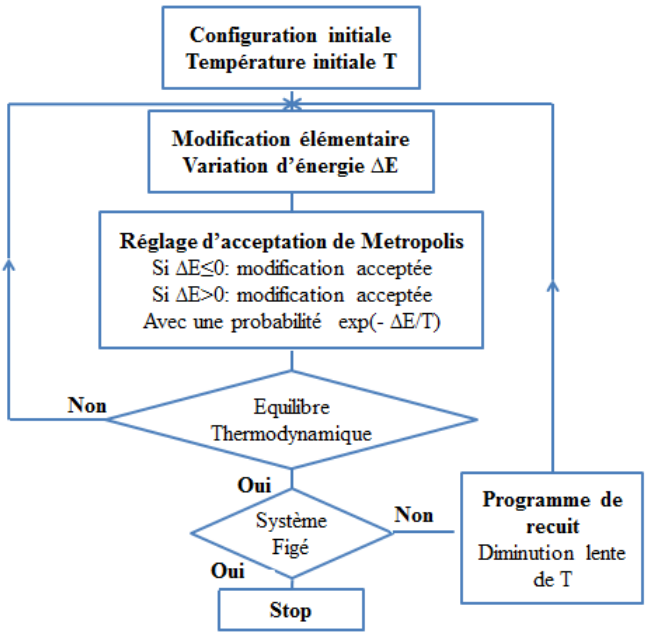


Figure 3.9 – Organigramme de l'algorithme du recuit simulé

3.4.4 Domaines d'application

la méthode du recuit simulé peut être utilisée pour résoudre plusieurs problèmes d'optimisation. En effet, par rapport aux autres algorithmes d'optimisation globale, il est perçu que cette méthode est plus facile à mettre en œuvre et moins encline à être piégée dans les optimums locaux.
Parmi les domaines d'applications les plus connus on peut citer :

- La conception des circuits intégrés (problème de placement et de répartition).
- Le routage des paquets dans les réseaux.
- La segmentation d'images.
- Le problème du voyageur de commerce.
- Le problème du sac à dos.
- ...etc.

3.4.5 Avantages et Inconvénients

Comme pour toute métaheuristique, le recuit simulé offre plusieurs avantages, et au même temps souffre de plusieurs défauts. Dans cette section on cite quelques avantages et inconvénients de cette méthode d'optimisation combinatoire.[55]

a) Avantages

Les principaux avantages de l'algorithme de recuit simulé (SA) peuvent être résumés comme suit :

- il peut traiter des systèmes arbitraires et des fonctions objectives, en termes de modèles hautement non linéaires et multimodaux, données bruitées ou soumises à de fortes contraintes.
- il converge vers un optimum global lorsque le nombre d'itérations tend vers l'infini (assez grand).
- il fournit des bonnes solutions pour la majorité des problèmes d'optimisation.
- il est relativement facile à mettre en œuvre et moins sensible à la taille de problème.
- c'est une technique d'optimisation généralisée, car elle ne repose pas sur toutes les propriétés restrictives du modèle.

b) Inconvénients

Il existe plusieurs défauts, parmi eux on peut citer les suivants :

- nombre important de paramètres(température initiale, taux de décroissance de la température, durée des paliers de la température, critère d'arrêt du programme).
- réglage souvent empirique des paramètres.
- temps de calcul excessif dans certaines application (problème de performance).
- l'impossibilité de savoir si la solution trouvée est optimale.

3.5 Recherche tabou

3.5.1 Principe de base

La méthode de recherche tabou ou « Tabu Search (TS) »[56] est une métaheuristique qui guide une procédure de recherche locale pour explorer l'espace de la solution au-delà de l'optimalité locale.

Cette méthode de recherche est basée sur deux principes : la mémoire adaptative et l'exploration réactive :[57]

- La fonction de mémoire adaptative de TS permet la mise en œuvre de procédures capables d'exploiter l'espace de solution de manière économique et efficace et cela en évitant les retours en arrières. Comme les choix locaux sont guidés par les informations recueillies lors de la recherche, TS contraste avec les conceptions sans mémoire qui s'appuient fortement sur des processus semi-aléatoires qui implémentent une forme d'échantillonnage.
- L'accent mis sur l'exploration réactive dans TS, que ce soit dans un déterministe ou probabiliste mise en œuvre, découle de la supposition qu'un mauvais choix stratégique peut souvent fournir plus d'informations qu'un bon choix aléatoire.

Contrairement aux autres procédures de recherche locales simples, TS est déterministe [2] où, à chaque itération, la meilleure solution de voisinage non interdite de la solution actuelle est sélectionnée, même si elle conduit à une pire solution par rapport à la fonction objectif. TS peut ainsi échapper aux optima locaux. La recherche s'arrête après un nombre fixe d'itérations ou un nombre maximum d'itérations continues sans améliorations de la solution la plus connue.[58]

3.5.2 Algorithme

L'algorithme de recherche tabou peut être résumé comme suit :

```
engendrer une configuration (solution) initiale s0 de s: S←S0
Répéter
    Voisinage s' de s n'appartient pas à la liste tabou
    Calculer ΔE=f(s)-f(s')  /* f fonction objective à maximiser
    Si ΔE<0 alors
        s←s'
    Fsi
    mettre à jour la liste tabou
Jusqu'à(critère d'arrêt)
Retourner(S)
```

Figure 3.10 – Algorithme de recherche tabou

2. Un algorithme est dit déterministe, s'il se comporte toujours de la même façon lorsqu'il est appliqué sur une instance donnée du problème

3.5.3 Pseudo-code

La figure suivante illustre la structure générale de la méthode recherche tabou :

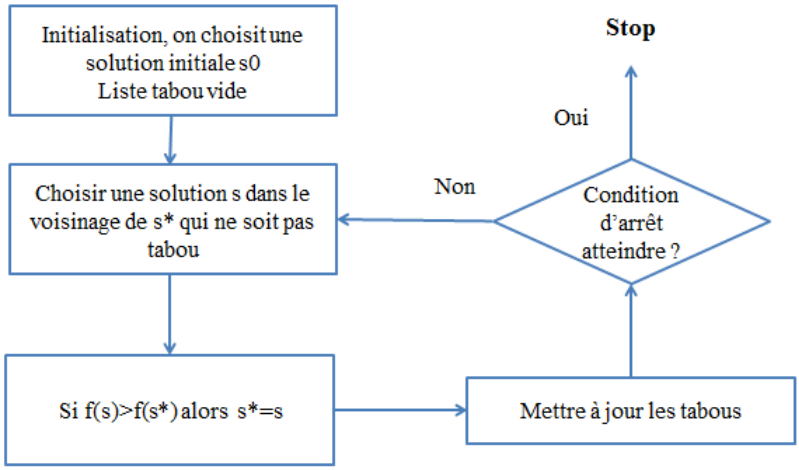


Figure 3.11 – Structure générale de la méthode recherche tabou

3.5.4 Domaines d'application

La méthode de recherche tabou est utilisée pour résoudre des problèmes complexes et/ou de très grande taille (NP-complet). Vu l'importance de cette technique de recherche, il existe plusieurs domaines qui s'intéressent à celui-ci. Parmi eux, on peut citer les suivants :

- Problèmes de transport.
- Planification et ordonnancement.
- Optimisation de la production et gestion des inventaires.
- Optimisation de graphes.
- Logique et intelligence artificielle.
- Télécommunications.
- Optimisation de structures.
- ...etc.

3.5.5 Avantages et Inconvénients

La méthode de recherche tabou fournit plusieurs avantages mais elle souffre aussi de beaucoup d'inconvénients. Les avantages et les inconvénients les plus importants sont les suivants :

a) Avantages

On peut résumer les avantages de la technique de recherche tabou dans deux points principaux qui sont :

- Grande efficacité.
- Fonctionnement simple à comprendre.

b) Inconvénients

Il existe plusieurs inconvénients pour la méthode RT, parmi eux :

- Paramètres peu intuitifs.
- Demande en ressources importantes si la liste des tabous est trop imposante.
- Aucune démonstration de la convergence.

3.6 Conclusion

A travers ce chapitre, nous avons abordé les définitions essentielles à la compréhension de travail du point de vue optimisation. Nous avons pu voir qu'un grand nombre de méthodes d'optimisation existe pour résoudre un problème combinatoire. Ces méthodes peuvent être exactes ou bien approchées où on distingue deux grandes familles : les heuristiques et les métaheuristiques qui sont détaillées dans la deuxième partie de ce chapitre.

Enfin, nous avons présenté deux techniques largement utilisées pour résoudre les problèmes d'optimisation qui sont le recuit simulé et la recherche tabou. Alors, on a donné pour chacune de ces méthodes sa définition, son algorithme, ses domaines d'application et nous avons terminé par ses avantages et inconvénients. Ces méthode seront utilisées dans ce travail pour optimiser les valeurs des poids qui relient les neurones d'une couche à l'autre dans le but d'augmenter la performance de classification du modèle MLP.

CHAPITRE 4

NOTRE MODÈLE DE DÉTECTION D'INTRUSIONS

4.1 Introduction

En fait, les réseaux de neurones sont considérés comme étant l'une des techniques les plus puissantes et les plus utilisées en classification des données, notamment le modèle MLP qu'on a déjà décrit dans le deuxième chapitre. Cette technique qui a la possibilité de classifier les données en classes, chacune comporte les objets les plus similaires entre eux, sera utilisée dans notre modèle pour classifier le trafic du réseau TCP/IP en trafic normal ou en trafic malveillant. La génération de notre modèle est basée sur la base NSL-KDD issue du jeu de données DARPA 1998.

Dans ce chapitre, nous présentons les différentes étapes du développement de notre travail en commençant par une simple description de la base de données NSL-KDD. Où, nous allons détailler les différentes phases de prétraitement qu'on a appliqué sur cette base, notamment la numérisation, la normalisation et la sélection des meilleurs attributs. Ensuite, nous allons présenter les techniques utilisées pour entrainer et tester notre modèle tout en illustrant les techniques d'optimisation que nous avons appliqué pour augmenter la performance de ce dernier.

4.2 Présentation de la base NSL-KDD

4.2.1 Historique

Le dataset NSL-KDD repose sur un autre dataset populaire, le KDD Cup 99, qui est à son tour extrait d'une autre base de données d'évaluation de systèmes de détection d'intrusions, DARPA [1]. Le KDD99 fut créé en 1999 pour une compétition de machine learning [60]. Le but de cette compétition était de classifier correctement des connexions réseau en 5 catégories : normal, déni de service (DoS), sonde réseau (probe), distant à local (R2L – Remote to Local) et utilisateur à root (U2R – User to Root).

NSL-KDD a été créé en 2009 pour résoudre certains problèmes inhérents à KDD Cup 99 [61]. Il reprend ainsi les mêmes données que ce dernier, mais le modifie grandement pour apporter ses corrections. Ainsi, les connexions redondantes ou dupliquées, qui composaient de 75%

1. Le premier événement IDS parrainé par la DARPA a été réalisé par le MIT Lincoln LAB en 1998 [59]. Dans cet événement DARPA, un scénario d'attaque à la base de l'Air-Force est simulé

à 78% du dataset, ont été supprimées.

Figure 4.1 – La relation entre les datasets

4.2.2 Description du dataset NSL-KDD

Bien qu'il soit assez vieux et non une représentation parfaite des réseaux réels existants, il est en continu un indice qui est utilisé pour comparer les systèmes de détection d'intrusions dans les recherches communes. Dans la littérature la plus récente [62, 63, 64], tous les chercheurs utilisent le NSL-KDD comme ensemble de données de référence.[65]

La base de données NSL-KDD contient des enregistrements de connexion TCP/IP (125973 pour l'apprentissage et 22544 pour le test), dont chaque enregistrement est constitué de 41 attributs caractérisant la connexion, et un attribut indiquant la nature de connexion s'il s'agit d'une attaque ou non. Les quatre catégories d'attaques existantes dans cette base sont :

- Dénis de Services
- Probing
- User to Root
- Remote to User

Ces types d'attaque sont détaillés dans la section 1.2.7 et le tableau suivant montre l'ensemble des attaques peuvent être inclut dans chaque type.

La classe	*Types d'attaques*
DoS	Apache2, Back, Land, Mailbomb, Neptune, Pod, Processtable, Smurf, Teardrop, Udpstrom
Probe	Mscan, Ipsweep, Nmap, Portsweep, Saint, Satan
R2L	Ftp_write, Guess_passwd, Imap, Multihop, Named, Phf, Dict, Snmpguess, Spy, Sqlattack, Warezclient, Warezmaster, Xlock, Xsnoop, Guest
U2R	Buffer_overflow, Httptunnel, Loadmodule, Xterm, Perl, Ps, Rootkit

Tableau 4.1 – Regroupement des attaques dans la base NSL-KDD

La distribution des connexions de la base NSL-KDD est illustrée dans le tableau ci-dessous :

Type	*Base d'apprentissage*		*Base de test*	
	Nbr connexions	*Pourcentage*	*Nbr connexions*	*Pourcentage*
Normal	67343	53.46%	9711	43.07%
DoS	45927	36.46%	7591	33.67%
Probe	11656	09.25%	2421	10.74%
R2L	995	00.79%	2754	12.22%
U2R	52	00.04%	67	00.30%
Total	125973	100.0%	22544	100.0%

Tableau 4.2 – Distribution des données dans la base NSL-KDD
[65]

4.2.3 Contenu de la base de données NSL-KDD

Cette base contient quatre groupes de données qui sont : [66][65]

- **KDDTrain^{+}** : qui contient toutes les données d'apprentissage du dataset NSL-KDD.
- **KDDTrain^{+}_20Percent** : qui représente seulement 20% des données d'apprentissage.
- **KDDTest^{+}** : ce groupe de données représente les données du test de la base NSL-KDD.
- **KDDTest^{-21}** : c'est un sous-ensemble du $KDDTest^{+}$ qui n'inclut pas les enregistrements ayant un niveau de difficulté de 21 sur 21.

Type de données	KDDTrain^{+}	KDDTrain^{+}_20Percent	KDDTest^{+}	KDDTest^{-21}
Nbr d' enregistrements	125,973	25192	22,554	11850

Tableau 4.3 – Contenu de la base NSL-KDD

4.2.4 Attributs de la base NSL-KDD

Dans la base NSL-KDD, chaque enregistrement est constitué de 41 attributs qui sont illustrés dans le tableau A.1 de l'annexe. Ces attributs peuvent être classés en quatre catégories :[66]

- **Les attributs de base :**qui sont les attributs de base des connexions TCP, telles que la durée, les hôtes source et destination, port et flag.
- **Les attributs du trafic :**qui sont les attributs calculés à l'aide d'une fenêtre de temps de deux secondes, tels que le nombre de connexions vers la même machine.

- **Les caractéristiques du contenu :**ces attributs sont construits à partir de la charge utile (Data) des paquets du trafic tels que le nombre d'échec de connexion et le nombre d'accès aux fichiers de contrôle.
- **Les caractéristiques de l'hôte :**ce sont les attributs conçus pour évaluer les attaques qui durent plus de deux secondes.

4.3 Processus de génération du modèle de classification

Dans ce travail, où on souhaite élaborer un modèle de détection d'intrusions puissant capable de classifier les connexions TCP/IP en deux catégories : normale ou attaque, on doit passer par les étapes principales que tout modèle de classification doit les suivre. Ces étapes sont résumées dans trois phases principales illustrées dans la figure 4.2 qui sont : le prétraitement, l'apprentissage et enfin la phase de test.

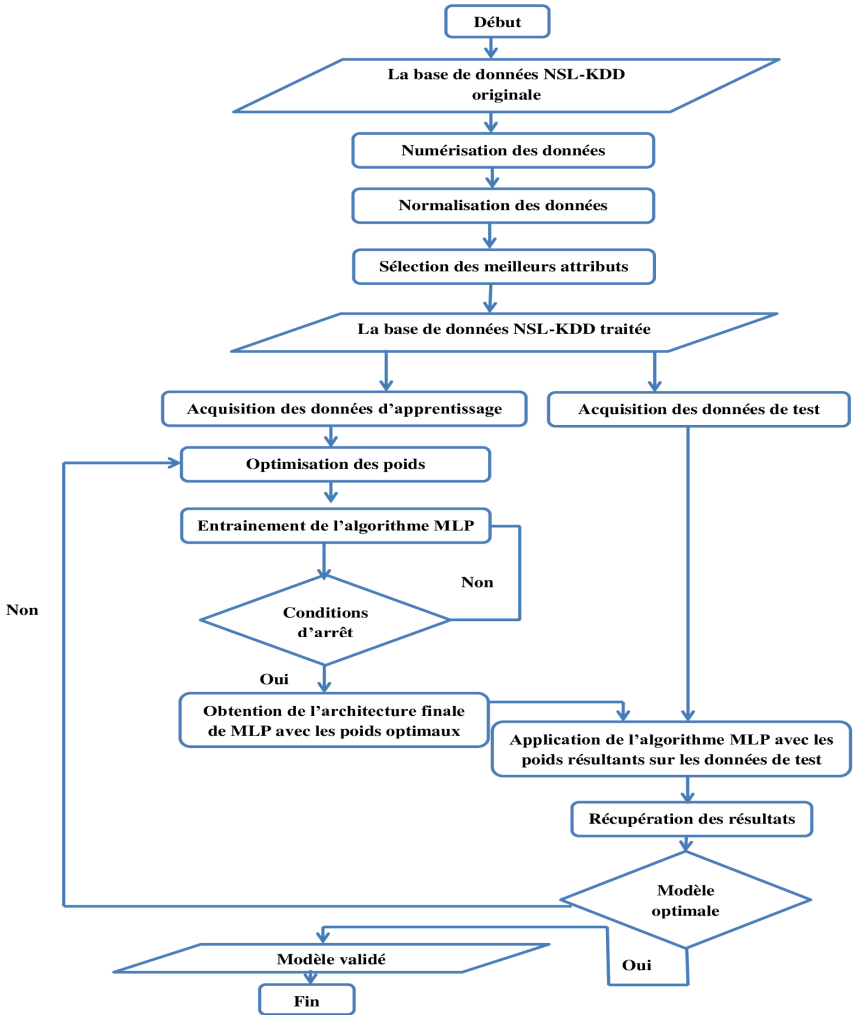


Figure 4.2 – Organigramme de fonctionnement de notre modèle de détection d'intrusion

4.3.1 Prétraitement des données

Le processus de prétraitement modifie le dataset pour le rendre lisible par certains algorithmes, comme le réseau de neurones. Cette phase est donc très importante pour générer un modèle de classification fiable et cohérent. En effet, trois étapes ont été réalisées sur la base NSL-KDD avant de l'exploiter. Ces étapes sont : la numérisation, la normalisation et enfin la sélection des attributs.

a) Numérisation des données

Afin d'adapter le dataset NSL-KDD avec les modèles de réseaux de neurones qui n'acceptent que des attributs numériques, et comme cette base contient 3 attributs alphabétiques qui sont : protocol_type, service et flag parmi ses 41 attributs, il est nécessaire de transformer toutes ces données catégoriques en données numériques via un encodage précis. Dans notre cas, chaque valeur alphabétique est remplacée par son entier équivalent, c'est-à-dire, s'il existe N valeurs possibles pour l'attribut X, ces valeurs sont remplacées par des valeurs entières compris entre 0 et N-1.

Si on prend par exemple le cas de l'attribut *protocol-type* qui peut prendre trois valeurs : tcp, udp ou bien icmp, le résultat de numérisation de cet attribut sera comme suit :

Avant numérisation	*Après numérisation*
TCP	0
UDP	1
ICMP	2

Tableau 4.4 – Exemple de numérisation

b) Normalisation des données

Pour garantir l'efficacité et améliorer les performances du modèle généré, il est très important d'ajuster les valeurs numériques obtenues après la phase de numérisation, puisqu'elles sont très variées et constituent un grand intervalle. Par exemple, certains attributs comme src_bytes et dst_types prennent des grandes valeurs tandis que d'autres comme serror_rate et same_srvrate ne prennent que des petites valeurs. Donc, pour éviter ce genre de problème, on est obligé d'effectuer une opération de transformation sur les données de la base NSL-KDD en utilisant une fonction bien choisie. Dans notre cas, la fonction utilisée est la fonction Min-Max décrite comme suit :

$$val_{nouv} = \frac{val_{anc} - Min_{anc}}{Max_{anc} - Min_{anc}}$$

Où :

— val_{anc} :est la valeur à normaliser.

— val_{nouv} :est la valeur après la normalisation.

— Min_{anc} :est la limite inférieure de l'intervalle à que val_{anc} appartient.

— Max_{anc} :est la limite supérieure de l'intervalle à que val_{anc} appartient.

En appliquant cette formule sur les données de la base NSL-KDD, on obtiendra une base normalisée dont toutes ses valeurs sont compris entre 0 et 1.

c) Sélection des meilleurs attributs

Comme nous avons vu précédemment, la base NSL-KDD contient 41 attributs qui sont censés être dans le cas normal les entrées de notre modèle neuronal. Cependant, si on utilise tout ces attributs, on risque d'affecter les performances de notre modèle de détection d'intrusions en terme de ressources utilisées ainsi que le temps de réponse. Donc, pour garantir l'efficacité de notre modèle, on doit sélectionner les attributs les plus significatives et les plus pertinents parmi tout ces attributs.

Il existe plusieurs techniques pour réaliser cette tâche, parmi-elles, celle qu'on a utilisé dans ce travail où on a choisi les attributs ayant les meilleurs gains d'informations parmi l'ensemble totale d'attributs et cela en suivant les étapes suivantes :

Étape 1 : Calcul de l'entropie pour chaque attribut en utilisant la formule suivante :

$$H(x_i) = -\sum_{j=1}^{n} p(x_j|c_1)\log_2 p(x_j|c_1) + p(x_j|c_2)\log_2 p(x_j|c_2)$$

Où :

— c_1, c_2 : dénotent les deux classes de classification (normale, attaque).

— x_i : représente un attribut.

— x_j : représente une valeur particulière de l'attribut x_i.

— n : dénote le nombre de valeurs de l'attribut x_i.

— p : la probabilité.

Étape 2 : Calcul de gain pour chaque attribut à l'aide la formule suivante :

$$Gain = Entropie_E - H(x_i)$$

Sachant que :

$$Entropie_E = \frac{nb_{normal}}{nb_{connexion}}\log_2(\frac{nb_{normal}}{nb_{connexion}}) + \frac{nb_{attack}}{nb_{connexion}}\log_2(\frac{nb_{attack}}{nb_{connexion}})$$

Où :

— nb_{normal} :présente le nombre des connexions qui sont classifiées comme normal.

— nb_{attack} :présente le nombre des connexions qui sont classifiées comme une tentative d'attaque.

— $nb_{connexion}$:présente le nombre total des connexions dans la base d'apprentissage..

Étape 3 : Élimination des attributs ayant un gain inférieur à un seuil donné (0.5 dans notre cas).

Étape 4 : Génération du modèle de classification en se basant sur l'ensemble des attributs restants après la troisième étape. Les attributs sélectionnées dans notre cas sont les suivants :

Numéro	*Attribut*	*Gain*
3	Service	0.672
4	Flag	0.519
5	Src-bytes	0.827
6	Dst-bytes	0.648
29	Same-srv-rate	0.510
30	Diff-srv-rate	0.519

Tableau 4.5 – Résultat de la phase de sélection d'attributs

4.3.2 Apprentissage et génération du modèle de classification

L'objectif de ce travail est d'établir un modèle de détection d'intrusions comportemental qui se base sur le perceptron multi-couches (MLP). Ce type de réseaux de neurones, qui contient un ensemble de neurones répartis en couches, utilise comme algorithme d'apprentissage celle de rétro-propagation du gradient, et puisque on a besoin de classifier nos données seulement en deux classes qui sont attaque ou normale, la couche de sortie va comporter un seul neurone qui peut prendre deux valeurs :

— 0 : trafic normal.

— 1 : attaque qui regroupe tous les types d'attaques : DOS, U2R, Probe et R2L.

La phase d'apprentissage consiste à entrainer le MLP en utilisant une base de données dite d'apprentissage. Dans notre cas, on utilise la base KDDTrain^{+}_20 Percent qui contient 25192 enregistrements comme base d'apprentissage. Ce processus se répète jusqu'à l'obtention d'une erreur quadratique inférieure à un seuil donné. A ce moment là, les poids optimaux obtenus après les mises à jour effectuées pendant l'entrainement seront utilisés pour calculer la valeur de sortie pour chaque enregistrement de la base de test.

4.3.3 Test et évaluation du modèle généré

Dans cette étape, on veut estimer la qualité de notre modèle de détection d'intrusions par rapport aux autres modèles déjà réalisés en se basant sur une base de données dite de test où toutes les instances sont déjà classées. En effet, pour comparer notre modèle avec les autres on a besoin de calculer quelques métriques lors de la phase de test telles que la matrice de confusion, la précision, le rappel...etc. Ces métriques sont définies comme suit : [67] [68]

- **La matrice de confusion :**est une matrice qui rassemble en lignes les observations et en colonnes les prédictions. Les éléments de la matrice représentent le nombre d'exemples correspondants à chaque cas.

		Classe détectée (prédite)	
		Normale	**Attaque**
Classe réelle	**Normale**	Vrai négatif **TN**(True Negative)	Faux positif **FP** (False Positive)
	Attaque	aux négatif **FN** (False Negative)	Vrai positif **TP** (True Positive)

Tableau 4.6 – Matrice de confusion

- Un vrai négatif (TN) est une activité normale correctement classée lors de test.
- Un faux positif (FP) est une activité normale mal classée pendant le test, c'est-à-dire, elle est considérée comme attaque.
- Un faux négatif (FN) est une attaque non détectée pendant le test, c'est-à-dire, elle est considérée comme trafic normal.
- Un vrai positif (TP) est une attaque correctement détectée.

- **L'exactitude (Accuracy) ou le taux de réussite :** c'est le rapport entre les enregistrements bien classés et la totalité d'enregistrements de test. Il sert à indiquer la façon dont la technique de détection est correcte.

$$Exactitude = \frac{TP + TN}{TP + TN + FP + FN} * 100\%$$

- **Le rappel :** c'est le taux des intrusions correctement détectées par rapport au nombre total d'intrusions. Il est calculé à partir de la formule suivante :

$$Rappel = \frac{TP}{TP + FN} * 100\%$$

- **La précision :**appelée aussi le taux de reconnaissance, est la proportion de prédictions de positifs qui sont en effet des positifs.

$$Prcision = \frac{TP}{TP + FP} * 100\%$$

- **Le taux des fausses alertes (taux des faux positifs) :** est la proportion des négatifs incorrectement détectés comme des positifs.

$$Taux - des - fausses - alertes = \frac{FP}{FP + TN} * 100\%$$

- **F-mesure (Moyenne harmonique) :** c'est une métrique qui combine la précision et le rappel en un nombre compris entre 0 et 1. Elle donne une évaluation de synthèse de la classification.

$$F - mesure = \frac{2 * Prcision * Rappel}{Prcision + Rappel}$$

4.3.4 Optimisation du modèle

Dans cette phase, nous nous intéressons par l'optimisation des poids du réseau en utilisant les algorithmes cités dans le troisième chapitre (le recuit simulé et la recherche tabou). Le principe de ces algorithmes est de générer une solution optimale à partir d'une solution initiale choisie aléatoirement en se basant sur le principe de voisinage qui est implémenté dans notre cas à l'aide de la méthode java *shuffle* qui fonctionne en permutant les éléments d'une liste spécifiée de telle sorte qu'on a obtenu une nouvelle solution extraite de la solution initiale.

a) Objectif

Dans ce travail, nous cherchons à maximiser le taux de réussite du modèle de détection d'intrusions établi, donc la fonction objectif sera comme suit :

$$f = Max(taux - de - russite)$$

b) Solution initiale

Dans notre cas, la solution initiale c'est un tableau des poids comme il est illustré dans cette figure :

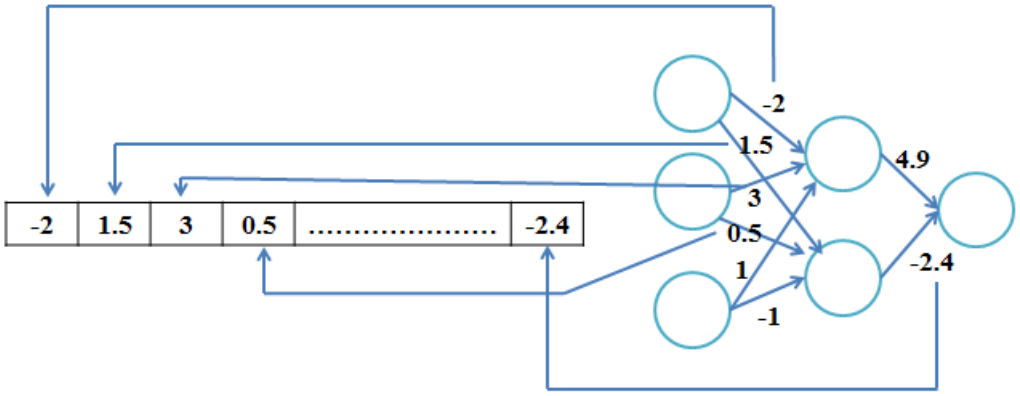


Figure 4.3 – Préparation de la solution initiale

c) Algorithme détaillé

Pour atteindre notre objectif, nous utilisons l'algorithme suivant :

```
Début
    Acquisition des données ;
    Numérisation ;
    Normalisation ;
    Sélection d'attributs ;
    Solution_initiale  S0=P0P1P2…….Pn ;  // n : le nombre des arcs du réseau
    Solution_finale=S0 ;
    f= l'exactitude (S0) ;
    Répéter
        S'= shuffle(Solution_finale) ; // pour la recherche tabou on doit vérifier
                                       que cette solution n'appartient pas à la liste taboue
        ΔE= f(Solution_finale)-f(S') ;
        Si (ΔE <0)
            Solution_finale= S' ;
        Sinon
            Tirer un nombre aléatoire u ∈ [0,1] ;
                Si  u<exp (-ΔE/T) alors ; //T : température initialisée à 1000 ;
                    Solution_finale=S' ;
                FSi
        FSI
        T=T*α ; // refroidissement
    Jusqu'à (critère d'arrêt) ;  // T<0.001(RS) ou bien Nbr_itération>MAX (RT)
    Entrainer _modèle (base d'apprentissage) ; //On utilise le vecteur des poids optimaux
    Tester_modèle(base de test) ;
    Extraction des résultats (matrice de confusion, rappel, exactitude,….) ;
Fin
```

Pour le recuit simulé

Figure 4.4 – Algorithme détaillé

4.4 Conclusion

Dans ce chapitre, nous avons introduit la base NSL-KDD qui a été utilisée pour entrainer et aussi pour tester notre modèle de détection d'intrusions qui est construit à base d'un réseau de neurones multi-couches. Tout d'abord, nous avons présenté l'historique de cette base ainsi que sa description, son contenu et ses différents attributs. Ensuite, nous avons détaillé notre processus de génération du modèle de classification en expliquant ses différentes phases notamment le prétraitement, l'apprentissage et le test.

Enfin, nous avons terminé ce chapitre avec une illustration de la phase la plus importante dans notre travail qui est l'optimisation du modèle. Où, nous avons montré l'utilisation du recuit simulé et recherche tabou dans notre problème de détection d'intrusions à base de réseau de neurones tout en donnant l'algorithme général utilisé.

CHAPITRE 5

IMPLÉMENTATION ET ANALYSE DES RÉSULTATS

5.1 Introduction

L'implémentation de notre modèle de détection d'intrusions consiste à trouver tout d'abord une architecture optimale qui donne la possibilité de détecter les attaques avec un taux de réussite élevé.

Afin d'obtenir les meilleures performances possibles, nous avons effectué plusieurs expérimentations avec des paramètres du réseau différents. Dans ce travail, nous nous intéressons beaucoup plus aux poids qui relient les arcs du réseau, donc nous essayons de trouver ses valeurs optimales qui peuvent perfectionner notre modèle. Les résultats de ces expérimentations ainsi que les valeurs des paramètres utilisées seront détaillés dans ce chapitre.

5.2 Environnement de programmation

Nous avons choisi l'environnement de programmation Eclipse Jee 2019 pour Windows afin d'implémenter les deux modules d'apprentissage et de test de notre modèle de détection d'intrusions. Le choix du langage java a été guidé par les avantages offerts par la programmation orientée objet d'une façon générale.

Les caractéristiques techniques de la machine sur laquelle le modèle est implémenté et testé sont résumées dans le tableau suivant :

Composantes	*Valeurs*
Processeur	Intel®Core™ i5-4300M CPU
Vitesse	2.60 GHz
Mémoire	8.00 Go
Système d'exploitation	Windows 7 64 bits

Tableau 5.1 – Caractéristiques techniques de l'ordinateur utilisé pour l'implémentation

5.3 Test et résultats Expérimentaux

5.3.1 Paramètres de test

Afin d'obtenir la meilleure exactitude(taux de réussite) possible, nous avons effectué plusieurs tests en changeant les paramètres du réseau à chaque fois(taux d'apprentissage, nombre de couches cachées, nombre de neurones par couche et l'intervalle des poids et biais initiaux). Donc, on a fait le test pour 2 couches cachées, 3, 4, 5 et 6 et à chaque fois on a changé le nombre de neurones jusqu'à l'obtention de sa valeur optimale qui donne les meilleurs résultats.

Les valeurs des autres paramètres communs dans les différentes expérimentations soit pour le réseau de neurones ou bien pour le recuit simulé sont indiquées dans le tableau ci-dessous :

Paramètres d'apprentissage		*Paramètres d'optimisation*	
Paramètres	**Valeurs**	*Paramètres*	**Valeurs**
Taux d'apprentissage	0.3	Température initiale	1000
Fonction de transfert	$1/(1+e^{-x})$	Taux de refroidissement	0.997
Intervalle des poids et biais initiaux	[-5,5]	Critère d'arrêt	Lorsque la solution reste inchangée 10 fois successives

Tableau 5.2 – Paramètres d'apprentissage et d'optimisation

5.3.2 Résultats obtenus

Pour montrer l'impact de recuit simulé et recherche tabou sur les performances du réseau de neurone, nous avons effectué trois expérimentations pour les mêmes valeurs des paramètres d'apprentissage. Pour le premier cas, nous nous basons uniquement sur l'opération d'apprentissage pour calculer les valeurs optimales des poids et biais qui seront utilisées pour classifier

les connexions dans le système de détection d'intrusions. Dans le second cas, on utilise le recuit simulé pour trouver la solution optimale qui perfectionne les performances du réseau. Tandis que dans le dernier cas, on utilise la méthode de recherche tabou au lieu de recuit simulé.

Les résultats obtenus pour les trois expérimentations sont montrés dans les tableaux et les graphes suivants :

Premier cas : sans méthodes d'optimisation

- Mesures de performance :

Nbr couches cachées	*Nbr d'attributs*	*Erreur*	*Exactitude %*	*Rappel %*	*Précision %*	*TFA %*	*F_score %*
2	7	0.1	**80.88**	68.52	97.01	2.79	80.31
3		0.1	**78.05**	63.23	97.25	2.35	76.63
4		0.1	**80.22**	67.15	97.26	2.49	79.45
5		0.1	**80.69**	68.49	96.61	3.17	80.16
6		0.1	**80.96**	68.55	97.16	2.64	80.39

Tableau 5.3 – Évaluation des résultats obtenus sans méthodes d'optimisation

Dans ce cas, on constate que la valeur d'exactitude ne dépasse pas 81% et le rappel aussi ne dépasse pas 69%. D'où la nécessité de l'utilisation des méthodes d'optimisation pour améliorer ces performances.

- Matrice de confusion :

	Nombre neurones/couche	*Réels*	*Calculés*	
			Attaque	*Normale*
MLP à 2 couches	2	*Attaque*	8794	4039
		Normale	271	9440
MLP à 3 couches	5	*Attaque*	8115	4718
		Normale	229	9482
MLP à 4 couches	5	*Attaque*	8618	4215
		Normale	242	9469
MLP à 5 couches	5	*Attaque*	8790	4043
		Normale	308	9403
MLP à 6 couches	5	*Attaque*	8798	4035
		Normale	257	9454

Tableau 5.4 – Matrice de confusion de modèle MLP sans méthodes d'optimisation

À partir de cette matrice on peut calculer les métriques précédents en appliquant les formules citées dans la section 4.3.3.

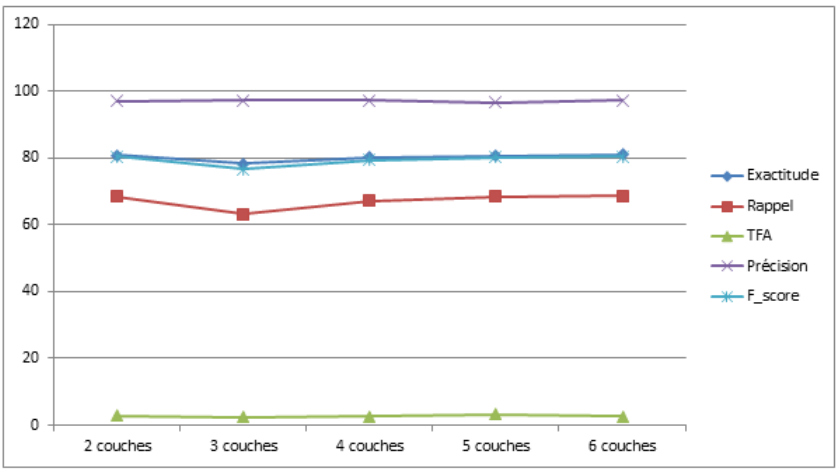


Figure 5.1 – Résultats obtenus sans méthodes d'optimisation

Cette figure montre clairement les résultats obtenus sans méthodes d'optimisation. Ces résultats nous ont incités à rechercher de nouvelles méthodes pour perfectionner notre modèle.

Deuxième cas : avec recherche tabou

- Mesures de performance :

Nbr couches cachées	*Nbr d'attributs*	*Erreur*	*Exactitude %*	*Rappel %*	*Précision %*	*TFA %*	*F_score %*
2	7	0.11	**85.04**	77.16	95.73	4.54	85.45
3		0.11	**84.94**	77.30	95.37	4.95	85.39
4		0.14	**85.96**	78.95	95.62	4.77	86.49
5		0.12	**85.94**	78.12	96.52	3.71	86.35
6		0.11	**86.10**	78.99	95.85	4.51	85.45

Tableau 5.5 – Évaluation des résultats obtenus avec recherche tabou

En utilisant la recherche tabou, les performances du modèle ont été mieux que celles obtenues sans méthodes d'optimisation. Donc, nous avons pu améliorer l'exactitude qui arrive jusqu'à 86,10% et le rappel qui atteint presque 79%.

- Matrice de confusion :

	Nombre neurones/couche	*Réels*	*Calculés*	
			Attaque	*Normale*
MLP à 2 couches	2	*Attaque*	9903	2930
		Normale	441	9270
MLP à 3 couches	5	*Attaque*	9920	2913
		Normale	481	9230
MLP à 4 couches	5	*Attaque*	10132	2701
		Normale	464	9247
MLP à 5 couches	5	*Attaque*	10026	2807
		Normale	361	9350
MLP à 6 couches	5	*Attaque*	10138	2695
		Normale	438	9273

Tableau 5.6 – Matrice de confusion de modèle MLP avec recherche tabou

D'après les résultats cités dans cette matrice, on peut montrer l'efficacité de recherche tabou qui nous a donné la possibilité de détecter plus d'attaque. Par exemple, pour un réseau de neurones de 5 couches cachées, le nombre des attaques bien détectées est 10026 alors que dans le premier cas, il a été 8790 seulement. Donc, nous avons pu détecter plus de 1200 nouvelles attaques.

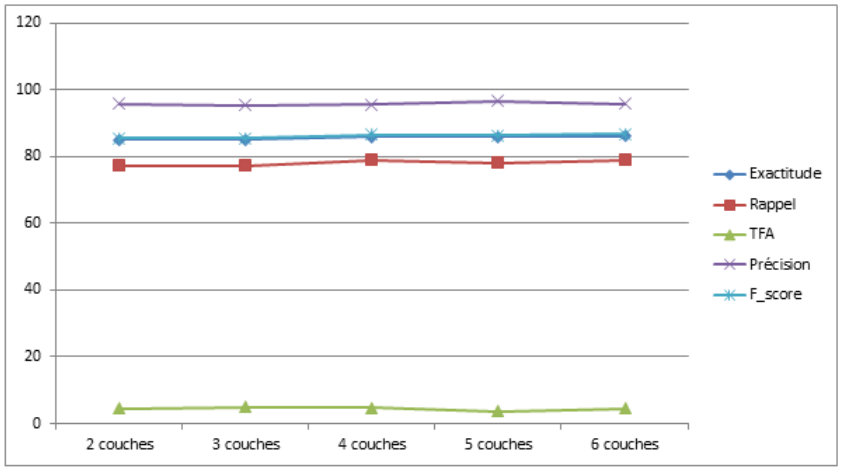


Figure 5.2 – Résultats obtenus avec recherche tabou

Cette figure montre l'amélioration apportée par l'algorithme de recherche tabou.

Troisième cas : avec recuit simulé

- Mesures de performance :

Nbr couches cachées	*Nbr d'attributs*	*Erreur*	*Exactitude %*	*Rappel %*	*Précision %*	*TEA %*	*F_score %*
2	7	0.11	**84.77**	76.27	96.18	3.99	85.08
3		0.11	**84.94**	77.02	95.68	4.59	85.34
4		0.14	**86.04**	79.24	95.45	4.98	86.60
5		0.11	**86.18**	78.57	96.49	3.76	86.62
6		0.12	**86.12**	78.60	96.33	3.95	86.57

Tableau 5.7 – Évaluation des résultats obtenus avec recuit simulé

Sachant que cette technique, recuit simulé, fournit des bonnes solutions pour la majorité des problèmes d'optimisation, dans ce travail, elle nous a donné aussi la meilleure exactitude qui égale à 86,18% pour un réseau de neurones de 5 couches cachées.

- Matrice de confusion :

	Nombre neurones/couche	*Réels*	*Calculés*	
			Attaque	*Normale*
MLP à 2 couches	2	*Attaque*	9789	3044
		Normale	388	9323
MLP à 3 couches	5	*Attaque*	9885	2948
		Normale	446	9265
MLP à 4 couches	5	*Attaque*	10170	2663
		Normale	484	9227
MLP à 5 couches	5	*Attaque*	10084	2749
		Normale	366	9345
MLP à 6 couches	5	*Attaque*	10088	2745
		Normale	384	9327

Tableau 5.8 – Matrice de confusion de modèle MLP avec recuit simulé

Si on prend le même exemple précédent, un réseau de neurones de 5 couches cachées, on trouve que le nombre des attaques bien détectées est 10084. Alors, on a plus de 50 nouvelles attaques ont été détectées par rapport à la méthode de recherche tabou et plus de 1250 par rapport à celle qui n'utilise aucune méthode d'optimisation.

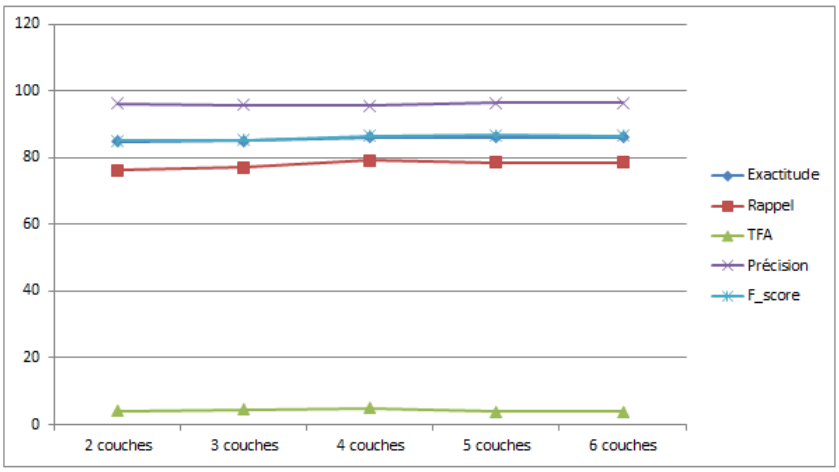


Figure 5.3 – Résultats obtenus avec recuit simulé

Cette figure donne une illustration des résultats obtenus en utilisant le recuit simulé qui nous a donné les meilleures résultats par rapport aux autres approches implémentées dans ce travail.

5.3.3 Analyse et comparaison des résultats

D'après les résultats présentés dans la section précédente, on montre que les performances de notre modèle de détection d'intrusions sont acceptables. Il peut détecter les intrusions y compris les nouvelles attaques avec un taux de réussite arrivant jusqu'à 86,18% et un taux de fausses alertes ne dépassant pas 5%. De plus, on peut montrer l'importance de l'algorithme recuit simulé et la méthode de recherche tabou qui peuvent améliorer les performances du modèle par l'optimisation de l'ensemble des poids et biais du réseau. Cette amélioration est montrée dans les tableaux suivants :

Nbr de couches cachées	*Exactitude*			*Rappel*			*Précision*		
	Sans RS	*Avec RS*	*différence*	*Sans RS*	*Avec RS*	*différence*	*Sans RS*	*Avec RS*	*différence*
2	80.88	84.77	**+3.89**	68.52	76.27	+7.75	97.01	96.18	-0.83
3	78.05	84.94	**+6.89**	63.23	77.02	+13.79	97.25	95.68	-1.57
4	80.22	86.04	**+5.82**	67.15	79.24	+12.09	97.26	95.45	-1.81
5	80.69	86.18	**+5.49**	68.49	78.57	+10.08	96.61	96.49	-0.12
6	80.96	86.12	**+5.16**	68.55	78.60	+10.05	97.16	96.33	-0.83

Tableau 5.9 – Comparaison des performances avec et sans recuit simulé

Nbr de couches cachées	*Exactitude*			*Rappel*			*Précision*		
	Sans RT	*Avec RT*	*différence*	*Sans RT*	*Avec RT*	*différence*	*Sans RT*	*Avec RT*	*différence*
2	80.88	85.04	**+4.16**	68.52	77.16	+8.64	97.01	95.73	-1.28
3	78.05	84.94	**+6.89**	63.23	77.30	+14.07	97.25	95.37	-1.88
4	80.22	85.96	**+5.74**	67.15	78.95	+11.8	97.26	95.62	-1.64
5	80.69	85.94	**+5.25**	68.49	78.12	+9.63	96.61	96.52	-0.09
6	80.96	86.10	**+5.14**	68.55	78.99	+10.44	97.16	95.85	-1.31

Tableau 5.10 – Comparaison des performances avec et sans recherche tabou

Selon les différentes expérimentations effectuées on peut constater l'impact du nombre de couches cachées sur les résultats obtenus. Dans notre cas, les meilleures performances calculées sont celles du réseau de neurones optimisé par le recuit simulé et qui contient 5 couches cachées.

Pour conclure, l'histogramme suivant montre clairement les améliorations que le recuit simulé et la recherche tabou apportent au réseau de neurones :

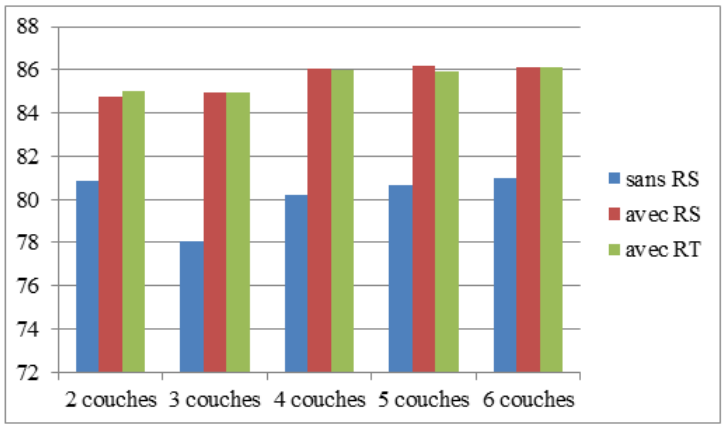


Figure 5.4 – Comparaison des taux de réussite calculés avec et sans méthodes d'optimisation

5.4 Conclusion

Dans ce dernier chapitre, nous avons présenté les résultats obtenus à partir de plusieurs expérimentations que nous avons effectué pour arriver au meilleur taux de réussite possible et les améliorations effectuées grâce aux algorithmes de recuit simulé et recherche tabou qui ont été utilisé comme des techniques d'optimisation sur les poids et les biais du MLP.

Les résultats obtenus ont montré l'efficacité des systèmes de détection d'intrusions basés sur les réseaux de neurones, et plus précisément le perceptron multi-couches, surtout lorsqu'il est combiné avec le recuit simulé ou la recherche tabou qui ont montré leur efficacité dans la résolution des problèmes NP-difficiles.

CONCLUSION GÉNÉRALE

Aujourd'hui, les attaques informatiques représentent un risque réel qui menace les systèmes informatiques et les réseaux des entreprises, ce qui nous a conduit à essayer, dans ce travail, de développer un modèle de sécurité capable de confronter cette menace en détectant toute tentative malveillante soit connue préalablement ou bien récente. Pour accomplir ce but, on a utilisé les réseaux de neurones et plus précisément le perceptron multi-couches, vu que c'est le modèle le plus adapté aux données non linéairement séparables, et cela pour classifier les connexions TCP/IP en deux catégories : normale ou attaque en se basant sur le benchmark NSL-KDD.

En effet, le choix des paramètres du réseau de neurones a une grande influence sur les performances de ces derniers. C'est pour cette raison que nous avons utilisé deux techniques d'optimisation qui sont le recuit simulé et la recherche tabou pour choisir les meilleurs valeurs des poids et biais du réseau qui nous donnent le meilleur taux de réussite possible. Tandis que le choix du nombre de couches, nombre de neurones par couche et le taux d'apprentissage a été fait manuellement en changeant ces valeurs dans chaque expérimentation jusqu'à l'obtention du modèle le plus performant.

Dans ce travail nous avons effectué une étude comparative entre trois modèles : celui qui se base uniquement sur l'algorithme du gradient, celui qui utilise le recuit simulé et enfin celui qui utilise la recherche tabou et cela pour trouver l'ensemble des poids et biais optimaux. Les résultats obtenus montrent l'influence positif des algorithmes d'optimisation sur les performances de réseau de neurones.

Enfin, puisque la recherche scientifique n'a pas des limites et malgré que nous avons obtenu des bons résultats, il existe des améliorations possibles pour perfectionner ce modèle telles que l'utilisation des métaheuristiques pour choisir les autres paramètres du réseau et aussi la réalisation d'une classification multi-classes qui donne de plus le type d'attaque détectée.

ANNEXE A

LES ATTRIBUTS DE LA BASE NSL-KDD

Les détails des attributs sont répertoriés dans le tableau suivant :[69]

N°	Nom de l'attribut	Description
01	duration	Durée de la connexion(nb de secondes)
02	protocol_type	Type du protocol : TCP, UDP ou ICMP
03	service	Service du réseau sollicité sur l'hôte de destination(ftp, http,...etc)
04	flag	Statut de la connexion(REJ, RSTO,...etc)
05	src_ bytes	Nbr d'octets envoyés de la source vers la destination
06	dst_ bytes	Nbr d'octets envoyés de la destination vers la source
07	land	Vaut 1 si la connexion est de/vers le même hôte/port, 0 sinon
08	wrong_ fragment	Nbr de fragments erronés
09	urgent	Nbr de paquets urgents
10	hot	Nbr d'indicateurs hot
11	num_ failed_ logins	Nbr de logins échoués
12	logged_ in	Vaut 1 si login réussi, 0 sinon
13	num_ compromised	Nbr de cas de compromission (compromised condition)
14	root_ shell	Vaut 1 si un shell root est obtenu, 0 sinon
15	su_ attempted	Vaut 1 si une commande super utilisateur est tentée, 0 sinon
16	num_ root	Nbr d'accès en mode root
17	num_ file_ creations	Nbr d'opérations en création de fichiers
18	num_ shells	Nbr de shells lancés
19	num_ access_ files	Nbr d'opérations d'accès aux fichiers de contrôle
20	num_ outbound_ cmds	Nbr de commandes non autorisées dans les sessions ftp
21	is_ host_ login	Vaut 1 si le login fait partie de la list hot, 0 sinon

N°	Nom de l'attribut	Description
22	is_ guest_ login	Vaut 1 si le login fait partie de la list guest, 0 sinon
23	count	Nbr de connexions pour le même hôte
24	srv_ count	Nbr de connexions pour le même service
25	serror_ rate	% de connexions pour le même hôte ayant l'erreur SYN
26	srv_ serror_ rate	% de connexions pour le même service ayant l'erreur SYN
27	rerror_ rate	% de connexions pour le même hôte ayant l'erreur REJ
28	srv_ rerror_ rate	% de connexions pour le même service ayant l'erreur REJ
29	same_ srv _rate	% de connexions pour le même hôte utilisant le même service
30	diff_ srv_ rate	% de connexions pour le même hôte utilisant différents services
31	srv_ diff_ host_ rate	% de connexions pour le même service utilisant différents hôtes
32	dst_ host_ count	Nbr de connexions pour le même hôte
33	dst_ host_ srv_ count	Nbr de connexions pour le même hôte utilisant le même service
34	dst_ host_ same_ srv_ rate	% de connexions pour le même hôte utilisant le même service
35	dst_ host_ diff_ srv_ rate	% de connexions pour le même hôte utilisant différents services
36	dst_host_same_src_port_rate	% de connexions pour le même hôte ayant le port src
37	dst_ host_ srv_ diff_ host_ rate	% de connexions pour le même hôte en provenance de différents hotes
38	dst_ host_ serror_ rate	% de connexions pour le même hôte ayant l'erreur SYN
39	dst_host_srv_serror_rate	% de connexions pour le même hôte et service ayant l'erreur SYN
40	dst_ host_ rerror_ rate	% de connexions pour le même hôte ayant l'erreur REJ
41	dst_ host_ srv_ rerror_ rate	% de connexions pour le même hôte et service ayant l'erreur REJ

Tableau A.1 – Liste des attributs de la base NSL-KDD

BIBLIOGRAPHIE

[1] B. Guttman and E. A. Roback, *An introduction to computer security : the NIST handbook*. Diane Publishing, 1995.

[2] W. Stallings, *Network security essentials : applications and standards*. Pearson, 2016.

[3] O. Lopez and F. Picard, *Cyber-assurance : nouveaux modèles pour quantifier l'impact économique des risques numériques*. No. 3, Association d'économie financière, 2019.

[4] J.-O. Gerphagnon, M. P. de Albuquerque, and M. P. de Albuquerque, "Attaques informatique," *CBPF-NT-007/00,Centre brésilien de recherche physique, Rio de Janeiro – RJ – Brazil*, 2004.

[5] S. Specht and R. Lee, "Taxonomies of distributed denial of service networks, attacks, tools and countermeasures," *CEL2003-03, Princeton University, Princeton, NJ, USA*, 2003.

[6] M. S. Hoque, M. Mukit, M. Bikas, A. Naser, *et al.*, "An implementation of intrusion detection system using genetic algorithm," *arXiv preprint arXiv :1204.1336*, 2012.

[7] S. Paliwal and R. Gupta, "Denial-of-service, probing & remote to user (r2l) attack detection using genetic algorithm," *International Journal of Computer Applications*, vol. 60, no. 19, pp. 57–62, 2012.

[8] R. Akimana, *Introduction à la sécurité informatique*. African Virtual University, 2018.

[9] R. Deal, *Cisco router firewall security*. Cisco Press, 2004.

[10] K. Salah, K. Sattar, Z. Baig, M. Sqalli, and P. Calyam, "Resiliency of open-source firewalls against remote discovery of last-matching rules," in *Proceedings of the 2nd International Conference on Security of Information and Networks*, SIN '09, (New York, NY, USA), p. 186–192, Association for Computing Machinery, 2009.

[11] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems : A survey of common practices," *ACM Comput. Surv.*, vol. 48, Sept. 2015.

[12] P. Biondi, "Architecture expérimentale pour la détection d'intrusions dans un système informatique," *Article de recherche,(Avril-Sptembre 2001)*, 2001.

[13] S. Mignault, *L'audit de sécurité et la protection des organisations*. Université de Montréal, 2009.

[14] G. Hiet, *Détection d'intrusions paramétrée par la politique de sécurité grâce au contrôle collaboratif des flux d'informations au sein du système d'exploitation et des applications : mise en œuvre sous Linux pour les programmes Java*. Theses, Université Rennes 1, Dec. 2008.

[15] N. Dagorn, *Détection et prévention d'intrusion : présentation et limites*. Laboratoire Lorrain de Recherche en Informatique et ses Applications(LORIA), 2006. https ://hal.inria.fr/inria-00084202.

[16] C. Frédéric, "Ids : Intrusion detection systems," *http ://www-igm.univ-mlv.fr/ dr/XPOSE2004/IDS/IDSSnort.html*, 2005.

[17] K. TABIA, "Développement de mécanismes de coopération entre algorithmes d'apprentissage automatique/classification dans un environnement incertain," *Mémoir de magistère, Université Mouloud Mammeri de Tizi ouzou*, 2005.

[18] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," in *Annales des télécommunications*, vol. 55, pp. 361–378, Springer, 2000.

[19] A. Phillip, "Porras and alfonso valdes "live traffic analysis of tcp/ip gateways"," in *Proceeding ISOC Symposium on Network and Distributed System Security, San Diego, CA, March1998*.

[20] R. Graham, "Faq : Network intrusion detection systems," *http ://www. robertgraham. com/pubs/network-intrusion-detection. html*, 2000.

[21] T. Stéphane, *Data mining et statistique décisionnelle : l'intelligence des données*. Editions Technip, 2012.

[22] N. Labroche, *Modelling of the chemical recognition system of ants for the unsupervised classification problem : application to web usage mining*. Theses, Université François Rabelais Tours, Dec. 2003.

[23] N. Monmarché, *Artificial ant based algorithms applied to clustering and optimization problems*. Theses, Université François Rabelais - Tours, Dec. 2000.

[24] G. Cleuziou, *A Clustering method for rules learning and information retrieval*. Theses, Université d'Orléans, Dec. 2004.

[25] Y. Fataicha, *Recherche d'information dans les images de documents*. PhD thesis, École de technologie supérieure - Montréal, 2005.

[26] N. B. Amor, S. Benferhat, and Z. Elouedi, "Réseaux bayésiens naïfs et arbres de décision dans les systèmes de détection d'intrusions," *Technique et Science Informatiques*, vol. 25, no. 2, pp. 167–196, 2006.

[27] R. Marée, *Classification automatique d'images par arbres de décision*. PhD thesis, University of Lìege-Electrical Engineering and Computer Science, 2005.

[28] A. DJEFFAL, *Utilisation des méthodes Support Vector Machine (SVM) dans l'analyse des bases de données*. PhD thesis, Université Mohamed Khider-Biskra, 2012.

[29] P. Wira, "Réseaux de neurones artificiels : architectures et applications," *Cours en ligne, Université de Haute-Alsace*, 2009.

[30] M. Y. Ammar, *Mise en œuvre de réseaux de neurones pour la modélisation de cinétiques réactionnelles en vue de la transposition batch/continu*. PhD thesis, Institut National Polytechnique de Toulouse, 2007.

[31] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks : A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[32] P. Borne, M. Benrejeb, and J. Haggège, *Les réseaux de neurones : présentation et applications*, vol. 15. Editions OPHRYS, 2007.

[33] A. Berkani and Y. Abdeessemed, "Métaheuristique hybride réseaux de neurones artificielspso du recuit simulé pour la commande d'un procédé industriel non-linéaire," *Mémoire de magistère en Électronique de l'université Batna*, 2013.

[34] G. Dreyfus *et al.*, "Les réseaux de neurones," *Mécanique industrielle et matériaux*, vol. 51, 1998.

[35] M. Parizeau, "Réseaux de neurones," *GIF-21140 et GIF-64326*, vol. 124, 2004.

[36] R. GHAYOULA, "Contribution à l'optimisation de la synthèse des antennes intelligentes par les réseaux de neurones," *Université de Tunis El Manar. Thèse de doctorat*, vol. 27, 2008.

[37] S. Gunadiz, *Algorithmes d'intellignece antificielle pour la classification d'attaquer réseau à partir de donnée TCP*. PhD thesis, Université de Boumerdès - M'hamed Bougara, 2011.

[38] S. Aissaoui, *Apprentissage automatique et sécurité des systèmes d'information. Application : un système de détection d'intrusion basé sur les séparateurs à vaste marg (svm)*. PhD thesis, Université d'Oran - Ahmed Ben Bella, 2008.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[40] A. Belgacem, *Classification des signaux EGC avec un système-multi-agent neuronale*. PhD thesis, Universite Abou Bekr Belkaid - Tlemcen, 2012.

[41] Y. HAMMOUCHE, *Comparaison de plusieurs méthodes pour la prédiction de la Charge Electrique Nationale*. PhD thesis, Université Badji Mokhtar - Annaba, 2009.

[42] C. Touzet, *LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME*. Collection de l'EERIE, EC2, 1992.

[43] M. R. Meireles, P. E. Almeida, and M. G. Simões, "A comprehensive review for industrial applicability of artificial neural networks," *IEEE transactions on industrial electronics*, vol. 50, no. 3, pp. 585–601, 2003.

[44] J.-K. Hao, P. Galinier, and M. Habib, "Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes," *Revue d'intelligence artificielle*, vol. 13, no. 2, pp. 283–324, 1999.

[45] E.-G. Talbi, *Metaheuristics : from design to implementation*, vol. 74. John Wiley & Sons, 2009.

[46] I. Boussaid, *Perfectionnement de métaheuristiques pour l'optimisation continue*. PhD thesis, Paris Est, 2013.

[47] J.-C. Boisson, *Modelling and Solving with cooperative metaheuristics : from atom to protein sequence*. Theses, Université Lille 1, Dec. 2008.

[48] L. Jourdan, *Métaheuristiques pour l'extraction de connaissances : Application à la génomique*. PhD thesis, Université des Sciences et Technologie de Lille-Lille I, 2003.

[49] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[50] W. Tfaili, "Conception d'un algorithme de colonie de fourmis pour l'optimisation continue dynamique," *Paris val of Marne university*, 2007.

[51] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[52] S. Kirkpatrick, "Optimization by simulated annealing : Quantitative studies," *Journal of statistical physics*, vol. 34, no. 5-6, pp. 975–986, 1984.

[53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[54] M. O'Keeffe and M. O. Cinnéide, "A stochastic approach to automated design improvement," in *Proceedings of the 2nd International Conference on Principles and Practice of Programming in Java*, PPPJ '03, (USA), p. 59–62, Computer Science Press, Inc., 2003.

[55] T. Sibalija, *Application of simulated annealing in process optimization : A review*, pp. 1–48. 01 2018.

[56] F. Glover, "Future paths for integer programming and links to ar tifi cial intelli g en ce," *Computers operations research*, vol. 13, no. 5, pp. 533–549, 1986.

[57] F. Glover, M. Laguna, and R. Marti, "Principles of tabu search," *Approximation algorithms and metaheuristics*, vol. 23, pp. 1–12, 2007.

[58] Y. Xu, *Metaheuristic approaches for QoS multicast routing problems*. PhD thesis, University of Nottingham Nottingham, 2011.

[59] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, and M. A. Zissman, "Evaluating intrusion detection systems without attacking your friends : The 1998 darpa intrusion detection evaluation," tech. rep., MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 1999.

[60] S. Stolfo, W. Fan, W. Lee, *et al.*, "Kdd-cup-99 task description," *http ://KDD. ics. uci. edu/databases/kddcup99/task.html*, 1999.

[61] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, IEEE, 2009.

[62] N. Paulauskas and J. Auskalnis, "Analysis of data pre-processing influence on intrusion detection using nsl-kdd dataset," in *2017 open conference of electrical, electronic and information sciences (eStream)*, pp. 1–5, IEEE, 2017.

[63] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[64] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[65] Y. Ding and Y. Zhai, "Intrusion detection system for nsl-kdd dataset using convolutional neural networks," in *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, CSAI '18, (New York, NY, USA), p. 81–85, Association for Computing Machinery, 2018.

[66] P. Aggarwal and S. K. Sharma, "Analysis of kdd dataset attributes-class wise for intrusion detection," *Procedia Computer Science*, vol. 57, pp. 842–851, 2015.

[67] A. DJEFFAL, *Cours Fouille de données avancée*. Université Mohamed Khider - Biskra, 2014-2015. www.abdelhamid-djeffal.net.

[68] M. LABONNE, A. OLIVEREAU, and D. ZEGHLACHE, "Automatisation du processus d'entraînement d'un ensemble d'algorithmes de machine learning optimisés pour la détection d'intrusion," 2018. cesar-conference.org.

[69] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

Résumé

Aujourd'hui, et à cause de l'évolution technologique et l'utilisation d'Internet à grande échelle, le fait de tout sécuriser devient une nécessité incontournable et un défi pour la plupart des entreprises. Et vu que les moyens traditionnels de sécurisation sont devenus insuffisants à cause de l'augmentation du nombre et types d'attaques informatiques qui apparaissent presque chaque jour, les chercheurs du domaine de la sécurité informatique s'occupent d'élaborer des outils de sécurité basés sur des notions de l'intelligence artificielle pour détecter les nouvelles attaques. Dans ce travail, on a réalisé un modèle de détection d'intrusions basé sur les réseaux de neurones multi-couches optimisés par le recuit simulé et la recherche tabou en utilisant le benchmark NSL-KDD pour générer et évaluer ce modèle.

Mots clés :intrusions,réseaux de neurones, intelligence artificiel, recuit simulé, NSL-KDD.

Abstract

Today, and due to technological developments and the use of the Internet on a large scale, making everything secure became an unavoidable necessity and a challenge for most companies. And since traditional means of security have become insufficient due to the increase in the number and types of computer attacks that appear almost every day, researchers in the field of computer security are busy developing security tools based on notions of artificial intelligence to detect new attacks. In this work, an intrusion detection model based on multi-layer neural networks optimized by simulated annealing and tabu search was produced using the NSL-KDD benchmark to generate and evaluate this model.

Keywords :intrusion, neural networks, artificial intelligence, simulated annealing, NSL-KDD.