REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – BISKRA

Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie

Département d'informatique

N° d'ordre : RTIC19/M2/2020

Mémoire

présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : Réseaux et technologies de l'information et de la communication

---

Algorithme NSGA-III pour la sélection

Des fonctionnalités utilisée dans un problème de classification déséquilibrée

---

Par :

FERHAT IKRAM

Soutenu le 20 septembre 2020, devant le jury composé de :

Slatnia Sihem	M.C.A	Encadreur

Dédicaces

A mes parents et ma famille ….

A mes Amis …

A tous ceux que j'aime et qui compte pour moi …

Remerciements

Je remercie ***Allah*** *le tout puissant de m'avoir donné le courage jusqu'à l'achèvement de ce mémoire.*

Au terme de ce travail, J'adresse ma profonde gratitude à Madame ***Slatnia Sihem****. Je la remercie pour l'aide inestimable, la disponibilité, la compréhension, la gentillesse, les conseils et les encouragements. J'ai eu le grand plaisir de travailler sous sa direction.*

Je remercie également ***Latreche Imene****, pour l'aide inestimable.*

Un grand merci à tous nos enseignants pour toutes les connaissances qu'ils nous ont inculquées tout au long des cinq années.

Mes remerciements s'adressent également à tous les membres du jury pour l'immense honneur qu'ils nous font en acceptant d'évaluer ce modeste travail.

Mes remerciements vont aussi à tous ceux qui ont contribué de près ou de loin à la concrétisation de ce travail pour leurs conseils, leurs encouragements et leurs soutiens. Qu'ils trouvent tous ici l'expression de ma gratitude.

Abstract

Feature selection can improve classification accuracy and decrease the computational complexity of classification. Data features in intrusion detection systems (IDS) always present the problem of imbalanced classification in which some classifications only have a few instances while others have many instances. This imbalance can obviously limit classification efficiency, but few effort s have been made to address it. In this paper, a scheme for the many-objective problem is proposed for feature selection in IDS, which uses two strategies, namely, a special domination method and a predefined multiple targeted search, for population evolution. It can differentiate traffic not only between normal and abnormal but also by ab-normality type. Based on our scheme, NSGA-III is used to obtain an adequate feature subset with good performance. Experimental results show that NSGA-III can alleviate the imbalance problem with higher classification accuracy for classes having fewer instances. Moreover, it can achieve both higher classification accuracy and lower computational complexity.

Keywords: Feature selection, Many-objective optimization, Network anomaly detection, IDS.

Résumé

La sélection de caractéristiques peut améliorer la précision de la classification et réduire la complexité de calcul de la classification. Les fonctionnalités de données dans les systèmes de détection d'intrusion (IDS) posent toujours le problème d'une classification déséquilibrée dans laquelle certaines classifications n'ont que quelques instances tandis que d'autres en ont de nombreuses. Ce déséquilibre peut évidemment limiter l'efficacité de la classification, mais peu d'efforts ont été faits pour y remédier. Dans cet article, un schéma pour le problème à objectifs multiples est proposé pour la sélection de caractéristiques dans l'IDS, qui utilise deux stratégies, à savoir, une méthode de domination spéciale et une recherche ciblée multiple prédéfinie, pour l'évolution de la population. Il peut différencier le trafic non seulement entre normal et anormal, mais aussi par type de normalité ab. Sur la base de notre schéma, NSGA-III est utilisé pour obtenir un sous-ensemble de fonctionnalités adéquat avec de bonnes performances. Les résultats expérimentaux montrent qu' NSGA-III peut atténuer le problème de déséquilibre avec une précision de classification plus élevée pour les classes ayant moins d'instances. De plus, il peut atteindre à la fois une précision de classification plus élevée et une complexité de calcul moindre.

Mots-clés: Sélection de fonctionnalité, Optimisation à plusieurs objectifs, Détection d'anomalies réseau, IDS.

Abréviations

IDS: Intrusion Detection System.

FA: Feature Selection.

NSGA-III: Non dominated Sorting Genetic Algorithm III.

GA: Genetic Algorithm.

HIDS: Host based Intrusion Detection System.

NIDS: Network based Intrusion Detection System.

KDD: Knowledge Discovery in Databases.

FN: False Negative.

FP: False Positive.

TP: True Positive.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRAL

Le réseau informatique s'est répandu rapidement ces dernières années. Internet est devenu un moyen important d'échange et de partage d'informations dans notre société. Internet à une énorme capacité d'information, une transmission à haut débit, une couverture mondiale, un degré élevé d'ouverture et d'interactivité. Il change donc profondément la manière de travailler et de vivre des gens et influe sur le développement politique, économique, militaire, culturel et technologique.

Avec le développement rapide du réseau informatique Les tentatives d'intrusions aux systèmes d'informations sont rendues plus facile grâce notamment aux failles, en constante croissance, découvertes dans les systèmes informatiques, augmentant de ce fait le risque d'attaques distantes.

La conception des mécanismes de sécurité de manière à empêcher l'accès non autorisé à des ressources système et de données est très importante. Plusieurs mécanismes de protection des réseaux et systèmes informatiques ont été proposés dans la littérature. Des techniques standards telles que les pare-feu sont incapables de prévenir des intrusions, et donc les systèmes de détection d'intrusion (systèmes de détection d'intrusion (IDS : Intrusion Détection System) sont nécessaires, soit pour détecter les atteintes à la sécurité externes ou internes.

L'IDS est un moyen de protéger un réseau informatique. Cette technologie permet aux utilisateurs d'un réseau d'être conscients des menaces entrantes émanant de l'Internet en observant et en analysant le trafic réseau.

D'une part Les caractéristiques des données dans les systèmes de détection d'intrusion (IDS) posent toujours le problème de la classification déséquilibrée dans laquelle certaines Classifications n'ont que quelques instances tandis que d'autres en ont plusieurs. Ce déséquilibre peut évidemment limiter l'efficacité de la classification, mais peu d'efforts ont été

faits pour y remédier. Pour cette raison, les méthodes de sélection peuvent être utilisées pour supprimer les fonctionnalités non pertinentes et redondantes sans réduire les performances.

Dans ce travail nous allons appliquer le métaheuristique NSGA-III (Algorithme génétique de trie non dominé III) au problème de la sélection des caractéristiques pour le sous-ensemble optimal d'attributs qui sera considéré dans l'étape de classification.

Structure du document

Le chapitre 1 : présente les concepts de base et les définitions nécessaires pour comprendre la sécurité informatique et le système de détection d'intrusion et la phase de la sélection des fonctionnalités.

Le chapitre 2 : présente les concepts de base de d'optimisation multiobjectif et l'algorithme NSGA III.

Le chapitre 3 : Présente une conception globale et détaillée du système par l'algorithme NSGA III.

Le chapitre 4 : Décrit les outils et l'environnement utilisés dans ce travail. Il va aussi présenter la réalisation de système et les résultats expérimentaux. Une conclusion à la fin de ce mémoire, permet de présenter un bilan sur l'approche adoptée et les extensions possible

CHAPITRE 1

LA SÉCURITÉ INFORMATIQUE ET LES SYSTÈMES DE DÉTECTION D'INTRUSIONS

1.1. Introduction

L'évolution actuelle des réseaux informatiques ne présente pas que des avantages pour leurs utilisateurs, mais aussi des menaces et des risques. Surtout avec le déploiement parallèle des outils dédiés pour le piratage et l'utilisation malveillante. De ce fait, la protection et l'assurance du bon fonctionnement des réseaux sont devenues une vraie nécessité. Pour cela, il existe plusieurs outils qui ont pour but d'assurer les trois impératifs, à savoir la confidentialité, l'intégrité et la disponibilité des données et des ressources dans un réseau. Parmi ces outils, il y a les systèmes de détection d'intrusions IDS.

1.2. La sécurité informatique

1.2.1. Définition

On peut définir la sécurité informatique comme étant le fait d'assurer le bon fonctionnement d'un système et de garantir les résultats attendus de sa conception.

Autrement dit, La sécurité informatique protège l'intégrité des technologies de l'information comme les systèmes, les réseaux et les données informatiques contre les attaques, les dommages ou les accès non autorisés.

1.2.2. Les services de sécurité

- **Confidentialité :** La confidentialité a été définie par l'Organisation Internationale de Normalisation (ISO) comme « le fait de s'assurer que l'information n'est seulement

accessible qu'à ceux dont l'accès est autorisé », et est une des pierres angulaires de la sécurité de l'information.

- **Intégrité :** Pour interdire ou pour connaitre les modifications et se préserver des pertes d'information [1].
- **Disponibilité :** Qui permet d'assurer un service en toutes circonstances [1].
- **Authentification :** Elle consiste à assurer l'identité d'un utilisateur, c'est-à dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être.
- **Non répudiation :** elle consiste à garantie qu'aucun des correspondants n'est en mesure de refuser la transaction.

1.2.3. Les attaques

1.2.3.1. Définition

Une attaque est définie comme une faute externe créée avec l'intention de nuire, y compris les attaques lancées par des outils automatiques : virus, etc. [2].

1.2.3.2. Les Types d'attaques [3]

- **Les attaques d'accès** : Ils attaquent la confidentialité des systèmes. Parmi les attaques les plus utilisées sont : Snooping, Ecoute, et Interception.
- **Les attaques de modification** : ces types des attaques tentent de modifier les informations du système dans tous les types de modification (modification, insertion, suppression).
- **Les attaques de déni de service** : les attaques par déni de service sont connues par le nom de Dos (Denial of service). Les attaques DoS cherchent à déni l'accès à l'information, aux applications, aux systèmes, et aux communications. Ce sont les attaques les plus utilisées par les pirates.
- **Les attaques de répudiation** : Ce type des attaques est dirigé contre la responsabilité et la réputation de l'entreprise. Ces types d'attaques tentent de donner une fausse vue sur le déroulement du système (négation d'un évènement, ou une transition s'est réellement produits).

1.2.4. Les moyens de sécurisé un réseau

1.2.4.1. Les Firewalls

Un pare-feu qui impose une politique de contrôle d'accès entre deux réseaux. Deux mécanismes sont utilisés : le premier consiste à interdire le trafic, le deuxième à l'autoriser [4].

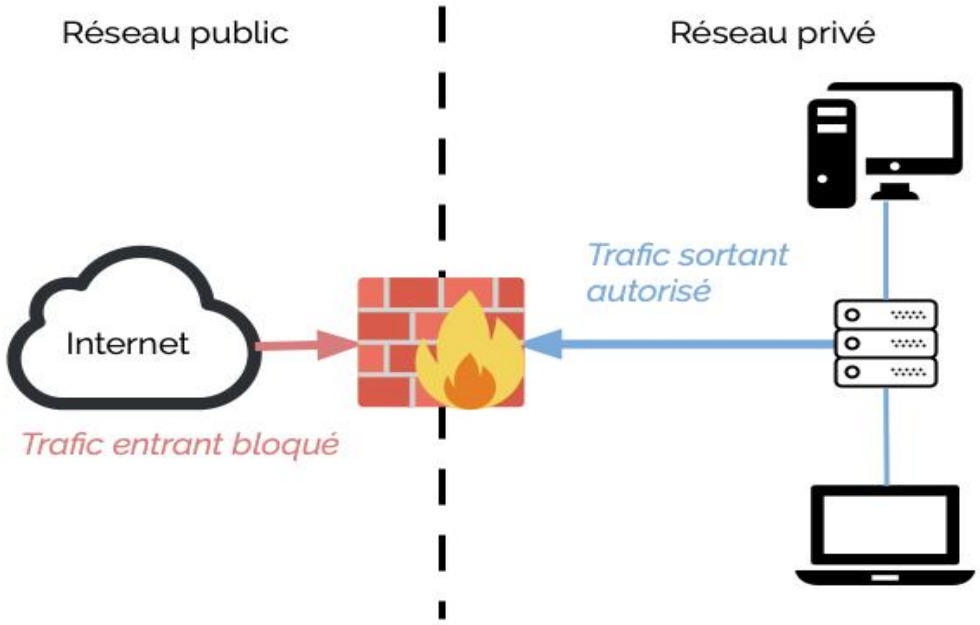

Figure 1.1 : Firewalls.

1.2.4.2. IDS

Le système de détection d'intrusion IDS est un programme ou d'un appareil pour surveiller et divulguer les données ou le comportement des utilisateurs afin d'identifier tout type de menace. Les buts sont nombreux :

• Collecter des informations sur les intrusions.

• Gestion centralisée des alertes.

1.2.4.3. IPS

L'objectif des IPS est de détecter les activités anormales avant de causer une menace ou un impact sur le système.

La technologie IPS «basée sur le contenu», c'est-à-dire qu'elle prend des décisions concernant ce qui est malveillant ou non, en fonction de l'analyse de protocole ou des capacités de correspondance de signature [5].

1.3. Les systèmes de détection d'intrusions

1.3.1. Qu'est-ce que la détection d'intrusion?

Un système de détection d'intrusion (IDS) est un appareil ou une application logicielle qui automatise des surveillances et les processus analysés [6]. Qui surveille un réseau pour détecter toute activité malveillante ou violation de politique. Toute activité malveillante ou violation est généralement signalée ou collectée de manière centralisée à l'aide d'un système de gestion des informations et des événements de sécurité.

Un IDS à quatre fonctions principales : l'analyse, la journalisation, la gestion et l'action [7]:

- Analyse : Analyse des journaux du système pour identifier des intentions dans la masse de données recueillie par l'IDS.
- Journalisation : Enregistrement des évènements dans un fichier de log.
- Gestion : Les IDS doivent être administrés de manière permanente.
- Action : Alerter l'administrateur quand une attaque dangereuse est détectée.

1.3.2. Présentation d'un système de détection d'intrusions

1.3.2.1. Architecture type d'un IDS

Un système de détection d'intrusion se compose de différents modules comme le montre la figure suivante [8] :

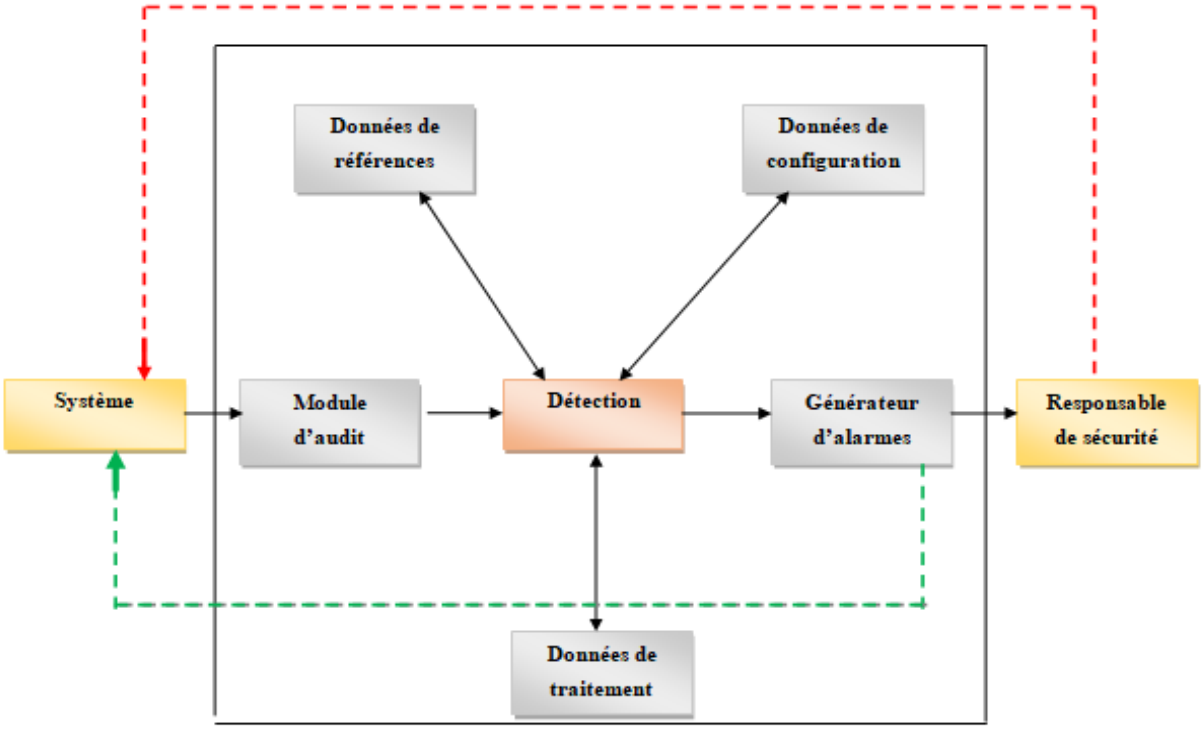

Figure 1.2 : Architecture type en modules d'un IDS.

- **Module d'audit :** Ce module se charge de la collecte et du stockage des données à analyser. Il collecte ces données de différentes sources [9].
- **Module de détection :** Ce module implémente les algorithmes de détection et après Ces algorithmes reçoivent en entrée les données issues de l'audit, produisent en sortie, analyse, les résultats qui seront utilisés par le générateur d'alertes [9].
- **Données de configuration :** Ce module contient des données de configuration, et c'est à travers ce module que le responsable de la sécurité contrôle l'IDS.
- **Données de référence :** Ce module contient les relevés de l'activité normale du système, mais aussi les signatures des intrusions. Ces données sont modifiées automatiquement.
- **Données de traitement** : C'est les données temporaires relatives à l'analyse du système et au traitement des données qui en découlent.
- **Générateur d'alertes** : Il se charge d'émettre des notifications au responsable de la sécurité en cas de détection d'une action malveillante.

1.3.2.2. Les différents types d'IDS

1.3.2.2.1. Les systèmes de détection d'intrusions réseau (NIDS)

Est un système qui analyse le trafic réseau entrant et génère des alertes s'il détecte des paquets malveillants.

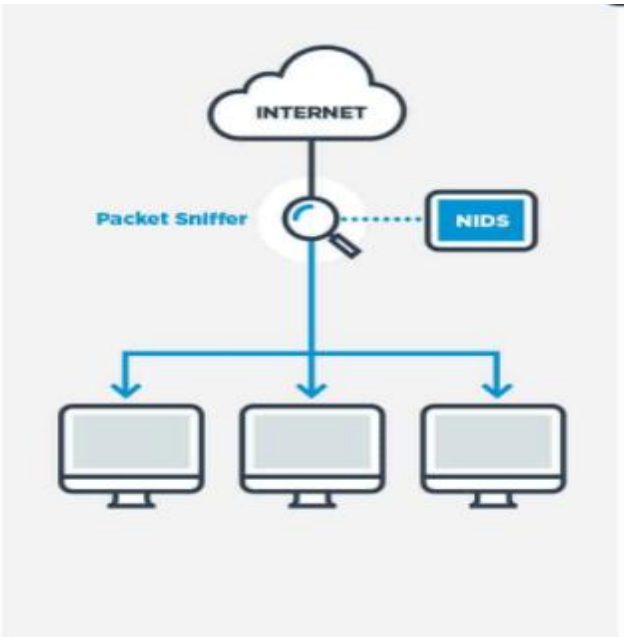

Figure 1.3 : IDS dans un réseau (NIDS).

1.3.2.2.2. Les systèmes de détection d'intrusion de type hôte (HIDS)

Un HIDS se trouve sur la machine, et de ce fait il analyse l'activité se passant sur cette machine et surveille les fichiers importants du système d'exploitation.

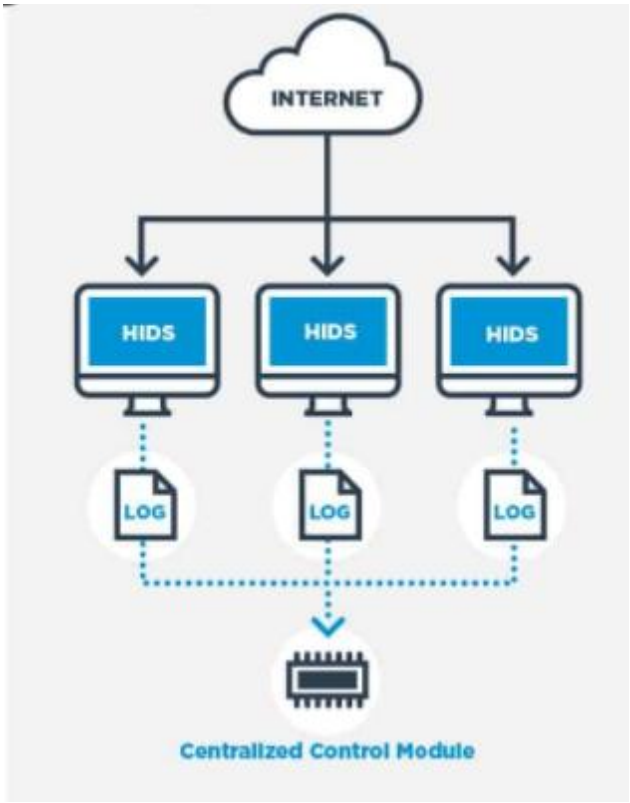

Figure 1.4 : IDS Niveau Système (HIDS).

1.3.2.2.3. Les systèmes de détection d'intrusions hybrides

On dit qu'ils sont « hybrides » du fait qu'ils sont capables de réunir aussi bien des informations provenant de systèmes HIDS que NIDS.

1.3.3. Classification des systèmes de détection d'intrusions

Le schéma suivant montre une classification hiérarchique des IDS [10] :

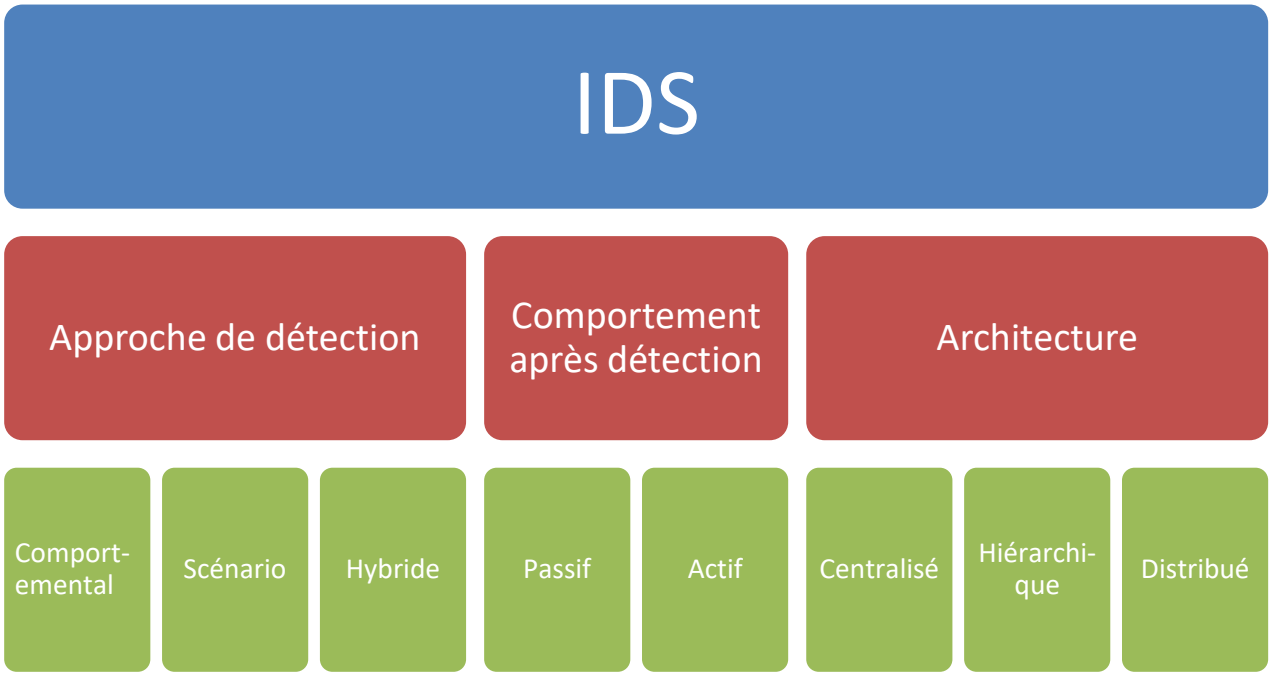

Figure 1.5 : Classification des IDS selon différents critères.

1.3.3.1. L'approche de détection

La recherche d'anomalies de L'IDS se base dans sa sur l'une des deux approches **[11]** :

1.3.3.1.1. L'approche comportementale

La détection d'anomalie essaie de quantifier le comportement habituel ou jugé acceptable et considère le comportement irrégulier comme une intrusion potentielle.

Une phase d'apprentissage : le système apprend le comportement normal d'un sujet (utilisateur ou système). Il crée ainsi "le profil normal" d'un utilisateur à partir des données collectées.

Une phase de détection : le système examine la trace d'audit courante ou l'information réseau et la compare au profil pour vérifier s'il n'y a pas d'activité anormale.

Les inconvénients :

- définir un profil n'est pas une tache facile. Il faut définir les bonnes variables à comparer entre le profil et les données d'audit ou données réseau. Durant une activité système normale, les valeurs des variables choisies doivent rester stables.
- S'il y a une attaque, elles doivent changer de manière significative.

Les avantages :

- Le principal avantage de l'approche comportementale est qu'elle est capable de détecter les attaques sans une connaissance a priori de ces attaques. Ainsi elle peut détecter des attaques inconnues.

1.3.3.1.2. L'approche par scénario

Utiliser une base de données, contenant des spécifications de scénarios d'attaques exploitant des vulnérabilités du système précédemment identifiées, c'est pourquoi on parle des signatures d'attaques et de base de signatures ou bien de modèle de scénarios. On compare ensuite, le comportement observé à cette base, s'il correspond à l'une des signatures alors une alerte sera levée.

Les inconvénients :

- Base de signatures difficiles à construire.
- Pas de détection d'attaques non connues.

Les avantages :

- Le principal avantage de cette approche est qu'elle peut détecter les attaques qui se sont produites par le passé et qu'elle peut prendre en compte les comportements exacts des attaquants potentiels.

1.3.3.1.3. L'approche hybride

D'abord l'approche comportementale cherche à trouver de possibles intrusions et après sont passées à l'approche par signatures pour la mise à jour de sa base.

1.3.3.2. Comportement après détection

- **Passif :** Un IDS passif est un système qui est configuré uniquement pour surveiller et analyser l'activité du trafic réseau et alerter un opérateur des vulnérabilités et attaques potentielles. Il n'est pas capable d'exécuter seul des fonctions de protection ou de correction.
- **Active :** Un IDS actif est un système qui est configuré pour bloquer automatiquement les attaques suspectes en cours sans aucune intervention requise par un opérateur.

1.3.3.3. Architecture

L'architecture IDS la plus courante est:

- **Centralisé** : Dans l'IDS centralisé, les données peuvent être collectées à partir de diverses sources (hôtes ou réseaux) mais est envoyé à un emplacement centralisé où il est analysé. De tels systèmes limiter l'évolutivité du système car il pourrait devenir un goulot d'étranglement sur l'augmentation du nombre de et représentent également un point de vulnérabilité unique.
- **Hiérarchique** : Dans l'IDS hiérarchique, certaines des données collectées à partir de plusieurs hôtes ou d'un seul hôte est transmis à travers les couches et est analysé à des degrés divers à chaque niveau.
- **Distribué :** Dans Distribue IDS, les données sont collectées et analysées sur l'ensemble du réseau suivi et les résultats sont ensuite envoyés à un emplacement centralisé. De tels systèmes sont évolutifs et ne sont pas soumis à un seul point de défaillance.

1.3.4. Les Types d'alarmes

Un IDS prend une entrée et la classe comme une attaque normale ou une attaque. Cette entrée peut prendre la forme de trafic réseau, d'appels système et de leurs séquences, de commandes et de leurs séquences, de comportement de l'utilisateur, de comportement du système et bien d'autres. L'IDS applique son algorithme de détection et classe l'entrée comme une normale ou une attaque et émet des alarmes en conséquence. Voici la liste des différentes alarmes avec leur signification [12].

a) **Faux positif :** L'entrée est normale et est détectée par IDS comme une attaque.
b) **Faux négatif :** On parle de faux négatif lorsqu'une tentative d'intrusion n'est pas détectée.
c) **Vrais positifs :** L'entrée est une attaque qui est détectée par IDS comme une attaque.
d) **Vrais négatifs :** L'entrée est normale, ce qui est détecté par IDS comme un trafic normal.

1.4. La sélection d'attributs

La détection d'intrusion est une tâche de classification. Elle consiste à créer un modèle prédictif permettant d'identifier les instances d'attaque.

D'une part, trop d'éléments ou d'attributs sont susceptibles de contenir une fausse corrélation. La classification des systèmes de détection d'intrusion d'anomalie est un travail complexe. En outre, de nombreuses fonctionnalités peuvent être non pertinentes ou redondantes. Pour cette raison, les méthodes de sélection des fonctionnalités peuvent être utilisées pour supprimer les fonctionnalités non pertinentes sans réduire les performances [12].

1.4.1. Définition

La sélection est le processus d'identification d'un sous-ensemble optimal des caractéristiques pertinentes qui représentent chaque classe mieux que l'ensemble d'origine. Dans de nombreux cas, la sélection des fonctionnalités est plus importante que le choix de l'algorithme de détection. L'utilisation d'un sous-ensemble optimal non seulement améliorer la précision du système, mais également réduire le temps de calcul et les résultats faussement positifs. La sélection ne crée pas de nouvelles fonctionnalités, mais sélectionne celles qui sont pertinentes et non redondantes. L'inclusion de caractéristiques non pertinentes et redondantes dans le processus de classification ou de regroupement peut entraîner une mauvaise généralisation et un sur ajustement [13]. Le processus de sélection des fonctionnalités comprend deux composants principaux: une stratégie de recherche et un critère d'évaluation.

La stratégie de recherche choisie est responsable de la sélection des caractéristiques à prendre en compte dans le sous-ensemble optimal. Le critère d'évaluation attribue un score à

chaque fonctionnalité. Si cette note dépasse un seuil, elle est considérée comme pertinente et incluse dans le sous-ensemble [14].

1.4.2. Etude du processus de sélection d'attributs

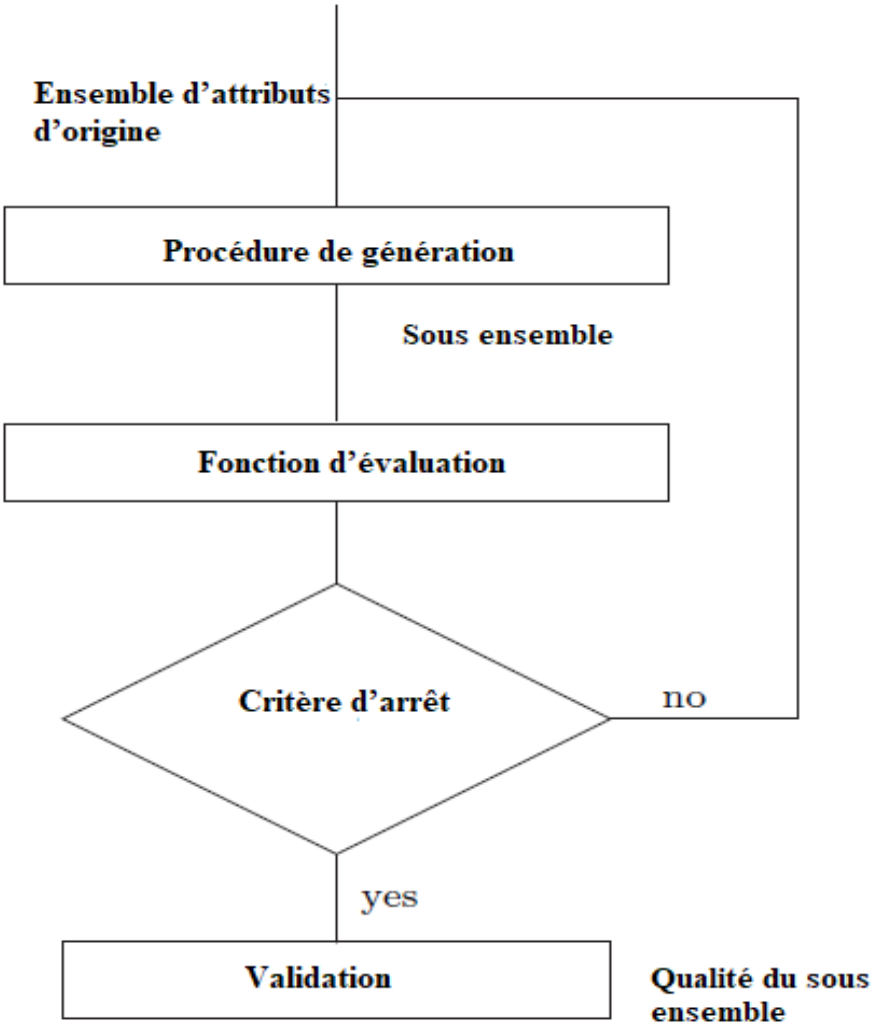

Figure 1.6 : Processus de sélection d'attributs.

1.4.2.1. Procédure de génération

Est un processus de recherche de sous-ensembles de caractéristiques, et le sous-ensemble obtenu sera utilisé comme entrée pour la fonction d'évaluation.

La sélection du sous-ensemble initial est le début de l'algorithme de sélection des caractéristiques et le point de départ du processus de génération de sous-ensemble, qui est divisé en trois catégories :

- Le sous-ensemble initial est vide. En cours de recherche, l'algorithme ajoute les caractéristiques candidates au sous-ensemble candidat une par une. Cette méthode s'appelle la recherche en aval.
- Le sous-ensemble initial est identique au jeu d'entités d'un jeu de données donné. Et il exclut pas à pas les fonctionnalités non pertinentes ou redondantes du sous-ensemble initial dans le processus de recherche, à savoir la recherche arrière.
- Le sous-ensemble initial est généré de manière aléatoire, puis l'entité est ajoutée ou supprimée un par un dans le processus de recherche [16, 17].

1.4.2.2. Fonction d'évaluation

Est utilisée pour évaluer les mérites du sous-ensemble de candidats obtenus par la recherche. Il comparera la valeur d'évaluation avec la meilleure valeur optimale stockée auparavant. Si la valeur d'évaluation est supérieure, le sous-ensemble candidat primaire sera remplacé [18, 19].

1.4.2.3. Critère d'arrêt

Fondés sur la fonction d'évaluation peuvent être la solution optimale trouvée ou ne pouvant pas obtenir une valeur d'évaluation supérieure en augmentant ou en diminuant le nombre de sous-ensembles de caractéristiques [20].

Les critères d'arrêt peuvent être :

- L'atteinte d'un nombre prédéfini d'attributs sélectionnés.
- L'atteinte d'un nombre prédéfini d'itérations.
- Si l'ajout ou la suppression d'attributs ne fournit pas un sous-ensemble d'attributs meilleur.
- Si on atteint un bon sous-ensemble d'attributs, qui répond aux contraintes de temps et de qualité imposées.

1.4.2.4. Procédure de validation

Vérifie l'efficacité de la classification des résultats de la sélection des caractéristiques dans certaines conditions. Cela ne fait pas partie du processus de sélection des fonctionnalités mais est nécessaire dans l'application pratique. La validation consiste généralement à former et tester le sous-ensemble de fonctionnalités dans une sorte de classifier et comparer les résultats de prédiction avec les résultats du jeu de données d'origine ou d'autres résultats de sélection d'entités [21].

1.4.3. Principales approches pour la sélection d'attributs

Il existe trois principales approches utilisées pour générer des sous-ensembles d'attributs : l'approche enveloppante et l'approche filtre et l'approche hybride.

1.4.3.1. L'approche enveloppante

Les enveloppantes ont été introduits par John et al. en 1994. La méthode enveloppante utilise un algorithme d'apprentissage pour évaluer le sous-ensemble de fonctionnalités sélectionnées.

L'algorithme d'apprentissage est utilisé comme boîte noire pour la recherche. La fonction objectif, une équation est utilisée pour évaluer un sous-ensemble de caractéristiques à l'aide de leur précision prédictive, qui est appelée taux de détection. Il s'agit d'une méthode de rétroaction basée sur deux éléments: la recherche et évaluation.

Le composant de recherche génère des réglages de paramètres qui sont ensuite évalués à l'aide du composant d'évaluation. Il utilise un algorithme d'apprentissage pour évaluer l'utilité des fonctionnalités et produit ainsi de meilleurs sous-ensembles de fonctionnalités.

Les caractéristiques sont sélectionnées en fonction de la précision du classificateur. Cette évaluation est faite par un calcul d'un score, par exemple un score d'un ensemble sera un compromis entre le nombre de variables éliminées et le taux de réussite de la classification sur un fichier de test. L'appel de l'algorithme de classification est fait plusieurs fois à chaque évaluation (c'est-à-dire à chaque sélection d'une variable, nous calculons le taux de

classification pour juger la pertinence d'une caractéristique) car un mécanisme de validation croisée est fréquemment utilisé.

Cependant, trois raisons font que les enveloppantes ne constituent pas une solution parfaite.

D'abord, ils n'apportent pas vraiment de justification théorique à la sélection et ils ne nous permettent pas de comprendre les relations de dépendances conditionnelles qu'il peut y avoir entre les variables.

D'autre part la procédure de sélection est spécifique à un algorithme de classification particulier et les sous ensembles trouvés ne sont pas forcément valides si nous changeons de méthode d'induction. Finalement, c'est l'inconvénient principale de la méthode, les calculs devient de plus en plus très longs, voir irréalisables lorsque le nombre de variables est très grand. [22].

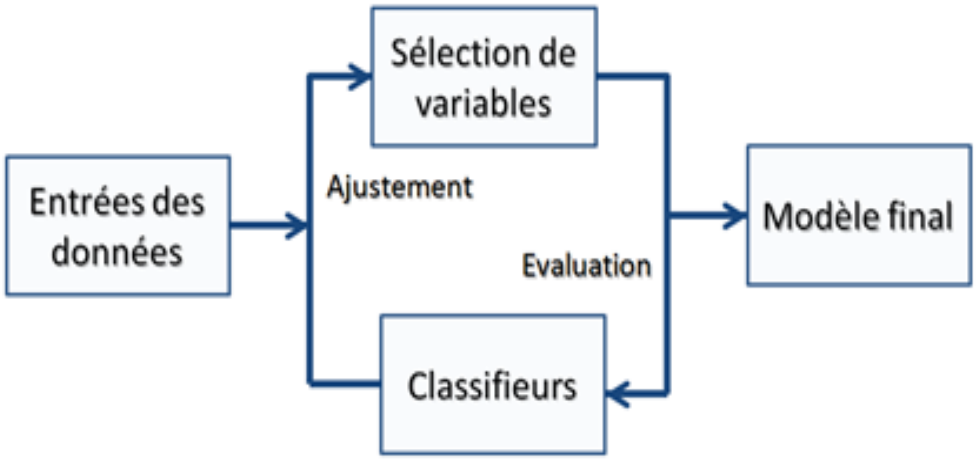

Figure 1.7 : L'approche enveloppante.

1.4.3.2. L'approche par filtre

Comme son nom l'indique, la méthode de filtrage filtre les fonctionnalités importantes qui ont moins d'options dans l'analyse des données.

Il n'utilise aucun algorithme d'apprentissage pour évaluer les caractéristiques [23]. Les entités sélectionnées sont évaluées en fonction des caractéristiques des données telles que

la distance, la cohérence, la corrélation et les mesures d'informations dans l'espace des entités.

La méthode de filtrage sélectionne des sous-ensembles avec un plus grand nombre de fonctionnalités, parfois même toutes les fonctionnalités, donc un seuil approprié est essentiel pour choisir un sous ensemble [22].

Figure 1.8 : L'approche filtre.

1.4.3.3. L'approche hybride

Les méthodes hybrides combinent la méthode du filtre et la méthode enveloppante et tirent parti des deux [24,25].Le mécanisme hybride se déroule généralement en deux étapes.

Dans un premier temps, les fonctionnalités candidates sont prétraitées par des méthodes de filtrage et les fonctionnalités non pertinentes sont supprimées. Ainsi, la dimension de l'ensemble de données pourrait être réduite. Ensuite, les méthodes hybrides sélectionnent les caractéristiques par les méthodes d'encapsulation et l'algorithme d'apprentissage de la classification est utilisé pour évaluer les sous-ensembles sélectionnés [26, 27].

1.4.4. Comparaison entre les approches

modèle	avantages	inconvénients
Filtre	rapide, évolutif, indépendant du classificateur, meilleure complexité de calcul	ignore l'interaction avec le classificateur
Enveloppante	simple, interagit avec le classificateur, les modèles présentent des dépendances, une bonne précision de classification, réduisent les coûts de calcul.	intensif en calcul
Hybride	interagit avec le classificateur, les modèles avec des dépendances de fonctionnalités, une meilleure complexité de calcul.	sélection dépendante du classificateur.

Tableau 1.1 : Comparaison entre les approches.

1.5. Problématique

La sélection des fonctionnalités peut améliorer la précision de la classification et diminuer la complexité de calcul de la classification. Les caractéristiques des données dans les systèmes de détection d'intrusion (IDS) posent toujours le problème de la classification déséquilibrée dans laquelle certaines classifications n'ont que quelques instances tandis que d'autres en ont plusieurs.

De plus, il existe de nombreuses redondances dans l'ensemble de fonctionnalités qui ralentissent le processus de calcul et même réduisent la précision de la classification. Pour éviter la redondance dans l'ensemble de fonctionnalités, la sélection de fonctionnalités pour obtenir un sous-ensemble de fonctionnalités optimal est nécessaire dans IDS.

Le sous-ensemble de fonctionnalités optimal dans un espace de dimension inférieure peut offrir différents avantages:

1) La malédiction de la dimensionnalité peut être évitée car les données sont situées dans un espace de dimension inférieure.

2) L'efficacité de calcul peut être améliorée en raison du plus petit nombre de fonctionnalités.

3) La visualisation des données devient plus intuitive en raison des espaces de dimension inférieure.

1.6. Travaux connexes

- **An improved NSGA-III algorithm for feature selection used in intrusion detection [28].**

Objectif : Dans cet article, un schéma pour le problème à plusieurs objectifs est proposé pour effectuer la sélection de caractéristiques pour IDS.

En outre, sur la base de NSGA-III, un algorithme amélioré appelé I-NSGA-III est proposé qui utilise la préservation de niche de guide qui utilise soit la sélection de biais basée sur les probabilités pour surmonter le problème de déséquilibre ou la sélection d'ajustement pour supprimer les fonctionnalités redondantes. La sélection de biais sélectionne l'individu avec le plus petit nombre de fonctions sélectionnées. Fit-sélection sélectionne l'individu avec le poids total maximum de ses objectifs.

- **A novel feature selection approach based on the cuttlefish optimization [29].**

Objectif : Cet article présente une nouvelle approche de sélection des fonctionnalités basée sur l'algorithme d'optimisation du poisson seiche qui est utilisé pour les systèmes de détection d'intrusion (IDS).

Le modèle proposé utilise l'algorithme de la seiche (CFA) comme stratégie de recherche pour déterminer le sous-ensemble optimal de caractéristiques et le classificateur d'arbre de décision (DT) comme jugement sur les caractéristiques sélectionnées qui sont produites par le CFA. L'ensemble de données KDD Cup 99 est utilisé pour évaluer le modèle proposé. Les résultats montrent que le sous-ensemble de fonctionnalités obtenu en utilisant CFA donne un taux de détection et un taux de précision plus élevés avec un taux de fausses alarmes inférieur, par rapport aux résultats obtenus en utilisant toutes les fonctionnalités.

- **A new feature selection IDS based on genetic algorithm and SVM [30].**

Objectif : Cet article a proposé un IDS basé sur les anomalies utilisant un algorithme génétique et une machine à vecteur de support (SVM) avec une nouvelle méthode de sélection des fonctionnalités. Le nouveau modèle a utilisé une méthode de sélection des fonctionnalités basée sur la génétique avec une innovation dans la fonction de remise en forme, réduire la dimension des données, augmenter la détection positive vraie et diminuer simultanément la détection de faux positifs. De plus, le temps de calcul pour la formation aura également une réduction remarquable. Les résultats montrent que la méthode proposée peut atteindre simultanément une grande précision et un faible taux de faux positifs (FPR), même si elle avait été précédemment réalisée séparément dans des études antérieures. Cette étude propose une méthode qui permet d'obtenir des caractéristiques plus stables par rapport à d'autres techniques. L'expérience et l'essai de modèle proposés sur les ensembles de données KDD CUP 99 et UNSW-NB15. Les résultats numériques et la comparaison avec d'autres modèles ont été présentés.

- **Intrusion detection model using machine learning algorithm on Big Data environment [31].**

Objectif : Cet article présente le modèle Spark-Chi-SVM pour la détection d'intrusion. Dans ce modèle, nous avons utilisé ChiSqSelector pour la sélection des fonctionnalités et construit un modèle de détection d'intrusion en utilisant le classificateur de machine à vecteur de support (SVM) sur la plate-forme Apache Spark Big Data. Nous avons utilisé KDD99 pour former et tester le modèle. Dans l'expérience, nous avons introduit une comparaison entre le classificateur Chi-SVM et le classificateur Chi-Logistic Regression. Les résultats de l'expérience ont montré que le modèle Spark-Chi-SVM a de hautes performances, réduit le temps de formation et est efficace pour le Big Data.

- **Novel Multi-Objective Artificial Bee Colony Optimization for Wrapper Based Feature Selection in Intrusion Detection [32].**

Objectif : Cette étude propose une nouvelle approche basée sur la colonie d'abeilles artificielles à objectifs multiples (ABC) pour la sélection des caractéristiques, en particulier

pour les systèmes de détection d'intrusion. L'approche est divisée en deux étapes: générer les sous-ensembles de caractéristiques du front de Pareto des solutions non dominées dans la première étape et utiliser l'hybride ABC et l'optimisation des essaims de particules (PSO) avec un réseau neuronal à action directe (FFNN) comme classificateur pour évaluer les sous-ensembles de fonctionnalités dans la deuxième étape. Ainsi, l'approche proposée comprend deux étapes: (1) l'utilisation d'une nouvelle technique de sélection de caractéristiques appelée sélection de caractéristiques ABC multi-objectifs pour réduire le nombre de caractéristiques des données de trafic réseau et (2) l'utilisation d'une nouvelle technique de classification appelée hybride ABC-PSO FFNN optimisé pour classer les données de sortie de l'étape précédente, déterminer un paquet d'intrus et détecter les intrus connus et inconnus.

- **An Intrusion Detection System Based on NSGA-II Algorithm [33].**

Objectif : Dans cet article, une méthode est proposée pour identifier les attaques dans IDS. Cette méthode génère des ensembles de règles pour le système de détection d'intrusion à l'aide d'un algorithme génétique de tri non dominé (NSGA-II). NSGA-II est un type d'algorithmes génétiques multi-objectifs. Cette méthode prend en compte les caractéristiques de connexion informatique et définit deux fonctions fitness différentes pour générer les règles. L'avantage de cette méthode par rapport aux méthodes précédentes qui appliquaient l'algorithme évolutionnaire. Comme certaines méthodes appliquaient une fonction de fitness ou convertissaient objectifs à objectif unique, ils ont perdu de nombreuses fonctionnalités.

1.7. Conclusion

Dans ce chapitre, nous avons vu un aperçu de la sécurité du réseau. Ensuite, nous avons abordé les systèmes de détection d'intrusions. Ensuite, nous avons abordé le problème de sélection d'attributs qui consiste à essayer de trouver des sous-ensembles d'attributs pertinents.

CHAPITRE 2

MÉTAHEURISTIQUES D'OPTIMISATION, ALGORITHME NSGA-III

2.1. Introduction

La plupart des problèmes d'optimisation pratiques sont multi-objectifs dans la nature. La nécessité d'optimiser plusieurs objectifs à la fois a été étudiée pour ans, et divers algorithmes ont été proposés.

Le résultat souhaité est un ensemble non dominé de solutions proches du véritable front Pareto-optimal, au lieu d'une solution optimale unique.

L'ensemble de solutions non dominées nous donne la possibilité de prendre une décision appropriée pour choisir une seule solution préférée suivant l'exécution de l'algorithme et fournit des informations utiles sur les solutions mal adaptées aux différentes préférences.

De plus, le décideur peut comparer les compromis entre les différentes solutions et donc justifier son / sa choix.

2.2. Heuristique

Une heuristique est une méthode, conçue pour un problème d'optimisation donné, qui produit une solution non nécessairement optimale lorsqu'on lui fournit une instance de ce problème [34].

2.3. Métahuristiques

Les métaheuristiques sont représentées essentiellement par les méthodes de voisinage comme le recuit simulé et la recherche tabou, et les algorithmes évolutifs comme les algorithmes génétiques et les stratégies d'évolution. Grâce à ces métaheuristiques, on peut proposer aujourd'hui des solutions approchées pour des problèmes d'optimisation classiques de plus grande taille et pour de très nombreuses applications qu'il était impossible de traiter auparavant [35].

2.4. Optimisation

Un problème d'optimisation en général est défini par un espace de recherche S et une fonction objective $\boldsymbol{f}$. Le but est de trouver la solution $\boldsymbol{s* \in S}$ de meilleure qualité $\boldsymbol{f(s*)}$. Suivant le problème posé, on cherche soit le minimum soit le maximum de la fonction *f*. L'équation (2.1) montre bien l'aspect de minimisation, maximiser une fonction *f* étant équivalent à minimiser *f*.

$$\boldsymbol{s}=\mathbf{mi}(\boldsymbol{f}(\boldsymbol{s}) \setminus \boldsymbol{s}\in\boldsymbol{S}) \qquad (2.1)$$

De plus, un problème d'optimisation peut présenter des contraintes d'égalité et/ou d'inégalité sur les solutions candidates $\mathbf{s \in S}$, être multi objectif si plusieurs fonctions objectifs doivent être optimisées.

Il existe de nombreuses méthodes déterministes (exactes) qui permettent de résoudre certains problèmes d'optimisation en un temps fini. Néanmoins, ces méthodes sollicitent que la fonction objectif présente un certain nombre de caractéristiques, telles que la convexité, la continuité ou encore la dérivabilité. Certains problèmes restent cependant trop compliqués à résoudre pour les méthodes exactes. Certaines caractéristiques peuvent poser des soucis pour ces méthodes. Dans ce cas, le problème d'optimisation est dit difficile, car aucune méthode déterministe ne peut résoudre ce problème en un temps raisonnable [36].

Les problèmes d'optimisation difficile se divisent en deux catégories : les problèmes à variables discrètes et les problèmes à variables continues. De façon générale, un problème d'optimisation à variables discrètes, ou combinatoire, consiste à trouver, dans un ensemble discret, la meilleure solution réalisable, au sens du problème défini par **(2.1)**. Le problème

majeur réside ici dans le fait que le nombre de solutions réalisables est généralement très élevé, donc il est très difficile de trouver la meilleure solution dans un temps raisonnable.

Les problèmes à variables continues sont, eux, moins formalisés que les précédents. En effet, la grande majorité des métaheuristiques existantes ont été créées pour résoudre des problèmes à variables discrètes [37].

2.5. Problème d'optimisation multi-objectif

L'optimisation multi-objectif consiste à trouver un vecteur de décisions qui satisfait les contraintes et optimise le vecteur objectif dont les éléments représentent les fonctions objectif, ces dernières forment une description mathématique du critère de performance, ces fonctions sont généralement en conflit. Le terme « optimiser » veut dire trouver une solution de telle façon que toutes les fonctions objectifs renvoient des valeurs acceptables. Un problème de minimisation est décrit comme ci-dessous [38] :

$$\begin{aligned} &minimiser \quad y= (f1\,(x),\ldots., fn(x)) \\ &Avec \qquad x= (x1,\ldots..,m) \in X \\ &\qquad\quad y= (y1,\ldots\ldots, yn) \in Y \end{aligned} \tag{2.2}$$

Où x est appelé vecteur de décision constitué de m variables de décision, dans un AGs, il représente un individu (solution potentielle). Et y vecteur objectif de n objectifs. X représente l'espace de décision, et Y l'espace des objectifs. L'optimisation multi objectif traite plusieurs fonctions objectifs en même temps, pour cela, elle utilise la notion de compromis 3 optimaux. Les solutions sont départagées en se basant sur la notion de dominance au sens de Pareto.

2.6. Optimisation combinatoire multi-objectif : problématique et cadre de travail

2.6.1. Qu'est-ce que l'optimisation combinatoire multi-objectif ?

L'optimisation combinatoire multi-objectif fait partie du domaine de l'optimisation combinatoire. Ainsi un certain nombre de définitions s'inspirent de l'optimisation combinatoire, mais différents concepts, septiques au multi-objectif, sont également introduits. En effet, la spécificité principale du multi-objectif étant l'existence de plusieurs fonctions à optimiser, il est en particulier nécessaire de revisiter la notion d'optimalité des solutions [39].

En plus du fait que l'optimisation soit multi-objectif, le nombre de solutions possibles est la plupart du temps gigantesque ce qui ne permet pas de créer des algorithmes qui recherchent les meilleurs solutions. Pour essayer tout de même de traiter au mieux ces problèmes on cherche donc à créer des heuristiques afin de s'approcher au mieux des optimums.

Ainsi, le but ne sera pas de trouver la meilleure solution mais de se rapprocher tant que possible des meilleures valeurs selon tous les critères afin de définir un front Pareto (meilleur front possible sans qu'aucune solution ne soit dominée par une autre) [40].

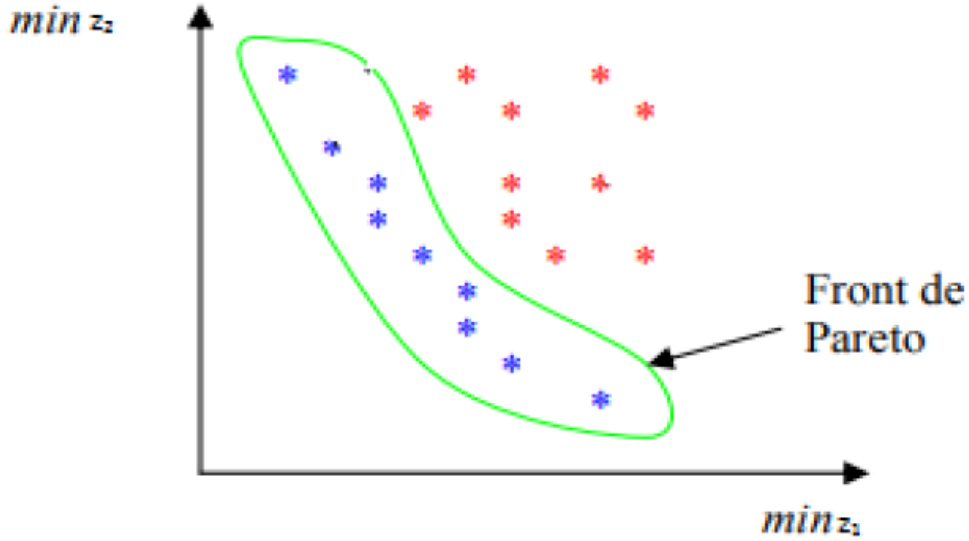

Figure 2.1 : Front de Pareto de min (z1; z2).

2.6.2. Problème d'optimisation combinatoire multi-objectif

Un problème d'optimisation combinatoire multi-objectif (PMO) (multi-objective combinat rial optimisation problème) peut être défini par :

$$(PMO) \quad \{ \quad \text{Optimiser } F(x) = (f1(x), f2(x), ..., fn(x)) \quad \text{Sous } x \in D\} \qquad (2.3)$$

ou n est le nombre d'objectifs ($n \geq 2$), x = (x1, x2, . . . , xk) est le vecteur représentant les variables de décision, ***D*** représente l'ensemble des solutions réalisables et chacune des fonctions ***Fi(x)*** est a optimiser, c'est-a-dire a minimiser ou a maximiser. Sans perte de généralité nous supposerons par la suite que nous considérons des problèmes de minimisation. Contrairement à l'optimisation mono objectif, la solution d'un problème multi-objectif n'est

pas unique, mais est un ensemble de solutions non dominées, connu comme l'ensemble des solutions Pareto Optimales ***(PO)*** [41].

2.6.3. La sélection d'attributs vue comme un problème d'optimisation combinatoire

Le problème de sélection d'attributs peut être vu comme une recherche dans un espace de solutions possibles. Etant donné un ensemble initial de N attributs, on aura 2N sous-ensembles possibles à explorer. Chaque sous ensemble est évalué à l'aide d'une fonction d'évaluation F. dans le cas d'une classification supervisée, cette fonction représente souvent la précision du classificateur construit à partir du sous ensemble sélectionné. Trouver alors un sous ensemble optimal pour la fonction F est un problème NP-complet.

Les problèmes NP-complets sont des problèmes auxquels on n'arrive pas à trouver une solution efficacement. De plus leur résolution se fait en un temps exponentiel à la taille de l'entrée. Beaucoup d'approches ont été mises au point pour contourner cette difficulté. Elles sont formalisées dans la définition suivante :

Soit X un ensemble d'attributs et soit F une mesure d'évaluation qui associe à tout sous-ensemble de X un score : F : $\bar{X} \subseteq X \rightarrow \mathbb{R}$. F doit être optimisé. On supposera dans la suite que F doit être maximisée.

La sélection d'un sous-ensemble d'attributs peut se faire suivant un des schémas suivants :

- Nombre d'attributs fixé : pour un nombre M fixé d'attributs tel que M < N, on cherche à trouver $\bar{X} \subset X$ tel que $|\bar{X}| = M$ et que F ($\bar{X}$) soit maximale.
- Seuil de performance fixé : on se donne une valeur seuil F_{opt}, c'est-à-dire le minimum acceptable pour F, et on cherche à trouver $\bar{X} \subseteq X$ tel que le cardinal de X soit le plus petit possible et que F ($\bar{X}$) $\geq F_{opt}$.
- Compromis performance et nombre d'attributs. Trouver un compromis entre le fait de minimiser le nombre d'attributs $|\bar{X}|$ et le fait d'optimiser F ($\bar{X}$).

2.6.4. Choix de la méthode d'aide a la décision

La résolution d'un problème multi-objectif menant à la détermination d'un ensemble de solutions Pareto, il est nécessaire de faire intervenir l'humain a travers un décideur, pour le

choix final de la solution a gardé. Ainsi, avant de se lancer dans la résolution d'un problème multi-objectif, il faut se poser la question du type de méthode d'optimisation à utiliser.

En effet, on peut repartir les méthodes de résolution de problèmes multi objectif en trois familles, en fonction du moment où intervient le décideur. Ainsi nous pouvons trouver les familles suivantes :

- **Les méthodes d'optimisation a priori** : dans ce cas, le compromis que l'on désire faire entre les objectifs à été défini avant l'exécution de la méthode. Ainsi une seule exécution permettra d'obtenir la solution recherchée. Cette approche est donc rapide, mais il faut cependant prendre en compte le temps de modélisation du compromis et la possibilité pour le décideur de ne pas être satisfait de la solution trouvée et de relancer la recherche avec un autre compromis.

- **Les méthodes d'optimisation progressives** : ici, le décideur intervient dans le processus de recherche de solutions en répondant à différentes questions afin d'orienter la recherche. Cette approche permet donc de bien prendre en compte les préférences du décideur, mais nécessite sa présence tout au long du processus de recherche.

- **Les méthodes d'optimisation a posteriori** : dans cette troisième famille de méthodes, on cherche à fournir au décideur un ensemble de bonnes solutions bien reparties. Il peut ensuite, au regard de l'ensemble des solutions, sélectionner celle qui lui semble la plus appropriée. Ainsi, il n'est plus nécessaire de modéliser les préférences du décideur (ce qui peut s'avérer être très difficile), mais il faut en contre partie fournir un ensemble de solutions bien reparties, ce qui peut également être difficile et requérir un temps de calcul important (mais ne nécessite pas la présence du décideur) [42].

2.7. Méthodes de résolution des problèmes d'optimisation combinatoire multi-Objectifs

2.7.1. Les Algorithmes génétiques

Les algorithmes génétiques [43, 44, 45] s'inspirent de la théorie de l'évolution et des règles de la génétique qui expliquent la capacité des espèces vivantes à s'adapter à leur environnement par la combinaison des mécanismes suivants :

- la sélection naturelle fait que les individus les mieux adaptés à l'environnement tendent à survivre plus longtemps et ont donc une plus grande probabilité de se reproduire.
- la reproduction par croisement fait qu'un individu hérite ses caractéristiques de ses parents, de sorte que le croisement de deux individus bien adaptés à leur environnement aura tendance à créer un nouvel individu bien adapté à l'environnement.
- la mutation fait que certaines caractéristiques peuvent apparaître ou disparaître de façon aléatoire, permettant ainsi d'introduire de nouvelles capacités d'adaptation à l'environnement, capacités qui pourront se propager grâce aux mécanismes de sélection et de croisement.

Les algorithmes génétiques reprennent ces mécanismes pour définir une méta-heuristique. L'idée est de faire évoluer une population de combinaisons, par sélection, croisement et mutation, la capacité d'adaptation d'une combinaison étant ici évaluée par la fonction objectif à optimiser. L'algorithme 1 décrit ce principe général.

Algorithme 1: Algorithme génétique
Initialiser la population avec un ensemble de combinaisons de E
Tant que critères d'arrêt non atteints faire
    Sélectionner des combinaisons de la population
    Créer de nouvelles combinaisons par recombinaison et mutation
    Mettre à jour la population
Retourner la meilleure combinaison ayant appartenu à la population

1. Sélection : cette étape consiste à choisir les combinaisons de la population qui seront ensuite candidates pour la recombinaison et la mutation. Il s'agit là de favoriser la sélection des meilleures combinaisons, tout en laissant une petite chance aux moins bonnes

combinaisons. Il existe de nombreuses façons de procéder à cette étape de sélection. Par exemple, la sélection par tournoi consiste à choisir aléatoirement deux combinaisons et à sélectionner la meilleure des deux (ou bien à sélectionner une des deux selon une probabilité dépendant de la fonction objectif). La sélection peut également être réalisée selon d'autres critères comme la diversité. Dans ce cas de figure, seuls les individus "distancés" sont autorisés à survivre et à se reproduire.

2. Recombinaison (croisement) : cette étape vise à créer de nouveaux individus par un mélange de combinaisons sélectionnées. L'objectif est de conduire la recherche dans une nouvelle zone de l'espace où de meilleures combinaisons peuvent être trouvées. A cette fin, la recombinaison doit être bien adaptée à la fonction à optimiser et capable de transmettre de bonnes propriétés des parents vers la combinaison créée. De plus, l'opérateur de recombinaison devrait idéalement permettre de créer des descendants diversifiés. Du point de vue de l'exploration et l'exploitation, la recombinaison est destinée à jouer un rôle de diversification stratégique avec un objectif à long terme de renforcement de l'intensification.

3. Mutation : cette opération consiste à modifier de façon aléatoire certains composants des combinaisons obtenues par croisement.

Exemple : Pour le voyageur de commerce, un opérateur de recombinaison simple consiste à recopier une sous-séquence du premier parent, et à compléter la permutation en plaçant les villes manquantes dans l'ordre où elles apparaissent dans le deuxième parent. Un opérateur de mutation classique consiste à choisir aléatoirement quelques villes et les échanger.

4. Mise à jour de la population : cette étape détermine quelles nouvelles combinaisons doivent devenir membre de la population et quelles anciennes combinaisons de la population doivent être remplacées. La politique de mise-à-jour est essentielle pour maintenir une diversité appropriée de la population, pour prévenir une convergence prématurée du processus de recherche, et pour permettre à l'algorithme de toujours découvrir de nouvelles zones prometteuses dans l'espace de recherche.

Ainsi, les décisions sont souvent prises selon des critères liés à la fois à la qualité et à la diversité. Par exemple, une règle bien connue de mise à jour basée uniquement sur la qualité consiste à remplacer la pire des combinaisons de la population, tandis qu'une règle fondée sur la diversité consiste à substituer les anciennes combinaisons de la population par de nouvelles combinaisons similaires, conformément à une mesure de similarité donnée.

5. Critères d'arrêt : le processus d'évolution est itéré, de génération en génération, jusqu'à ce qu'un critère d'arrêt soit atteint. Il peut s'agir d'un nombre maximum de générations, un nombre maximum d'évaluations, un nombre maximum de générations sans améliorer la meilleure solution, une qualité de solution atteinte ou encore une diversité de population inférieure à un seuil donné.

2.7.2. Résolution exacte

Le principe des méthodes exactes consiste à chercher la meilleure solution dans l'ensemble des solutions réalisables d'un problème.

L'optimisation exacte concerne toutes les méthodes permettant d'obtenir un résultat dont on sait qu'il est optimal à un problème précis. On peut les classer en trois grandes classes :

- La programmation dynamique.
- La programmation linéaire et quadratique.
- Méthode de séparation et évaluation [46].

2.7.3. Résolution approchée

Dans les méthodes exactes, le temps de calcul est généralement exponentiel, ce qui explique qu'elles ne sont utilisables que sur des problèmes de petite, voire de taille moyenne.

Ainsi, pour les problèmes de grande taille, ces méthodes ne sont pas envisageables. Il est dans ce cas préférable d'utiliser des méthodes approximatives qui donnent des solutions approchées ou *sub-optimales*, c'est-à-dire qu'elles ne fournissent pas de solution optimale, mais essayent de s'en approcher. On choisit ainsi ces méthodes pour leur rapidité d'exécution.

Les solutions obtenues peuvent ensuite servir de solutions initiales pour les méthodes amélioratrices. Les algorithmes qui fournissent des solutions approchées sont appelés : les heuristiques et les metaheuristiques [47].

2.8. Optimisation multi-objectif par algorithmes génétiques

2.8.1. Méthodes Pareto

Soit un ensemble de vecteurs de décision $X' \subseteq X$, X' est dit un ensemble optimal de Pareto local si et seulement si :

$$X' = \{\forall\ a' \in X' : \nexists a \in X : a < d a' \wedge \|a - a'\| < \varepsilon\}$$

X' est dit un ensemble optimal de Pareto local s'il n'existe pas de solutions $a \in X$ qui dominent les solutions a' ($a' \in '$) dans le voisinage $\|a - a'\| < \varepsilon$, sans le que reste de l'espace de décision ne soit vérifié. Rappelons que X représente l'espace de décision.

X'' est dit un ensemble optimal de Pareto global si et seulement si :

$$X'' = \{\forall\ a'' \in X'' : \nexists a \in X : a < d a''\}$$

X' est dit un ensemble optimal de Pareto global s'il n'existe pas de solutions $a \in X$ qui dominent les solutions a'' ($a' \in ''$) sur tout l'espace de décision X.

L'image de l'ensemble de décision $X*$ par la fonction objectif représente le front de Pareto optimal (global ou local), ce dernier est fait de telle façon qu'on ne peut améliorer une performance (objectif) sans détériorer les autres performances [13]. La figure suivante illustre l'espace objectif contenant les images des solutions non-dominées, et les solutions dominées dans l'espace objectif, ainsi que le front de Pareto pour un problème de minimisation à deux objectifs :

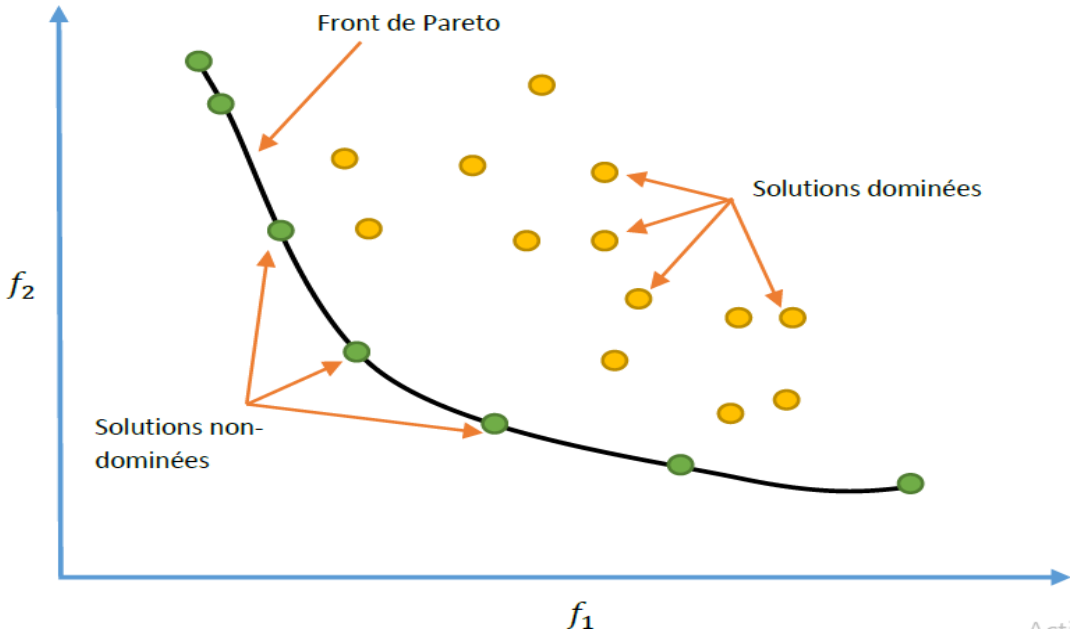

Figure 2.2 : Front de Pareto.

Exemple : Pour mieux expliciter les notions précédemment définies, considérons l'exemple suivant :

Soient six vecteurs de décision :

a=(a1 , a2), b=(b1,b2),c= (c1 ,c2),d= (d1 ,d2),e= (e1 ,e2), f =(f1 ,f2)

Soit la fonction objectif F x = [*f1*(x), *f2*(x)], dans cet exemple on peut écrire :

- *f* : est dominé par tous les autres vecteurs.
- *b* : est dominé uniquement par c.
- *d* : est dominé par *c* est *e*.
- *a*, *c* et *e* : ne sont dominés par aucun vecteur.

Par conséquent les solutions (*a*, *c*, *e*) sont Pareto optimaux. Donc le front de Pareto est l'ensemble : {a, c, e}.

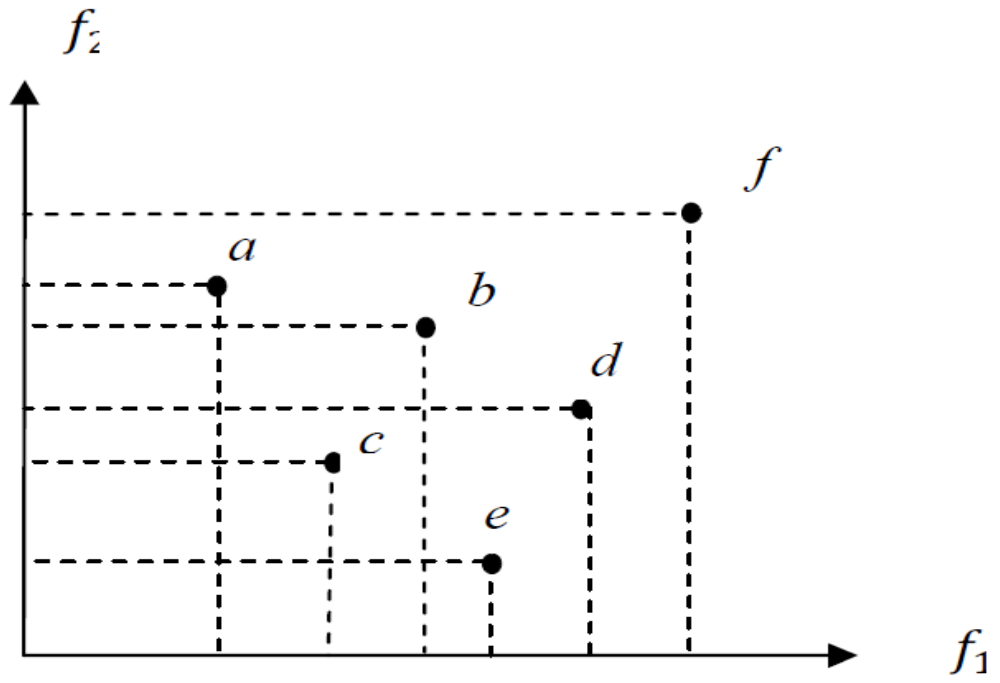

Figure 2.3 : Exemple de dominance et d'optimalité au sens de Pareto.

2.8.2. Méthode agrégée

Ces approches basées sur la transformation du problème multi objectif en un problème uni objectif. Cette classe d'approches comprend par exemple les méthodes basées sur l'agrégation qui combine entre les différentes fonctions objectives fi en une seule fonction objectif $\boldsymbol{f}$. Ces approches nécessitent d'avoir une bonne connaissance du problème [49].

2.8.3. Méthodes non-agrégées non-Pareto

Dans ce type d'approche, la population est utilisée pour diversifier la recherche, mais le principe de Pareto n'est pas directement incorporé dans le processus de sélection. Le modèle le plus représentatif de cette méthode est le Vector Evaluated Genetic Algorithm (VEGA).

- **Vector Evaluated Genetic Algorithm (VEGA) :** Proposé par Schaffer en 1985 [50], le VEGA est un AGs multi-objectif avec un processus de sélection particulier. À chaque génération, une sous-population est générée à partir de la population initiale pour chaque objectif, donc pour n objectifs, n sous-populations de p/n individus sont générées (p le nombre d'individus de la population initiale). Ensuite, ces sous-

populations sont rassemblées pour obtenir une nouvelle population sur laquelle les opérateurs génétiques seront appliqués (figure 2.4).

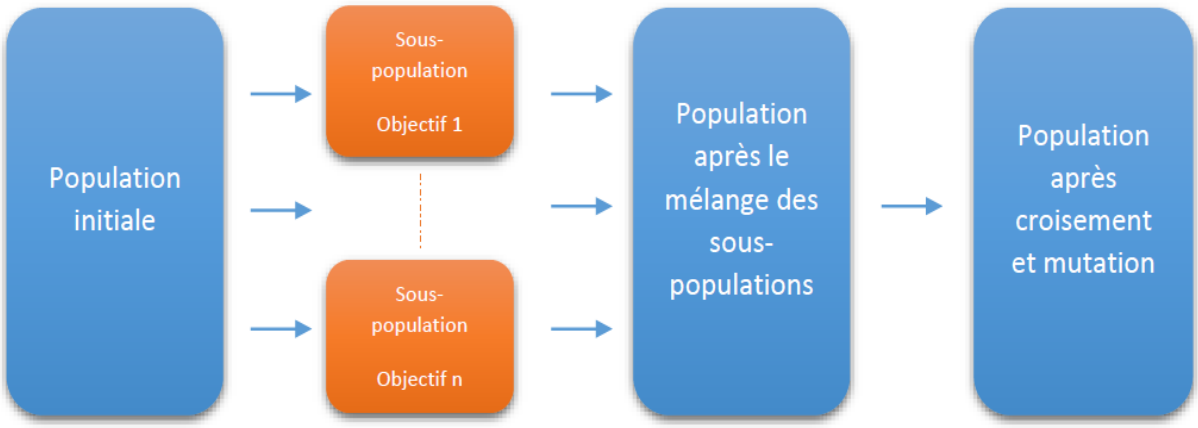

Figure 2.4: Vector Evaluated Genetic Algorithm (VEGA).

2.9. Description détaillée NSGA-III

2.9.1. Définition

Dans cette section, nous présentons le NSGA - III qui a été désigné comme une réponse aux difficultés qu'ont les algorithmes évolutionnaires pour traiter plusieurs objectifs en même temps (plus de deux). Cet algorithme a été proposé par Deb et al. [51] et il s'appuie sur la même structure que son prédécesseur le NSGA - II avec quelques changements importants dans le mécanisme de sélection. Dans l'algorithme NSGA - II, le classement de la population est fait par différents niveaux de solutions non-dominées (fronts, F1; F2, etc.).

Ensuite les différents opérateurs génétiques sont appliqués pour créer la population de la génération suivante. Et selon le nombre de solutions par front et le nombre d'individus à garder, nous prenons en compte la crowding distance comme un critère de sélection. De cette manière les solutions avec les valeurs de crowding distance les plus grandes seront choisies.

- Les étapes du NSGA III:

1. La NSGA-III commence par définir un ensemble de points de référence.

2. Ensuite, une population initiale avec N membres est générée de manière aléatoire, N étant la taille de la population.

3. Les étapes suivantes sont itérées jusqu'à ce que le critère de terminaison soit satisfait.

2.9.2. Fonctionnement du NSGA III

Le cadre de base de NSGA-III reste similaire à la NSGA-II établie avec des changements importants dans son mécanisme de sélection. La procédure principale de NSGA-III peut être brièvement décrite ci-dessous. NSGA-III commence par la définition d'un ensemble de points de référence. Ensuite, une population initiale avec N membres est générée de manière aléatoire, où N est la taille de la population. Les étapes suivantes sont répétées jusqu'à ce que le critère de terminaison soit satisfait.

À la dixième génération, la population parent actuelle Pt est utilisée pour produire une population de descendants Qt en utilisant une sélection aléatoire, un opérateur de croisement binaire simulé (SBX) et une mutation polynomiale. Les tailles de Pt et Qt sont toutes deux N. Par la suite, les deux populations Pt et Qt sont fusionnées pour former une nouvelle population $Rt = Pt \cup Qt$ (de taille 2N).

Pour choisir les meilleurs N membres de Rt pour la prochaine génération, le tri non dominé basé sur le principe de domination habituel est d'abord utilisé, qui classe Rt en différents niveaux de non-domination (F1, F2, etc.). Ensuite, une nouvelle population St est construite en remplissant les membres de différents niveaux de non-domination un par un, à partir de F1, jusqu'à ce que la taille de St soit égale à N ou pour la première fois devienne supérieure à N. Supposons que le dernier niveau inclus est le l ème niveau.

Par conséquent, les solutions à partir du niveau l + 1 sont simplement rejetées. Les membres de St \ Fl sont déjà choisis pour Pt + 1, et les tranches de population restantes sont choisies dans Fl de telle sorte qu'une diversité souhaitée soit maintenue dans la population. Dans le NSGA-II d'origine, les solutions Fl avec les plus grandes valeurs de distance d'encombrement est sélectionné [52].

Algorithme 2 : NSGA-III pour la sélection des fonctionnalités [51]

Entrée: Z^s *ou* Z^r % H points de référence structurés ou points d'aspiration fournis

Sortie: front de Pareto de l'index jaccard

1: Générer une population initiale et évaluer des valeurs objectives pour chaque individu

2: boucle NSGA-III

3: Utiliser des opérateurs évolutifs pour générer une population d'enfants

4: Utilisez le tri non dominé pour la combinaison des populations père et enfant

5: Elite-sélection aux fronts de la population de combinaison

6: si le dernier front est partiellement accepté,

7: Normaliser les objectifs pour accepter le front, y compris le dernier front

8: Créer le jeu de référence Z^r selon Z^s ou Z^a

9: Associer des individus à des points de référence

10: Utilisez Niche-Preservation pour générer une nouvelle population

11**: fin** si

12: **Fin** de la boucle NSGA-III

Algorithme 3 : Procédure de normalisation $(\boldsymbol{f^n}, S_t, Z^s, Z^r, /Z^a)$ [51]

Entrée: S_t , Z^s (points structurés) ou Z^a) (points fournis)

Sortie: $\boldsymbol{f^n}$, Z^r (points de référence sur l′hyperplan normalisé)

1: pour $j = 1$ à M do

2: Calculer le point idéal: $Z_j^{min} = \min_{S \in S_t} fj\,(s)$

3: Traduire les objectifs: $f_j(s) =: f_j(s) - Z_j^{min}\ \forall s \in S_t$

4: Calcul des points extrêmes: $Z^{j,max} = S$:

$\text{argmins}_{S \in S_t} ASF\,(s, w^j)$, où $w^j = (\epsilon, \ldots, \epsilon)^T$

$\epsilon = 10^{-6}$, et $w_j^j = 1$

5: fin pour

6: Calculer les interceptions a_j pour j = 1,. . ., M

7: Normaliser les objectifs ($\boldsymbol{f^n}$) en utilisant l'équation 5

8: si Z^a est donné,

9: Cartographier chaque point (d'aspiration) sur un hyperplan normalisé

en utilisant l'équation 5 et enregistrer les points dans l'ensemble Z^r

10: sinon

11: $Z^r = Z^s$

12: fin si

Algorithme 4 : Procédure association (S_t ,Z^r) [51]

Entrée: Z^r, S_t

Sortie: $\pi (s \in S_t), d (s \in S_t)$

1: pour chaque point de référence z% Z^r do

2: Calculer la ligne de référence $w = z$

3: fin pour

4: pour chaque $s \in S_t$ do

5: pour chaque $w \in Z^r$ do

6: Calculer $d \in (s, w) = s - w^T s/||w||$

7: fin pour

8: Attribuer $(s) = w: \text{argmin}_{w \in Z^{rd}} \perp (s, w)$

9: Attribuer $d (s) = d^{\perp} (s, \pi(s))$

10: fin pour

Algorithme 5 : procédure Niching (K, $\boldsymbol{\rho_j}$, π, d, Z^r , $\boldsymbol{F_l}$, : P_{t+1}) [17]

Entrée: **K**, $\boldsymbol{\rho_j}$, $\pi (s \in S_t)$, $d(s \in S_t)$, Z^r, $\mathbf{F_l}$,

Sortie: : P_{t+1}

1: k = 1

2: tant que $k \leq \mathbf{K}$ do

3: $J_{min} = \{ j : \text{argmin}_{j \in Z^r} \rho_j \}$

4: $\bar{j}$ = random (J_{min})

5: $I_{\bar{j}} = \{s: \pi(s) = \bar{j}, s \in \mathbf{F_l}\}$

6: si $I_{\bar{j}} \neq \emptyset$ alors

7: si $\rho_{\bar{j}} = 0$ alors

8: $P_{t+1} = P_{t+1} \cup \left(s : \text{argmin}_{s \in I_{\bar{j}}} d(s)\right)$

9: sinon

10: $P_{t+1} = P_{t+1} \cup \text{random}(I_{\bar{j}})$

11: fin si

12: = $\rho_{\bar{j}} = \rho_{\bar{j}} + 1$, $F_l = F_l \setminus s$

13: k = k + 1

14: sinon

15: $Z^r = Z^r / \{\bar{j}\}$

16: fin si

17: fin tant que

Algorithme 6 : la fonction fast_nondominated_sort()

Pour chaque p ∈ Population

Pour chaque q ∈ Population

Si (p domine q) Alors

Sp = Sp ∪{q}

Sinon

Si (q domine p) Alors np = np + 1

FinSi

FinPour

Si np = 0 Alors F1 = F1 ∪ {p}

FinPour

i = 1

Tant que Fi ≠ ∅

H = ∅

```
Pour chaque p ∈ Fi
Pour chaque h q ∈ Sp
 nq = nq - 1
Si nq = 0 Alors H = H ∪ {q}
FinPour
FinPour
 i = i + 1
Fi = H
FinTantQue
```

2.10. Conclusion

Dans ce chapitre, nous avons présenté les Problèmes d'optimisation multi-objectifs et et des techniques génétiques et des techniques d'optimisation multi-objectifs ainsi que leur application dans le domaine de la sélection de classificateurs et de la sélection de caractéristiques.

Après avoir donné une brève introduction sur les algorithmes génétiques et leur manière d'optimiser un objectif et après avoir présenté les différentes techniques d'optimisation multi-objectifs par agrégation et par méthodes de Pareto, dans le domaine de la sélection.

Enfin nous avons détaillée algorithme NSGA-III.

CHAPITRE 3

CONCEPTION ET RÉALISATION

3.1. Introduction

La sélection d'attributs consiste à trouver un sous-ensemble d'attributs, parmi les 2N sous-ensembles d'un ensemble de grande taille, qui permettra de mener à bien la classification supervisée. Dans le cadre de notre travail, il s'agit de fournir ce sous-ensemble à un IDS pour qu'il puisse distinguer au mieux les enregistrements sains de ceux qui ne le sont pas.

Dans ce qui suit, nous commençons par présenter l'architecture générale de notre solution puis nous en parlerons plus en détails. Ensuite, nous exposerons la conception de l'algorithme NSGA-III pour performer notre solution.

3.2. Conception globale

3.2.1. Architecture globale du système

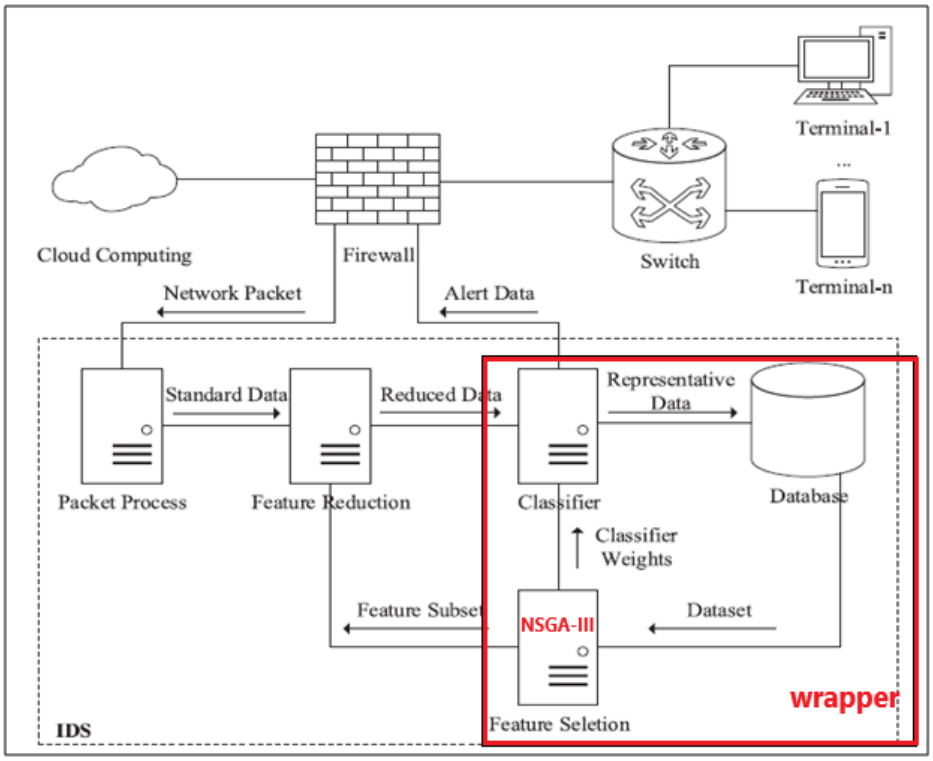

Figure 3.1 : Montre l'architecture globale du système.

3.2.1.1. Le module de traitement

Reçoit des paquets du réseau et transforme ces paquets en données normalisées selon l'ensemble de fonctionnalités par défaut.

3.2.1.2. Le module de réduction de fonctionnalités

Pour améliorer l'efficacité, le module de réduction de fonctionnalités supprime les fonctionnalités redondantes des données standardisées selon un sous-ensemble de fonctionnalités reçu du module de sélection de fonctionnalités, puis sort les données réduites, ce qui signifie que les données sont présentées avec moins de fonctionnalités.

3.2.1.3. Utilisation de l'approche enveloppent

3.2.1.3.1. L'architecture détaillée de l'approche enveloppent

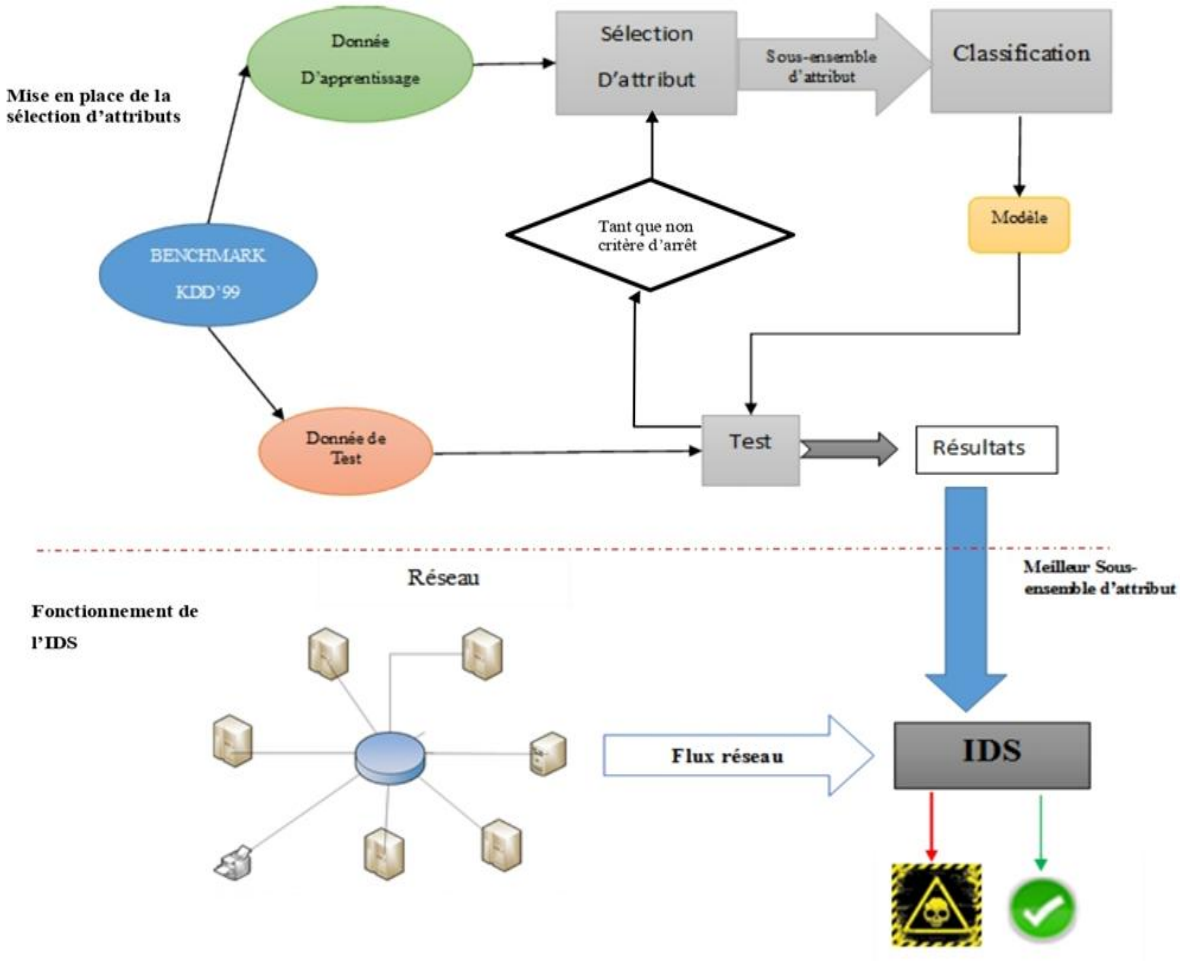

Figure 3.2 : Montre l'architecture détaillée de l'approche enveloppent.

3.2.1.3.2. Description de Système

Cette architecture se compose de deux phases :

La phase 1: (la mise en place de la sélection d'attributs), se devise en quatre grandes étapes :

- L'étape 1

La première étape consiste à séparer les données en deux sous-ensembles, un pour l'apprentissage (90% de l'ensemble) et l'autre pour le test de validation.

- L'étape 2

Cette étape consiste à utiliser un algorithme de génération de sous-ensembles d'attributs. Ce processus peut être vu aussi comme une recherche d'un sous-ensemble dans un espace de solutions possibles (les sous-ensembles d'attributs). Nous utiliserons pour cette phase un algorithme métaheuristiques NSGA-III.

- L'étape 3

Cette étape consiste à faire un apprentissage en effectuant une classification des individus de l'ensemble d'apprentissage.

- L'étape 4

Cette étape consiste à faire un test de validation de la phase d'apprentissage, en essayant de classifier les individus de l'ensemble de test. La précision de cette classification déterminera le sous-ensemble d'attributs considéré comme le meilleur parmi tous les sous-ensembles générés et sera transmis à l'IDS.

La phase 2: La seconde partie, bien que ne faisant pas partie de notre projet et qui pourrait faire l'objet d'un travail futur, décrit le comportement de l'IDS, qui va utiliser le sous-ensemble d'attributs fourni par la partie précédente pour mieux appréhender et détecter le trafic malsain sur le réseau et cela dans un temps moindre.

3.2.1.3.3. La sélection d'attribut (NSGA-III)

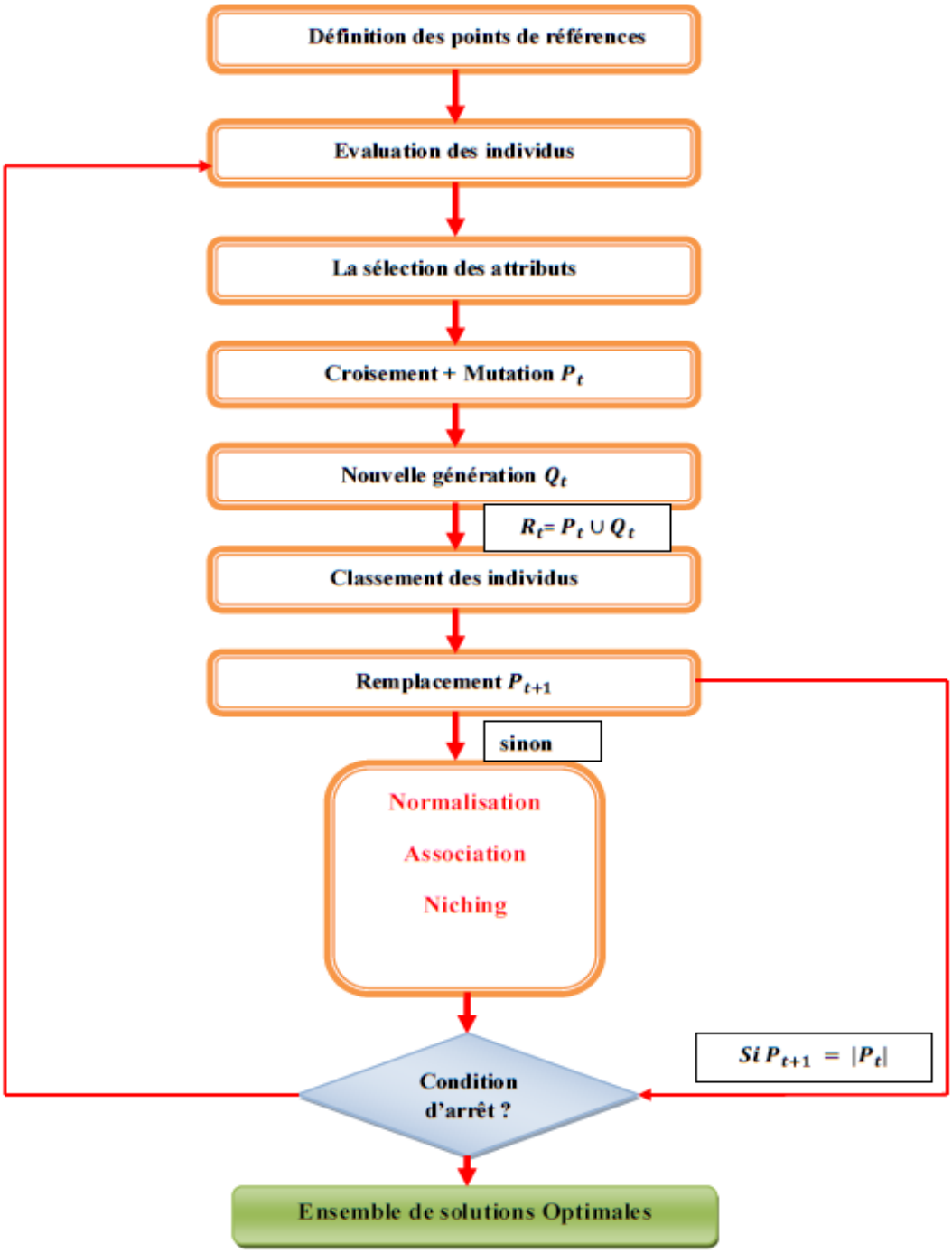

Figure 3.3 : Processus de traitement du NSGA-III.

3.3. Conception détaillée

3.3.1. Adaptation de l'algorithme génétique à la sélection d'attributs

- **Population :** est un ensemble de chromosomes et qui représente un ensemble de sous-ensembles d'attributs.
- **Un chromosome (ou individu) :** est une représentation d'une solution possible de l'espace des solutions, constitué d'un ensemble de gènes (le vecteur binaire représentant un sous-ensemble d'attributs).
- **Un gène** est un élément atomique permettant le codage de la représentation du chromosome (e.g. un bit dans une chaîne de bits).
- **La reproduction :** qui est la phase de génération de chromosomes.

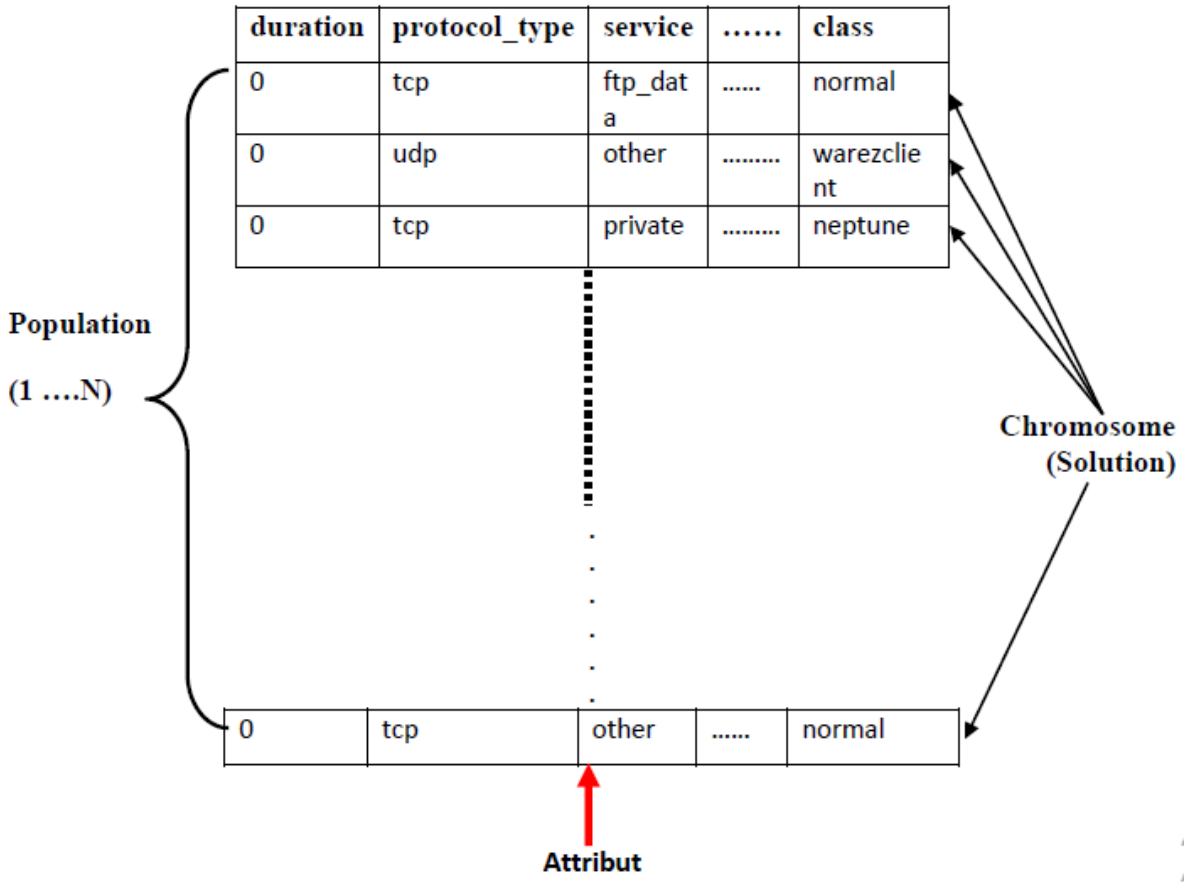

Figure 3.4 : Représentation d'une solution.

- **Sélection de caractéristiques par un algorithme génétique :** En 2000, Kudo et Sklansky [53] ont montre la possibilité d'utiliser les AGs pour la sélection sur des ensembles de grande échelle (50 caractéristiques et plus) en ajustant les paramètres de l'AG (le nombre de générations, la taille de la population et les probabilités des opérations génétiques) d'un côté et la fonction d'évaluation de l'autre côté. Une fois que les paramètres ont été bien fixés et la fonction d'évaluation bien définie, ils ont montre que les résultats de l'AG, sont meilleurs que ceux des méthodes basées sur la recherche séquentielle. La procédure de sélection par un algorithme génétique est illustrée par la figure (3.5).

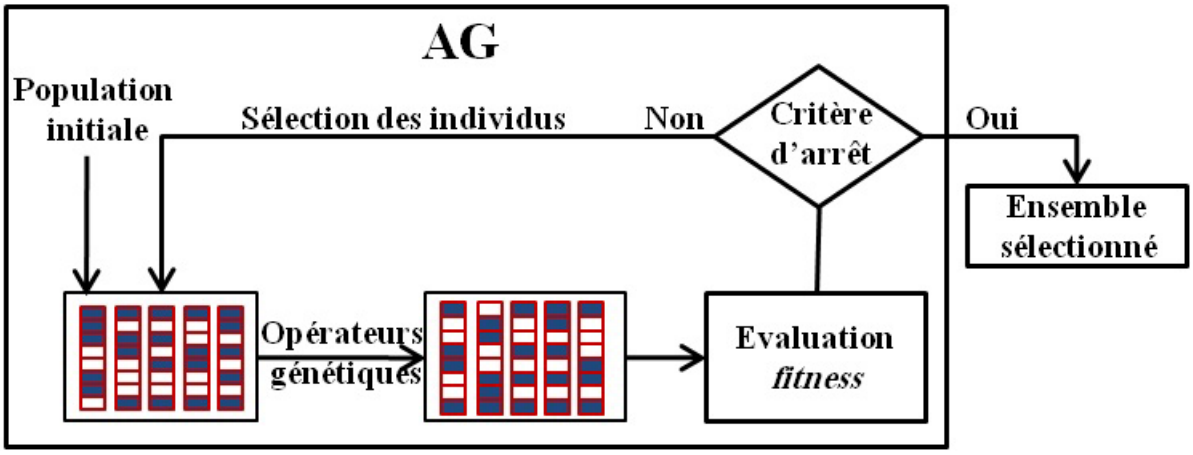

Figure 9.5 : Sélection de caractéristiques par un algorithme génétique.

3.3.2. L'Optimisation par l'algorithme NSGA-III

3.3.2.1. Population initial

Les Chromosomes de la population initiale sont générés d'une manière aléatoire et la taille N choisis par l'utilisateur.

3.3.2.2. Les points de références

NSGA-III utilise un ensemble prédéfini de points de référence pour garantir la diversité des solutions obtenues. Le nombre de ces points est calculé à partir du nombre des objectifs |M| et le nombre de division |P| [50].

Les points de référence choisis peuvent être prédéfinis de manière structurée ou fournis préférentiellement par l'utilisateur. Un exemple des points de référence est illustré à la Figure.3.6 Dans un problème à trois objectifs.

(M = 3), les points de référence sont créés sur un triangle avec des sommets à (1, 0,0), (0, 1,0) et (0, 0,1).

Si quatre divisions (P = 3) sont choisies pour chaque axe d'objectif, H = (3 + 3 −1 3) = 10 points de référence seront créés.

$$H = M + P - 1\ P \qquad (3.1)$$

Où

• H est le nombre total de points de référence dans un problème d'objectif M.

• M est le nombre d'espace objectif.

• P est le nombre de divisions sur chaque dimension objective.

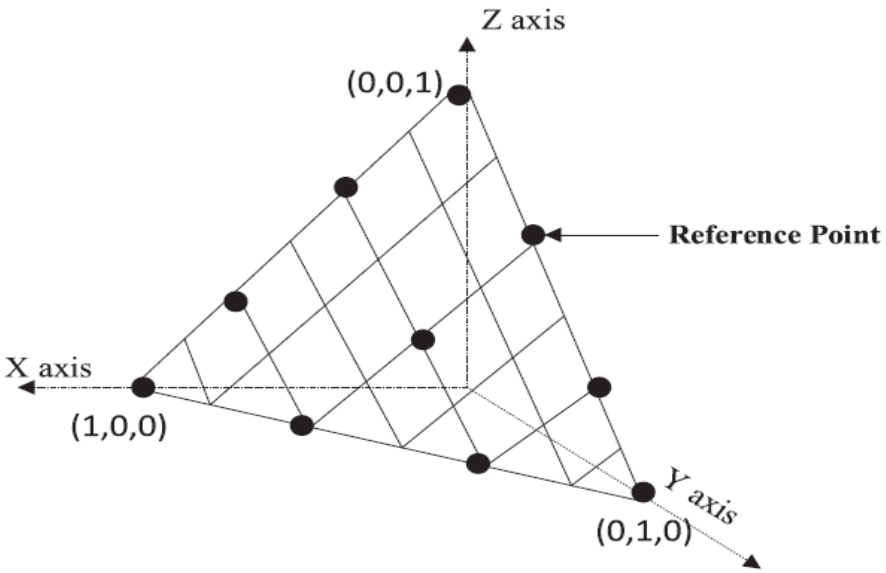

Figure 3.6 : Les points de références.

3.3.2.3. Evaluation des individus [28]

Evaluer chaque individu de la population par la fonction d'évaluation appropriée au problème (fonction de fitness).

- Jaccard index : L'indice de Jaccard est utilisé comme mesure pour évaluer l'objectif. Il mesure la similitude et la diversité entre les ensembles de données. Soit L_r le véritable label fourni par l'ensemble des données, et soit L_d le label détecté résultant. L'indice de Jaccard J (c) est indiqué dans l'équation (3.2) :

$$J(c) = \frac{P_c}{P_c + fp_c + fn_c} \qquad (3.2)$$

Alors que :

- **c** indique une étiquette de trafic qui est considérée comme classe = normale ou un type d'attaque, c'est-à-dire DOS, SONDE, U2R ou R2L.
- P_c est le nombre d'éléments classés avec succès (vrais positifs).
- fn_c c'est le nombre d'éléments étiquetés comme a ε L_r dans l'ensemble de données, mais non étiquetés comme ε L_d par le classifieur (faux négatifs).
- fp_c c'est le nombre de non étiquetés comme a ε L_r dans l'ensemble de données et étiquetés comme a ε L_d par le classifieur (faux positifs).

Dans l'équation (3.2), lorsque l'indice de Jaccard d'une classe atteint sa valeur maximale, le nombre de vrais résultats positifs atteint la valeur maximale et ses résultats faux positifs et faux négatifs atteignent la valeur minimale. Ainsi, le coefficient peut être utilisé pour sélectionner le meilleur sous-ensemble de caractéristiques.

3.3.2.4. La sélection des attributs

Dans cette étape, on sélectionne les sous-ensembles d'attributs prometteurs qui représentent les solutions non dominées Pt de la population initiale qui est basée sur l'équation (3.2).

3.3.2.5. Croisement

La stratégie de croisement utilisée est le croisement uniforme qui consiste à générer deux fils, en prenant aléatoirement des composantes soit de l'un des parents ou de l'autre. Le second vecteur sera complémentaire au premier.

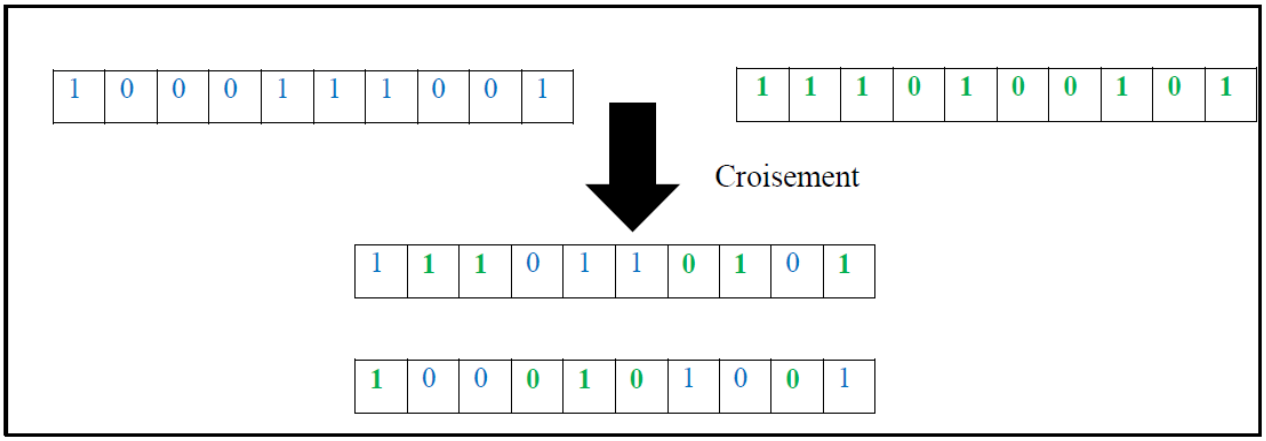

Figure 3.7 : Méthode de croisement.

3.3.2.6. Mutation

La phase de mutation a pour but diversifié les solutions pour d'accélérer le processus de recherche du meilleur sous-ensemble.

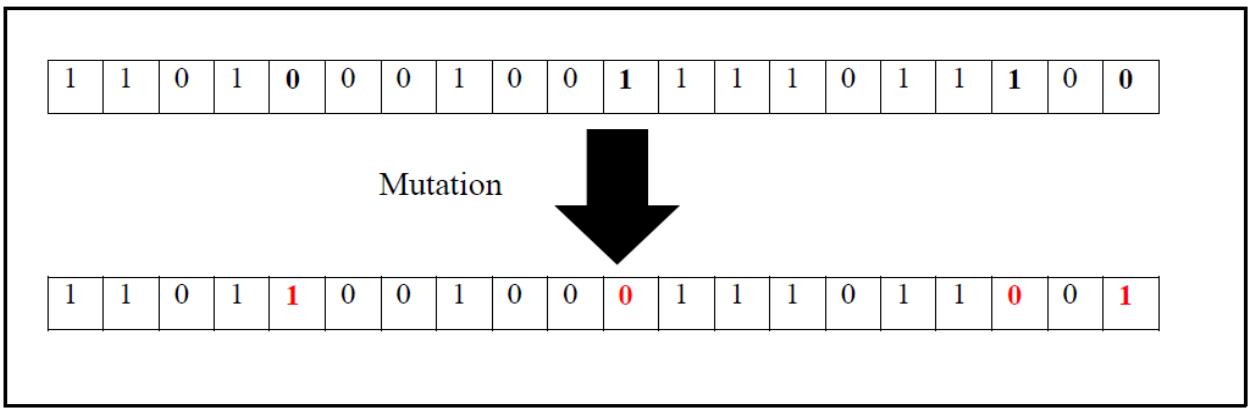

Figure 3.8 : Méthode de mutation.

3.3.2.7. Classement des individus

Dans l'optimisation multi-objectif, Après l'évaluation on doit classer les individus selon leur dominance par rapport aux fonctions objectives et établira les fronts de Pareto [50].

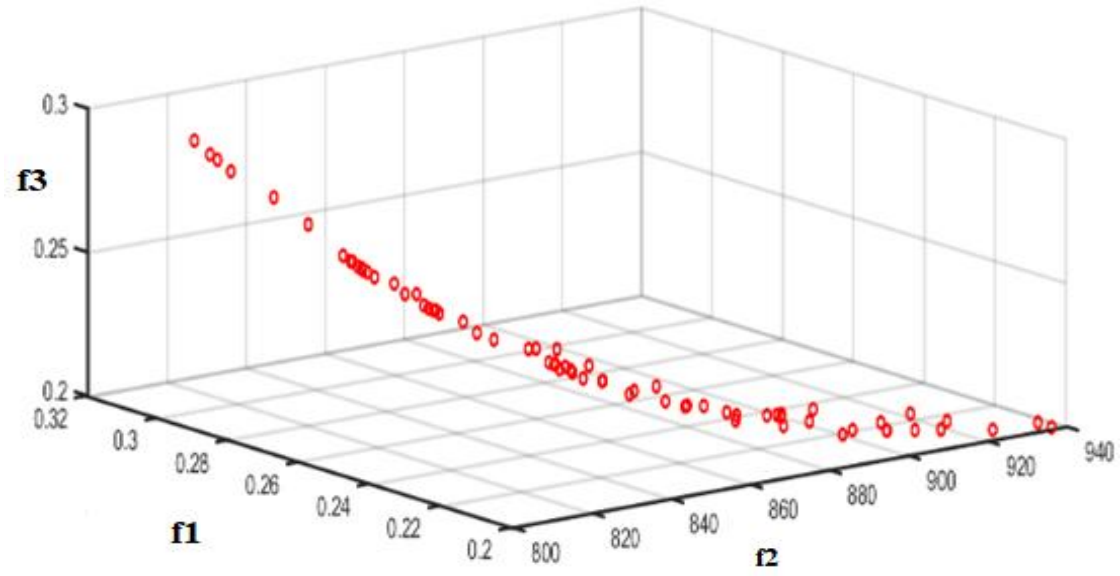

Figure 3.9 : Classement des individus.

3.3.2.8. Remplacement

On remplace l'ancien **Pt** par le nouveau $\boldsymbol{P_{t+1}}$ pour la prochaine itération. Le $\boldsymbol{P_{t+1}}$ contient les meilleurs sous-ensembles d'attributs de **Pt** et de la nouvelle population candidate. De plus, on classe les solutions de **Pt** et de nouvelle population candidatée en fonction de Jaccard index (3.2). Ensuite, on incorpore les meilleures solutions dans la population $\boldsymbol{P_{t+1}}$ [50].

3.3.2.9. Normalisation

Dans cette phase les objectifs de chaque individu sont normalisés et compris entre 0 et 1 les individus vont être distribués dans un hyper plan à 3 axes (1, 0,0) (0, 1,0) (0, 0,1). Les étapes de la normalisation dans la Figure (3.10) [50] :

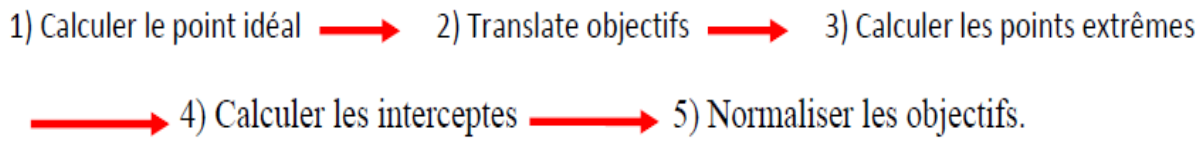

Figure 3.10 : Les étapes de la normalisation.

3.3.2.10. Association

Les étapes de l'association :

1. calculer les lignes de références entre chaque point de référence et le point idéal.
2. calculer la distance entre chaque individu normalisé de St et les lignes de références.
3. le point de référence dont la ligne est la plus proche d'un membre de la population lui sera associée.

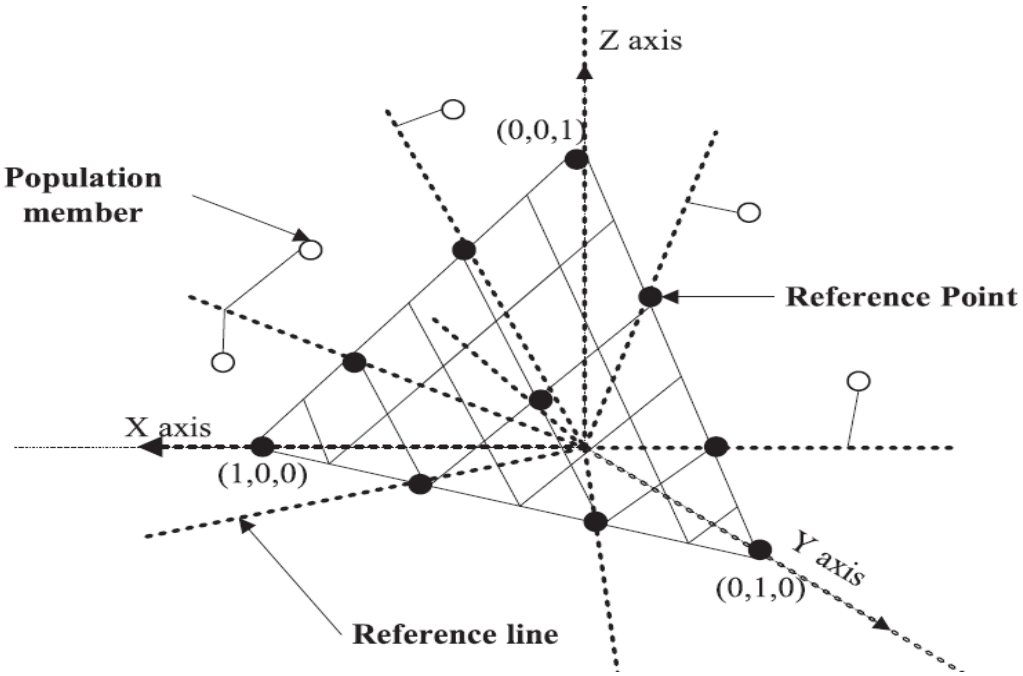

Figure 3.11 : Association de membres de la population avec des points de référence.

3.3.2.11. Niching

Dans le NSGA-III, le Niching est réalisé en triant les groupes obtenus au stade de l'association selon sa cardinalité dans l'ordre croissant. L'élément ayant la distance la plus faible à la ligne induite dans chaque groupe est sélectionnée, et l'algorithme continue avec le groupe suivant jusqu'à ce que la population soit remplie [50].

3.3.2.12. Critère d'arrêt

Le critère d'arrêt sert à stopper le processus de sélection d'attributs en fixant des conditions (tant que le critère d'arrêt qui est le nombre de génération prédéfinie n'est pas satisfait, algorithme reste en fonction).

3.3.2.13. Les solutions optimales

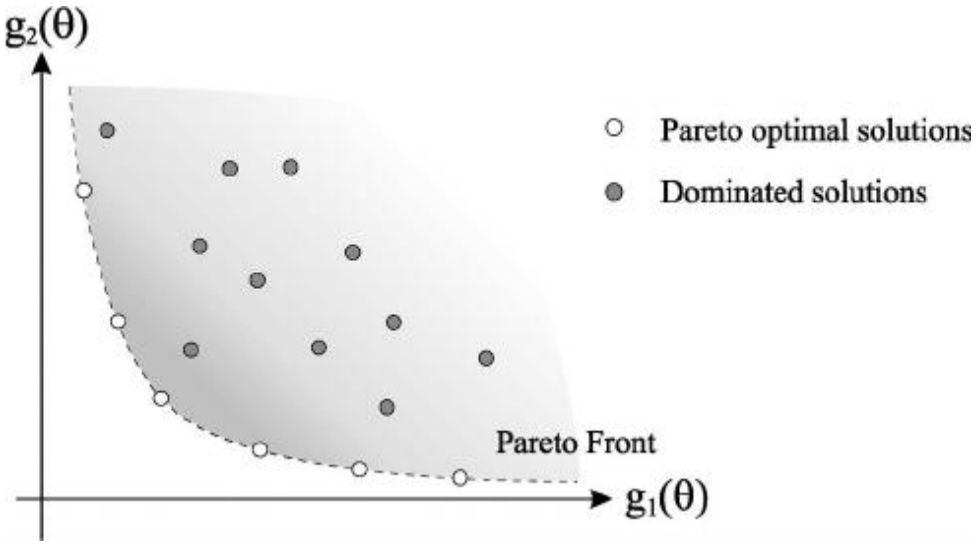

Figure 3.12 : Les solutions optimales.

3.4. Conclusion

Dans ce chapitre, toutes les questions concernant la manière de réaliser le système à développer ont été élucidées. Comme nous pouvons le constater, l'activité de la conception a facilité la compréhension de notre système, qui ébauche vers l'activité d'implémentation.

CHAPITRE 4

IMPLÉMENTATION ET RÉSULTAT

4.1. Introduction

Dans ce chapitre, nous présentons les expériences que nous avons menées pour valider notre méthode de sélection. Nous commençons par décrire les différents outils de développement, moyens et techniques qui ont été utilisés dans l'implémentation de notre application. Nous procédons ensuite à l'étude des paramètres de la méthode et notamment des paramètres de l'algorithme génétique.

Nous présentons enfin les résultats de la sélection par notre méthode et leur comparaison avec d'autres méthodes de la littérature.

4.2. Matériel et outils de développement

4.2.1. C'est quoi java?

Java est un langage de programmation et une plate-forme informatique qui ont été créés par Sun Microsystems en 1995. Beaucoup d'applications et de sites Web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour. Java est rapide, sécurisé et fiable. Des ordinateurs portables aux centres de données, des consoles de jeux aux superordinateurs scientifiques, des téléphones portables à Internet, la technologie Java est présente sur tous les fronts [54].

4.2.2. Eclipse

Eclipse IDE est un environnement de développement intégré libre (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent,

permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions [55].

4.2.3. La Bibliothèque WEKA

Weka (Waikato Environment for KnowledgeAnalysis) est un ensemble d'outils permettant de manipuler et d'analyser des fichiers de données, implémentant la plupart des algorithmes d'intelligence artificielle, entre autres, les arbres de décision et les réseaux de neurones ,elle est développé à l'Université de WAIKATO en Nouvelle Zélande . Il se compose principalement :

– De classes Java permettant de charger et de manipuler les données.
– De classes pour les principaux algorithmes de classification supervisée ou non supervisée.
– D'outils de sélection d'attributs, de statistiques sur ces attributs.
– De classes permettant de visualiser les résultats [56].

4.2.4. Weka dans Eclipse

- Sous eclipse nous avons créé un projet qui aura besoin de Weka.
- Project −! Properties −! Onglet Librairies.
- Addexternal jar (ici /opt/weka/weka-3-4-13/weka.jar).

4.2.5. Le pc de traitement

Notre traitement est fait dans un PC portable de marque DEL, avec un processeur : Intel (R) Core (TM) i3-3230M CPU 2.60GHz 2.60GHz, une Mémoire installé (RAM) : 4 Go (3.88 Go utilisable) , un système d'exploitation 64 bits , processeur x64 , mémoire graphique total disponible : 5571 Mo .

4.2.6. La base de données NSL-KDD

A. La base de données KDD CUP 99

L'ensemble de données KDD Cup 1999 a été créé par le laboratoire MIT Lincon. Chaque paquet de l'ensemble des données de KDD Cup 99 est constitué de 41 champs et est labellisés comme paquet normal ou paquet anormal avec les types d'attaques. Parmi ces champs, 37 sont des champs de type numérique et 4 sont des champs de type non numérique [57].

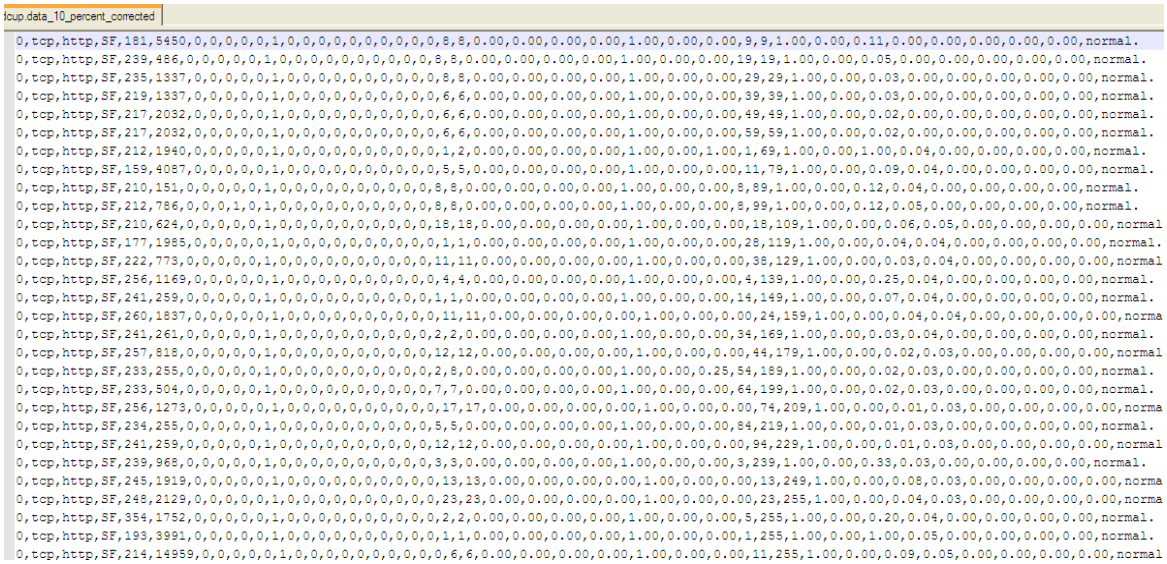

Figure 4.10 : Base de données KDD99 fichier texte.

B. Description de la base de données NSL-KDD

Les inconvénients inhérents au jeu de données KDD cup 99 ont été révélés par diverses analyses statistiques et ont affecté la précision de détection de nombreux IDS modélisés par les chercheurs. L'ensemble de données NSL-KDD est une version raffinée de son prédécesseur. Il contient les enregistrements essentiels de l'ensemble de données KDD complet. Une collection de fichiers téléchargeables est à la disposition des chercheurs. L'ensemble de données contient 5 classes (DoS, Normal, U2R, R2L, Probe) :

- **Denial of Service Attack (DoS)**: est une attaque dans laquelle l'attaquant rend une ressource informatique ou mémoire trop occupée ou trop pleine pour gérer les

demandes légitimes, ou refuse aux utilisateurs légitimes l'accès à une machine (par exemple, syn flood).

- **Remote to Local Attack (R2L)**: se produit lorsqu'un attaquant qui a la capacité d'envoyer des paquets à une machine sur un réseau mais qui n'a pas de compte sur cette machine exploite une certaine vulnérabilité pour obtenir un accès local en tant qu'utilisateur de cette machine (par exemple: deviner le mot de passe).
- **User to Root Attack (U2R):** est une classe d'exploit dans laquelle l'attaquant commence par accéder à un compte d'utilisateur normal sur le système (peut-être obtenu en reniflant des mots de passe, une attaque par dictionnaire ou l'ingénierie sociale) et est capable d'exploiter certains vulnérabilité pour obtenir un accès root au système.
- **Probing Attack**: est une tentative de recueillir des informations sur un réseau d'ordinateurs dans le but apparent de contourner ses contrôles de sécurité (par exemple: analyse des ports).

Nom du fichier	Description	Normal	DoS	Probe	R2L	U2R
KDDTrain+20%	L'ensemble complet de trains NSL-KDD, y compris les étiquettes de type d'attaque et le niveau de difficulté au format CSV	13449 (53.39 %)	9234 (36.65 %)	2289 (9.09%)	209 (0.83 %)	11 (0.04%)
KDDTrain+	Un sous-ensemble de 20% du fichier KDDTrain + .txt	67343 (53.46 %)	45927 (36.46)	11656 (9.25%)	995 (0.79 %)	52 (0.04 %)
KDDTest+	L'ensemble de test NSL-KDD complet, y compris le type d'attaque étiquettes et niveau de difficulté au format CSV.	9711 (43.08 %)	7458 (33.08 %)	2421 (10.74%)	2754 (12.22 %)	200 (0.89 %)
KDDTest-21	Un sous-ensemble du fichier KDD Test.txt qui n'inclut pas les enregistrements avec un niveau de difficulté de 21 sur 21	2152 (18.16 %)	4342 (36.64 %)	2402 (20.27%)	2754 (23.24 %)	200 (1.69 %)

Tableau 4.1 : Distribution les fichier et les classes de NSL-KDD.

S/N	Name	Type
1	Back	Dos
2	buffer_overflow	u2r
3	ftp_write	r2l
4	guess_passwd	r2l
5	Imap	r2l
6	Ipsweep	probe
7	Land	Dos
8	Loadmodule	u2r
9	Multihop	r2l
10	Neptune	Dos
11	Nmap	probe
12	Perl	u2r
13	Phf	r2l
14	Pod	Dos
15	Portsweep	probe
16	Rootkit	u2r
17	Satan	probe
18	Smurf	Dos
19	Spy	r2l
20	Teardrop	Dos
21	Warezclient	r2l
22	Warezmaster	r2l

Tableau 4.2 : Les types d'attaques.

Nom d'attributs	Description	Type
Attributs de trafic de connexion pendant une fenêtre de deux secondes		
Count	Nombre de connexion à la même machine que la connexion courante durant les dernières secondes	Continu
Attributs des connexions de même machine		
Serror_ rate	Nombres de connexions qui ont des erreurs de SYN	Continu
Rerror_rate	Nombres de connexions qui ont des erreurs de	Continu
Same_srv_rate	Nombre de connexions au même service	Continu
Diff_svv_rate	Nombre de connexions au service différents	Continu
Srv_count	Nombre de connexions au même service que le raccordement courant dans les dernières deux secondes	Continu
Attributs des connexions TCP individuelles		
Durée	Longueur (nombre de secondes) de la connexion	Continu
Protocol_type	Type du protocole , par exemple TCP,UDP	Discret
Service	Service de réseau pour la destination ,par exemple http ,Telnet, ect..	Discret
Src_bytes	Nombre de bytes de donnée de la source a la destination à la source	Continu
Dst_bytes	Nombre de bytes de données de la destination à la source	Continu
Flag	Statut normal ou erreur de la connexion	Discret
Land	1 si la connexion est from /to même host/port ; 0 autrement	Discret
Wrong_fragment	Nombre de « faux » fragments	Continu

Urgent	Nombres de fragments	Continu
Attribut de contenu		
Hot	Nombre de hot indicateurs	Continu
Num_faild_logins	Nombre tentatives d'ouverture Echouées	Continu
Lgged_in	1 si enté avec succès ;0 autrement	Discret
Num_compromised	Le nombre de conditions compromises	continu
Root_shell	1 si rootshell est obtenue ;0 autrement	discret
Su _attempted	1 si la commande su root racine a été essayée ; 0 autrement	discret
Num_root	Nombre d'accès root	continu
Num_file_creations	Nombre d'opérations de création de dossier	continu
Num_shells	Nombre de shell sollicités	continu
Num_access_files	Nombre d'opérations sur des dossiers de contrôle d'accès	continu
Num_outbound_cm	Nombre de commandes venant	continu

Tableau 4.3 : Les différents attributs des données NSL KDD.

4.3. Description de l'application

4.3.1. Prétraitement des données

Le prétraitement porte sur les champs non numériques. Les champs non numériques sont : le type de protocole (TCP, UDP, ICMP), le type de service (aol, auth, bgp, … Z39_50), le drapeau (OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2, S3, SF, SH) et la classe du paquet (Normal ou Anormal).

Pour le type de protocole, nous affectons les valeurs numériques suivantes : TCP=1, UDP=2 et ICMP=3. Nous affectons 1 aux paquets normaux et 0 aux paquets anormaux.

Pour les champs de types service et drapeau, nous pouvons leur attribuer les valeurs numériques par ordre croissant ou décroissant de leur nombre total [58]. A montré les limites

d'une telle approche, il propose de donner les valeurs aléatoires à ces champs. Nous avons dans nos travaux donnés les valeurs aléatoires entre 1 et 10 aux champs de type drapeaux et les valeurs aléatoires entre 1 et 65 aux champs de type services.

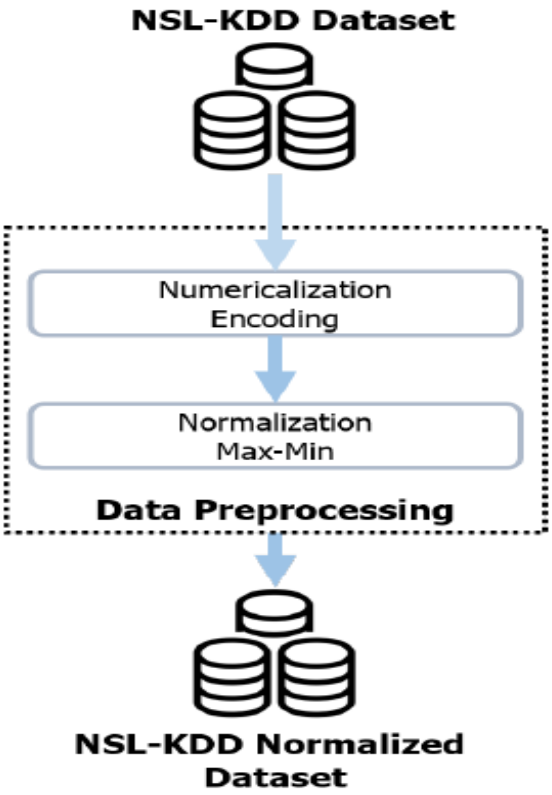

Figure 4.2 : Prétraitement des données.

4.3.2. Formation du classificateur

Le classifier NaiveBayes : La classification NaiveBayes est un type de classification bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un NaiveBayes, ou classifieur naïf de Bayes, appartenant à la famille des classifieurs linéaires (dont le rôle est de classer dans des groupes (des classes) les échantillons qui ont des propriétés similaires, mesurées sur des observations) [59].

```
40      Instances data = new Instances(buf);
41      data.setClassIndex(data.numAttributes() -1);
42      buf.close();
43      NaiveBayes Classifier =new NaiveBayes();
44      Classifier.buildClassifier(data);
45
46      Evaluation Evaluation = new Evaluation(data);
47      Evaluation.crossValidateModel(Classifier, data, 10, new Random(1));
48
49
```

Figure 4.3 : Le classifier NaiveBayes.

4.3.3. Évaluation du Jaccard Index

4.3.3.1. Matrice de confusion

Pour évaluer un système de classification (NaiveBayes), nous utilisons un corpus étiqueté (corpus d'apprentissage et même pour le test) pour lequel on connaît la vraie catégorie de chaque connexion ou instance, et le résultat obtenu par le classifier. Pour ce corpus, nous pouvons construire la matrice de contingence pour chaque solution (Voir Table 4.4), qui fournit 4 informations essentielles :

	Classé comme intrusion	**Classé comme non-intrusion**
Réellement Intrusion	**VP**	**FN**
Réellement Non-Intrusion	**FP**	**VN**

Tableau 4.4 : Matrice de confusion (IDS).

Le Taux de Vrais Positifs (TVP) : correctement prévu pour être une classe positive.

Le Taux de Faux Positifs(TFP) : incorrectement prédit comme une classe positive.

Le Taux de Vrais Positifs (TFN) : incorrectement prévu pour ne pas être une classe positive.

```
689
690     eval.falsePositiveRate(3);
691     eval.falseNegativeRate(3);
692     eval.truePositiveRate(3);
```

Figure 4.4 : Matrice de confusion.

Les sous ensembles d'attributs de la population Pt et la nouvelle population candidate Qt sont évalués à l'aide de la fonction fitness (jaccard index) :

$$\textbf{Jaccard index} = \frac{TP}{TP+FN+FP}$$

4.3.4. Implémentation de NSGA-III

4.3.4.1. Les structures de données

- **Population**

```
ArrayList<Solution> population;
```

Figure 4.5 : Population.

- **Solution**

```
new solution<KDD99.size>();
```

Figure 4.6 : Solution.

- **Les points de références**

```
ArrayList<refpoint> referencePoints=new ArrayList<>();
```

Figure 4.7 : Les points de références.

4.3.4.2. Algorithmes utilisés

On a utilisé un ensemble de procédures pour implémenter NSGA-III, dans ce qui suit, on va présenter les plus importants :

1) Procédure 1 (les points de références)

```
public  void ReferencePoints(ArrayList<refpoint> referencePoints)
    {
          refpoint p1=new refpoint(1, 0, 0);
          refpoint p2=new refpoint(0.66,0.33, 0);
          refpoint p3=new refpoint(0.33,0.66, 0);
          refpoint p4=new refpoint(0,1,0);
          refpoint p5=new refpoint(0.33,0.33,0.33);
          refpoint p6=new refpoint(0,0.66,0.33);
          refpoint p7=new refpoint(0,0.33,0.66);
          refpoint p8=new refpoint(0,0,1);
          refpoint p9=new refpoint(0.33,0,0.66);
          refpoint p10=new refpoint(0.66,0,0.33);
referencePoints.add(p1);referencePoints.add(p2);referencePoints.add(p3);referencePoints.add(p4);referencePoints.add(p5);
referencePoints.add(p6);referencePoints.add(p7);referencePoints.add(p8);referencePoints.add(p9);referencePoints.add(p10);
      }
```

Figure 11.8 : Procédure (GenerateReferencePoints).

2) Procédure 2 (Evaluation des populations)

```
/**================================================ evaluation population ================================================
          public void evaluation(int Tpop)
          { int t=0; int uu=0;
               for(int i =0;i<Tpop;i++)
                  {    Instances data;
                  try {
                      data = new Instances(solution.get(i));
                       data.setClassIndex(data.numAttributes() -1);
                       solution.get(i).close();
                       NaiveBayes b =new NaiveBayes();
                       try {
                          b.buildClassifier(data);
                       } catch (Exception e) {
                          e.printStackTrace();
                       }
                        Evaluation eval = null;
                       try {
                           eval = new Evaluation(data);
                       } catch (Exception e1) {

                           e1.printStackTrace();
                       }
                        try {
                          eval.crossValidateModel(b, data, 10, new Random(1));
                       } catch (Exception e) {e.printStackTrace();
                       }
                        rescla.append(eval.toSummaryString(" result...",true));
        rescla.append(eval.precision(1)+"\n"+eval.falseNegativeRate(1)+"  "+eval.falsePositiveRate(1)+"  "+eval.truePositiveRate(1)+"
fppp= fppp+eval.falsePositiveRate(1); fnnn= fnnn+eval.falseNegativeRate(1);tppp= tppp+eval.truePositiveRate(1);
    pop1.add(eval.falsePositiveRate(1));pop1.add(eval.falseNegativeRate(1));pop1.add(eval.truePositiveRate(1));
                  } catch (IOException e) {
```

Figure 4.9 : Procédure évaluation des populations.

3) Procédure 3 (Domination)

```
public void dominate()
{   int mm=0; int j=0;int i=0;

while(i<rt.size())
 {
    j=0; mm=0;
    while(j<rt.size())
     { if((rt.get(j)<rt.get(i)) && (rt.get(j+1)<rt.get(i+1)) && (rt.get(j+2)<rt.get(i+2)))
        {
            mm=1;
        }
        j=j+3;
     }
     if(mm==0){
         nondominated.add(rt.get(i));
         nondominated.add(rt.get(i+1));
         nondominated.add(rt.get(i+2)); }
     else{
         dominated.add(rt.get(i));
         dominated.add(rt.get(i+1));
         dominated.add(rt.get(i+2)); }i=i+3;
 }
 fronts.add(a,nondominated);

}
```

Figure 4.10 : Procédure la dominance.

4) Procédure 4 (Front du Pareto)

```
    void front()
    {
ArrayList<Double> indice;int mm=0; int j=0; int i=0;

while(dominated.size()!=0)
       { mm=0;  j=0;   i=0;
            nondominated1 = new ArrayList<>(); indice=new ArrayList<>();
           while(i<dominated.size())
           {  j=0; mm=0;
           while(j<dominated.size())
            {
   if((dominated.get(j)<dominated.get(i)) && (dominated.get(j+1)<dominated.get(i+1)) && (dominated.get(j+2)<dominated.get(i+2)))
               {
                   mm=1;
               }
               j=j+3; }
            if(mm==0)
            {   nondominated1.add(dominated.get(i)); nondominated1.add(dominated.get(i+1)); nondominated1.add(dominated.get(i+2));
                indice.add(dominated.get(i)); indice.add(dominated.get(i+1)); indice.add(dominated.get(i+2));        }
            i=i+3;
           }
           fronts.add(a,nondominated1);
            if(indice.size()!=0)
             {
               for(int t=0;t<indice.size();t++)
                 {   for(int i1=0;i1<dominated.size();i1++)
                    {  if(indice.get(t)==dominated.get(i1))
                          {
                           dominated.remove(i1); } }
                            } }
            else
             {  System.out.println(" no solution"); }
            a++;
```

Figure 4.11 : Procédure Pareto fronts

5) Procédure 5 (Normalisation)

```
public void normalize(ArrayList<Double> Stfit)
{
    ArrayList<Double> translateobjectifs=translateobjectif(Stfit);
    ArrayList<ArrayList<Double>> extremepoints=findExtremePoints(translateobjectifs);
    ArrayList<Double> intercepts=constructHyperplane(extremepoints);
  double f1,f2,f3; int i=0;
  while(i<Stfit.size())
  {
     f1=translateobjectifs.get(i)/Math.abs(intercepts.get(0)-z.get(0));

    f2=translateobjectifs.get(i+1)/Math.abs(intercepts.get(1)-z.get(1));

    f3=translateobjectifs.get(i+2)/Math.abs(intercepts.get(2)-z.get(2));

    Stfit.set(i, f1);
    Stfit.set(i+1, f2);
    Stfit.set(i+2, f3);
   i=i+3;
  }
```

Figure 4.12 : Procédure Normalisation.

6) Procédure 6 (Idéal point)

```
void idealpoint()
{ double min1; double min2;
  double min3 ;
        min1=fronts.get(0).get(0); int i=3;
        while(i<fronts.get(0).size())
        {
            if(fronts.get(0).get(i)<min1 )
            {
              min1=fronts.get(0).get(i);
            }
            i=i+3;
        }

        min2=fronts.get(0).get(1);i=4;
        while(i<fronts.get(0).size())
        {
            if(fronts.get(0).get(i)<min2 )
            {
              min2=fronts.get(0).get(i);
            }
            i=i+3;
        }

        min3=fronts.get(0).get(2);i=5;
        while(i<fronts.get(0).size())
        {
            if(fronts.get(0).get(i)<min3)
            {
              min3=fronts.get(0).get(i);
            }
            i=i+3;
        }
        z.add(min1); z.add(min2); z.add(min3);      }
```

Figure 4.13: Procédure Idéal Point.

7) Fonction 1 (transmission les objectifs)

```
ArrayList<Double> translateobjectif(ArrayList<Double> Stfit)
{   Double X, Y ,Z ;
   Double x1 ,y1,z1;
    idealpoint();
    x1=z.get(0);
    y1=z.get(1);
    z1=z.get(2);
    int i=0;
    while(i<Stfit.size()) {
        X =Stfit.get(i)-x1;
        Y = Stfit.get(i+1)-y1;
        Z =Stfit.get(i+2)-z1;

        Stfit.set(i, X);  Stfit.set(i+1, Y);  Stfit.set(i+2, Z);
    i=i+3;
    }
    return Stfit;
}
```

Figure 4.14 : Fonction (translate objectif).

8) Fonction 2 (trouver des points extrêmes)

```
   ArrayList<ArrayList<Double>> findExtremePoints(ArrayList<Double> population)
   {
ArrayList<ArrayList<Double>> extremePoints =new ArrayList<ArrayList<Double>>(3);
      ArrayList<Double> min_indv = null;
      for (int k=0;k< 3; k++)
      {   double min_ASF = Double.MAX_VALUE; int i=0;
          while( i < population.size())
            {
              double asf = ASF(population.get(i),population.get(i+1),population.get(i+2), k);
                if ( asf < min_ASF )
                {   min_indv = new ArrayList<Double>();
                    min_ASF = asf;
                    min_indv.add(population.get(i));
                    min_indv.add(population.get(i+1));
                    min_indv.add(population.get(i+2));
                }
                i=i+3 ;
            }
          extremePoints.add(min_indv);
          System.out.println("ext point= "+min_indv.get(0) +" "+min_indv.get(1)+" "+min_indv.get(2));
      }
      System.out.println("ext point size= "+extremePoints.size());
          return extremePoints;
   }
```

Figure 4.15 : Fonction (find extrême points).

9) Fonction 3 (association)

```
public ArrayList<assoc> associate(ArrayList<Double> population)
{
    ArrayList<Integer>pi=new ArrayList<>();
    ArrayList<Double>dis=new ArrayList<>();
    ArrayList<assoc>association=new ArrayList<>();assoc ac;
    int t = 0;
    while( t <population.size())
    { int min_rp = -1;
      double min_dist = Double.MAX_VALUE;
      for (int r = 0; r < referencePoints.size(); r++)
      { double d = perpendicularDistance(referencePoints.get(r),population.get(t),population.get(t+1),population.get(t+2));

        if (d < min_dist)
        {min_dist=d;
         min_rp = r;
        }
      }
      ac=new assoc(t, min_rp,min_dist);
      association.add(ac);
     t=t+3;
    }

    System.out.println("-************** Resultat de l'association **************- \n");
    System.out.println(" ID    Reference point   Distance \n");
    for(int i=0;i<association.size();i++)
    {
        System.out.println( " "+association.get(i).id_sol+"\t  "+  association.get(i).pref+"\t\t"+   association.get(i).dis+" \r\n"
    }

    return association;
}
```

Figure 4.16 : Fonction (associate).

10) Fonction 4 (niching)

```
public void niching(ArrayList<Double>St,int K,ArrayList<assoc>association,int ptsize)
    {   ArrayList<Integer>p=new ArrayList<>(); int k=0;ArrayList<Integer>Jmin=new ArrayList<>(); ArrayList<refpoint>copyreferencepoint=
        copy_refpoint(referencePoints, copyreferencepoint);create_p(association, copyreferencepoint, ptsize, p);// create p
       System.out.println("-************ La Selection des  "+K+" solutions  *************- \r\n");
        while (k<K/3)
        { Jmin=minimum_p(p);int jbar=choisir_jbar(Jmin); int indice = 0;
          if(p.get(jbar)==0)
          {   if(copyreferencepoint.get(jbar).membres_fl.size()!=0)
              {double mindis=copyreferencepoint.get(jbar).membres_fl.get(0).dist;
                for(int i=1;i<copyreferencepoint.get(jbar).membres_fl.size();i++)
                 {  if(copyreferencepoint.get(jbar).membres_fl.get(i).dist<mindis)
                    { mindis=copyreferencepoint.get(jbar).membres_fl.get(i).dist;indice=i;} }
                pop1.add(rt.get(copyreferencepoint.get(jbar).membres_fl.get(indice).id_sol));
                pop1.add(rt.get((copyreferencepoint.get(jbar).membres_fl.get(indice).id_sol)+1));
                pop1.add(rt.get((copyreferencepoint.get(jbar).membres_fl.get(indice).id_sol)+2));
                copyreferencepoint.get(jbar).membres_fl.remove(indice);
                p.set(jbar,p.get(jbar)+1);
                k++; } else
             {   copyreferencepoint.remove(jbar);
                 p.remove(jbar);
              }
          }else if(p.get(jbar)>=1)
          {    if(copyreferencepoint.get(jbar).membres_fl.size()!=0)
              {  Random r = new Random();
                 int nbr=r.nextInt(copyreferencepoint.get(jbar).membres_fl.size());
                 pop1.add(rt.get(copyreferencepoint.get(jbar).membres_fl.get(nbr).id_sol));
                 pop1.add(rt.get((copyreferencepoint.get(jbar).membres_fl.get(nbr).id_sol)+1));
                 pop1.add(rt.get((copyreferencepoint.get(jbar).membres_fl.get(nbr).id_sol)+2));
                 p.set(jbar,p.get(jbar)+1); copyreferencepoint.get(jbar).membres_fl.remove(nbr); k++;}
          else{ copyreferencepoint.remove(jbar);
                p.remove(jbar);
              }
         }
```

Figure 4.17 : Fonction (Niching).

11) Procédure 7 (Remplacement)

```
void remplacement()
    {   ArrayList<Double>translate=new ArrayList<Double>(); int K; int i=0; int N=pop1.size();
       do{
           for(int j=0;j<fronts.get(i).size();j++)
           {
            st.add(fronts.get(i).get(j));
            translate.add(fronts.get(i).get(j));
           } i++;
         }while(st.size()!=N && st.size()<N);
         if(st.size()==N)
       {   for(int ind=0;ind<st.size();ind++)
           {  pop1.add(st.get(ind));    }
       } else if(st.size()>N)
        {   int in=0;
             for(int ind=0;ind<i;ind++)
            {   for(int j=0;j<fronts.get(ind).size();j++)
               {
              translate.set(in, fronts.get(ind).get(j));
               in++;
               } }
            int index=0;
           for(int ind=0;ind<i-1;ind++)
           {    for(int j=0;j<fronts.get(ind).size();j++)
               {
                pop1.add(fronts.get(ind).get(j));
                index++;
               }                   }
           K=N-index;
         System.out.println("------------------------------NORMALISATION------------------------------\n ");
         normalize(translate);
        System.out.println("-------------------------------ASSOCIACION--------------------------------\n");
       ArrayList<assoc>association=associate(translate);
        niching(translate, K, association, pop1.size());
```

Figure 4.112 : Procédure (Remplacement).

12) Ecrire dans le fichier texte

```
try {
      File file = new File("C:\\Users\\13\\Desktop\\projet_master\\sol.txt");
      File file1 = new File("C:\\Users\\13\\Desktop\\projet_master\\solution_final.txt");
      File file2 = new File("C:\\Users\\13\\Desktop\\projet_master\\solk1.txt");

      // créer le fichier s'il n'existe pas
      if (!file.exists()) {
       file.createNewFile();
      }
      if (!file2.exists()) {
           file2.createNewFile();
          }

      BufferedWriter bw = new BufferedWriter(new FileWriter(file));
      BufferedWriter bw2 = new BufferedWriter(new FileWriter(file2));
      BufferedWriter bw1 = new BufferedWriter(new FileWriter(file1));

     bw2.write("FP /    \t\t\t\t\t FN /  \t\t\t\t\t TP   \n");
      bw1.write("FP /    \t\t\t\t\t FN /  \t\t\t\t\t TP   \n");
      bw.newLine();

for(int i1=0;i1< solution_optimal.size();i1++)
{  int j1=0;
bw.write("---------------------------------------Solution Optimal du generation nbr "+i1+"---------------------------------------
bw.newLine();
bw.write("FP    \t\t\t\t\t FN    \t\t\t\t\t TP   \n");
while(j1< solution_optimal.get(i1).size())
{

        Double fp4=solution_optimal.get(i1).get(j1);
        Double fn4=solution_optimal.get(i1).get(j1+1);
```

Figure 4.113 : Ecrire dans le fichier texte.

4.4. Interface graphique

Au lancement de notre application, une interface graphique apparait en premier aux utilisateurs, elle est composé de :

1. Une partie appelées NSL KDD 99 : Dans cette interface nous pouvons :
- Ouvrire data.
- Prétraitement data.
2. Une partie appelées Paramètres NSGA-III qui permettent a l'utilisateur de spécifiée ses propres paramètres.
3. Une partie appelées résultat qui affiche :
    - les résultats du clasifieur naivebayes.
    - les résultats de la sélection avec NSGA-III
    - le nombre de solutions optimales finales.
    - le nombre de génération.
    - Le temps d'exécution.
    - Un tableau affiché les solutions optimales.

4. Quatre buttons :
    - Un button pour Sélectionner les fichiers enregistrés.
    - Un button pour afficher les fronts.
    - Un button pour afficher les solutions optimales.
    - Un button pour afficher Jaccard Index pour toutes les solutions.

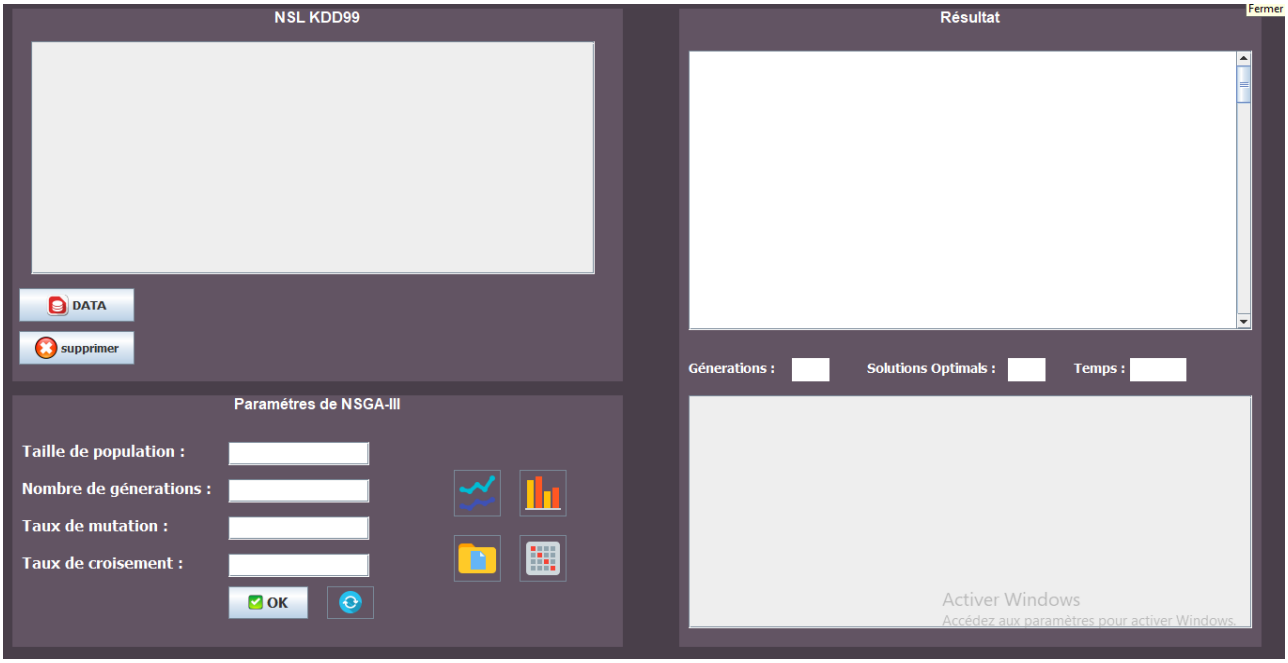

Figure 4.20 : Interface graphique (avant l'exécution).

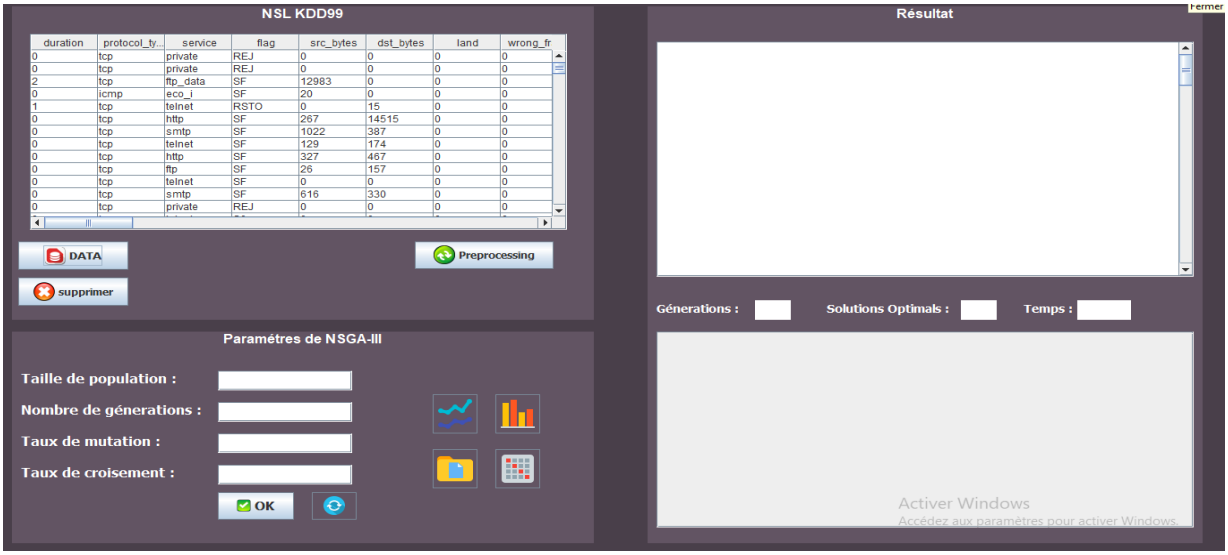

Figure 4.21 : Import Data.

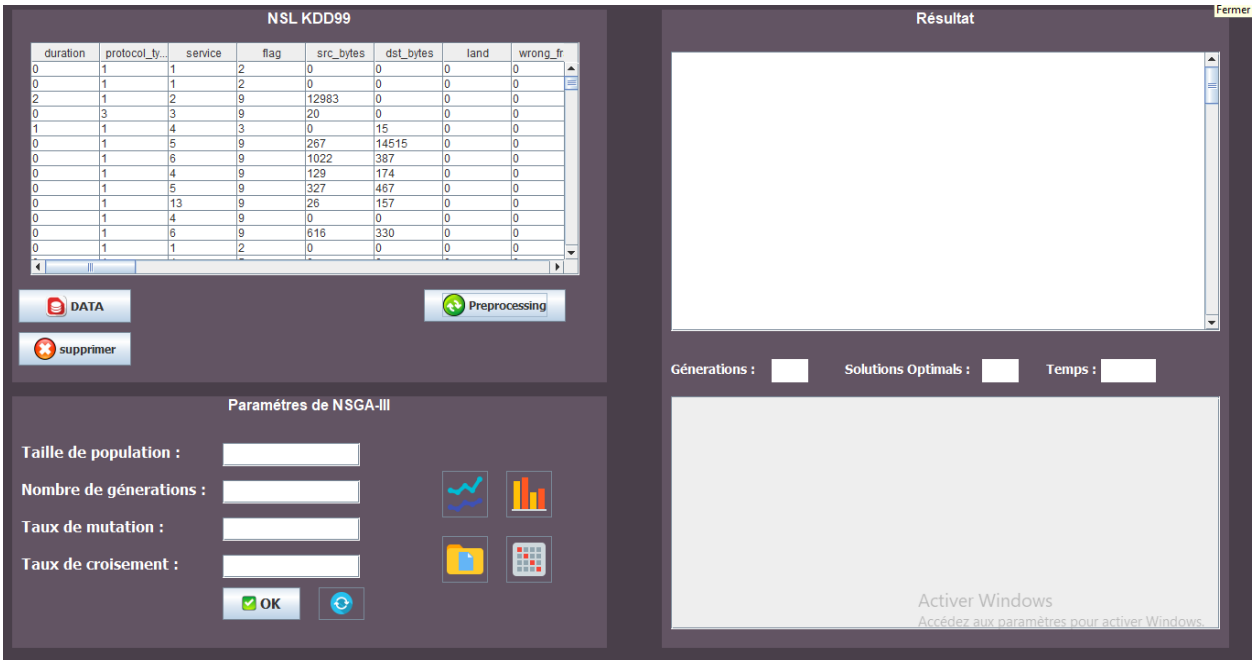

Figure 4.22 : Prétraitement des données.

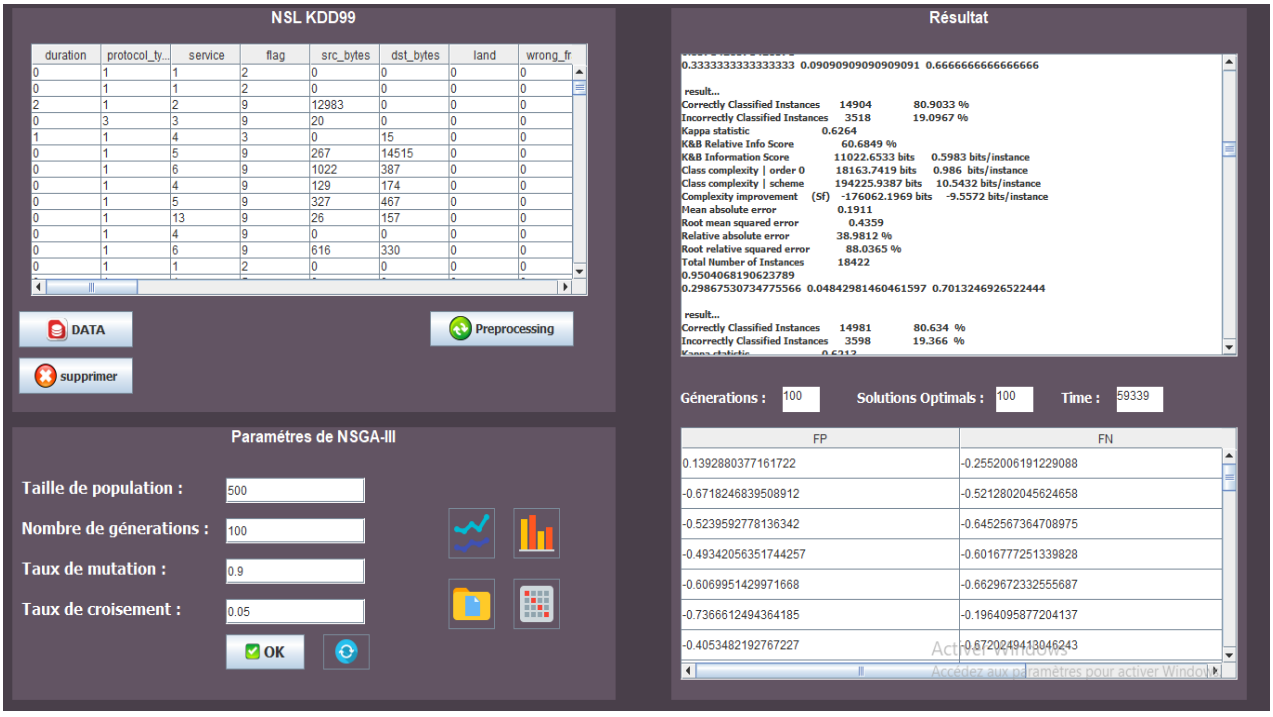

Figure 4.23 : Interface graphique (Après l'exécution).

```
Correctly Classified Instances      113858               90.3829 %
Incorrectly Classified Instances     12115                9.6171 %
Kappa statistic                          0.806
K&B Relative Info Score            10150194.7334 %
K&B Information Score               101151.3878 bits      0.803  bits/instance
Class complexity | order 0          125537.9399 bits      0.9965 bits/instance
Class complexity | scheme           737014.4962 bits      5.8506 bits/instance
Complexity improvement     (Sf)    -611476.5563 bits     -4.854  bits/instance
Mean absolute error                      0.0965
Root mean squared error                  0.3058
Relative absolute error                 19.3947 %
Root relative squared error             61.3067 %
Total Number of Instances           125973

------------------------------------------------------------------------------------

      falseNegativeRate      |       falsePositiveRate      |      trueNegativeRate

------------------------------------------------------------------------------------

    0.13358348968105066       0.063599780229571        0.936400219770429
------------------------------------------------------------------------------------
```

Figure 4.24 : Résultat du NaiveBayes.

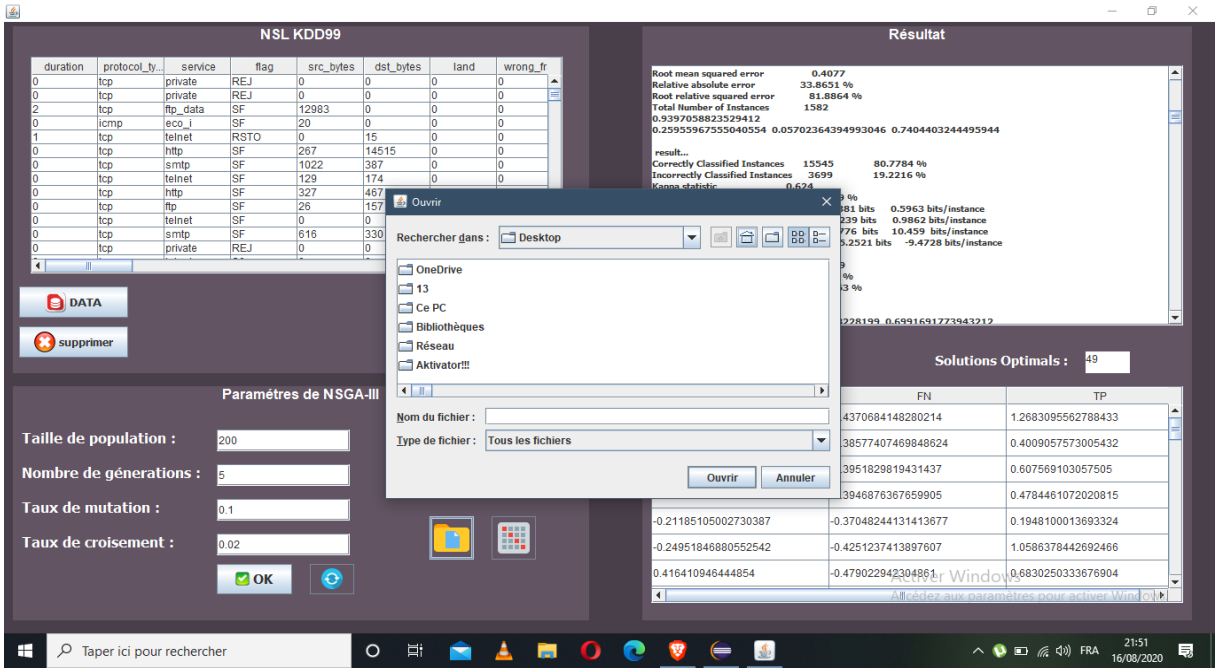

Figure 4.25 : Sélectionné les fichiers enregistrés.

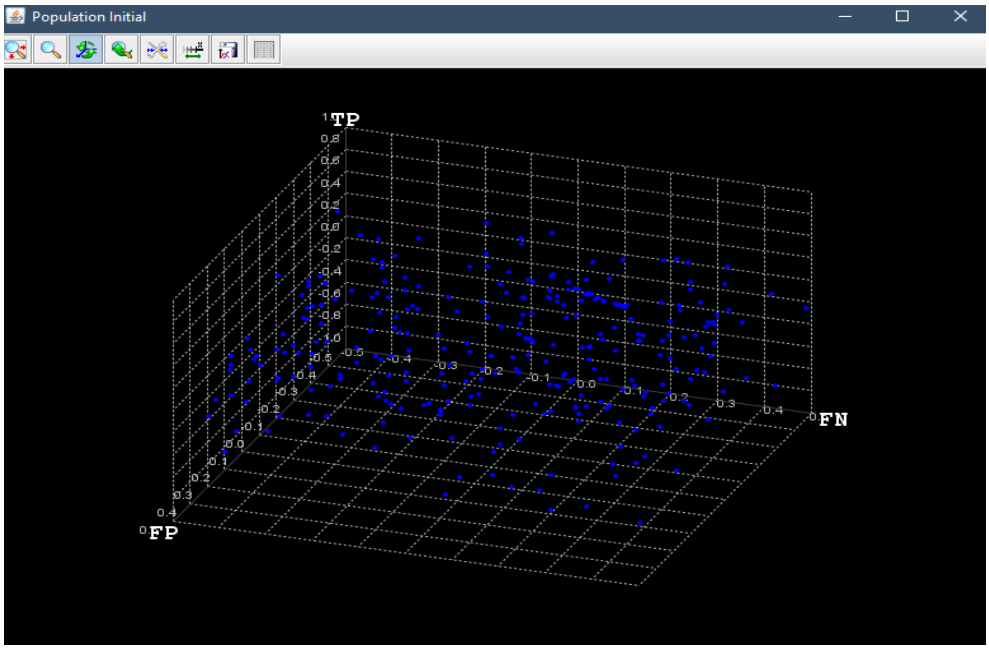

Figure 4.26 : Représentation la population Initial.

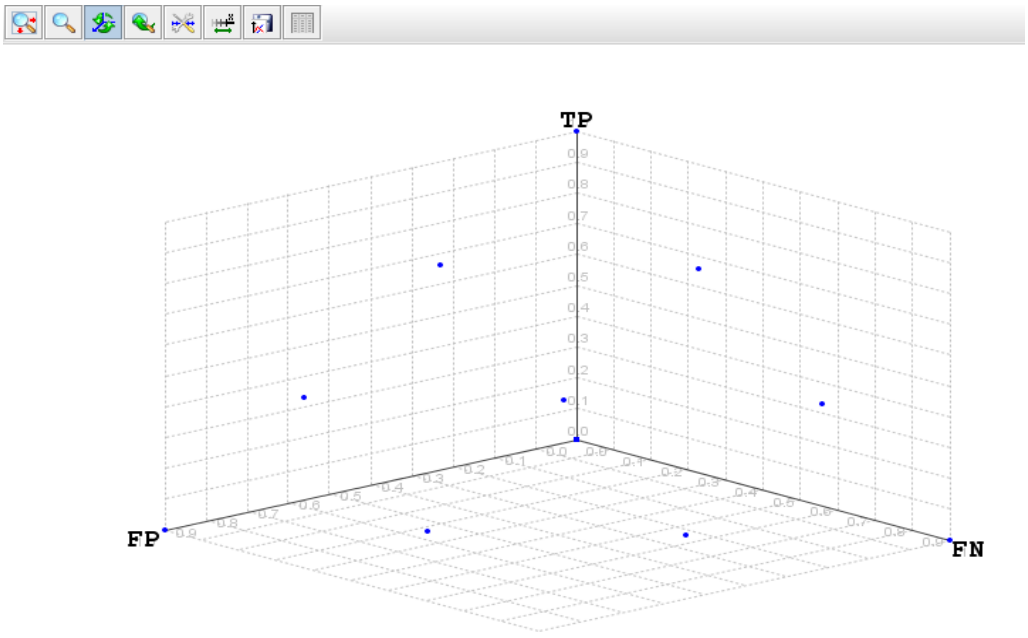

Figure 4.27 : Représentation Les points de références.

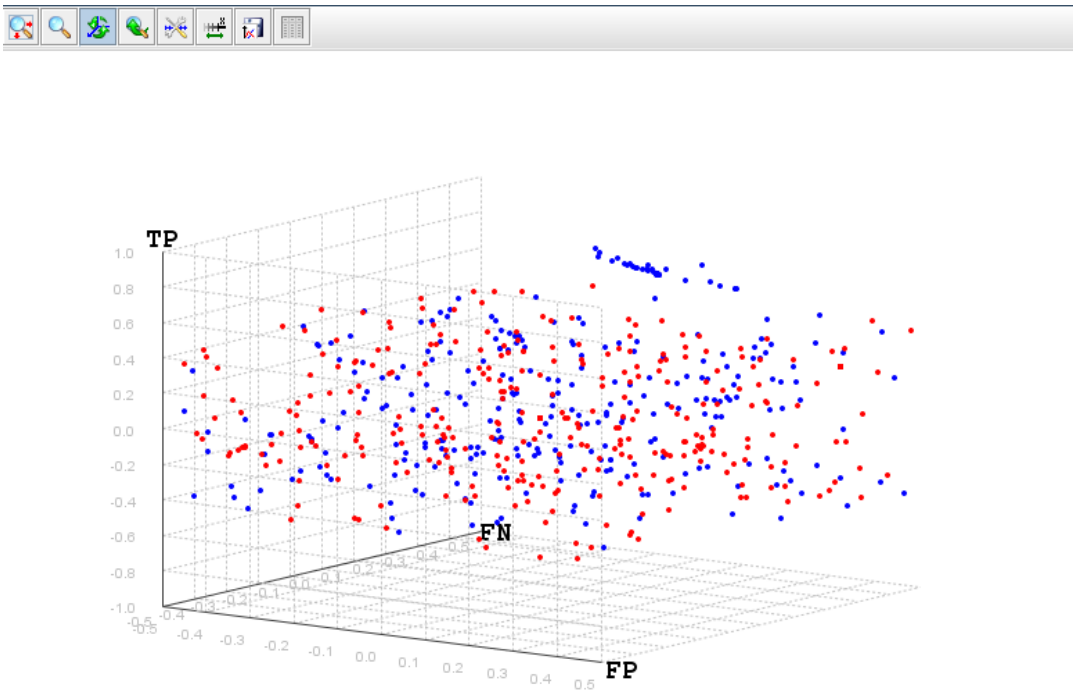

Figure 4.28 : Représentation la population RT (Pt+Qt).

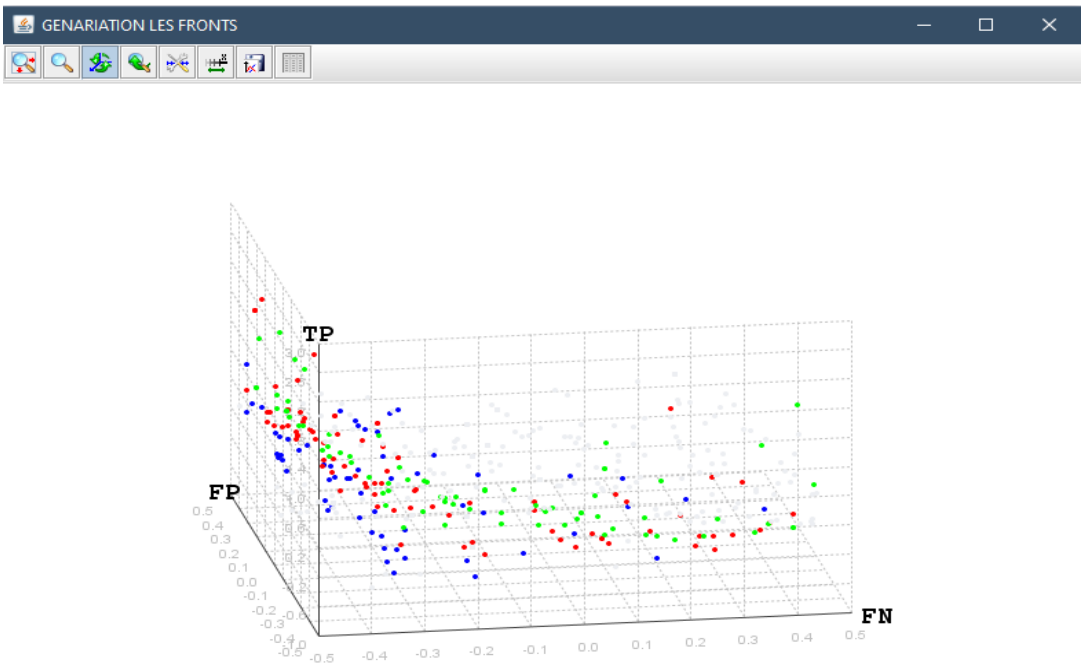

Figure 4.29 : Représentation des fronts.

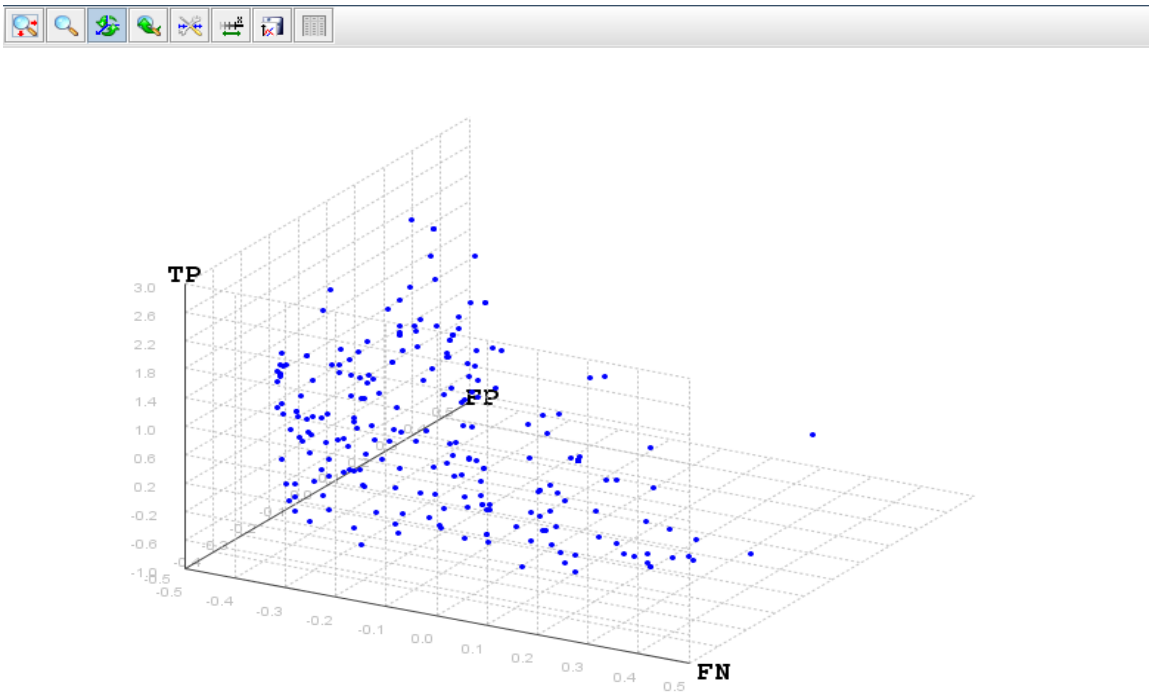

Figure 4.30 : St avant normalisation.

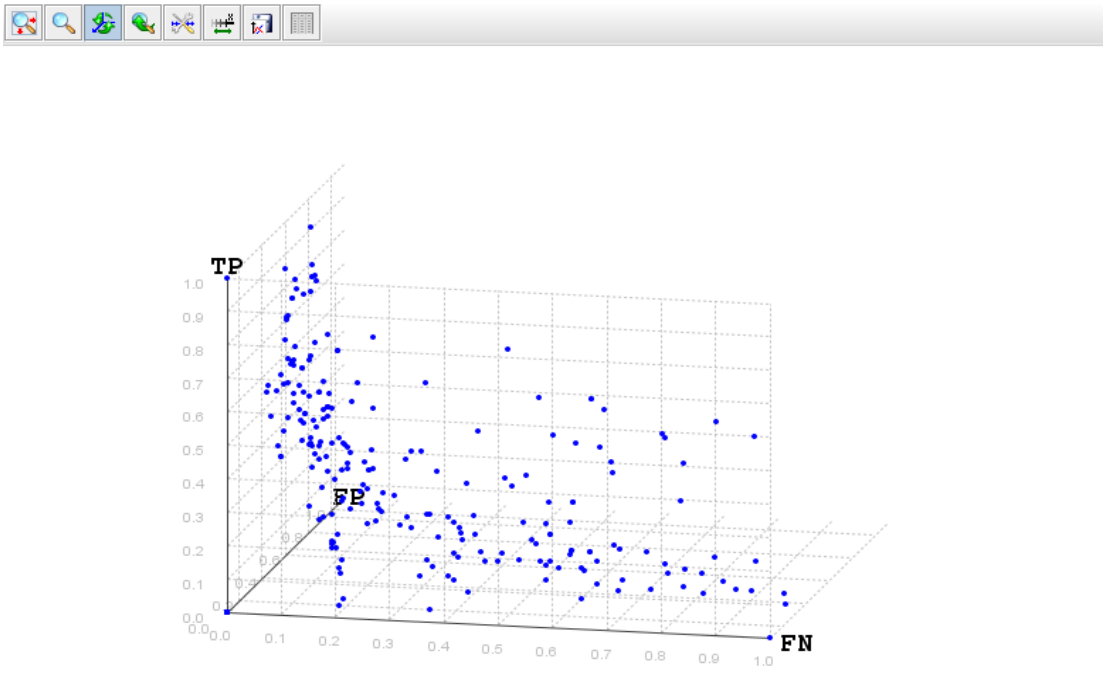

Figure 4.31 : St après normalisation.

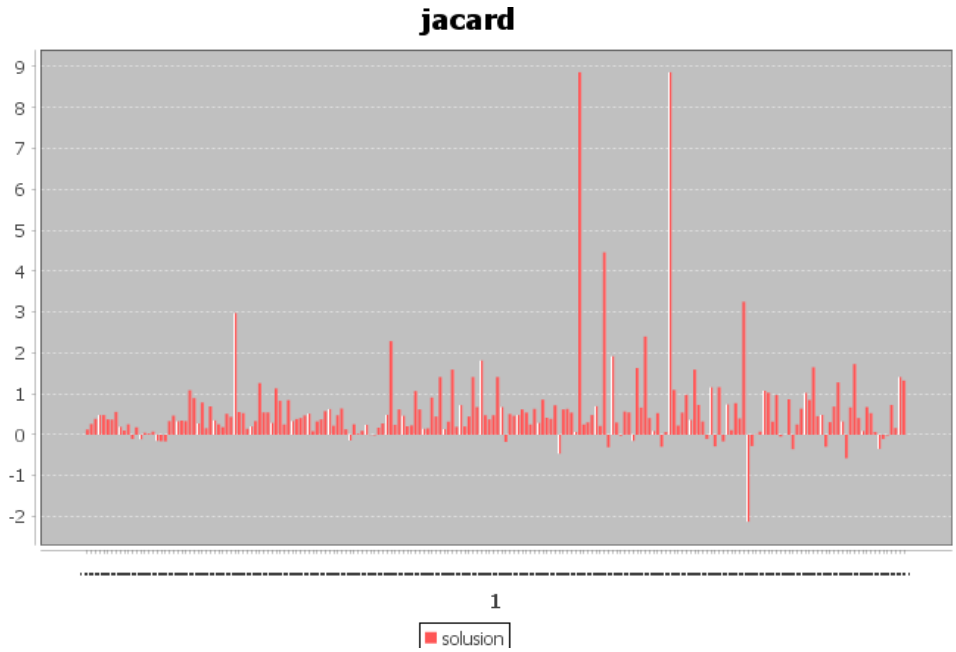

Figure 4.32 : Interface statique.

4.5. Résultats obtenus

Dans cette section, nous montrons les résultats obtenus par l'algorithme NSGA-3 est utilisée pour obtenir un sous-ensemble de fonctionnalités pour IDS.

La phase 1 : il est nécessaire de déterminer les sous-ensembles de fonctionnalités optimaux dans les fronts Pareto. Nous utilisons NSGA-III pour entraîner le classifieur avec validation croisée afin d'obtenir des sous-ensembles de fonctionnalités avec des index Jaccard.

Pour nous expérience nous avons utilisées :

Parameters	Valeur
La taille de population	10-50-500
Nombre de generation	10-50-500
Probabilité de mutation	0.05
Propabilité de croisement	0.6

Tableau 4.5 : Paramètres des algorithmes.

- la taille de population =10 / Génération =10

Génération	Les solutions optimales		
	FP	**FN**	**TP**
1	0.05119351448731422	0.30408438061041293	0.6959156193895871
	0.04945054945054945	0.1981981981981982	0.8018018018018018
	0.04797047970479705	0.22508038585209003	0.77491961414791
	0.04995274740110706	0.2946245383668445	0.7053754616331556
	0.1962892697732388	-0.3020646753222138	0.3634049218142418
	0.05219432300318494	-0.16331398064224456	0.47589263950556615
	0.17133201020289468	0.15577461701959883	0.16936400485741943
10	0.2808935325248648	-0.20061589281161732	0.18681957817576605
	-0.2601150951585723	-0.39796805817460464	0.1824650041042521
	0.3327747265282245	-0.3133480094536172	0.18042151930433196
	-0.3275230820371690	-0.12519586826110154	0.1545687707661168
	-0.3984343686659968	-0.04706596188640255	0.1888995899904349
	0.15694074044499262	-0.18043205111886385	0.40858548302748976
	-0.2390155906981383	-0.14357729945736586	0.04220653256637408

Tableau 4.6 : les solutions initiales et finales pour la taille de population = 10.

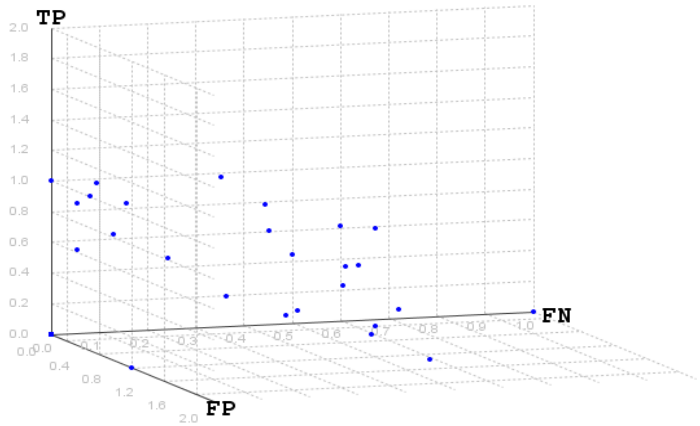

Figure 4.33 : les solutions finales pour la taille de population = 10.

- la taille de population =50 / Générations = 50

Génerétion	**Les solutions optimales**		
	FP	**FN**	**TP**
1	-0.2984334641843034	-0.3204487095562799,	0.23879098560699863
	-0.0241586704660408	0.4226579422138048	-0.1035088812174255
	-0.2092562494989137	-0.19782235959596206	0.1087255992086471
	-0.0135556452364191	-0.20713260901156672	0.08880559716386249
	-0.3642652993950608	-0.3471094376947583	0.33518249900625163
	-0.3447499224388837	0.33945560142724984	0.08658644128997561
	0.11013968835119481	0.46760927004335373	0.281491827661163
	0.4101650951192538	-0.3244446697404517	-0.1443948883115926
	0.29309881653200853	0.06697968865003456	-0.4028163436571889
	0.2637727233092594	0.15159616931695574	0.1285675439297227
	0.14615787215595022	-0.199895233717313	0.08229732298906978
	-0.3417604622541578	0.455309149967282	0.003897748214683139
	-0.2340239372080358	-0.08811094947920739	0.012538580486424089

50	-0.5744696482657065	-0.2056751007313581	0.2909771054828734
	-0.0402676062919076	0.08789215793150859	-0.5344227995130519
	-0.073660746671399	-0.5123181459303813	-0.437095568147489
	0.19844481766103347	-0.15050955110943712	-0.4731369425114799
	-0.0905677046878693	-0.5144453500858384	-0.31262771464438815
	-0.4190377505055972	-0.46642289342959986	-0.33992605152507666
	-0.5485535961470067	-0.2961391029026993	0.2571074167274057
	0.0956434036647850	-0.4908155628012229	0.057214464717597543
	-0.446052475258396	-0.1948274381694666,	0.24133848758041215
	-0.4667467731333155	-0.0895202754593907	-0.4925380611905516
	-0.5791628832704003	0.24035027579544074	-0.06838993037014458
	0.38568890312118864	-0.16393201768221932	-0.42713713374825557
	-0.170436002256386	-0.44601866040790084	-0.406797322074794
	-0.1113075866572814	-0.47150186922663684	-0.3243411390306771
	-0.4851747598182057	-0.10140433940973548	-0.27108735871271294
	-0.2228733417414993	0.3466371345684016	-0.4953005338827383
	-0.373444501191743	-0.13389848365050339	-0.4104408982109389

Tableau 4.7 : les solutions initiales et finales pour la taille de population = 50.

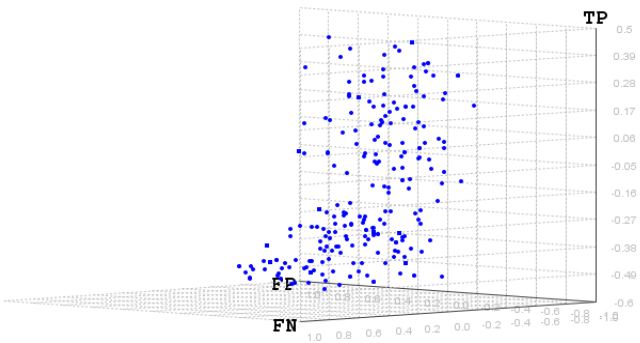

Figure 4.34 : les solutions finales pour la taille de population = 50.

- la taille de population =500 / Génération = 500

Génération	**Les solutions optimales**		
	FP	**FN**	**TP**
1	-0.4645062677265984	-0.0556707315692212	-0.3239087812033118
	-0.3680968262653007	-0.4116252930693004	0.05254128308421946
	-0.4927677073790188	-0.2124488824585894	0.35185627606341785
	-0.4769794476880886	-0.3778480027068255	-0.2127020149998758
	-0.4934933604412814	0.217942291404728	-0.1091322793302022
	0.3124888125473535	-0.3147638371800674	-0.49882129840736
	0.04266974936585699	-0.3190472550753807	-0.4684941458861408
	0.067085460201314	-0.4258722977627686	-0.3888059526968569
	-0.2343443106897574	-0.4792292313523574	-0.16311309280397424
	-0.3268807990411493	-0.4898893999623424	0.4705024093886846
	-0.3199681886387028	-0.4279611300630105	-0.14335829749206985
	0.04500924141596685	-0.4191309830646832	-0.4738384660939545
	0.08111949568589749	-0.44965837480834714	-0.2561552218565568
	0.44054276083036426	-0.39751542373451354	-0.4781648929428408
	-0.49069063355653186	0.449067495525989	-0.30437620970960155
	-0.09104341326781451	-0.4815435565912749	0.4129610726131008
	-0.2545113662823013	-0.42069056322224696	-0.26628105545248415
	-0.23898720700134402	-0.4755842955110785	0.22318888456600927
	-0.4781728585773708	0.03353288870639559	-0.4141741205186984
	-0.48973570073824746	-0.3214266097509759	0.4935850208995647
	-0.3957578208297925	-0.4095964230492546	-0.4631736626874102
	0.4579311896482775	-0.46067347331177233	-0.35771640730269016
	-0.4945710606205407	0.18606713732401803	0.24538774967271526
	-0.35439264913947544	0.09560440831531491	-0.46348843263748307
	0.4224081290920154	-0.4984030736719969	-0.2968686918927459
	0.13646016995957055	-0.3068364696382385	-0.49936328793866236
	-0.47729417201105084	0.09472934468050498	-0.43986917331208775
	0.17081050476677007	-0.45899445103488257	-0.3197599205530641
	0.4717157213934222	-0.4607746418329609	-0.4140234934731022

	-0.4812819544872169	-0.13439592632385067	-0.29950127382707215
	-0.4135394097478472	-0.3534303380039172	-0.3327933030257164
	-0.12303999274724442	-0.4639532311248493	-0.2180847201669489
	-0.02943282263902846	-0.46969253770831076	-0.18532543476942087
	-0.24458762176501103	-0.3082557481546381	-0.49434371106436836
	-0.45841556516738047	-0.4410196194820042	0.3153756088752365
	-0.35032832241428014	-0.46213270637031356	0.09548203862022775
	0.11896818023947553	-0.4717754109145349	-0.3159079085956756
	0.17081050476677007	-0.45899445103488257	-0.3197599205530641
	0.4717157213934222	-0.4607746418329609	-0.4140234934731022
	-0.4812819544872169	-0.13439592632385067	-0.29950127382707215
	-0.4135394097478472	-0.3534303380039172	-0.3327933030257164
	-0.12303999274724442	-0.4639532311248493	-0.2180847201669489
	-0.02943282263902846	-0.46969253770831076,	-0.18532543476942087
	-0.24458762176501103	-0.3082557481546381	-0.49434371106436836
	-0.45841556516738047	-0.4410196194820042,	0.3153756088752365
	-0.35032832241428014	-0.46213270637031356	0.09548203862022775
	0.11896818023947553	-0.4717754109145349	-0.315907908595675
	0.17081050476677007	-0.45899445103488257	-0.3197599205530641
	0.4717157213934222	-0.4607746418329609	-0.4140234934731022
	-0.4812819544872169	-0.13439592632385067	-0.29950127382707215
	-0.4135394097478472	-0.3534303380039172	-0.3327933030257164
	-0.12303999274724442	-0.4639532311248493	-0.2180847201669489
	0.029432822639028466	-0.46969253770831076	-0.18532543476942087
	-0.24458762176501103	-0.3082557481546381	-0.49434371106436836
	-0.45841556516738047	-0.4410196194820042	0.3153756088752365
500	-0.3438800555121922	-1.0387730885280282	-0.2719153468761697
	-1.020861400057072	-0.3225839659742921	-0.1406216695426042
	-0.4035117355556357	-1.0195136995128529	-0.4157909422460791
	-0.9102160385309533	-0.6810327420190323	-0.6439122010367102
	-0.0064985826323975	-0.8368827802634727	-0.7462014270415476
	-0.3552301543869725	-0.8408518089905748,	-0.6463190332872911

	-0.6639796280513427	-0.9611602293590435,	-0.5362653908659089
	-0.8293297929228998	-0.7127392436307917,	-0.6191327714821514
	-0.7506236473028657	-0.6336712218154718	-0.7453330606486354
	-0.1182327599087698	-0.9653833449837048	-0.6574248827317216
	-0.0944406562019236	-0.9905793351263474	-0.6126317064032271
	-0.9972395899881367	-0.7253399271309697	0.03952922199372447
	-0.9809890105671775	-0.7184667545967346	-0.0883774253520863
	-0.3903240257831847	-0.8025432067873093	-0.7064390829662642
	-0.1999312708429777	-0.8725196184238578	-0.7451111454676232
	-0.9935527458413481	-0.10790277135866866	-0.5221388515982502
	-0.9446991500135006	-0.9230583802389665	-0.5839686893588467
	-0.7922243834390347	-0.9323980727583989	-0.6289000122286228
	-0.8799598430481052	-0.06004731408713182	-0.723793975797145
	-0.8863347244398458	-0.9953988313824049	-0.2999921853859582
	-0.7834864461732347	-0.948330413386871	-0.363512089706076
	-0.5580489294983656	-0.9995480265743507	0.13407605509562698
	-0.3266519729188144	-0.8895887107994793	-0.7137462265201784
	-0.8539682856020582	-0.06252642480101878	-0.7373861854823457
	-0.4891264672593695	-0.9696274867430142	-0.3024282140279606
	-0.6852383917276308	-0.7531148800449575	-0.704816567604724
	-0.8144329145053736	-0.5595862134656607	-0.698171896471406
	-0.988862741936387	-0.9506790812982459	0.06415849283690103
	-0.3089505216490144	-0.40914349207655987	-0.7485352912295178
	-0.972635214488809	-0.6336976825994819	-0.1997230677596527
	-0.2013421616601058	-0.9651590137203367	-0.6498312152238572
	-0.9371572263382772	-0.6859643983512075,	-0.6438510967934405
	-0.9936288495422965	-0.12527245039577617	-0.1937838812600516
	-0.9555768289466231	-0.8317837995464091	-0.4320166731463734
	-0.9458848893475108	-0.6960035199166436	-0.5085505853379542
	-0.9719043757998794	-0.6150353396455693	-0.641143841694059
	-0.4035117355556357	-1.0195136995128529	-0.4157909422460791
	-1.020861400057072	-0.3225839659742921	-0.1406216695426042
	-0.3438800555121922	-1.0387730885280282	-0.2719153468761697

	-0.9102160385309533	-0.6810327420190323	-0.6439122010367102
	-0.7925810792501382	-0.6265591283844918	-0.6730247647205561
	-0.872153610622858	-0.8555982905683608	-0.5846641819931974

Tableau 4.8 : les solutions initiales et finales pour la taille de population = 500.

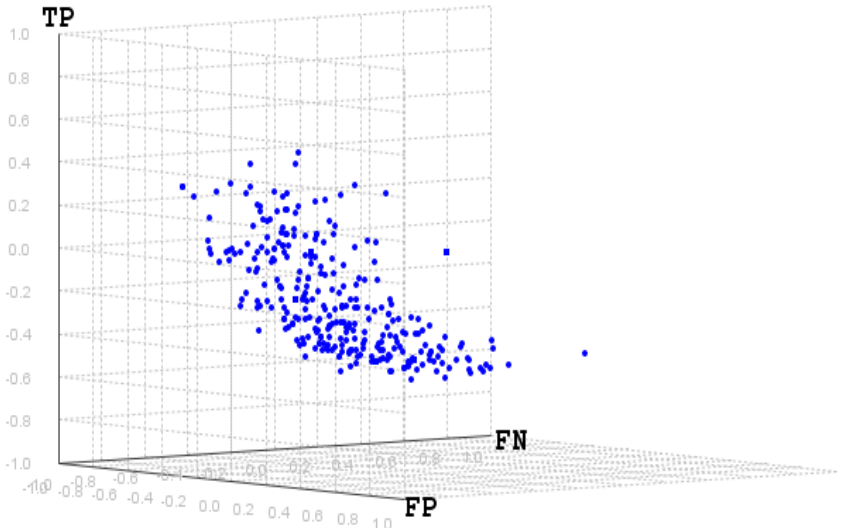

Figure 4.35 : les solutions finales pour la taille de population = 500.

Résultat 1 :

Les sous ensemble obtenus après l'application de l'algorithme NSGA-III sont caractérisés par:

- la réduction des solutions redondantes.

-la réduction des solutions dominantes.

-la réduction de la dimensionnalité.

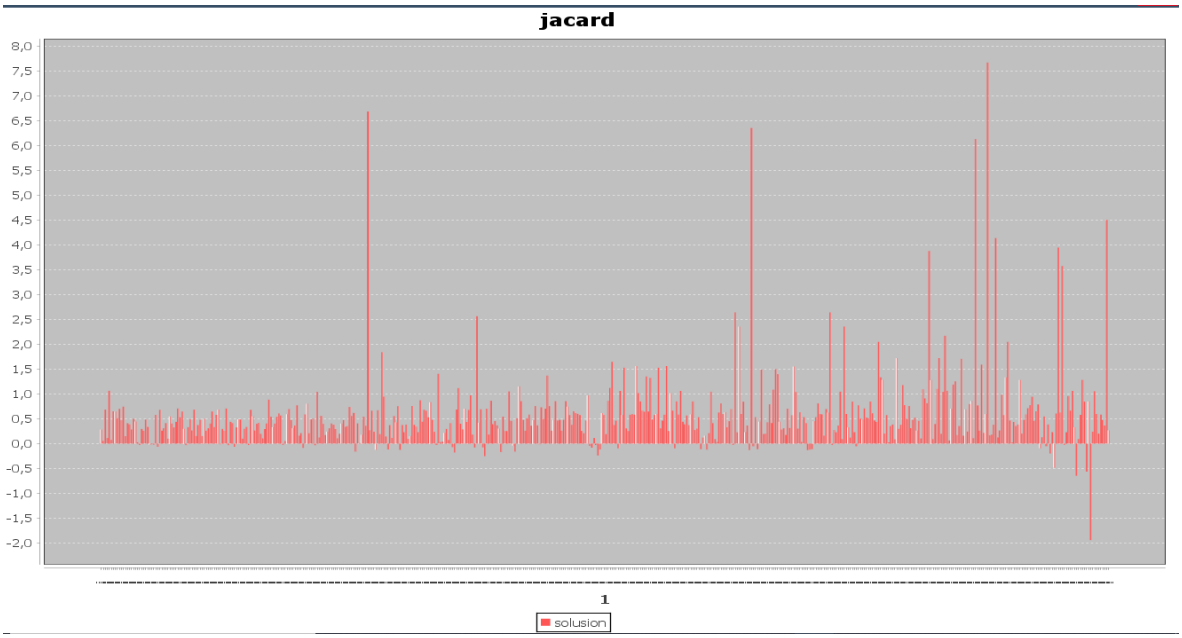

Figure 4.36 : Jaccard index pour toutes les solutions.

La phase 2 : Les sous-ensembles de fonctionnalités qui ont les plus grandes valeurs d'index Jaccard pour NORMAL, DOS, PROBE, U2R et R2L sont sélectionnés comme sous-ensembles de fonctionnalités optimaux.

Les sous-ensembles de caractéristiques S1, S2, S3, S4 et S5 sont déterminés dans le front de Pareto. Les index jaccard moyens de ceux-ci sont représentés sur la figure 4.37 à des fins de comparaison.

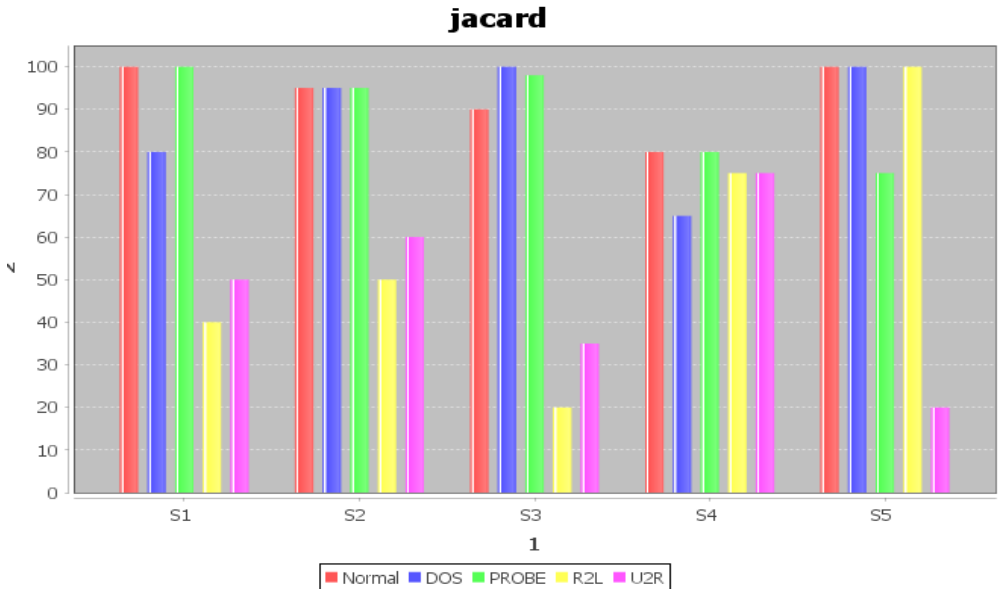

Figure 4.37 : Index Jaccard utilisant des sous-ensembles de fonctionnalités optimaux et toutes les fonctionnalités pour les attaques normales, DOS, PROBE, U2R et R2L.

Résultat 2 :

Les sous-ensembles de fonctionnalités optimaux offrent des performances plus élevées pour les classes ayant moins d'instances. Par exemple, pour U2R et R2L, le sous-ensemble de fonctionnalités optimales S4 fournit des index jaccard plus élevés que les autres sous-ensembles. Ainsi, NSGA-III offre une précision de classification plus élevée pour les classes ayant moins d'instances.

4.6. Conclusion

Dans ce chapitre nous avons exposé les différents tests, ajustements et résultats de la solution proposée pour la résolution du problème de sélection d'attributs en utilisant les meta-heuristiques NSGA-III et justifié l'utilisation du classificateur Naivbayes et la base de données NSL KDD.et nous avons présenté langage de programmation et la bibliothèque utilisée.

CONCLUSION GÉNÉRAL ET PERSPECTIVES

La sélection d'attributs, qui est une discipline du data mining, se propose d'épurer les ensembles de données qui vont permettre de tester la capacité de détection d'un IDS. Cette épuration consiste à éliminer les données redondantes et non pertinentes. Cette tâche n'est pas toujours facile, étant donné l'accroissement rapide du trafic réseau et la diversification des sources et des types de données. Voilà pourquoi nous avons essayé de mettre en place une stratégie de réduction de la dimensionnalité des données, en concevant un algorithme de sélection d'attributs (NSGA-III), un algorithme génétique multi-objectif établi basé sur des mécanismes de tri non dominés.

Par rapport à d'autres algorithmes avec des schémas multi-objectifs, notre système a deux stratégies supplémentaires importantes pour l'évolution de la population, la méthode de domination spéciale et la recherche ciblée multiple prédéfinie, et il peut gérer quatre objectifs ou plus.

Le problème de sélection d'attributs pouvant être formulé sous forme d'un problème d'optimisation combinatoire, cela nous a permis d'intégrer les métaheuristiques au processus de sélection d'attributs pour la recherche de sous-ensemble. Et comme nous l'avons vu, cela a donné des résultats très intéressants et très prometteurs pour les travaux futurs.

Les résultats obtenus montrent que l'utilisation de la métaheuristique NSGA-III pour guider le processus de sélection d'attributs est très intéressante.

Au terme du travail de que nous avons réalisé dans le cadre de ce projet de fin d'étude, nous proposons quelques améliorations pour les travaux futur.

- NSGA-III peut être testé sur d'autres types de problèmes réels contraints et de représentations de données.
- L'application NSGA-III avec plus de 3 valeurs d'IDS.

- NSGA-III peut également être améliorée en notant la distribution des solutions non dominées afin de trouver des techniques plus appropriées pour les problèmes d'optimisation contraints.

ANNEXE

LES RÉSULTATS COMPLETS DES EXPÉRIENCES

Generation	Les solutions optimales		
	FP	**FN**	**TP**
1	-0.02188868909893504	-0.496976793825380	-0.2369133110571766
	-0.3811108226534171	0.12032643256783804	-0.002610712072477095
	-0.4506324581957335	-0.01473246393039795	0.10871126722117797
	-0.15360599084612503	-0.45870637818788484	-0.02723249606597533
	-0.3397839113996133	-0.24449588951678847	-0.3177524849456581
	-0.4898103033716179	0.0468896911006186	0.22773377985072696
	-0.34700530392024775	-0.3509973705328411	-0.15777249332943377
	-0.3927657762144272	0.16739107448233892	-0.13544472088428816
	-0.1796399817236186	-0.170616811325218	-0.3447311563777574
	-0.05156789339843648	-0.00163497747552033	-0.42123511595490104
	-0.4330121468694166	-0.04310360519378942	0.18386222502388283
	-0.2905942164289089	-0.4075952490929188	-0.1454394399001523
	-0.43426619342413686	0.45183877744096057	0.0389749162465709
	0.11898045155224246	0.28991107034893093	-0.44990259517198383
	0.25418110144386075	0.14487720633702816	-0.453428983173777
	-0.29481275850527866	-0.3897853937568839	0.13022738064393402
	0.4178424640117413	-0.39509493277975016	-0.3023506881364013
	-0.16425081918793993	0.10517287585830015	-0.4175099552472361
	-0.24962702058545438	0.39546603879322173	-0.40062592852084367
	-0.2143772891276633	-0.29384512371568006	-0.272557582478581

	-0.2503730113734183	0.28695415923958456	-0.3226638202643627
	-0.2370937683647829	-0.331810805763341	-0.15894517118210438
	0.04567618952502428	-0.3802032057564445	-0.30064600353494886
	-0.40631728481895435	-0.2682425207944764	0.22352715666675027
	-0.22374628473078673	0.37756603226282626	-0.4451522275954706
	0.042240733889403415	-0.36389003815318754	-0.3954586401364083
	0.2595707319665781	0.4618901787522597	-0.4762358129755968
	-0.48122229058214294	0.007550267245043596	0.4820610335452735
10	-0.18011262573387773	-0.3539593822687248	-0.4742150941426571
	-0.44092667235881955	0.09766036828964048	-0.56545418358403
	-0.5852292765620357	-0.514051690543179	-0.08416842288184022
	-0.15774524699889378	-0.4258163246855531	-0.3311442567664576
	-0.18011262573387773	-0.3539593822687248	-0.4742150941426571
	0.030250266618168375	-0.5725115484063967	-0.25979289289417473
	-0.2911970826289348	-0.6499451373324665	0.32722980494973286
	-0.4836265733864036	-0.33094391611388535	-0.1093020127349821
	-0.24092776541169075	-0.6555598039713934	0.26945215786790977
	0.04578658755237336	-0.50373052852476	-0.5066835866278668
	0.2751721534055199	-0.6846774850459201	0.36045935395978523
	-0.6379323725618986	0.20406100376101308	-0.49620121150173346
	-0.40944725522909786	-0.5986109103254104	-0.09339504816891467
	-0.5405648946687266	-0.3083984105710719	-0.19826044646639795
	-0.2362818996832901	-0.19032971356364864	-0.5122247724814226
	-0.614833504936679	-0.2757869115751583	-0.33284356966747997
	-0.2573671154500157	-0.5494831097172014	-0.31687132814797325
	-0.4562889340385027	-0.16387796910690183	-0.4277471649545642
	-0.02070844804209720	-0.6437175128455926	-0.16714585875825375
	-0.5865811842482676	-0.4069429253835164	0.006142587046180176
	-0.6605251789597775	-0.4272894976839009	0.01765164711709874
	0.062352065039846194	-0.1457886722601383	-0.5319812772801559
	-0.15774524699889378	-0.4258163246855531	-0.3311442567664576
	-0.4219178562912955	-0.37165257925369866	-0.17163741057585208
	-0.36293218563990115	-0.3405905010570104	-0.20850168574234357

	-0.3514266228501912	-0.4774360882133112	-0.28325954072677145
	-0.5308610966511893	-0.02778810351835661	-0.4441993511265094
	-0.5942961080381618	-0.350286202167852	-0.04593900956948234
	-0.16510708571777047	-0.42125034803367123	-0.41748053554362347
	-0.15774524699889378	-0.4258163246855531	-0.3311442567664576
	-0.3488209492991189	-6.242400144227389	-0.45870894251629446
	-0.40997668985252533	-0.43853819995336984	-0.10458204206112975
	-0.39599901553693884	-0.40001916723235975	-0.1137844725258993
	-0.42858961055828726	-0.29250269043820476	-0.25976626948669634
	-0.3638925098375182	-0.316846654493033	-0.3701498382181819
100	-0.36141615950515626	-0.7947536870222449	-0.3403268830715038
	0.09907595825351329	-0.821809685684654	-0.6649062482809596
	-0.21847299790939378	-0.8505451066020143	-0.6141293458233783
	0.04886956287997371	-0.893420946530617	-0.4464824270931188
	0.05479889974682234	-0.8987922698542996	-0.3254363894097697
	0.04685462630250306	-0.8842013749180972	-0.44522264060532213
	-0.23975923472097804	-0.7715489676310786	-0.4188575136350998
	-0.3116978927964794	-0.39868395856022554	-0.6785529184069631
	-0.17267863526155847	-0.7364391879170145	-0.6674760366302357
	-0.3589321295581176	-0.708884365759579	-0.5528196710586087
	-0.33773330024193415	-0.5569920857236963	-0.5722221292214147
	-0.24846888072925863	-0.6622330388140136	-0.6196380185103867
	0.04685462630250306	-0.8842013749180972	-0.44522264060532213
	-0.17267863526155847	-0.7364391879170145	-0.6674760366302357
	0.09907595825351329	-0.821809685684654	-0.6649062482809596
	0.05479889974682234	-0.8987922698542996	-0.3254363894097697
	-0.3116978927964794	-0.39868395856022554	-0.6785529184069631
	0.04886956287997371	-0.893420946530617	-0.4464824270931188
	-0.7271750976644492	-0.7820397391496262	0.1616952969086634
	-0.7517869369957586	-0.6863614107592649	0.1596741790693234
	-0.3345091774532235	-0.13933239290842755	-0.5728809109331421
	0.05479889974682234	-0.8987922698542996	-0.3254363894097697
	-0.3116978927964794	-0.39868395856022554	-0.6785529184069631

	-0.6039264253071328	-0.4182628044734319	0.13240227780541935
	-0.5024839647146895	-0.4355305709498083	-0.25219161117308997
	-0.53064554414609	0.11838008900700775	-0.4534463802422981
	-0.6812523930343253	-0.2160904864796384	0.022218606027720716
	-0.6039264253071328	-0.4182628044734319	0.13240227780541935
	-0.44256190956479435	-0.4504326837943713	-0.15875273314880195
	-0.5518696298752259	-0.12815639477097665	-0.4253116009832392
	-0.4778449112546139	-0.27308868130021724	-0.4586629355076585
	-0.5693538606699442	-0.4209300599183412	-0.18823805684533274
	-0.5784677338359725	-0.35497491849065177	-0.41887872386881103
	-0.5902985937860681	0.20228652772042607	-0.4240931778337849
	-0.6039264253071328	-0.4182628044734319	0.13240227780541935
	-0.3116978927964794	-0.39868395856022554	-0.6785529184069631
	-0.43401583143133926	-0.47825385035609425	-0.4939527390668279
	-0.48849169256160074	-0.3153282202947657	-0.45409165971993815
1000	-0.7155882166517769	-0.7123595159949738	-0.3524463503622205
	-0.5878515847104726	-0.8030039629643668	-0.025228710235297103
	-0.42216509545125824	-0.8188067592488367	-0.43151678129483395
	-0.6772334685812308	-0.7433960508010449	-0.380949198221777
	-0.21028413418461955	-0.7986033770640734	-0.4633435127299872
	-0.5906840395746907	-0.8008259506083278	0.1389737048831775
	-0.6204481762756332	-0.8082272786649467	0.1779366126753179
	-0.37450848920649005	-0.6166454224487484	-0.5060190381111599
	-0.7155882166517769	-0.7123595159949738	-0.3524463503622205
	-0.6453647664111074	-0.7828094127168294	-0.10726996903723739
	-0.6868594031215401	-0.20464274012030922	-0.5225673558967339
	-0.593778656281806	-0.17548753109826254	-0.5400138368038274
	-0.7563650245534373	-0.4357065931562264	0.05036896767990659
	-0.6563434665049275	-0.5268926458241076	-0.448411425778363
	-0.7722376382687853	0.0892696214245064	-0.06106372330779525
	-0.5927711229473398	-0.3297408386706827	-0.6324590413718401
	-0.576186285766811	-0.5134547523960139	-0.4689092448032705
	0.1753459825224105	-0.550062751629536	-0.6262749833192847

	-0.13923704903600606	-0.6182607776492837	-0.5814642425445941
	-0.7888863865785891	0.003481715999126861	0.09082268174151813
	0.1953404804423496	-0.604066470255025	-0.6256099182625768
	-0.4369693385001231	-0.3866505921851594	-0.495446986071662
	0.02482488606321201	-0.4777700071300247	-0.5967252378133207
	-0.6563434665049275	-0.5268926458241076	-0.448411425778363
	-0.620448176275633	-0.8082272786649467	0.1779366126753179
	-0.7722376382687853	0.0892696214245064	-0.06106372330779525
	0.07857264972925632	-0.6289161349833042	-0.5682946963334465
	-0.6934835120791827	0.2373938666615075	-0.4136528072996397
	-0.32599779809590096	-0.6125496468128648	-0.5107292202567241
	-0.524023329051363	-0.36444764281398123	-0.49578763983570595
	-0.7722376382687853	-0.7722376382687853	0.0892696214245064
	-0.06106372330779525	-0.3952031023546657	-0.3952031023546657
	-0.4018092748510702	-0.47293173472676736	-0.3952031023546657
	-0.401809274851070	-0.47293173472676736	-0.755496540997084
	-0.6326548899988859	0.1388457076775192	-0.5659349235544222
	-0.36276797170210573	-0.5945424774156784	-0.5523069880159049
	-0.7799962631167691	-0.5479807675666486	-0.16675039338906844
	-0.4687839278821968	-0.8332026483760422	-0.2114451377212862
	-0.8120403290255063	-0.5625582740447602	-0.05984090282666671
	-0.6919941569376351	-0.6594541627308873	-0.138068075047375
	-0.5801465129132432	-0.8712394562948335	0.12922846857550863
	-0.3892787848845948	-0.7611318710430414	-0.43665520417693016
	-0.6763019889046317	-0.6984200009383467	-0.3174275845212243
	-0.4268528524130264	-0.3772835824213594	-0.6703168167621111
	-0.7533871728100204	-0.7068247573674049	0.031718473664800545
	-0.7314453662287744	0.0621864732275656	-0.49745365836956995
	-0.7592109437325346	-0.18942055470288063	-0.3119171093360576
	-0.6860690129954792	-0.4006998483651363	-0.40100655535921886
	0.10851407360701198	-0.41230779141136586	-0.5849985962714924
	-0.025431915396584154	-0.7901613366842095	-0.35879578021353414
	-0.4073285651342581	-0.4728891480906558	-0.3916184676150805

	-0.2405919917866561	-0.699609942724289	-0.5346003933921876
	-0.0717588647662473	-0.6439701889381446	-0.5370931667501008
	-0.5825995192999228	-0.406294564279425	-0.5052087427166484
	-0.40410557002408753	-0.678695232742227	-0.43167332569191197
	-0.6733173278460343	-0.4341758902758799	-0.3676710758603301
	-0.5106876538038644	-0.5997644760686703	-0.3535555017956381
	-0.5187090181331018	-0.5422737734747992	-0.34510457061891703
	-0.5579339028193954	-0.49543414012726994	-0.529937015296895
	-0.7235402496332803	-0.4549848627739257	-0.6386881762862261
	-0.7937301985698161	-0.23621209026615042	0.0023403109322995413
	-0.4025507723132178	-0.8200874213704535	-0.027087623628692714
	-0.027087623628692714	-0.37764238478978174	-0.7215692922548136
	-0.4485033672421736	-0.7390967983209326	-0.13269495217733315
	-0.4578219459597362	-0.764054307192169	-0.7127427315146593
	-0.3343556409870429	-0.20319695697787318	-0.35447975981129365
	-0.7770667085844498	-0.4978053539840085	0.3379636001418374
	-0.4758913887690023	-0.6370644478365406	-0.4714314938730591
	0.028287337528086792	-0.765763231340078	-0.5788237103319386
	-0.5309911540510092	-0.694564402116153	-0.3878625995757768
	-0.5570659649630668	-0.7589717775850889	-0.05335653621243218
	-0.35726339501316307	-0.7933291393475641	-0.11091909018417544
	-0.23121508266310625	-0.6337617572986526	-0.5665858884311127
	-0.6749798863565983	-0.5019875362120808	-0.37744441565245646
	-0.6934787008728169	-0.629745794291251	-0.3481767282847318
	-0.2631966253995708	-0.549498960999338	-0.4719847433553717

BIBLIOGRAPHIE

[1] Philippe Atelin, José Dordoigne , Réseaux informatiques : notions fondamentales : normes, architecture, modèle OSI, TCP/IP, Ethernet, Wi-Fi, Editions ENI, 2006 - 452 pages.

[2] Philippe Biondi, Architecture expérimentale pour la détection d'intrusions dans un système informatique, Article de recherche, Avril-Sptembre 2001.

[3] Eric Maiwald, *"Sécurité des réseaux"*, Publié par Campus Press, ISBN : 2-7440-1240-8, paris 2001.

[4] A. Bouhoula, Z. Trabelsi, E. Barka, and M.-A. Benelbahri, "Firewall filtering rules analysis for anomalies detection," Int. J. Secur. Netw., vol. 3, no. 3, pp. 161–172, 2008.

[5] Shon Harris, *"CISSP Certification ALL-in-one Exam Guide, 6th Edition"*, Publisher: Tata Graw-Hill, ISBN: 978-0-07-178173-2, 2013.

[6] M.Tran Van Tay, le système de détection des intrusions et le système d'empêchement des intrusions (ZERO DAY), Rapport de stage de fin d'étude, institut de la francophonie pour l'informatique, université de Québec à Montréal, Féverier 2005.

[7] Farhaoui, Yousef. (2015). Evaluation des systèmes de détections et de prévention des intrusions et conception d'une nouvelle architecture des IDS.

[8] Briffaut, Jeremy. (2007). Formalization and guaranty of system security properties : application to the detection of intrusions.

[9] Karim Tabia,Modèles graphiques et approches comportementales pour la détection d'intrusions. Thèse de doctorat de l'Université d'Artois, 218 pages, Novembre 2008.

[10] Pietro R. Di, Mancini L. V., A. Mei, A. Panconesi, and J. Radhakrishnan, "Redoubtable Sensor Networks," ACM Transactions on Information and System Security, vol. 11, no. 3, pp. 1–22, Mar. 2008.

[11] Boukhlouf Djemaa , "Cours Sécurité des systèmes d'Information et Web ",cour Université Mohamed KheiderBiskra, 25/11/2018.

[12] G. Giacinto, F. Roli, and L. Didaci, “Fusion of multiple classifiers for intrusiondetection in computer networks,” *Pattern recognition letters*, vol. 24, no. 12,pp. 1795–1803, 2003.

[13] Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods.

[14] Antonia Nisioti, Alexios Mylonas , Paul D.Yoo, Vasilios Katos, << From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods>> ,IEEE Communications Surveys & Tutorials (Volume: 20 , Issue: 4 , Fourth quarter 2018).

[15] V. R. Balasaraswathi, M. Sugumaran, Y. Hamid. Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms [J]. Journal of communications and information networks, 2017.

[16] P. Langley, “Selection of relevant features in machine learning,” *Proceedings ofthe Aaai Fall Symposium on Relevance*, pp. 140–144, 1994.

[17] W. Siedlecki and J. Sklansky, “On automatic feature selection,” *International Journal of Pattern Recognition & Artificial Intelligence*, vol. 2, no. 2, 2011.

[18] D. Tian, J. Keane, and X. J. Zeng, “Evaluating the effect of rough set feature selection on the performance of decision trees,” in *Granular Computing, 2006 IEEE International Conference on*, 2006, pp. 57–62.

[19] H. Liu and L. Yu, “Toward integrating feature selection algorithms for classification and clustering,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 4, pp. 491–502, 2005.

[20] S.W. Lin, K. C. Ying, S. C. Chen, and Z. J. Lee, “Particle swarm optimization for parameter determination and feature selection of support vector machines.”*Expert Systems with Applications*, vol. 35, no. 4, pp. 1817–1824, 2008.

[21] Y. L. Wu, C. Y. Tang, M. K. Hor, and P. F. Wu, “Feature selection using genetic algorithm and cluster validation,” *Expert Systems with Applications*, vol. 38, no. 3, pp. 2727–2732, 2011.

[22] V. R. Balasaraswathi, M. Sugumaran, Y. Hamid. Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms [J]. Journal of communications and information networks, 2017, 2(4): 107-119.

[23] S. Binitha, S. S. Sathya. A survey of bio inspired optimization algorithms [J],International journal of soft computing and engineering, 2012, 2(2): 137-151.

[24] M. C. Lee, “Using support vector machine with a hybrid feature selection method to

the stock trend prediction," *Expert Systems with Applications*, vol. 36, no. 8, pp. 10 896–10 904, 2009.

[25] Y. L. Murphey and H. Guo, "Automatic feature selection-a hybrid statistical approach," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 2. IEEE, 2000, pp. 382–385.

[26] H.-H. Hsu, C.-W. Hsieh, and M.-D. Lu, "Hybrid feature selection by combining filters and wrappers," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8144–8150, 2011.

[27] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper–filter feature selection algorithm using a memetic framework," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 1, pp. 70–76, 2007.

[28] Zhu, Yingying & Liang, Junwei & Chen, Jianyong & Ming, Zhong. (2016). An improved NSGA-III algorithm for feature selection used in intrusion detection. Knowledge-Based Systems. 116. 10.1016/j.knosys.2016.10.030.

[29] Eesa, Adel & Orman, Zeynep & Mohsin Abdulazeez, Adnan. (2015). A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems.

[30] H. Gharaee and H. Hosseinvand, "A new feature selection IDS based on genetic algorithm and SVM," *2016 8th International Symposium on Telecommunications (IST)*, Tehran, 2016, pp. 139-144. doi: 10.1109/ISTEL.2016.7881798

[31] Othman, S.M., Ba-Alwi, F.M., Alsohybe, N.T. *et al.* Intrusion detection model using machine learning algorithm on Big Data environment. *J Big Data* **5,** 34 (2018). https://doi.org/10.1186/s40537-018-0145-4

[32] Ghanem, Waheed & Jantan, Aman. (2016). Novel multi-objective artificial bee colony optimization for wrapper based feature selection in intruction detectoin. 8. 70-81.

[33] Tamimi, Ali et al. "An Intrusion Detection System Based on NSGA-II Algorithm." 2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec) (2015): 58-61.

[34] HAO J. - K. , GALINIER P., HABIB M. « Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes ». Revue d'Intelligence Artificielle. 1999. Vol. 13 n°2 p. 283-324.

[35] G. LAPORTE, I.H. OSMAN, Metaheuristics in combinatorial optimization, Annals of Operations Research 63, J.C. Baltzer Science Publishers, Basel, Switzerland, 1996.

[36] J. Dréo, Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical. Thèse de doctorat en sciences, Soutenue le 13 décembre 2004. Université Paris 12. Val de Marne. UFR de Sciences.

[37] P. Siarry, J. Dréo, A. Pétrowski, E. Taillard. (2006). Métaheuristiques pour l'optimisation difficile, 2003, Eyrolles. ISBN : 2-212-11368-4.

[38] Zitzler, E., Deb, K. & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation, 8(2), 173-195.

[39] C. Dhaenens. Optimisation Combinatoire Multi-Objectif : Apport des méthodes coopératives et contribution à l.extraction de connaissances. PhD thesis, Université de Lille, 2005.

[40] Y. Dufresne. Pji - algorithmes de recherche locale pour l'optimisation combinatoire Multi-objectif Technical report, 2012.

[41] RABIA Mohamed Akli , HAROUNI Youcef ,Approche de résolution d'un problème de sac a dos bi-objectif en variables binaires , Mémoire de magister, Université M'hamed Bougara Boumerdes,2017.

[42] John H. Holland. Adaptation and artificial systems. University of Michigan Press, 1975.

[43] David. E. Goldberg. Genetic algorithms in search, optimization and machine learning. AddisonWesley, 1989.

[44] Agoston E. Eiben and Jim E. Smith. Introduction to Evolutionary Computing. Springer, 2003.

[45] Dihya AISSAOUI ,Souad BENHAMA, Modélisation mathématique et techniques de décision, Mémoire de Magister , Université Abderrahmane Mira, Béjaia.

[46] BENHAMA Abdelouhab, BENKHELOUF Yacine Etude comparative des algorithmes génétiques multi-objectifs, Mémoire de magister , Université Abderrahmane MIRA de Bejaia ,2015.

[47] Hafid Zidani. Représentation de solution en optimisation continue, multi-objectif et applications ,Thèse de Doctorat , Université Mohammed V-Agdal (Rabat, Maroc), 2013.

[48] Schaffer, J. D. (1985, January). Multiple objective optimization with vector evaluated genetic algorithms. In Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985 (pp. 93-100).

[49] Srivinas N. and Deb K., "Multiobjective Optimization using Nondominated Sorting in Genetic Algorithms", Technical Report, Department of Mechanical Engineering, Institue of Technology, India, 1993.

[50] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I : solving problems with box constraints. Evolutionary Computation, IEEE Transactions on, 18 (4) :577 _ 601, 2014.

[51] Yuan, Yuan & Xu, Hua & Wang, Bo. (2014). An improved NSGA-III procedure for evolutionary many-objective optimization. GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference. 10.1145/2576768.2598342.

[52] Blank J., Deb K., Roy P.C. (2019) Investigating the Normalization Procedure of NSGA-III. In: Deb K. et al. (eds) Evolutionary Multi-Criterion Optimization. EMO 2019. Lecture Notes in Computer Science, vol 11411. Springer, Cham. https://doi.org/10.1007/978-3-030-12598-1_19.

[53] Kudo, M. et Sklansky, J. (2000). Comparison of algorithms that select features for pattern classiers. Pattern Recognition, 33(1):25 - 41.

[54] java, oracle: site officiel de la technologie java : https://www.java.com.

[55] Eclipse : site officiel de l'environnement eclipse : https://www.eclipse.org.

[56] Allam Karima, Amel Hamidi Zoulikha,Meta-heuristique pour la détection d'intrusion : Application de la PSO, Mémoire de magister, Université Dr Tahar Moulay de Saida.

[57] Berlin Hervé Djionang Lekagning, Gilbert Tindo. VERS UNE NOUVELLE ARCHITECTURE DE DETECTION D'INTRUSION RESEAUX A BASE DE RESEAUX NEURONAUX. 2016. ffhal01304514f.

[58] Aslihan Ozkaya && Bekir Karlik "Protocole Type Based Intrusion Detection Using RBF Neural Network" International Journal of Artificial Intelligence and Expert Systems (IJAE), Volume(3): Issue(4):2012.

[59] Ahmed Chaouki LOKBANI, Le problème de sécurité par le DataMining ; Thèse de doctorat, Université Djillali Liabes - Sidi Bel Abbes,2017.