République Algérienne Démocratique et populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université M'hamed BOUGARA de Boumerdes

Faculté des sciences

Département d'informatique

MEMOIRE DE MAGISTERE

Spécialité : informatique

Option : Spécification de logiciel et traitement de l'information

Ecole doctorale

Présenté par :

Gunadiz Safia

Thème :

Algorithmes d'intelligence artificielle pour la classification d'attaques réseaux à partir de données TCP

Devant le jury de soutenance composé de :

Mr : MEZGHICHE Mohamed	Professeur	UMBB	Président
Mr : DJOUADI Yassine	Maître de conférence	UMMTO	Examinateur
Mr : BABA-ALI Ahmed Riadh	Maître de conférence	USTHB	Examinateur
Mr : TAMINE Karim	HDR	Université de limoges	Rapporteur

Année universitaire : 2010/2011

REMERCIMENT

En premier lieu, je remercie DIEU de m'avoir aidé et donner la force et la volonté pour achever ce modeste travail.

Par la suite ce travail a été réalisé sous la direction du monsieur Karim TAMINE qu'il trouve ici ma profonde reconnaissance et mes sincères remerciements, pour ses encouragements, son aide ses conseils précieux et aussi ses idées pour la réalisation de ce mémoire.

J'adresse mes vifs remerciements à Monsieur MOHAMED MEZGHICHE, professeur à l'université de BOUMERDES (U.M.B.B) pour l'honneur qu'il me fait de présider le jury.

Je tiens aussi à remercier, les membres du jury qui ont accepté de juger ce travail.

Je remercie aussi l'ensemble des enseignants ayant intervenu aux cours de notre première année de post-graduation, trouvent ici l'expression de ma gratitude.

Également, je dois à mes camarades en PG et à l'ensemble de mes collègues surtout Basma une reconnaissance que je veux à la hauteur du soutien et encouragement indéfectibles qu'ils n'ont jamais cessés de témoigner à mon égard.

Et enfin, je tien à remercier mes chères parents, mon frère Samir et mon mari pour leurs encouragements, leurs aides et leurs grande patience avec moi.

Dédicace

A mes très, très, très chères parents.

Ames frères Dahmen et Samir.

A mes soeuurs Naima et Sadjia.

A ma petite famille, mon mari Brahim et ma chère fille Rahma.

A mes chères amies Amina,Amel,Souhila,Zineb et Basma.

Safia

Résumé

Les réseaux et systèmes informatiques sont devenus aujourd'hui des outils indispensables pour le bon fonctionnement et l'évolution de la plupart des entreprises. Ainsi, les systèmes et réseaux informatiques sont déployés dans différents domaines comme la banque, la médecine ou encore le domaine militaire. L'accroissement de l'interconnexion de ces divers systèmes et réseaux les a rendus accessibles par une population diversifiée d'utilisateurs qui ne cesse d'augmenter. Ces utilisateurs, connus ou non, ne sont pas forcément pleins de bonnes intentions vis-à-vis de ces réseaux. En effet, ils peuvent essayer d'accéder à des informations sensibles pour les lire, les modifier ou les détruire ou encore tout simplement pour porter atteinte au bon fonctionnement du système. Dès lors que ces réseaux sont apparus comme des cibles d'attaques potentielles, les sécuriser est devenu un enjeu incontournable.

La détection d'intrusions consiste à découvrir ou identifier l'utilisation d'un système informatique à d'autres fins que celles prévues. De nombreux mécanismes ont été développés pour assurer la sécurité des systèmes informatiques et tout particulièrement pour prévenir les intrusions, malheureusement, ces mécanismes ont des limitations. En effet, les systèmes informatiques présentent des failles qui permettent à des attaquants de contourner les mécanismes de prévention. Pour cela une seconde ligne de défense est nécessaire, la détection d'intrusions.

L'intelligence artificielle a vu naître de nombreuses méthodes telles que les algorithmes génétiques, les réseaux de neurones, les algorithmes de fourmis virtuelles, etc.… qui peuvent être implémentées et utilisées pour la détection d'intrusions.

Dans ce mémoire on implémente deux systèmes de détection d'intrusions, l'un est basé sur les réseaux de neurones et l'autre sur l'algorithme ANTClass. Les systèmes sont implémenté avec JBuilder X, et testé sur les données de la base de données KDD'99.

Mots clés : classification, système de détection d'intrusions, réseaux de neurones, algorithme de fourmis, algorithme génétique.

Summary

Networks and information technology systems have become nowadays essential tools for good functioning and evolution of most enterprises. Thus, they are deployed in different fields such as Banks, medical field, or military. The growth of these networks and systems made them accessible by a very wide number of users, which is still growing. These users, whether they are known or not, do not necessarily use these systems in a friendly manner. In fact, they can try to access vital information to read, modify, or even destroy the information or the systems. Since these networks and systems have become targeted, securing them has become very essential.

Intrusion detection is basically the process of detecting and identifying attempts to use the systems for unwanted purposes. A number of mechanisms have been developed to protect information technology systems and specially to prevent intrusions. Unfortunately, these mechanisms have limitations. In fact, Information systems present some weak points that permit to the attackers to go around the prevention mechanisms. For this, a second line of defense is necessary, Intrusion detection.

Artificial intelligence has seen many methods developed such as genetic algorithms, neural networks, Virtual Ant algorithms … which can be implemented and used for intrusion detection.

In this project, We implement two intrusion detection systems, one is based on Neural Networks, and the other on ANTClass algorithm.The systems are implemented under JBuilder X, and tested on the data from the database KDD'99.

Keywords: clustering, Intrusion detection systems, neural networks, Virtual Ant algorithms, genetic algorithms.

ملخص

أصبحت اليوم أنظمة وشبكات الأعلام الآلي أجهزة لا يمكن الاستغناء عنها لتسيير و تطوير معظم المؤسسات, كما أنها تستغل في مختلف المجالات, في البنوك, الطب و حتى في المجال العسكري. إن تزايد التوصيل بين هذه الأنظمة و الشبكات جعلها في متناول فئات مختلفة من المستعملين لها والتي تتزايد باستمرار. هؤلاء المستعملين, سواء كانوا معروفين أم لا, لا يستغلون بالضرورة هذه الشبكات لأهداف حسنة. و بالتالي يستطيعون محاولة الوصول إلى معلومات حساسة لقراءتها, لتغييرها أو لإتلافها أو بكل بساطة لتعطيل النظام. منذ أن أصبحت هذه الشبكات أهداف للتدخلات الغير سليمة و الهجمات,أصبح من الضروري تأمينها منها.

الكشف على التدخلات الغير سليمة يقتضي إيجاد أو التعرف على كل استعمال لجهاز الإعلام الآلي لأهداف غير التي وضع لأجلها. العديد من التقنيات طورت لضمان أمان أنظمة الإعلام الآلي و خاصة لتوقع الهجمات قبل حدوثها. لكن هذه التقنيات تبقى محدودة لأن أنظمة الإعلام الآلي تعرف نقاط ضعف و التي تسمح للمهاجمين بتفادي تقنيات التوقع. لهذا فانه من الضروري إضافة خط ثاني للدفاع و هو الكشف عن التدخلات.

عرف الذكاء الصناعي ميلاد العديد من النماذج كالخوارزميات الجينية, الشبكات العصبية, الخوارزميات التي تعتمد علي مبدأ التصنيف لدى النمل...الخ.و التي بإمكانها أن تطور و تستعمل للكشف عن التعديات.

في هذه المذكرة ننجز نظامين للكشف عن التدخلات, الأول يتركز على الشبكات العصبية والثاني على نظام التصنيف المتبع لدى النمل.النظامين تم إعدادهما بJBUI L DER X , و تم اختبارهما بمعلومات قاعدة المعلومات KDD'99 .

الكلمات المفاتيح: التصنيف, أنظمة الكشف عن التعديات, الشبكات العصبية, خوارزمية النمل, خوارزمية الجينات.

Table des matières

Chapitre III

La base de données KDD'99

Chapitre IV

Application de ANTclass sur la base KDD'99.

Chapitre V

Application des réseaux de neurones sur la base KDD'99

LISTES DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES.

Introduction générale

La sécurité des systèmes informatiques est un problème sensible et préoccupant. Le progrès spectaculaire des technologies de l'information et de communication offre actuellement des facilités incontournables en matière de transfert de fichiers, messageries et bien d'autres formes d'échange d'informations. Le développement de l'informatisation des échanges s'est accompagné malheureusement du développement d'activités malveillantes dont les motivations sont aussi nombreuses que dangereuses et évoluent dans le temps.

Profitant de la connectivité croissante des systèmes d'informations, à l'Internet notamment, les possibilités d'attaques à distance sont plus grandes et plus menaçantes. La connectivité croissante des systèmes d'informations, les vulnérabilités et failles continuellement découvertes dans les systèmes informatiques (systèmes d'exploitation, applications, protocoles de communication, etc.), augmentent les risques d'attaques à distance. Actuellement, les pare-feux permettent de réduire partiellement ce risque, cependant un réseau protégé par un pare-feu demeure tout de même pénétrable. En plus, un attaquant interne peut abuser de ses privilèges et attaquer des systèmes au sein du même réseau local. La détection d'intrusions réseaux trouve alors tous ses justificatifs en ce sens qu'à défaut de se protéger de façon sûre contre les attaques.

Afin de détecter toute tentative de violation des mécanismes de la sécurité, une surveillance permanente ou régulière des systèmes peut être mise en place : ce sont les Systèmes de Détection d'Intrusions (IDS).

La détection d'intrusions consiste à scruter le trafic réseau, collecter tous les événements, les analyser et générer des alarmes en cas d'identification de tentatives malveillantes.

Ces système sont devenus très largement déployés dans les systèmes informatiques et ils ont gagné une place importante dans la conception de la stratégie de sécurité. Ils sont généralement utilisés pour surveiller l'accès et le flux d'information, dans le but de déterminer tout comportement malicieux, que se soit de l'intérieur ou de l'extérieur de système d'informations, et rendre cette information disponible aux administrateurs de la sécurité. En option, les systèmes de détection d'intrusions peuvent réagir contre ces comportements malicieux et prendre des contre-mesures.

Il existe actuellement deux principales approches mises en oeuvre pour élaborer des système de détection d'intrusions : l'approche par scénario et l'approche comportementale. La première se pose sur des signatures d'attaques déjà connues et qui peuvent être fournies par

l’expert du domaine ou extraites à l’aide de techniques inductives à partir de données intrusives. L’inconvénient majeur de cette approche est son incapacité à détecter les nouvelles attaques (dont les signatures sont encore inconnues).l’approche comportementale suppose que l’activité normale est différente de l’activité intrusive. Il suffit alors d’élaborer un profil pour l’activité normale et un mécanisme permettant de comparer l’activité courante au profil établi pour détecter des écarts sensibles qui seront considérés des éventuelles intrusions. Un tel profil peut être obtenu en observant, pour une durée suffisante, l’activité normale au sein du système d’informations ou selon les consignes de la politique de sécurité mise en place sur ce système. C’est en ce sens que l’approche comportementale peut détecter les nouvelles attaques.

Cette approche sera ,dans ce qui suit, l’approche sur laquelle on se base pour proposer des modèles des systèmes de détection d’intrusions comportementaux utilisant des méthodes d’intelligence artificielle pour l’élaboration du profil normal.

Dans ce mémoire, nous proposons d’implémenter quelques méthodes de classification non supervisée pour élaborer le profil normal des utilisateurs, qui sera ensuite utilisé pour distingué entre les déviations et les activités normales.

Les systèmes proposés seront élaborés et testés à l’aide de la base de données KDD’99, qui est une base de connexions TCP/IP, et qui comporte deux types de données, les données d’apprentissage qui sont utilisées pour l’élaboration des modèles de détection d’intrusions, et les données de test qui sert à évaluer les performances des systèmes élaborés.

Notre première contribution est de proposer et implémenter deux systèmes de détection d’intrusions, le premier se base sur l’algorithme ANTClass et le deuxième sur la carte de Kohonen, puis nous donnons des résultats expérimentaux des deux systèmes. La comparaison des deux modèles sera faite à la fin de ce mémoire.

Motivation :

L'agence américaine DARPA a initié un programme d'une série d'évaluations des systèmes de détection d'intrusions annuelles dont la finalité est l'élaboration d'une méthodologie d'évaluation standard.DARPA'98, eut lieu en 1998 et permit de construire un premier corpus d'évaluation. Avec des objectifs limités et un nombre modeste de participants, cette campagne a suscité un grand intérêt de la part des différentes communautés de détection d'intrusions.

C'est dans cette optique que s'inscrit notre travail qui porte sur l'application des méthodes d'intelligence artificielle sur la base KDD'99 dans le but de développer des systèmes de détection d'intrusions que nous désirons d'être efficaces et capables de détecter le maximum d'attaques en générant le minimum de fausses alertes.

L'algorithme ANTClass proposé par Nicolas Monmarché dans [001] est l'un des algorithmes rarement appliqué sur la base KDD'99, pour cela nous le choisissons, en premier lieu, pour concevoir et développer un système de détection d'intrusion qui sera expérimenté sur les données de cette base puis les résultats fournis seront comparés aux résultats donnés par l'application d'un autre système, utilisant les réseaux de neurones artificiels, sur les mêmes données.

Notre travail consistera principalement à :

1. Proposer une architecture appliquant l'algorithme ANTClass sur la base KDD'99 et une autre appliquant les réseaux de neurones sur la même base.
2. Développer deux systèmes à base de ces deux architectures.
3. Expérimenter ces deux systèmes sur la base de données KDD'99.
4. Interpréter, analyser et expliquer les résultats.
5. Comparer les résultats fournis par les deux systèmes.

Introduction

Dans un contexte de connectivité croissante des réseaux informatiques, la sécurisation des systèmes d'informations est devenue un enjeu majeur. Ainsi, les systèmes et réseaux informatiques sont déployés dans différents domaines comme la banque, les assurances, la médecine ou encore le domaine militaire. L'accroissement de l'interconnexion de ces divers systèmes et réseaux, les a rendus accessibles par une population diversifiée d'utilisateurs qui ne cesse d'augmenter. Ces utilisateurs, connus ou non, ne sont pas forcément pleins de bonnes intentions vis-à-vis de ces réseaux. En effet, ils peuvent essayer d'accéder à des informations sensibles pour les lire, les modifier ou les détruire ou encore tout simplement pour porter atteinte au bon fonctionnement du système. Dès lors que ces réseaux sont apparus comme des cibles d'attaques potentielles, les sécuriser est devenu un enjeu incontournable.

De nombreux mécanismes ont été développés pour assurer la sécurité des systèmes d'informations et tout particulièrement pour prévenir les intrusions.

Malheureusement, ces mécanismes ont des limitations. En effet, les systèmes informatiques présentent des failles de conception, d'implémentation et de configuration qui permettent à des attaquants de contourner les mécanismes de prévention. Pour cela une seconde ligne de défense est nécessaire, la détection d'intrusions.

Ce chapitre est subdivisé en deux partie, la première représente des définitions et des stratégies de la sécurité, dans la deuxième sera posé le problème de la détection d'intrusions, les systèmes de détection d'intrusions seront ensuite abordés en présentant les approches de détections, la classification des IDS selon différents critères, les limites des IDS et quelques propriétés requises des IDS.

I Généralités sur la sécurité informatique :

I.1 La sécurité informatique :

La sécurité informatique est l'ensemble des moyens matériels et logiciels mis en œuvre pour minimiser la vulnérabilité d'un système contre des menaces accidentelles ou intentionnelles, permettant ainsi aux systèmes informatiques de fonctionner normalement.

Elle consiste, aussi, à s'assurer que celui qui modifie ou consulte les données du système en a l'autorisation et qu'il peut le faire car le service est disponible. [003]

Sécuriser les données, c'est garantir : [003]

- **L'authentification :** Le but de l'authentification est de garantir l'origine:
  - D'une information : Prouver qu'une information provient de la source annoncée (auteur, émetteur).
  - D'une personne (ou machine, groupe ou organisation): Prouver que l'identité est bien celle annoncée.
- **L'intégrité :** L'intégrité est d'assurer que les données n'ont pas été modifiées et empêcher toute modification (intentionnelle ou accidentelle) non explicitement requise par une entité habilitée. Cela permet, par exemple, au récepteur d'un message d'être raisonnablement assuré que le message reçu est le même que le message envoyé. Donc, l'intégrité des données est la propriété qui assure qu'une information n'est modifiée que dans des conditions prédéfinies (selon des contraintes précises).
- **La confidentialité :** La confidentialité est de garder secret le contenu de l'information et empêcher (ou prévenir) sa divulgation à des entités (sites, organisation, personnes, etc.) non habilitées à le connaître. Seuls les destinataires prédéterminés doivent être capables de lire le contenu du message.
- **La non répudiation** pour éviter la contestation par l'émetteur de l'envoi de données, la non répudiation est une propriété qui assure que l'auteur d'un acte ne peut ensuite nier l'avoir effectué (signature de l'acte) et que le récepteur ne peut ultérieurement dénier avoir reçu un message (exemple exécution d'un ordre boursier, d'une commande…).

Définir une politique de sécurité pour un système d'information revient à élaborer l'organisation et les mécanismes à mettre en place et l'ensemble des mesures à prendre en vue de :

- réduire dans la mesure du possible l'utilisation frauduleuse, l'altération et l'accès non autorisé au système et ses ressources (machines, réseau, etc.).

- Détecter aussi rapidement que possible toute activité, malveillante ou accidentelle, pouvant porter atteinte à la confidentialité, intégrité et disponibilités des ressources et services.
- Prendre aussi efficacement que possible des contre-mesures afin de limiter les conséquences et poursuivre éventuellement l'auteur du forfait.

I.2 Protection des systèmes d'informations:[003]

Pour atteindre les objectifs de la sécurité, une stratégie de sécurité doit minutieusement évaluer ce qui est à protéger dans le système d'informations, les risques potentiels, les vulnérabilités du système à sécuriser, pour dégager de cette analyse les mécanismes et moyens nécessaires, tenant compte bien entendu d'autres considérations telles que le coût de déploiement et la complexité de cette stratégie. Entre autres mécanismes utilisés pour la sécurité des systèmes d'informations, ci-après les plus répandus :

- **Authentification et contrôles d'accès aux ressources** : il s'agit en premier lieu de la sécurité physique des équipements et installations. L'authentification est un mécanisme permettant de prouver l'identité d'un utilisateur (mots de passe, cartes à puce, méthodes biométriques, etc.) et de lui accorder uniquement les privilèges nécessaires pour l'accomplissement de ses tâches.
- **Scanners de vulnérabilités** : Les scanners de vulnérabilités automatisent la découverte des failles de sécurité. Ils sont utilisés par les attaquants pour localiser les faiblesses du réseau cible. De plus les administrateurs peuvent en tirer profit pour corriger les vulnérabilités de leurs systèmes informatique. Cependant, malgré le grand nombre de vulnérabilités détectées, les scanners d'aujourd'hui sont inaptes à déterminer toutes les faiblesses possibles. De plus, la mise à jour de ces produits ne suit pas le rythme de la découverte des nouvelles vulnérabilités.
- **La cryptographie** : les informations sensibles et confidentielles sont souvent cryptées pour empêcher leur lecture même si elles étaient dérobées ou accédées frauduleusement. La cryptographie garantit la confidentialité, l'intégrité, la non répudiation et l'authenticité des données mais elle ne constitue pas une solution unique et suffisante de sécurité. Effectivement, diverses implémentations des protocoles de sécurité se sont révélées vulnérables. De plus la sécurité peut être rompue via plusieurs types d'attaques (par exemple :l'homme du milieu (MITM),les courts et les simples mots de passes utilisés sont facilement cassables, le risque de vol des clés privées).

- **Les pare-feu [Firewall]** : un pare-feu (ou coupe-feu) est un dispositif matériel et logiciel d'interfaçage entre le réseau à sécuriser et un autre réseau externe, jugé moins sûr, et potentiellement source d'attaques. Un pare-feu assure d'abord une fonction de filtrage des flux entrants et sortants puisqu'il est un passage obligé pour tout échange. Le risque d'attaques distantes est alors réduit puisque le pare-feu ne laisse passer que le trafic d'une certaine plage d'adresses IP et relatif à certains ports et services. Malgré leurs grands intérêts, les pare feux présentent quelques lacunes. En effet, un attaquant peut exploiter les ports laissés ouverts pour pénétrer au réseau local. Les scripts constituent aussi des sources d'intrusion que les pare feux échouent à détecter. Ainsi l'opération supplémentaire d'encapsulation/décapsulation des données permet à l'attaquant de contourner le pare feu.
- **Protection anti-virus** : les logiciels anti-virus sont largement utilisés pour les stations de travail et ordinateurs personnels. Ils détectent et protègent contre les virus pouvant se propager à travers des fichiers, courrier électronique, etc.
- **Audit de sécurité et détection d'intrusions** : la plupart des systèmes d'exploitations et applications sont dotés de mécanismes d'audit permettant d'enregistrer tout ou une partie des événements (exécutions de commandes, utilisation de ressources, etc.) ayant lieu sur le système. Ces données peuvent faire l'objet d'une analyse en temps réel ou en différé pour détecter des activités malveillantes, suspectes ou toute activité pouvant violer la politique de sécurité. Cependant, ces données contiennent beaucoup d'informations normales et anormales. La taille énorme des fichiers contenant ces données pose souvent des problèmes de stockage et d'exploration du contenu. Les administrateurs fournissent aussi l'effort pour localiser les activités anormales, comprendre les objectifs des attaquants et déterminer les vulnérabilités exploitées du système.

I.3 Nécessité de la détection d'intrusions :

L'importance particulière qu'il convient selon [007] d'apporter à la détection d'intrusions tient à trois raisons :

- De nombreux systèmes existants sont vulnérables aux intrusions externes (attaques), et internes (abus de la part d'utilisateurs habilités à se servir du système).
- Les systèmes existants ne sont pas toujours remplaçables par des systèmes plus sécurisés, soit pour des raisons de coût (développer un système hautement sécurisé est une tache difficile), soit car l'apport de sécurité se fait au détriment du confort d'utilisation du système (voire de sa philosophie, c'est notamment le cas du système UNIX).

- Même les systèmes les plus sécurisés sont vulnérables aux abus de la part des utilisateurs habilités et aux canaux cachés. Il est important de noter que ces deux risques, ne violant aucune des règles de la politique de sécurité, ne sont détectables que par audit.

II Introduction aux systèmes de détection d'intrusions :

Les attaquants suivent une stratégie d'attaque pour réussir leurs exploits. Ils disposent de plusieurs sources d'information et de divers outils pour compromettre le système informatique. Par conséquent, les administrateurs déploient des solutions de sécurité efficaces capables de protéger le réseau de l'entreprise. Dans ce contexte, les systèmes de détection d'intrusions constituent une bonne alternative pour mieux sécuriser le réseau informatique.

II.1 Terminologies et concepts de base :

Il convient avant d'aborder certains aspects de la détection d'intrusions de revenir sur certaines définitions qui prêtent parfois à confusion. Nous nous sommes référés pour cela aux définitions données par les RFC2828 [041]. Ce sont ces définitions qui seront utilisées dans le reste de ce document.

- **Attaque :** tentative d'assaut sous forme d'une action ou combinaison d'actions intelligentes et délibérées dont l'objectif est la violation de la politique de sécurité.
- **Intrusion :** événement ou combinaison d'événements permettant d'avoir indûment accès (sans autorisation) à un système et ses ressources.
- **Vulnérabilité :** défaut ou faiblesse dans la conception d'un système, son implémentation, fonctionnement ou administration et qui pourrait être exploité pour violer la politique de sécurité.
- **Menace :** danger potentiel pouvant exploiter une vulnérabilité pour violer la politique de sécurité causant éventuellement des dégâts. Une menace peut être réelle ou non fondée
- **Détection d'intrusions :** analyse des événements ayant lieu dans un système dans le but de trouver en temps réel, en quasi temps réel ou en différé des tentatives d'accès non autorisé et aussitôt de notifier ces tentatives à l'administrateur du système.
- **faux-positif :** détection en absence d'attaque, alarme générée par un IDS pour un évènement légal.
- **faux-négatif :** absence de détection en présence d'attaque, non génération d'alarme par un IDS pour un événement illégal.

II.2 Les systèmes de détection d'intrusions :

Le concept de système de détection d'intrusions a été introduit en 1980 par James Anderson [004]. Mais le sujet n'a pas eu beaucoup de succès. Il a fallu attendre la publication d'un modèle de détection d'intrusions par Denning en 1987 [005] pour marquer réellement le départ du domaine.

La recherche dans le domaine s'est ensuite développée, le nombre de prototypes s'est énormément accru. Le gouvernement des États-Unis a investi des millions de dollars dans ce type de recherches dans le but d'accroître la sécurité de ses machines. La détection d'intrusion est devenue une industrie mature et une technologie éprouvée : à peu près tous les problèmes simples ont été résolus, et aucune grande avancée n'a été effectuée dans ce domaine ces dernières années , les éditeurs de logiciels se concentrant plus à perfectionner les techniques de détection existantes

Un IDS est un système informatique, composé généralement de logiciel et éventuellement de matériel, dont le rôle est la détection d'intrusions. Par définition, un IDS n'a pas de vocation préventive ou réactive dans la mesure où il n'empêche pas une intrusion de produire.

Il se contente plutôt d'analyser certaines informations en vue de détecter d'éventuelles activités malveillantes qu'il aura à notifier dans les plus brefs délais au responsable de la sécurité du système. C'est pour cette raison que la majorité des IDS opèrent en temps réel.

Toutefois, il y'a des IDS qui réagissent suite à la détection d'une intrusion en mettant fin par exemple à une connexion suspecte.

II.3. Trafic réseau : [003]

Les réseaux étendus et Internet reposent sur les protocoles TCP/IP. L'audit réseau concerne la capture du trafic transitant sur un point du réseau. Il s'agit de capturer ici les paquets TCP/IP, mais aussi, les paquets UDP, ICMP, etc. qui constituent les entités d'échanges élémentaires entre deux machines. Ce sont ces paquets (ou les connexions construites sur la base de ces paquets) qui sont analysés, selon l'approche de détection adoptée, afin de distinguer entre trafic légitime et trafic intrusif. Le trafic réseau peut être analysé à plusieurs niveaux de granularité. Souvent, l'analyse porte sur des paquets individuellement où chaque paquet est classé normal ou intrusif sur la base des informations constituant l'entête du paquet (information de contrôle) ou de son contenu (informations utiles). Le trafic réseau peut également être analysé au niveau « connexion » relative à un certain service. En plus de la complexité et variabilité qui caractérisent le trafic réseau, ce dernier contient énormément d'aléas et anomalies. Bellovin [096] et Paxson [095] révèlent que le trafic dans un réseau étendu contient plusieurs irrégularités telles que les adresses IP privées, des paquets de

synchronisation (SYN) avec des drapeaux d'urgence, des paquets d'acquittement (ACK) des paquets jamais reçus.

II.4. Approches de détection d'intrusions :

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

Un système de détection d'intrusions par recherche de signatures connaît ce qui est mal, alors qu'un système de détection d'intrusions par analyse de comportement connaît ce qui est bien.

On parle de la détection de malveillances (approche par scénario) et la détection d'anomalies (approche comportementale).

II.4.1 L'approche comportementale :(Anomaly Detection) [007]

Une approche proposée par James Anderson [004] puis reprise et étendue par Denning [005] consiste a utiliser des méthodes basées sur l'hypothèse selon laquelle l'exploitation d'une vulnérabilité du système implique un usage anormal de celui-ci. Une intrusion est donc identifiable en tant que déviation par rapport au comportement habituel d'un utilisateur. Bien sur une telle déviation peut avoir une autre cause qu'une attaque du système par exemple un changement de fonction de l'utilisateur au sein de l'entreprise. On s'attachera donc a trouver des méthodes possédant le plus fort taux de discrimination possible (c'est-à-dire ayant le plus fort taux de détection d'intrusions et le plus faible taux de fausses alarmes). De plus on se référera à un seuil au-delà duquel on considérera que le comportement est intrusif.

Cette approche, dont la question de base est « le comportement actuel de l'utilisateur est-il cohérent avec son comportement passé ? », est appelée approche comportementale. Pour caractériser le comportement normal d'un utilisateur (on parle de *profil*), des techniques employées pour modéliser le comportement se basent sur des méthodes statistiques, il est également possible d'envisager l'utilisation de systèmes experts ou de réseaux de neurones.

II.4.2 L'approche par scénario (Misuse Detection) : [007]

S'il n'est pas envisageable de décrire statistiquement le comportement d'un attaquant (on manque de données pour ce faire), il est possible de donner des règles sur sa manière de procéder. Ces règles prennent la forme de scénarios d'attaque exploitant des vulnérabilités du système précédemment identifiée. On parle parfois, pour cette approche, dont la question de base est « le comportement de

l'utilisateur correspond-il à une attaque connue ? », de modèle de scénario (signature d'attaque). Avec de tels systèmes, toute attaque non prévue reste indétectée.

Pour cette approche, les techniques de modélisation et d'implémentation les plus utilisées pour élaborer des IDS ayant connu un succès dans la pratique sont les systèmes experts, les algorithmes génétiques et d'autres.

II.4.3 Critiques des approches de détection:

i. L'approche comportementale :

De manière générale, cette approche présente plusieurs avantages. Le profil spécifie le comportement «normal», en revanche, il ne fait pas d'hypothèse sur les «anomalies» qui le violent : autrement dit, en signalant toute déviation par rapport au profil, il est possible de détecter *a priori* toute attaque qui viole ce profil, même dans le cas où cette attaque n'était pas connue au moment de la construction du profil. Cette capacité à détecter de «nouvelles attaques» constitue la principale force de la détection d'anomalies.

Cependant, l'approche comportementale souffre de quelques défauts intrinsèques :

- Les données utilisées en apprentissage doivent être exemptes d'attaques.
- En cas de modifications subites de l'environnement de l'entité modélisée, cette entité changera sans doute brutalement de comportement. Des alarmes seront donc levées. Pour autant, ce n'est peut-être qu'une réaction normale à la modification de l'environnement.
- Enfin, un utilisateur malicieux peut habituer le système (soit pendant la phase d'apprentissage, soit en exploitation si l'apprentissage est continu) à des actions malveillantes, qui ne donneront donc plus lieu à des alertes.

ii. L'approche par scénario :

Contrairement à un système de détection d'anomalies, ce type de détecteur d'intrusions nécessite une maintenance active : puisque par nature il ne peut détecter que les attaques dont les signatures sont dans sa base, cette base doit être régulièrement (sans doute quotidiennement) mise à jour en fonction de la découverte de nouvelles attaques. Aucune nouvelle attaque ne peut par définition être détectée, ce qui implique un taux plus élevé de faux négatifs.

De manière générale, les détecteurs de scénario se montrent fiables pour signaler les attaques référencées dans la base. Théoriquement, leur taux de faux positifs devrait rester très faible, car par définition une alerte n'est levée que dans le cas où la signature d'une attaque est observée.

Cependant, pour des raisons de performance, les signatures sont souvent trop simples, peuvent donc leur correspondent des actions tout a fait légitimes. Le taux de faux positif reste donc élevé avec les outils existant aujourd'hui.

De plus, une éventuelle connaissance de la base de signatures (particulièrement dans le cas des *patterns*) permet en principe à l'attaquant de construire précisément un scénario non détectable, cela ne fait que renforcer encore l'exigence de maintenir régulièrement la base.

Ce type de détecteur reste assez facile à mettre en oeuvre, ne nécessitant pas de phase d'apprentissage.

II.5 Classification des systèmes de détection d'intrusions : [024][042]

Suite au premier modèle de la détection d'intrusions proposé par Denning, plusieurs approches ont été proposées. Elles reposent sur diverses techniques de détection. Etant donnée la diversité de ces méthodes, des schémas de classification ont été proposés par Lunt [026], Sundavar [036], Debar [037][038], Axelsson [039] et Bace [040]. Debar [038] utilise cinq critères pour classer les systèmes de détection d'intrusions :

- ✓ **Méthode de détection** : deux principes de détections existent. L'approche comportementale modélise le comportement normal des utilisateurs, du système informatique et de l'activité réseau. Ensuite, toute déviation par rapport à la normale constitue un événement suspect. La deuxième approche, appelée détection par connaissances, recherche explicitement les signatures des attaques connues dans les fichiers de sécurité et le trafic réseau.
- ✓ **Source d'information** : les données analysées partagent les systèmes de détection d'intrusions en deux catégories : les systèmes de détection d'intrusions réseaux NIDS et les systèmes de détection d'intrusions hôtes HIDS. La première catégorie des IDS filtre le trafic réseau et se déploie généralement dans des endroits précis du réseau. Quelques exemples de NIDS : NetRanger [029], NFR [030], Snort [031], DTK [032], ISS RealSecure [033]. La deuxième catégorie des IDS analyse les données des journaux de sécurité établis par les systèmes d'exploitation et les applications qui tournent sur les machines. Ces IDS sont déployés directement sur les hôtes du réseau, Swatch[027], Tripwire[028], Security Manager[034] sont des exemples de HIDS.

Une version d'*IDS hybride* est possible et désormais supportée par différentes offres commerciales. Même si la distinction entre HIDS et NIDS est encore courante, certains HIDS possèdent maintenant les fonctionnalités de base des NIDS. Des IDS bien connus comme ISS RealSecure se nomment aujourd'hui "IDS hôte et réseau". Dans un futur proche, la différence entre les deux familles devrait s'estomper de plus en plus (ces systèmes devraient évoluer ensemble). De ces deux familles principales, de nombreuses variantes sont issues :

- IDS de noeud réseau NNIDS (Network Node IDS).
- IDS basé sur une application ABIDS (Application-Based IDS).
- IDS basé sur la pile SBID (Stack-Based IDS).
- Pot de miel (honeypot).
- Systèmes "capitonnés" (padded cell systems).

- ✓ **Réponses des IDS** : les systèmes de détection d'intrusions émettent des réponses actives qui influent directement la source d'attaque (Bloquer le compte d'un utilisateur, Suspendre des processus malveillants, Arrêter la machine, Déconnecter la machine du réseau, etc.), comme ils peuvent se restreindre à des réponses passives qui inscrivent l'événement suspect (Emettre un rapport, Générer une alarme, Créer des fichiers de sauvegarde, etc.)
- ✓ **Paradigme de détection** : la détection d'intrusions s'effectue en analysant l'état courant du système ou en supervisant les transitions des états normaux aux états dangereux. Durant ces deux types d'inspection, l'IDS récupère les informations en interrogeant directement le système ou en écoutant passivement les événements.
- ✓ **Mode de supervision** : l'analyse assurée par un système de détection d'intrusions peut être continue ou périodique dans le temps.

II.6 Evaluation des IDS : [003]

L'efficacité d'un IDS dépend de plusieurs facteurs. Nous nous limitons ici aux paramètres quantitatifs permettant d'analyser la qualité de détection. Il est d'abord question du taux de détection et du taux d'erreurs de détection.

Le taux de détection [Detection Rate] est le pourcentage des intrusions correctement détectées par rapport au nombre total d'intrusions signalées par l'IDS. Ce taux est également nommé probabilité de détection.

Par opposition, le taux d'erreur de détection [Detection Error Rate] est donc la proportion des intrusions dont la détection est erronée par rapport à la totalité des intrusions signalées.

D'autres métriques sont plus au moins utilisées, selon les besoins et les approches, et qui sont généralement fonction des paramètres schématiquement résumés dans la matrice de confusion ci-après.

Une matrice de confusion est un tableau bidimensionnel où les lignes représentent les vraies classes des données analysées tandis que les colonnes représentent les classes prédites par le système de détection.

		Classe détectée (prédite)	
	→	Normale	Attaque
Classe réelle	Normale	Vrai négatif TN [True Négative]	Faux positif FP [False positive]
	Attaques	Faux négatif FN [False Négative]	Vrai positif TP [True Positive]

TAB I.1 Sémantique d'une matrice de confusion

De cette matrice de confusion, il ressort que :

- un vrai négatif TN [True Negative] est une activité normale considérée en tant que telle par l'IDS.
- une fausse alerte, appelée dans le jargon de détection d'intrusions faux positif, FP [False Positive], est une activité normale considérée comme une attaque par l'IDS.
- un faux négatif FN [False Negative] est une attaque ratée (considérée activité normale) par l'IDS.
- un vrai positif TP [True Positive] est une attaque correctement détectée par l'IDS.

On trouve dans le jargon des classificateurs des métriques parfois utilisées pour l'évaluation des IDS notamment lorsque la détection d'intrusions est vue comme un problème de classification. Ainsi, la précision et le rappel, concernant les intrusions, se définissent par les équations qui suivent :

$$\text{Pr}\,écision = \frac{TP}{(TP + FP)} \qquad \text{(I.1)}$$

$$Rappel = \frac{TP}{(TP + FN)} \qquad \text{(I.2)}$$

La précision renseigne sur l'exactitude des intrusions signalées. Quant au rappel, Il traduit le taux d'intrusions correctement détectées par rapport au nombre total d'intrusions existantes (TP+FN). Le taux des fausses alertes (taux des faux positifs) correspond au nombre de faux positifs par rapports au nombre total d'intrusions signalées :

$$Taux\_de\_faux\_positifs = \frac{FP}{(TP + FP)} \qquad \text{(I.3)}$$

Pour avoir une idée sur la charge de travail nécessaire pour l'administration d'un IDS dans sa phase opérationnelle, le taux des fausses alertes est souvent exprimé par le nombre des faux positifs durant un intervalle de temps (ex. 10 fausses alertes par jour).

Une erreur de détection (les cases grises sur la matrice ci-dessus) peut être une fausse alerte, ou une intrusion non détectée. Un taux de faux positifs trop élevé nuit à la crédibilité des résultats et provoquera une baisse de vigilance du côté des responsables concernés. Un taux de faux négatifs élevé rend, quant à lui, l'IDS inefficace dans la mesure où la détection de la majorité des intrusions est ratée. Étant donné qu'il est difficile d'optimiser ces deux paramètres en même temps, on recherche, selon les cas, le meilleur compromis taux de fausses alertes/taux de vrais négatifs sous-jacent. Il y a lieu également de signaler que du point de vue classification, un faux positif ou faux négatif est une erreur de classification. Cependant, une erreur de classification peut ne pas être une erreur de détection : signaler une intrusion donnée alors qu'il s'agit réellement d'une autre intrusion est une erreur de classification mais pas une erreur de détection (en effet, c'est une erreur d'identification de l'attaque par son nom).

II.7 Limites des IDS actuels : [035]

La plupart des IDS souffrent actuellement de quelques problèmes :

- Dans la plupart des cas, les systèmes de détection d'intrusions sont faits d'un seul bloc ou module qui se charge de toute l'analyse. Ce système monolithique demande qu'on lui fournisse beaucoup de données d'audit, ce qui utilise beaucoup de ressources de la machine surveillée. L'aspect monolithique pose également des problèmes de mises à jour et constitue un point d'attaque unique pour ceux qui veulent s'introduire dans le système d'informations.
- Même en implémentant les deux types d'approches, certaines attaques sont indécelables et les systèmes de détection sont eux-mêmes attaquables. Les approches comportementale et par scénarios ont elles-mêmes leurs limites.

- Les mises à jour de profils, de signatures d'attaques ou de façon de spécifier des règles sont généralement difficiles. De plus, les systèmes de détection d'intrusions demande de plus en plus de compétence à celui qui administre le système de sécurité.

II.8 Qualités requises des systèmes de détection d'intrusions : [003]

Les systèmes de détection d'intrusions actuels tendent à garantir les cinq propriétés suivantes :

- ✓ Exactitude de détection : elle se traduit par une détection parfaite des attaques avec un risque minimal de faux positifs.
- ✓ Performance : une détection rapide des intrusions avec une analyse approfondie des événements est indispensable pour mener une détection efficace en temps réel.
- ✓ Complétude : une détection exhaustive des attaques connues et inconnues.
- ✓ Tolérance aux fautes : les systèmes de détection d'intrusions doivent résister aux attaques ainsi qu'à leurs conséquences.
- ✓ Rapidité : une analyse rapide des données permet d'entreprendre instantanément les contre mesures nécessaires pour stopper l'attaque et protéger les ressources du réseau et du système de détection d'intrusions.

Plus de ces critères d'autres caractéristiques sont fondamentales

- ✓ Capacité de fonctionner continuellement et avec un minimum d'intervention humaine.
- ✓ Difficulté pour un attaquant de le désactiver ou modifier sa configuration.
- ✓ Capacité de se contrôler lui-même et de détecter s'il vient de faire l'objet de manipulation de la part d'un attaquant.
- ✓ Utilisation minimale de ressources (de calcul, stockage, etc.) sur le système sur lequel il est installé.
- ✓ Capacité d'accepter des mises à jour et des modifications de configuration pour rendre compte des nouvelles dispositions de la politique de sécurité et les changements susceptibles de s'opérer dans l'organisation(nouvelles acquisitions, restructuration, etc.).
- ✓ Facilité et simplicité de déploiement : facilité d'installation et de configuration, portabilité, etc.).
- ✓ Interopérabilité avec d'autres systèmes et outils de la sécurité informatique.

Conclusion :

La détection d’intrusions joue un rôle complémentaire indispensable par rapport aux mécanismes préventifs. Elle est une discipline relativement jeune et manque encore de maturité. Les systèmes de détection d’intrusions actuels souffrent soit de l’incapacité de détecter suffisamment les nouvelles attaques, soit d’un taux de fausses alertes trop élever qui les rend inefficaces dans la pratique. Le point clé est d’améliorer la capacité de détection de nouvelles attaques tout en générant le plus petit nombre possible de fausses alertes.

Nous avons présenté dans ce chapitre les qualités requises des systèmes de détection d'intrusions. Afin de remplir ces objectifs, diverses méthodes de détection d'intrusions ont été proposées. Elles se basent principalement sur deux principes de détection : la détection par anomalie et la détection par la connaissance. Nous avons expliqué ces deux principes de détection. Par ailleurs nous avons souligné les limites des systèmes de détection d'intrusions actuels. Afin de résoudre ces limites, nous proposons par la suite deux architectures basées sur la classification non supervisée de données.

Dans le chapitre suivant nous présentons un état de l’art des méthodes d’intelligence artificielle, plus particulièrement les méthodes de classification non supervisée, qui ont été utilisées pour la détection d’intrusions.

Introduction :

La détection des tentatives d'attaques sur un réseau est une problématique très importante dans le domaine de la sécurité informatique. Les technologies classiques de protection des réseaux de type Firewall filtrant sont en effet inefficaces contre la plupart des attaques actuelles. Aussi sont apparus de nouveaux équipements réseaux pour prendre en compte ces carences, systèmes de détection d'intrusions réseaux, dont le but est de détecter les tentatives d'attaque qu'un firewall ne peut pas bloquer. Malheureusement, en pratique, les IDS soit génèrent un nombre trop élevé d'alarmes où ils sont incapables de détecter de nouvelles attaques. L'utilisation des méthodes de la classification non supervisée pour modéliser le comportement des utilisateurs peut être efficace pour construire des IDS générant le minimum de fausses alertes. De tels modèles ont déjà fait leur apparition dans des applications similaires en se basant sur différentes méthodes de l'intelligence artificielle.

Nous reviendrons donc,dans ce chapitre, tout d'abord sur le problème de partitionnement de données, nous passerons ensuite à la présentation de quelques méthodes de la classification non supervisée, et nous terminons par une exposition des travaux liés à ce domaine.

I. Problème de la classification :

La classification est le processus qui permet de regrouper un ensemble d'objets en sous-ensembles de tel sorte que les objets appartenant au même sous ensemble présentent la plus grande similarité, et que deux objets de sous groupes différents soient le moins similaire possible ou le plus dissimilaire [012].

Par classification, on entend l'identification de la classe d'un objet à partir des attributs le décrivant. Il s'agit, à titre d'exemple, d'identifier la maladie (classe) d'un patient (objet) à partir des symptômes (attributs descriptifs) qu'il manifeste. On peut l'assimiler à une fonction associant à une description d'un objet, la classe correspondante [003].

La classification est à la base de la plus part des applications en reconnaissance des formes. Celles-ci sont nombreuses et s'intéressent à des domaines aussi variés que l'analyse de l'écriture, la détection automatique d'objets particuliers dans des images (des visages par exemple), l'analyse des données bancaires, du marketing, la mesure d'audiences sur Internet ou enfin la détection d'intrusions dans des réseaux informatiques qui est le propos du présent ouvrages.

La classification, bien que primordiale dans toutes les applications citées précédemment, n'est pas la seule phase à considérer avec attention lors du développement d'un outil d'extraction et d'analyse de l'information [062]. La figure suivante montre le parcours de l'information à classifier.

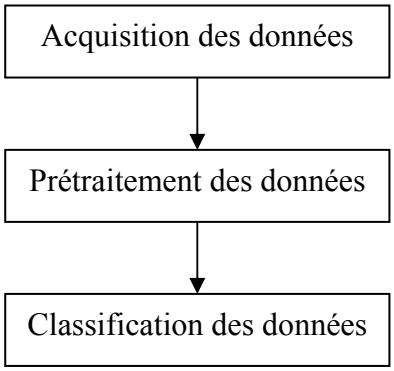

FIG II.1 Le parcours de l'information à classifier

- **Acquisition des données :** d'une manière générale, il s'agit à ce niveau de mettre en place l'ensemble de l'instrumentation (capteurs, matériel d'acquisition,…), de façons à reproduire le phénomène observé le plus fidèlement possible. C'est l'opération de transformation de l'information à traiter en signaux numériques manipulables par ordinateurs ou bien la

numérisation de l'information. Dans notre cas, il s'agit de placer des sondes pour écouter le trafic réseau.

- **Prétraitement des données :** Cette phase correspond au filtrage des informations en ne conservant que ce qui est pertinent dans le contexte d'étude, c'est la partie segmentation dans la classification d'images. Dans notre cas, il s'agit de l'enregistrement des connexions TCP/IP.
- **Classification des données :** Elle correspond à l'étape de décision et pour cela plusieurs méthodes se présentent pour la résolution. Pour la classification des connexions TCP/IP nous appliquerons un algorithme basé sur le fonctionnement des fourmis artificielles, que nous détaillerons dans le paragraphe suivant.

Le problème de la classification peut être modélisé comme suit : étant donné un ensemble $O = \{o_1, o_2,\ldots,o_N\}$ de N objets sélectionnés par l'expert du domaine sur lesquels on veut effectuer notre classification. Chaque objet o_i correspond à un vecteur de M valeurs numériques (M attributs numériques) $a_1, a_2, \ldots, a_M$. Le tableau suivant donne un exemple de données possibles.

Objets	Attributs
	a_1 , …, a_M
O_1	1.32 ….. 3.67
⋮	⋮ ⋮ ⋮
O_N	8.98 …… 2.75

FIG II.2 Exemple de données numériques

Le problème de la classification consiste alors à la recherche d'une partition de ces points en K classes de telle sorte que les points d'un même groupe soient plus similaires entre eux qu'avec les points d'autres groupes.

II. Taxonomie des méthodes de classification :

Il existe à peu prés autant de problèmes de classification qu'il y a de domaines d'applications envisageables. De ce fait, un grand nombre de méthodes différentes a été établi pour tous les résoudre. Cependant du fait que certaines de ces approches partages des caractéristiques communes, soit dans la façons d'appréhender le problème « apprentissage supervisé ou non », soit par exemple dans la sortie réalisée « groupes disjoints ou classification floue », il a été possible de les regrouper

sous la forme d'une hiérarchie de méthodes appelée taxonomie. Nous présentons ci-après une taxonomie dérivée de celle de Jain et Dubes [067]

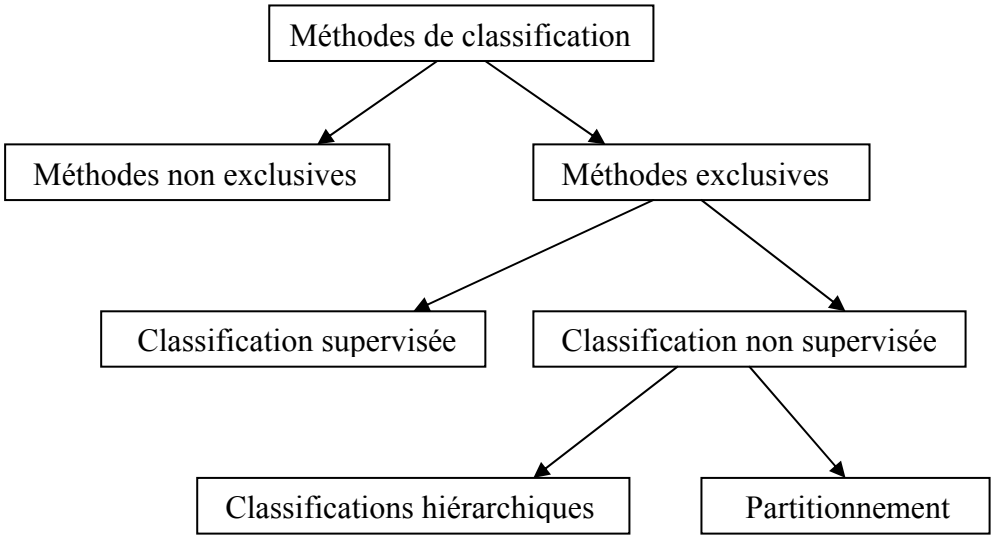

FIG II.3 Les méthodes de classification [001]

- **Classification exclusive/non exclusive** : Une classification exclusive est un partitionnement des objets : un objet n'appartient qu'à une classe et une seule. Au contraire, une classification non exclusive autorise qu'un objet appartienne à plusieurs classes simultanément. Les classes peuvent alors se recouvrir, on parle de la classification floue.
- **Classification supervisée / non supervisée :** Dans la classification non supervisée, aucune information n'est fournie à la méthode (les objets ne sont pas étiquetés), ici on cherche à découvrir de nouveaux groupes d'objets. En classification supervisée, les objets sont étiquetés, il s'agit d'utiliser les groupes connus pour classer les nouveaux objets.
- **classification hiérarchique/partitionnement :** Une méthode de classification hiérarchique construit une séquence de partitions imbriqués, que l'on visualise par exemple par un dendogramme (figure II.4) , alors qu'un *partitionnement* ne construit qu'une seule partition des données.

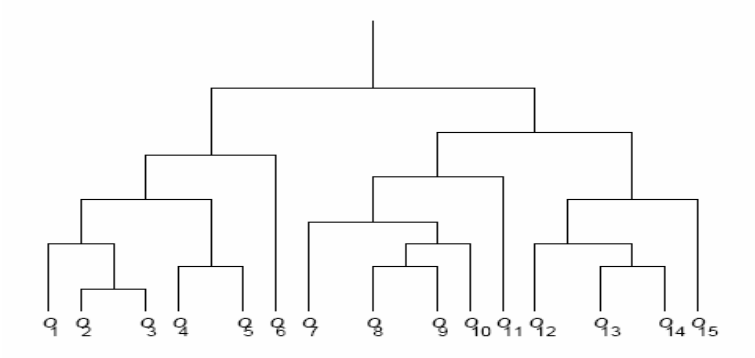

FIG II.4 Exemple d'un dendogramme.

III. Classification supervisée et classification non supervisée : **[062]**

La classification peut être appliquée selon deux motivations différentes. Elle peut être utilisée tout d'abord comme un outil de partitionnement. Dans ce cas le bénéfice de la classification et d'autoriser une étude plus concise de données initiales en essayant de trouver le plus petit nombre de groupes de données qui va le mieux possibles conserver ou mettre en lumière l'information retenue dans ces même données. Les groupes alors formés sont les plus cohérents et homogènes possibles et sont appelés des *clusters*. Dans ce cas, le problème est alors de définir la similarité entre objets. Typiquement, la similarité entre objets est estimée par une fonction calculant la distance entre ces objets. Une fois cette fonction distance définie, la tâche de *clustering* consiste à réduire au maximum la distance entre membres d'un même cluster, tout en augmentant au maximum la distance entre clusters.

Ces systèmes de classification ne reçoivent aucune information extérieure (provenant d'un opérateur humain par exemple) pour créer des partitions. Cette approche porte en conséquence la dénomination de *classification non supervisée.* De nombreuses techniques peuvent être envisagées: classification hiérarchique, classification par partition, classification statistique, classification floue (Fuzzy clustering), classification par réseaux de neurones ou classification par algorithmes génétiques….

La deuxième approche nécessite l'intervention d'un système extérieure (qui se compose le plus souvent d'un expert humain) pour étiqueter les informations avant leur classification. Dans ce cas, la classification va apprendre à trouver les groupes de données en fonction des étiquetages précédents et va donc s'entraîner à reconnaître les données. L'objectif de cette démarche est ensuite de pouvoir fournir au système entraîné des données non étiquetées pour qu'il puisse continuer à les répartir dans

les groupes adéquats sans autre intervention humaine. On parle alors d'une approche *d'apprentissage supervisé*. On peut citer comme méthodes de classification supervisées toutes les méthodes basées sur un ensemble d'apprentissage tel que la plupart des méthodes du Data Mining (Arbre de décision, Raisonnement à base de cas, Réseaux de neurones…).

IV. Le codage de l'information :

A l'origine, l'information obtenue n'est pas forcément manipulable. Pour cette raison, on doit apporter quelques modifications (formaliser et unifier les données) sur ces informations afin qu'elles soient traitable.

IV.1. Les types de données :

En classification, on distingue deux types de données :

- Les données individus-variables : Elles se présentent sous forme d'un tableau qui présente un ensemble de variables observées sur plusieurs unités statiques (individus). Les types des variables peuvent être quantitatives, c'est-à-dire qu'elles prennent des valeurs numériques ou bien qualitatives, dans ce cas, elles prennent des valeurs symboliques qui déterminent des catégories.

- Les données de proximité : elles présentent les distances ou les similitudes entre les éléments d'un groupe, c'est une matrice symétrique par rapport à la diagonale.

IV.2. Notion de similarité sur les variables :

La similarité est un facteur qui se mesure pour chaque couple de données et varie entre 0 et 1, il est égal à 1 si les données sont similaires et 0 sinon. Et cela selon le type :

- Les variables disjonctives : elle est égale à 1 si les deux objets présentent la même caractéristique.
- Les variables qualitatives : elle est égale à 1 si les deux objets présentent la même variable (couleur, sexe,…).
- Les variables quantitatives : elle mesure l'écart entre les deux variables de manière relative à l'intervalle des variables.

IV.3. La relation entre la similarité et la distance :

La notion de similarité est complémentaire à la notion de distance, comme le montre la formule suivante :

Distance (A, B) = 1 – Similarité (A, B) (II.1)

Deux objets similaires veut dire que la distance qui les sépare est nulle.

V. Méthode de la classification :

V.1. Les centres mobiles :

L'algorithme des centres mobiles est un algorithme de partitionnement itératif connu sous le nom de *K-Means.* Il a été introduit par MacQueen en 1967. Cet algorithme peut être utilisé sur des jeux de données volumineux. L'algorithme K_Means utilise l'erreur quadratique comme critère d'évaluation des partitions. En premier temps les objets sont regroupés autour de K centres arbitraires en affectant chacun des objets au centre de gravité le plus proche. On calcule les nouveaux centres de gravité et on refait l'opération d'affectation jusqu'à ce que les objets se stabilisent [062]. La figure suivante donne un exemple d'une partition associée à trois centres dans le plan.

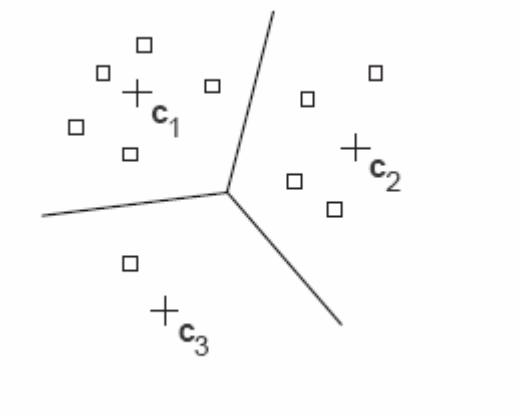

FIG II.5 Exemple de partition obtenue par les centres mobiles

Algorithme II.1 : Algorithme des centres mobiles K-Means

(1) **tantque** l'inertie intraclasse ne s'est pas stabilisée **faire**
(2) Générer une nouvelle partition *P'* en affectant chaque objet à la classe dont le centre est le plus proche
(3) Calculer les centres de gravité des classes de la nouvelle partition *P'*
(4) $P \leftarrow P'$
(5) **fintantque**
(6) **retourner** *P*

D'autres algorithmes ont été proposés tel que H-Means, KH-Means, K-Harmonic Means introduite dans [011], l'algorithme SBKM [010] qui signifie « symmetry –based K-Means », et d'autres variants tel que J-Means [009] qui apporte la notion d'objets occupés ou non occupés et minimise l'erreur quadratique comme le fait K-Means, l'hybridation de cet algorithme avec les K-Means ou les KH-Means donne une autre approche qui est J-Means+. WaveFront [008] et K-D Means [006] sont aussi des approches inspirées des K-Means dans le but de pallier certaines lacunes des K-Means original.

V.2. Les fourmis artificielles :

L'informatique et notamment l'intelligence artificielle ont connus quelques modèles inspirés de la biologie pour résoudre des problèmes d'optimisation, de classification ou d'autres.

Dans la nature, les fourmis offrent un modèle stimulant pour le problème du partitionnement. L'exemple du tri collectif du couvain ou de la constitution de cimetières est le plus marquant. Certains travaux expérimentaux montrent que certaines espèces de fourmis sont capables d'organiser spatialement divers éléments du couvain : les oeufs, les larves et les nymphes.

Le modèle de règles utilisé est relativement simple : [001]

- Lorsqu'une fourmi rencontre un élément du couvain, la probabilité qu'elle s'en empare est d'autant plus grande que cet élément est isolé.

- Lorsqu'une fourmi transporte un élément du couvain, elle le dépose avec une probabilité d'autant plus grande que la densité d'éléments du même type dans le voisinage est grande.

Les probabilités de ramasser un objet (p_p) et de le déposer (p_d) sont données par les équations suivantes :

$$P_p = \left(\frac{k_1}{k_1 + f} \right)^2 \qquad \text{(II.2)}$$

$$P_d = \left(\frac{f}{k_2 + f} \right)^2 \qquad \text{(II.3)}$$

Où k_1 et k_2 sont des constantes positives et f correspond à la proportion d'éléments perçus dans le voisinage de la fourmi.

Quand il y a peu d'objets dans le voisinage de l'objet convoité par la fourmi, $f << k_1$ ce qui signifie que p_p est proche de 1 et l'objet a beaucoup de chance d'être ramassé. Inversement, quand le voisinage est dense en éléments, $f >> k_1$ et alors p_p est proche de 0.

f correspond au nombre d'objets rencontrés durant les T derniers déplacements divisé par le nombre maximum d'objets qui auraient pu être rencontrés. Les résultats obtenus par simulation montrent l'apparition de groupes d'objets, les agents ainsi définis permettent donc de ranger une surface sur laquelle des objets ont été éparpillés.

Le comportement de rassemblement se transforme alors en tri, ce qui se rapproche de notre problème de partitionnement.

V.2.1. Algorithme de LUMER et FAIETA :

LUMER et FAIETA ont proposé un algorithme inspiré du comportement des fourmis réelles, cet algorithme utilise une mesure de dissimilarité entre les objets (sous la forme d'une distance euclidienne). Les objets correspondent à des points d'un espace numérique à M dimensions sont plongés dans un espace discret de dimension moindre (typiquement de dimension 2), Ce nouvel espace ressemble alors à une grille G dont chaque case peut contenir un objet. Les fourmis se déplacent sur la grille et chacune d'elles perçoit une région R_s de s ×s cases dans son voisinage. La figure suivante donne un exemple d'une grille avec une fourmi représentée par ×, son périmètre de détection en trait épis et les autres formes représentent les objets.

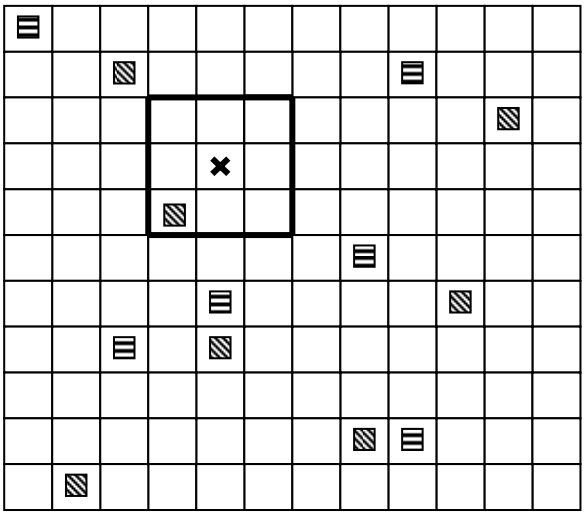

FIG II.6 Grille utilisée par l'algorithme de LUMER et FAEITA

Les probabilités de prendre un objet et de le déposer sont modifiées de la façon suivante :

$$P_p(o_i) = \left(\frac{k_1}{k_1 + f(o_i)} \right)^2 \qquad \text{(II.4)}$$

$$P_d(o_i) = \begin{cases} 2f(o_i) & \text{si } f(o_i) < k_2 \\ \\ 1 & \text{si } f(o_i) \geq k_2 s \end{cases} \qquad \text{(II.5)}$$

La fonction de densité locale dépend de l'objet considéré o_i et de sa position sur la grille $r(o_i)$. Elle est calculée de la manière suivante :

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum_{o_j \in R_s(r(o_i))} 1 - \frac{d(o_i, o_j)}{\alpha} & \text{Si } f \rangle 0 \\ \\ 0 & \text{Sinon} \end{cases} \qquad \text{(II.6)}$$

Où $r(o_i)$: la position de l'objet o_i sur la grille G.

S : taille du voisinage.

o_i : Les objets présents dans le voisinage de o_i.

α : Facteur d'échelle déterminant dans quelle mesure la dissimilarité entre objets est prise en compte. Si α est top grand, il y aura peu de groupe car les objets seront considérés comme similaires et inversement si α est petit.

L'algorithme suivant donne les étapes de la méthode en utilisant A fourmis $\{a_1, a_2, \ldots a_A\}$.

Algorithme II.2 : Algorithme de LUMER et FAEITA

(1) Placer aléatoirement les *N* objets *o*1, . . . , *oN* sur la grille *G*
(2) **pour** *T* = 1 à *T*max **faire**
(3) **pour tout** *aj* ∈ *{a*1, . . . , *aA}* **faire**
(4) **si** la fourmi *aj* ne transporte pas d'objet et *r*(*oi*) = *r*(*aj*) **alors**
(5) Calculer *f*(*oi*) et *pp*(*oi*)
(6) La fourmi *aj* ramasse l'objet *oi* suivant la probabilit´e *pp*(*oi*)
(7) **sinon**
(8) **si** la fourmi *aj* transporte l'objet *oi* et la case *r*(*aj*) est vide **alors**
(9) Calculer *f*(*oi*) et *pd*(*oi*)
(10) La fourmi *aj* dépose l'objet *oi* sur la case *r*(*aj*) avec une probabilité *pd*(*oi*)
(11) **finsi**
(12) **finsi**
(13) Déplacer la fourmi *aj* sur une case voisine non occup´ee par une autre fourmi
(14) **finpour**
(15) **finpour**
(16) **retourner** l'emplacement des objets sur la grille

Généralement l'algorithme obtient plus de classes qu'il n'en existe dans la distribution initiale. Lumer et Faieta ont apporté un certain nombre d'améliorations pour réduire cette tendance :

- chaque fourmi a été dotée d'une vitesse *v* pouvant varier entre 1 et $V_{\max}$ d'une fourmi à l'autre. Le calcul de $f(o_i)$ a été modifié pour que les fourmis les plus rapides soient moins sensibles à la dissimilarité entre deux objets que les plus lentes.

– chaque fourmi dispose d'une mémoire à court terme lui permettant de se rappeler de l'emplacement des *m* derniers objets qu'elle a déposés. A chaque objet ramassé, la fourmi compare les caractéristiques de cet objet aux *m* objets de sa mémoire et se dirige alors vers le plus similaire. Cette technique permet de réduire le nombre de groupes d'objets équivalents.

– comme les objets ont de moins en moins de chance d'être déplacés, un mécanisme de redémarrage a été ajouté : si une fourmi n'a pas effectué d'action depuis un certain temps, elle peut détruire un groupe en ramassant un objet.

V.2.2. L'algorithme ANTClass :

Proposé par N. Monmarché [001], cette méthode s'inspire de celle de Lumer et Faieta en apportant quelques modifications pour pallier aux inconvénients de l'ancienne méthode. L'algorithme ANTCLass utilise une grille toroïdale, ce qui signifie que les fourmis passent d'un coté de la grille à l'autre en un seul pas, sa taille est déterminée automatiquement en fonction du nombre d'objets à traiter . Si N représente le nombre d'objets, G comporte L cases par coté : $L = \sqrt{2N}$

Chaque fourmi a la possibilité de transporter plusieurs objets à la fois, comme un élément de la grille (tas) peut contenir plusieurs objets. La méthode inclut aussi une hybridation avec la méthode du centre mobile (K-Means) afin d'améliorer le résultat obtenu par les fourmis. L'algorithme se compose alors de deux parties : la procédure Ants et la procédure du K-Means.

Le positionnement des objets à partitionné peut se fait d'une manière aléatoire ou suivant le résultat obtenu par une analyse factorielle des correspondances AFC.

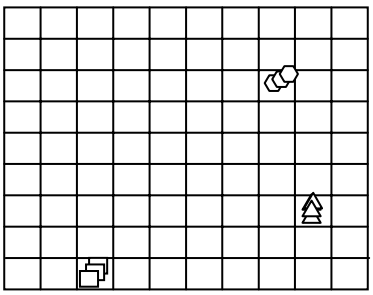

FIG II.7 Grille utilisée par l'algorithme ANTClass

Initialement, les A fourmis $\{a_1, \ldots, a_A\}$ sont disposées aléatoirement sur la grille en vérifiant qu'une case ne peut accueillir qu'une seule fourmi. A chaque itération de l'algorithme, chaque fourmi a_i se déplace aléatoirement sur la grille à une vitesse v (a_i). Concrètement, si ($x(a_i)$, $y(a_i)$) sont les coordonnées de la fourmi a_i sur G, après un déplacement, la nouvelle position sera dans

l'intervalle ([$x(a_i)-v(a_i)$, $x(a_i)+v(a_i)$], [$y(a_i)-v(a_i)$, $y(a_i)+v(a_i)$]) en tenant compte du fait que G est toroïdale. Enfin la direction choisie par une fourmi dépend de sa direction précédente : elle a une probabilité égale à 0.6 de continuer tout droit et de 0.4 de changer de direction[2]. Dans ce cas, elle a une chance sur deux de tourner de 45 degrés à gauche ou à droite.
Chaque fourmi est dotée d'une capacité de transport c (a_i).

Afin de déterminer les règles que les fourmis vont utiliser sur la grille pour manipuler les objets, il faut leur donner certaines mesures :
– la distance maximale entre deux objets de l'ensemble O :

$$d^*(O) = \max_{(i,j)\in\{1,\ldots,N\}^2} \{d(x_i, x_j)\} \quad \text{(II.7)}$$

– La distance moyenne entre les objets de l'ensemble O :

$$\overline{d(O)} = \frac{2}{N(N-1)} \sum_{(i,j)\in\{1,\ldots,N\}^2, i<j} d(x_i, x_j) \quad \text{(II.8)}$$

– La distance maximale entre les objets d'un tas T_j et son centre de gravité g_j :

$$d_g^*(T_j) = \max_{x_i \in T_j} \{d(x_i, g_j)\} \quad \text{(II.9)}$$

– La distance moyenne entre les objets d'un tas T_j et son centre de gravité g_j :

$$\overline{d_g}(T_j) = \frac{1}{|T_j|} \sum_{x_i \in T_j} d(x_i, g_j) \quad \text{(II.10)}$$

Ramassage d'objets

Si la fourmi *ai* ne transporte pas d'objet et qu'elle se trouve sur une case contenant un objet ou un tas d'objets T_j , elle a une probabilité p_p de ramasser un objet :

$$P_p(T_j)=\begin{cases}1 & \text{Si } |T_j|=1\\ \min\left(\left(\dfrac{\overline{d_g}(T_j)}{\overline{d(o)}}\right)^{k_1},1\right) & \text{Si } |T_j|=2\\ 1-0.9\left(\dfrac{\varepsilon+\overline{d_g(T_j)}}{\varepsilon+d_g^*(T_j)}\right)^{k_1} & \text{Sinon}\end{cases} \qquad \text{(II.11)}$$

Où ε est une petite valeur positive (10^{-5}) et $|T_j|$ le nombre d'objets dans le tas. La fourmi ramasse les objets jusqu'à ce que sa capacité soit atteinte en choisissant à chaque fois l'objet *oi* le plus éloigné du centre de gravité du tas. Si la case ne contient qu'un seul objet ($|T_j|$ = 1), il est systématiquement ramassé par la fourmi. Si la case contient un tas de deux objets ($|T_j|$ = 2 et $T_j=\{o_u,o_v\}$), la probabilité de ramasser l'un des objets dépend de la distance entre les deux objets et le centre de gravité g_j $\left(d(x_u,g_j)=d(x_v,g_j)=\overline{d_g}(T_j)\right)$ et de la distance moyenne entre tous les objets $\left(\overline{d}(O)\right)$.

Enfin, si le tas se compose de plus de deux objets, *pp* est proche de 1 quand la distance moyenne au centre est négligeable devant la distance au centre de l'objet le plus éloigné.k_1 est un paramètre réel positif permettant de contrôler la forme de la densité de $p_p(T_j)$ quand $|T_j|\rangle 2$ .Si la capacité de la fourmi est supérieure à 1, elle ramasse autant d'objets que sa capacité le lui permet.

Dépôt d'objets :

Si la fourmi transporte un objet, et qu'elle se trouve sur une case contenant un ou plusieurs objets, sa probabilité de déposer l'objet o_i sur le tas T_j est donnée par :

$$p_d(o_i,T_j)=\begin{cases}1 & \text{Si } d(x_i,g_j)\le d_g^*(T_j)\\ 1-0.9\min\left(\left(\dfrac{d(x_i,g_j)}{\overline{d(o)}}\right)^{k_2},1\right) & \text{Sinon}\end{cases} \qquad \text{(II.12)}$$

Où k_2 est un paramètre réel positif permettant de contrôler la forme de la densité de $p_d(o_i, T_j)$ quand $d(x_i, g_j) > d^*g(T_j)$. Si l'objet o_i transporté par la fourmi est moins éloigné du centre g_j du tas T_j que l'objet du tas le plus éloigné de ce centre, elle dépose systématiquement o_i. Sinon, plus la distance entre o_i et g_j ($d(x_i, g_j)$) est grande par rapport à la distance moyenne entre les objets de la base ($\overline{d}(O)$), plus la probabilité de déposer sera faible.

Si la capacité de la fourmi est supérieure à 1 et qu'elle transporte plusieurs objets, la probabilité de déposer le tas T_j qu'elle transporte sur le tas T_j est calculée de la même façon que pour un objet unique en remplaçant x_i par le centre de gravité g_j des objets transportés.

Patience des fourmis

S'il y a trop de fourmis, on peut rencontrer le problème suivant : tous les objets sont transportés, ce qui n'offre plus la possibilité de dépôt pour les fourmis (absence de tas).

Dans le cas où la capacité des fourmis est égale à 1, il suffit de choisir un nombre de fourmis inférieur aux nombres d'objets. Dans le cas contraire (capacité des fourmis supérieur à 1), on dote les fourmis d'une patience $p(a_i)$. Quand une fourmi effectue un nombre de déplacement supérieur à $p(a_i)$ sans réussir à déposer les objets qu'elle transporte,elle les dépose sur la case où elle se trouve si elle est vide ou l'une de son voisinage dans le cas contraire.

Mémoire des fourmis :

Les fourmis sont dotées d'une mémoire local $m(a_i)$, elles sauvegardent les centres de gravité des objets déposés puisque les fourmis peuvent transporter plusieurs objets.

Concernant la taille de la mémoire à utiliser et en dehors de toute expérimentation, on peut envisager qu'une grande mémoire sera coûteuse à gérer du point de vue du temps de calcul puisqu'à chaque ramassage d'un ou plusieurs objets la fourmi calcule la distance entre le centre de gravité de ce qu'elle vient de ramasser et le centre de gravité de chacun des tas mémorisés. D'un autre côté, si la mémoire est trop petite, comme la fourmi choisit de se diriger vers le tas dont le centre de gravité est le plus proche, son éventail de choix pourrait être trop restreint. Ce qui signifie que son déplacement pourrait être inutile si elle ne dépose rien sur le tas qu'elle a choisi d'atteindre.

Les fourmis gèrent leurs mémoires sous la forme d'une liste FIFO où les positions les plus anciennes sont oubliées quand la fourmi mémorise de nouvelles positions. Cette mémorisation est effectuée quand la fourmi dépose un ou plusieurs objets sur un tas.

Hybridation avec les centres mobiles :

Le partitionnement construit par les fourmis n'est pas obligatoirement optimal au sens de l'inertie intra-classe même si les règles de ramassage et de dépôt des objets tendent à agglomérer les objets à des tas dont le centre est proche. L'algorithme des centres mobiles (ou plus couramment K-means) offre une réponse simple à la non-uniformité de la partition construite par les fourmis. Comme les fourmis agissent localement il peut rester un certain nombre d'objets « méritant » d'appartenir à une classe dont le centre est beaucoup plus proche. K-means est alors utilisé pour corriger ce type de défaillance.

Dans les versions précédentes de AntClass les objets isolés (c'est-à-dire se trouvant sur une fourmi ou seul sur la grille) sont affectés au tas dont le centre de gravité était le plus proche. Cela est intéressant quand une fourmi est interrompue dans une opération qu'elle pouvait accomplir ou pour les objets qui n'auraient pas été visités sur la grille. Ce deuxième point peut être évité en donnant plus d'itérations aux fourmis, certes au détriment du temps de calcul. Cependant, on peut raisonnablement penser qu'un objet dont les paramètres sont trop bruités ait du mal à trouver une place sur un tas, il a donc plus de chances d'être seul sur la grille (si une fourmi impatiente l'a finalement laissé tomber) ou bien transporté par une fourmi au moment de l'interruption. Il peut être alors intéressant qu'il reste isolé afin d'attirer l'attention, l'algorithme K-means ayant beaucoup de chance de le laisser seul dans sa classe.

Algorithme :

L'algorithme AntClass désigne une succession d'itérations des fourmis et des centres mobiles. Les fourmis ont pour tache de réduire le nombre de classes et les centres mobiles d'améliorer globalement la partition découverte par les fourmis.

L'algorithme suivant donne la structure générale de la méthode de classification par les fourmis (appelé Ants).

Algorithme Ants : regroupement des objets par les fourmis. Les indications entre crochets concernent le cas où les fourmis ont une capacité de transport supérieure `a 1.

Algorithme II.3 : L'algorithme Ants(Grille *G*)

(1) **pour** *t* = 1 à *T* **faire**
(2) **pour** *k* = 1 à *A* **faire**
(3) Déplacer la fourmi *ak* sur une case non occupée par une autre fourmi
(4) **si** il y a un tas d'objets *Tj* sur la même case que *ak* **alors**
(5) **si** la fourmi *ak* transporte un objet *oi* [un tas d'objets *Ti*] **alors**
(6) Déposer l'objet *oi* [le tas *Ti*] transporté par la fourmi sur le tas *Tj* suivant la probabilité *pd*(*oi, Tj*) [*pd*(*Ti, Tj*)] (équation 4.7)
(7) **sinon**
(8) /* La fourmi ne transporte pas d'objet */ Ramasser l'objet *oi* le plus dissimilaire du tas *Tj* [jusqu'à ce que la capacité *c*(*ak*) de la fourmi soit atteinte ou que le tas soit vide] selon la probabilité *pp*(*Tj*) (équation 4.6)
(9) **finsi**
(10) **finsi**
(11) **finpour**
(12) **finpour retourner** la grille *G*

Paramètre	Description
A	Nombre de fourmis
T	Nombre de déplacements de chaque fourmi
c(ai) $\forall$ i $\in$ {1, . . ., A}	Capacité de transport des fourmis
m(ai) $\forall$ i $\in$ {1, . . ., A}	Taille de la mémoire de chaque fourmi
v(ai) $\forall$ i $\in$ {1, . . ., A}	Vitesse sur la grille de chaque fourmi
p(ai) $\forall$ i $\in$ {1, . . ., A}	Patience de chaque fourmi
k1, k2	Paramètres de calcul des probabilités p_p et p_d

TAB II.1 paramètres de l'algorithme Ants

L'algorithme suivant donne le schéma général de AntClass. L'algorithme K-means est initialisé avec la partition obtenue par Ants.

Algorithme II.4 : L'algorithme ANTClass.

(1) Soit P_0 la partition initiale form´ee de N classes.
(2) **pour** $t = 1$ à T_{AntClass} **faire**
(3) Initialiser la grille G à partir de la partition P_{t-1} (un tas par classe)
(4) $G' \leftarrow$Ants(G)
(5) Construire la partition P' associée à la grille G'
(6) $P_t \leftarrow$K-means(P')
(7) **finpour**
(8) **retourner** la partition $P_{T_{\text{AntClass}}}$

V.3. Réseau de neurones artificiels :

V.3.1. Présentation :

Un réseau de neurones est défini comme étant un modèle mathématique, s'inspirant des modèles biologiques. A l'origine, cette méthode conçue essentiellement pour la reconnaissance de l'écriture, la synthèse de la parole, la vision (traitement d'image), elles ont connu leur essor à partir des années 1980. Aujourd'hui et depuis les années 1990, ces méthodes sont exploitées dans des domaines tels que le datamining [049].

Un neurone est une unité de calcul élémentaire qui combine des entrées réelles $X_1, X_2, ...X_n$ en une sortie réelle S. Les entrées n'ont pas toute la même importance et à chaque entrée X_i est associé un poids (ou coefficient synaptique) W_i. En règle générale, pour le neurone formel, l'activité en entrée est mesurée par la somme pondérée des entrées $\sum_i W_i X_i$.

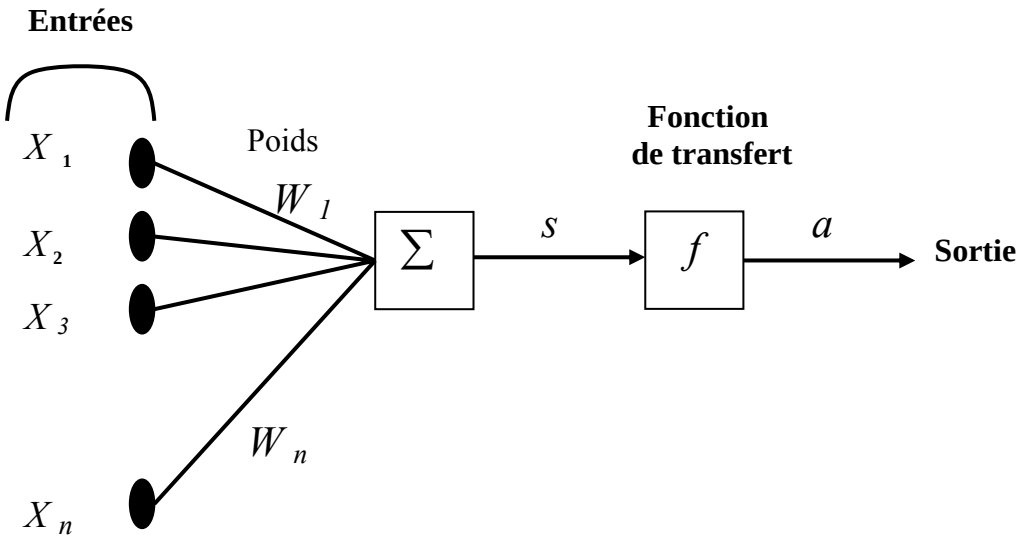

FIG II.8 Neurone artificiel.

Puis on applique une fonction de transfert *f* (appelée fonction d'activation, de sortie ou d'état) au résultat afin d'obtenir la sortie, Il existe de nombreuses formes possibles pour la fonction de transfert, elle peut être une simple fonction d'identité ou une fonction Gaussienne, linéaire, sigmoïde ou autre forme.

Un réseau de neurone se compose de multiples neurones interconnectés, les structures qui peuvent être utilisées sont très variées. La structure la plus utilisée est la structure de réseau à couches. On distingue quatre types de connexion : réseau multicouche, réseau à connexions locales, réseau à connexions récurrentes et réseau à connexions complètes.

i. Apprentissage des RNAs :

Les réseaux de neurones ont la capacité d'apprendre à partir d'exemples de manière assez analogue à celle des humains basée sur leur expérience [050], cet apprentissage peut être de différents modes :

- **Le mode supervisé :** Dans le cas de l'apprentissage supervisé, l'adaptation intervient directement lorsque le système compare la réponse qu'il a calculée avec la réponse attendue (ou désirée) en fonction des entrées fournies. La performance de l'apprentissage est déterminée par l'intermédiaire d'un critère à optimiser.

- **Le mode non-supervisé (auto-organisationnel) :** Dans l'apprentissage non supervisé, le réseau modifie ses paramètres en tenant compte seulement des informations locales. Ces méthodes n'ont pas besoin de sorties désirées préétablies.

- **Le renforcement :** Le renforcement est en fait une sorte d'apprentissage supervisé et certains auteurs le classe d'ailleurs. Dans cette approche le réseau doit apprendre la corrélation entrée/sortie via une estimation de son erreur, c'est-à-dire du rapport échec/succès. Le réseau va donc tendre à maximiser un index de performance [016] qui lui est fourni, appelé signale de renforcement. Le système étant capable ici de savoir si la réponse qu'il fournit est correcte ou non, mais il ne connaît pas la bonne réponse.

- **Le mode hybride :** Le mode hybride prend en fait les deux autres approches, puisque une partie des poids doit être déterminée par apprentissage supervisé et l'autre partie par apprentissage non-supervisé.

ii. Type des RNAs :

Il n'est pas possible d'énumérer l'ensemble des types de réseaux de neurones disponibles à ce jour. Cependant, à titre illustratif sont présentés familles parmi les plus populaires. Les chercheurs n'ont de cesse que d'inventer de nouveaux types de réseaux toujours mieux adaptés à la recherche de solutions de problèmes particuliers.

- **Le perceptron Multicouches (PMC) :**

  Le perceptron multicouches est le type le plus connu des réseaux de neurones. C'est une extension du perceptron monocouches, avec une ou plusieurs couches cachées entre l'entrée et la sortie. Chaque neurone dans une couche est connecté à tous les neurones de la couche précédente et de la couche suivante (excepté pour les couches d'entrée et de sortie) et il n'y a pas des connexions entre les cellules d'une même couche. Les fonctions d'activation utilisée sont principalement les fonctions à seuil ou sigmoïdes. Il peu résoudre des problèmes non linéairement séparables. Lorsque le perceptron multicouches est utilisé en tant que classificateur, la couche de sortie comporte un neurone par classe. En phase d'apprentissage, les valeurs désirées pour ces sorties sont alors:

  - 1 pour le neurone qui correspond à la classe de l'objet présenté à l'entrée du réseau.
  - 0 pour tous les autres neurones.

  Il suit aussi un apprentissage supervisé selon la règle de correction de l'erreur.

- **Les cartes auto-organisatrices de Kohonen** : [061][089]

  Le réseau de kohonen est à la fois une méthode de classification et de visualisation sous forme de carte hypertextuelle [091] [090]. Le réseau se répartit en deux couches : une couche d'entrée formée par les vecteurs d'entrée et une couche de sortie qui est la couche de Kohonen. Une distance euclidienne est utilisée comme fonction de transfert. Les réseaux de Kohonen décrivent en fait trois familles de réseaux de neurones :

  - *VQ (Vector Quantization) :* Introduite par Grossberg (1976), la quantification vectorielle est une méthode d'apprentissage non supervisé. Elle permet de retrouver des groupes sur un ensemble de données,
  - *SOM (Self Organizing Map) :* Les SOM sont issus des travaux de Fausett (1994) et Kohonen (1995). Ces réseaux sont très utilisés pour l'analyse de données. Ils permettent de cartographier en deux dimensions et de distinguer des groupes dans des ensembles de données.
  - *LVQ (Learning Vector Quantization) :* Les réseaux utilisant la méthode LVQ ont été proposés par Kohonen (1988). Le LVQ est une méthode adaptée à la classification de données par "recherche du plus proche voisin".

- **Les Réseaux de Hopfield :** [061]

  Ces réseaux sont des réseaux récursifs, un peu plus complexes que les perceptrons multicouches. Chaque cellule est connectée à toutes les autres et il n'y a aucune différenciation entre les neurones d'entrée et de sortie, et les changements de valeurs de cellules s'enchaînent en cascade jusqu'à aboutir à un état stable. Ces réseaux sont bien adaptés à la reconnaissance de formes. Le mode d'apprentissage utilisé est le mode non supervisé.

V.3.2. Réseau de neurone et la classification de données : [001]

La classification non supervisée peut être traitée par les réseaux de neurones artificiels (RNA). Les cellules d'entrée du réseau correspondent chacune à un attribut des objets. Les cellules de sortie donnent la classe de l'objet. La figure suivante présente un réseau pouvant accepter des objets à cinq attributs (cellules noires) et peut fournir de une à trois classes (cellules blanches).

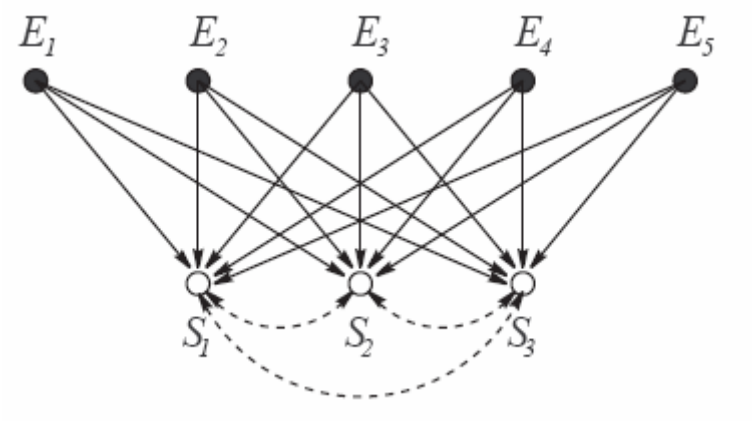

FIG II.9 Réseau de neurone artificiels : 5 cellules d'entrée et 3 cellules de sortie.

Les cartes de Kohonen [091] (*Self Organizing Maps*) sont assez proches de l'algorithme K_means. Les exemples sont présentés au réseau dans un ordre aléatoire et sont affectés à la cellule dont le représentant $\omega j = (\omega 1j, \ldots, \omega Mj)$ est le plus proche, tout comme les objets sont affectés à la classe dont le centre de gravité est le plus proche dans le cas des centres mobiles. Ensuite, les poids sont mis à jour non seulement pour la cellule *Sj* choisie mais aussi pour les cellules voisines de *Sj* . Cette structure de voisinage est le plus souvent représentée sous la forme d'une grille à deux dimensions ce qui explique l'analogie avec une carte.

Les principaux inconvénients de cette méthode sont que la convergence dépend de l'ordre dans lequel les exemples sont présentés ainsi que la difficulté du choix de la taille du voisinage.

V.3.3. Classification non supervisée par la carte de KOHONEN :

Nous avons vue comment les réseaux de neurones peuvent être utilisés pour traiter le problème de la classification de données, que se soit en modes supervisé ou non supervisé. Le modèle des réseaux neuronaux le plus connu dans le domaine de partitionnement de données (classification non supervisée) est le réseau de Kohonen, ou encore carte auto-organisée de Kohonen. Il s'agit d'un réseau non supervisé avec un apprentissage compétitif où l'on apprend non seulement à modéliser l'espace des entrées avec des prototypes, mais également à construire une carte à une ou deux dimensions permettant de structurer cet espace.

a) **Définition :**

Un réseau de Kohonen est illustré à la figure II.10. Les neurones de ce réseau sont constitués d'un vecteur de poids dans l'espace des entrées. La carte des neurones (figure II.10 b) définit quant à elle des relations de voisinage entre les neurones. Par exemple, la figure II.11 montre la forme «carrée»

de voisinage qui est la plus souvent utilisée (les neurones y sont représentés par des cercles vides). On voit sur cette figure que les neurones adjacents sont liés entre eux par des arêtes (on pourrait aussi avoir des arêtes diagonales). Ce sont les voisins immédiats spécifiés par le voisinage $vi = k$ du neurone i, c'est-à-dire l'ensemble des neurones liés au neurone i par des chemins dans le graphe de la carte contenant au plus k arêtes.

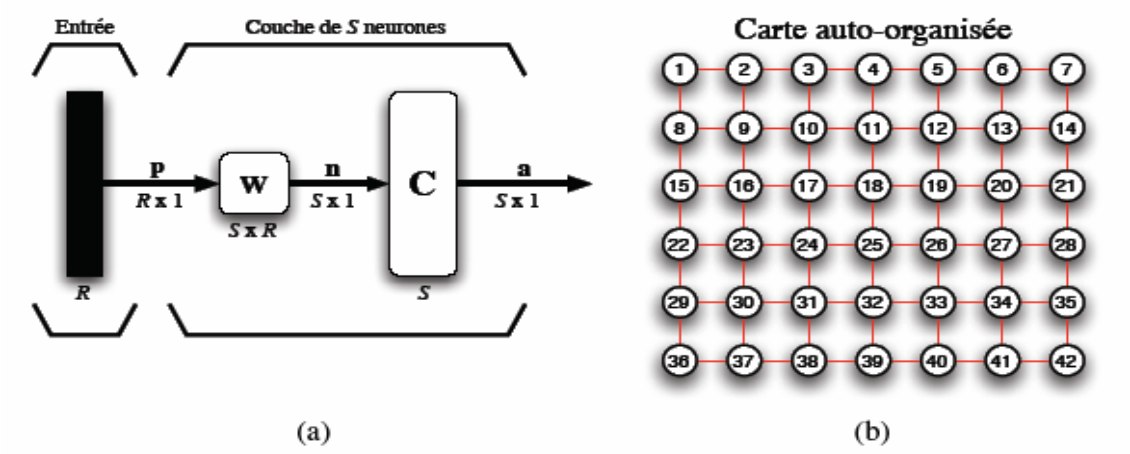

FIG II.10 Réseau de Kohonen avec carte rectangulaire

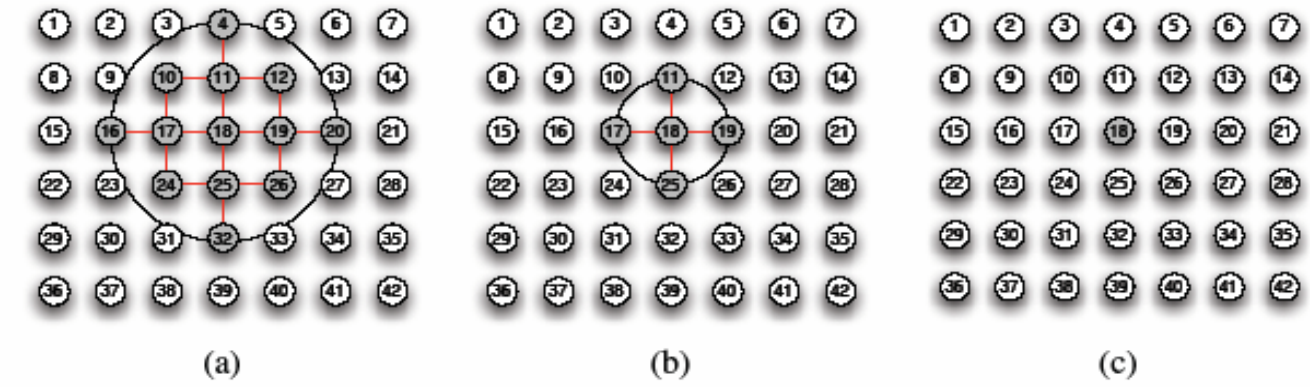

FIG II.11 Topologie de voisinage (quatre voisins) pour une carte à deux dimensions :
(a) V18 = 2 ;(b) V18 = 1 ; et (c) V18 = 0.

b) Apprentissage de la carte de Kohonen :

L'algorithme d'apprentissage du réseau de Kohonen est de type compétitif. La mise à jour des poids $\Delta_i w(t)$ du neurone i au temps t s'exprime de la façon suivante :

$$\Delta_i w(t) = \begin{cases} A(t)\left[p(t) - iw(t)\right] & \text{si } i \in vg(t) \\ 0 & \text{autrement} \end{cases} \qquad \text{(II.13)}$$

Où $p(t)$ désigne le stimulus d'apprentissage au temps t et $vg(t)$ représente le voisinage au temps t du neurone gagnant g. Ce dernier est déterminé simplement en retenant l'indice du neurone pour lequel la distance avec le stimulus p est minimum :

$$g: \quad \arg\min_i \quad \|iw - p\| \le \|jw - p\| \qquad \forall j \qquad \text{(II.14)}$$

A l'équation (II.13), il importe de remarquer que le taux d'apprentissage A et le voisinage du neurone gagnant Vg dépendent tous deux du temps. L'idée étant d'employer au départ un grand taux d'apprentissage ainsi qu'un grand voisinage, et de réduire ceux-ci au fur et à mesure que le temps (donc l'apprentissage) progresse. On utilise souvent une décroissance linéaire pour le taux d'apprentissage :

$$A(t) = \begin{cases} A0 - \left(\dfrac{A0 - A\tau}{\tau}\right)t & \text{si} \quad t < \tau \\ A\tau & \text{autrement} \end{cases} \qquad \text{(II.15)}$$

Où A_0 est le taux d'apprentissage initial, $A\tau$ est le taux d'apprentissage final, et où τ délimite la frontière entre deux phases d'apprentissage. De même, pour le voisinage, on peut aussi utiliser une décroissance linéaire :

$$V(t) = \begin{cases} v0\left(1 - \dfrac{t}{\tau}\right) & \text{si} \quad t < \tau \\ 0 & \text{autrement} \end{cases} \qquad \text{(II.16)}$$

Une variante que l'on peut aussi rencontrer consiste à remplacer l'équation II.13 par l'expression suivante :

$$\Delta_i w(t) = \alpha(t) v(j,g,t)\left[p(t) - {}_i w(t)\right] \qquad \text{(II.17)}$$

Où :

$\alpha(t)$ est le taux d'apprentissage et $v(j,g,t)$ la fonction de voisinage.

Le taux d'apprentissage est une fonction qui diminue avec le temps

$$\alpha(t) = \frac{\alpha_0}{1 + K_\alpha t} \qquad \text{(II.18)}$$

La fonction de voisinage $v(j,g,t)$ est une gaussienne évoluant avec le temps de la forme

$$v(j,g,t) = e^{-\frac{d^2(j,g)}{2\sigma^2(t)}} \qquad \text{(II.19)}$$

La distance $d(j,g)$ entre l'élu (d'indice *g*) et le neurone d'indice *j* peut être définie de plusieurs manières. Nous avons retenu la suivante [069] :

$$d(j,g) = \left|m_j - m_g\right| + \left|n_j - n_g\right| \qquad \text{(II.20)}$$

pour une carte de dimension 2 (*m* et *n*).

$\sigma(t)$ est définie par :

$$\sigma(t) = \sigma_{initial}\left(\frac{\sigma_{final}}{\sigma_{initial}}\right)^{\frac{t}{t_{\max}}} \qquad \text{(II.21)}$$

c) Algorithme d'apprentissage :

L'algorithme de base de Kohonen est résumé à la figure suivante. Il consiste à échantillonner les stimuli d'apprentissage jusqu'à l'atteinte d'un certain critère d'arrêt. Celui-ci est le plus souvent spécifié par un nombre maximum d'itération $t_{\max}$ mais peut aussi tenir compte de la stabilité des poids. Par exemple, lorsque $\max_i \left\|\Delta_i w\right\| < \varepsilon$ durant plusieurs itérations, on peut décider arbitrairement de stopper l'apprentissage puisque le réseau modélise suffisamment bien l'espace des entrées. À chaque itération, on détermine le neurone gagnant (le plus proche) et on adapte le poids de son voisinage.

Algorithme II.5 : Apprentissage de la carte de Kohonen

1. Initialiser les poids $_i w(0)$ avec de petites valeurs aléatoires.
2. Fixer A_0, A_τ, τ et v_0.
3. t = 1.
4. Répéter tant que $t < t_{\max}$:

(a) Choisir aléatoirement un stimulus p(t) parmi l'ensemble d'apprentissage.

(b) Déterminer le neurone gagnant g(p) à l'aide de l'équation II.14 :

$$g : \quad \|iw - p\| \le \|jw - p\| \qquad \forall j$$

(c) Mettre à jour les poids à l'aide de l'équation II.13 :

$$\Delta_i w(t) = \begin{cases} A(t)[p(t) - {}_i w(t)] & \text{si } i \in vg(t) \\ 0 & \text{Autrement} \end{cases}$$

Où $A(t)$ correspond au taux d'apprentissage et $vg(t)$ à un voisinage autour du neurone gagnant g.

(d) t = t + 1 ;

V.3. Algorithmes génétiques :

V.3.1. Présentation des AGs : [043][046][007]

Les premiers travaux sur les algorithmes génétiques ont commencé dans les années cinquante lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis entre 1960 et 1970, John Holland [044], sur la base des travaux précédents, développa les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation mathématique. L'ouvrage de Goldberg [045] qui décrit l'utilisation des algorithmes génétiques dans le cadre de résolution de problèmes concrets a permis de mieux faire connaître ces derniers et a marqué le début d'un nouvel intérêt pour ces techniques.

Un algorithme génétique est un ``*algorithme d'optimisation stochastique itératif*'' qui opère sur des ensembles de points *codés*, à partir d'une *population initiale*, et qui est bâti à l'aide de trois opérateurs : *croisement*, *mutation*, *sélection*. Les deux premiers sont des opérateurs d'exploration de l'espace, tandis que le dernier fait évoluer la population vers les optima d'un problème.

La fonction dont on recherche l'optimum est dite fonction objective. On ne fait aucune hypothèse sur cette fonction, en particulier elle n'a pas à être dérivable, ce qui représente un avantage sur certaines méthodes de recherche d'extremum (par exemple les méthodes basées sur le gradient).

Pour résoudre un problème quelconque par approche génétique, on devra l'exprimer sous la forme d'une fonction objective.

Un algorithme génétique manipule une population de taille constante, formée d'individus représentant chacun le codage d'une solution potentielle au problème à résoudre. Chaque individu prend la forme d'une chaîne de caractères. Dans les cas classiques, l'alphabet chromosomique est binaire (les deux allèles possibles sont 0 et 1) mais parfois les spécificités du problème à résoudre amènent à choisir un alphabet de cardinalité supérieure à2.

L'algorithme génétique dispose à l'instant t d'une population Pt appelée génération : $Pt=\{x^t_1,x^t_2,\ldots\ldots,x^t_n\}$, l'application des différents opérateurs génétiques va permettre de créer de nouveaux chromosomes (et en faire disparaître d'autres, la taille de la population étant la même).

Chaque solution x^t_i doit être évaluée pour donner son adaptation ou *fitness,* ces valeurs vont permettre de choisir les meilleurs individus de cette population, généralement la fitness sera calculée par la fonction d'évaluation (fonction sélective) en ce point.

L'application des opérateurs génétiques sur des individus jugés par une fonction sélective particulière permet d'explorer l'espace des solutions à la recherche d'un extremum. Cette recherche se fait avec un très bon équilibre entre l'exploitation des résultats déjà acquis et l'exploration de nouvelles régions de l'espace, ce qui permet d'obtenir un très faible rapport entre le nombre de points échantillonnés et le nombre de points total dans l'espace de recherche.

Algorithme II.6 : Algorithme génétique.

```
Initialisation : Générer une population initiale de N individus.
Tant que critère d'arrêt n'est pas satisfait faire
    Evaluation de chacun des individus par la fonction d'adaptation.
    Appliquer sur la population :
                ❖  Sélection
                ❖  Croisement
                ❖  Mutation
Fin tant que
```

V.4.2. Algorithmes génétiques et la classification de données :[001]

Les premiers travaux proposant un algorithme génétique pour le problème de classification sont dus à *Raghavan* et *Birchard* en 1979. Un des points clés qui détermine l'efficacité d'un algorithme génétique ainsi que sa complexité est la méthode qu'il utilise pour coder ou décoder les données sous

forme de chromosome. Nous présentons des approches génétiques de classification ainsi que les méthodes de codage associées dans ce paragraphe extraites de [013]:

i. SICM :

Cette méthode signifie « simultaneously clustering méthod » et fait l'hypothèse que 2 objets au moins sont nécessaires pour créer un cluster. Si N est le nombre d'objets à classer, et si on choisit de coder pour chaque objet l'appartenance à un cluster par un bit (mis à 1 en cas d'appartenance et 0 sinon), alors il faudrait représenter chaque chromosome avec dans le pire de cas, N (N/2) variables, cela n'est pas possibles pour des valeurs de N importantes. Par exemple, si N=10 alors il faudrait un chromosome de 50 bits et pour N=60 il faudrait utiliser 1800 bits par chromosome.

Une solution pour compresser cette information d'appartenance à un cluster consiste à coder l'indice d'un cluster en binaire par un nombre de bits dépend du nombre de cluster, puis on décrit une partition une partition en concaténant pour chacun des objets, l'indice de son cluster d'appartenance codé en binaire. Prenons par exemple, un jeu de données de 6 objets le nombre maximal de classes=3. Le codage d'une classe sera sur 2 bits.

C1= {o2, o5}, C2=Φ, C3= {o1, o3, o4, o6}.

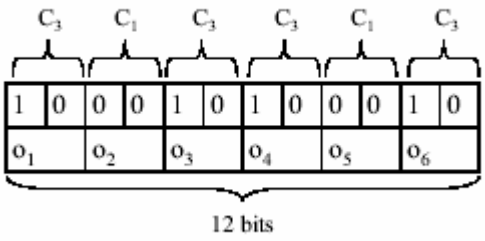

FIG II.12 Codage du chromosome obtenu avec l'approche SICM

ii. STCM :

L'algorithme STCM pour *Step Wise clustering Méthod* réalise une décomposition arborescente des objets pour en définir une partition. Chaque nœud fils de l'arbre représente un sous ensemble des données de départ et est obtenu par la scission de son père à l'aide d'un algorithme génétique. La racine de l'arbre regroupe l'ensemble des objets et ses feuilles forme la partition recherchée.

Cette approche est capable de séparer un groupe d'objet en deux sous ensembles. Pour ce faire, chaque chromosome associe les valeurs 0 et 1 à chacun de ces objets, après l'exécution de l'algorithme génétique deux groupes sont formés, l'un contenant des objets étiquetés 1 et l'autre contenant des objets étiquetés 0.

Le groupe « 1 » est lui-même scindé en deux jusqu'à ce que l'algorithme génétique ait atteint une décomposition optimale des objets. A ce moment, les mêmes opérations sont appliquées sur les autres branches de l'arbre qui n'ont pas été traités pour améliorer les résultats.

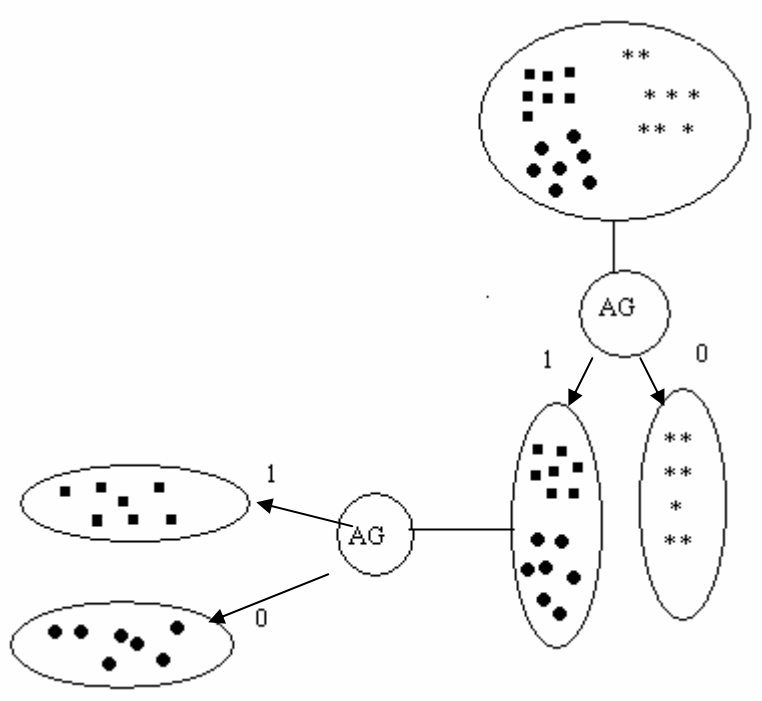

FIG II.13 Illustration du fonctionnement de l'approche STCM.

iii. SCPM :

Pour « cluster seed points méthod », est une approche hybride dans laquelle un algorithme génétique détermine la partition initiale optimale ie les grains de clusters pour un algorithme de classification de type K-means.

Chaque chromosome est codé sous la forme d'un vecteur à N composantes (N est le nombre d'objets), pouvant prendre les valeurs 0 ou 1. Lorsqu'une composante prend la valeur 1, l'objet correspondant est utilisé comme centre d'un cluster dans la partition recherchée. Les composantes initialisées à 0 ne sont pas utilisées comme centre de cluster et vont donc etre rattachées au centre dont elles sont le plus proche en terme de distance euclidienne. La fitness est fonction de l'erreur quadratique de la partition ainsi obtenue.

Un certain nombre de travaux ont aussi appliqué les algorithmes génétiques à des problèmes de partitionnement. Nous donnons ici à titre d'exemple les travaux présentés dans [085].

Chaque individu de la population représente une partition des objets. Une partition est alors représentée par une chaîne de N entiers $(e_1, e_2, ...e_N)$ où $e_i \in \{1,2,....K\} \forall i$. Si $e_i = e_j$ alors les objets o_i et *oj* se trouvent dans le même groupe. Si N = 6, (2, 1, 2, 3, 3, 2) donne la partition *{o*2*}, {o*1*, o*3*, o*6*}, {o*4*, o*5*}*. L'opérateur de croisement consiste à choisir aléatoirement un groupe d'un parent *p*1 et à le recopier dans un parent *p*2. Les objets qui appartenaient à ce groupe dans *p*2 mais pas dans *p*1 sont redistribués aléatoirement parmi les groupes de *p*2. La mutation consiste à échanger deux valeurs *ei* et *ej* dans une partition.

D'autres représentations ont été proposées (Jones et Beltrano dans [084]), par exemple en introduisant des séparateurs dans la chaîne d'entiers représentant une partition :
Par exemple si N = 6, (4, 3, 7, 1, 5, 2, 8, 6) donne la partition *{o*4*, o*3*}, {o*1*, o*5*, o*2*}, {o*6*}* car 7 et 8 représentent des séparateurs. Faulkenauer a fait remarquer que ces représentations prenaient en compte uniquement l'affectation des objets aux classes et propose un codage des solutions incluant les groupes [083]. Par exemple, les deux partitions suivantes, *{o*2*}, {o*1*, o*3*, o*6*}, {o*4*, o*5*}* et *{o*2*, o*3*, o*4*}, {o*1*, o*5*, o*6*}*, seraient codées par (2, 1, 2, 3, 3,2 : 2, 1, 3) et (1, 2, 2, 2, 1,1 : 1, 2) où la première partie (avant les deux points) représente l'affectation des objets aux groupes et où la deuxième représente les groupes présents. Le principal avantage de cette représentation apparaît lorsque l'opérateur de croisement est appliqué : seules les deuxièmes parties sont croisées, les premières parties en subissant les conséquences.

Plusieurs présentations des travaux associant les algorithmes génétiques et les problèmes de classification peuvent être trouvées dans [082] [081].

VI. travaux relatifs :

VI.1. La classification :

La classification est principalement employée dans la détection des modèles dans le trafic réseau. Portnoy et *al.* [002] utilise la classification hiérarchique pour séparer les profils normaux et anormaux du trafic de la base de données KDD'99. Cette méthode peut détecter automatiquement les intrusions connues ainsi que des nouvelles intrusions. Leur algorithme suit l'approche agglomérative typique, *c.-à-d* Chaque nouvel exemple recherché de la base de données est comparé aux clusters existants, et suivant la distance entre l'exemple et le cluster, l'exemple est ajouté au cluster le plus proche. Les auteurs ont trouvé un taux de détection approximativement de 50% et le taux de faux positif de en dessous de 1%.

Leung et leckie [068] présentent une approche de classification à base de grille pour une détection non supervisée d'anomalie. Prenant des données non étiquetées comme entrée, la technique essaye

de trouver les intrusions incluses dans les données, fondées sur les deux hypothèses de Portnoy *et al.* [002] et Javitz et Vadles [094]. Le premier a observé que la plupart des réseaux représentent un comportement normal, alors que le deuxième a montré que le trafic due à une attaque est statistiquement différent que le trafic normal. C'est cette dernière propriété qui est importante pour les techniques traditionnelles de partitionnement et de classification. Le bénéfice d'un tel système est sa capacité de détecter les attaques qui étaient inconnues auparavant. Les auteurs ont aussi réclamé que l'apprentissage non supervisé avec des données non étiquetées a amélioré l'efficacité du système de détection car il enlève le besoin d'une base de données des signatures. L'évaluation sur les données réelles d'un réseau suggère que les résultats ont été consistant avec des études similaires par Oldmeadow et *al.* [018] et Eskin et *al.* [023]. Les auteurs donnent l'exemple de détection des attaques de la bande passante DoS (Denial of service), ou il y a autant d'intrusions que du comportement normal. Les mêmes principes sont vrais pour plusieurs techniques de cette nature, et elles souffrent de l'inconvénient général de la détection des anomalies, qui est celui d'indiquer de faux positives.

Leon *et al.* [066] a abordé la détection d'intrusion par une approche basée sur Unsupervised niching clustering (UNC). Il a décrit UNC comme technique qui peut automatiquement déterminer le nombre de clusters dans un système. Niching est une autre variété de mécanisme qui est capable de maintenir plusieurs optima des techniques de recherche génétiques [014]. Leon et *al*. a combiné UNC avec La théorie des ensembles flous avec une fonction d'appartenance associée avec chacune des clusters.

Hu et Zhan [017] ont aussi considéré une approche hiérarchique, mais vont vers le développement d'une technique pour optimiser l'allocation des clusters en identifiant les clusters d'anomalies avec des degrés d'appartenance, et optimiser le système pour maximiser le taux de détection et minimiser le taux de fausses alarmes. Les auteurs valident leurs travail sur les données DARPA, et on trouvé un taux de détection au delà de 70% pour les attaques connues et 50% pour les attaques inconnues.

Récemment, Benferhat et *al* [080], a expérimenté deux algorithmes, NaïveBayes [079] et C4.5 [071] sur la base de données KDD'99 [086]. Ils concluent, d'une part, que ces deux algorithmes sont compétitifs avec l'algorithme gagnant de la compétition mondiale KDD'99 [070], et d'autre part, que NaïveBayes, utilisant la méthode du noyau pour le traitement des attributs numériques, est meilleur dans la détection de certaines catégories d'attaques faibles de la base de données KDD'99, à savoir les attaques R2L et Probes.

La classification également fait partie d'un certain nombre d'approches hybrides à la détection d'intrusion. Par exemple, Yang *et al.* [092] a proposé un hybride d'analyse de cluster et de systèmes

experts, Marin *et al.* [093] présente une technique hybride qui applique les règles expertes pour réduire de la dimensionnalité des données, suivie d'une classification initiale, Et le raffinement ultérieur des endroits des clusters en utilisant un réseau compétitif, Learning Vector Quantisation (LVQ). Soroush et *al.* [097] a utilisé les systèmes de Data Mining basés sur les colonies de fourmis, pour extraire des règles de classification.

VI.2. Réseaux neuronaux :

On peut envisager l'application des réseaux de neurones à la détection d'intrusions de plusieurs manières :

- Pour donner une modélisation statistique du comportement des utilisateurs [059].
- Pour classifier le comportement des utilisateurs [059][051] par un algorithme de type « carte de Kohonen » (classification automatique et non supervisée).
- Pour prédire le comportement des utilisateurs [052][060]. Le réseau apprend les séquences de commandes usuelles à chaque utilisateur. Il lui est alors possible, après chaque commande passée par l'utilisateur, de prédire la commande suivante sur la base de ce qu'il a appris. En cas de déviation entre la prévision et la réalité, une alarme est émise.

Fox [051] propose l'utilisation des réseaux de neurones pour prévoir le comportement des utilisateurs. Les réseaux de neurones sont ensuite utilisés dans divers travaux [052][053][054]. Ghost et ses co-auteurs [055] emploient le modèle d'Elman des réseaux de neurones pour tenir compte des séquences récurrentes dans les exécutions des programmes.

Récemment, quelques travaux proposent l'utilisation des cartes auto-organisatrices de Kohonen (SOM) pour assurer l'apprentissage automatique. Lichodzijewsk [057][056] construit des profils de connexions réseaux et utilise les multi SOM pour assurer une détection d'intrusions basée hôte. Par ailleurs Ramadas [058] applique le même concept mais en distinguant les services réseaux. Les réseaux de Kohonen évitent l'apprentissage supervisé et présentent une forte sensibilité aux données fréquentes. Ils réduisent également la complexité calculatoire. En revanche ils peuvent présenter un taux d'erreur élevé et ne garantissent pas la convergence sur des réseaux de grande dimension.

Lei et Ghorbani [099] décrivent une technique pour détecter des intrusions réseau basées sur réseaux d'apprentissage compétitifs standard « Standard Competitive Learning Network » (SCLN). C'est essentiellement un réseau monocouche, avec des neurones de sortie entièrement reliés aux neurones d'entrer. Il y a, donc, une certaine similarité avec les notions de la carte d'organisation SOM. Les performances de l'approche appliquée à l'ensemble des données de la base KDD'99, ont montré des taux de détections similaire dans les deux cas de SCLN et SOM.

Ghosh et Schwartzbar [065] ont utilisé un réseau de neurones pour apprendre le profil du comportement normal d'un système. Cela a été utilisé ensuite pour la détection des anomalies et au mauvais usage. Leur approche est basé sur un processus et capable de généraliser a partir des attaques observées auparavant aux nouvelles attaques. Une approche adaptive de neurones pour la détection des anomalies a été présenté Cannady [064] qui été capable d'apprendre les nouvelles attaques sans avoir besoin d'un apprentissage complet. Elle a aussi offert la possibilité d'améliorer par rapport aux analyses précédentes.

Jing Xin et *al.* [063], motivés par le besoin de minimiser le taux des faux positives, ont appliqué un classificateur basé sur les réseaux de neurones, dans un système modulaire. Ils suggèrent qu'un système de détection d'intrusion basé sur les réseaux de neurones doit inclure :

- Un moniteur de réseaux pour capturer les paquets d'entrée du système de détection.
- Un mécanisme d'extraction des paramètres pour transformer les données en un vecteur de paramètres pour la classification.
- Un classificateur basé sur les réseaux de neurones pour déduire, à partir du vecteur des paramètres, la présence ou l'absence d'une intrusion.
- Un mécanisme d'interface avec l'utilisateur.
- Une base de données contenant des échantillons d'intrusion pour l'apprentissage.
- Une base de données d'événements pour enregistrer les informations lors des intrusions.

Leur approche a adopté un classificateur de rétro propagation (Back Propagation BP) et, même si elle a réduit le taux des faux positives comparé aux systèmes de détection traditionnels, ils ont identifié une limitation sur la capacité de généralisation du classificateur.

Chen et *al*. [048] a remarqué qu'il est très lourd d'examiner tous les paramètres dans un ensemble de données pour détecter les signatures d'attaques. Ceci les a pris à proposer un mécanisme basé sur Flexible Neural Tree (FNT) amélioré pour identifier les paramètres importants pour construire un système de détection qui est à la fois efficaces et réalisables du côté des calcules. Un algorithme évolutionnaire a été utilisé pour développer la structure FNT, tandis que Particle Swarm Optimisation a été utilisée pour optimiser les paramètres. L'exploration de l'approche a suggéré qu'il est possible de réduire l'espace des paramètres, et que ceci mène à une amélioration de la performance.

Depern et *al.* [098] a introduit une architecture qui combine un module de détection d'anomalie et un autre de détection de signature avec un système de décision qui combine les résultats des mécanismes de détection individuels. La détection d'anomalie utilise SOM pour modéliser le comportement normal. Ce que SOM fait essentiellement est de transformer les données d'entrée de

grandes dimensions à un espace de sortie de petite dimension (souvent de dimension 2), avec des neurones de sortie différents représentant les classifications des données d'entrée. La classification est basée sur la similarité, c'est-à-dire grouper les entités qui sont proches géométriquement. Cela peut aussi être utilisé comme un moyen de visualisation des ensembles de données complexes.

Hoglund et Hatonen [025] par exemple présentent un système qui fournit la détection d'anomalies automatisée et la visualisation du comportement de l'utilisateur. Cette étude est concentrée en particulier sur l'aspect de visualisation, qui est atteint en utilisant un SOM (Self Organising Map) de dimension 2. Le module de détection de signature d'attaque de Depren et *al.* utilise l'algorithme de décision J.48 pour classifier les différents types d'attaques. Ensuite, un système de décision basé sur une règle interprète les résultats des deux modules. Evaluation de la performance basée sur l'application des données KDD'99 suggère que la combinaison est plus efficace que chacun des deux modules utilisés seul.

Eskin et *al.* [023] a présenté un algorithme pour la détection des anomalies en utilisant la comparaison des paramètres. Deux plans ont été créés, un pour la connexion réseau et l'autre pour les traces des appels. Une variété d'algorithmes ont été appliqué à la carte des paramètres dans le but d'identifier les régions denses/claires de la carte qui représentent les anomalies. Une évaluation sur les données de KDD'99 a montré que l'approche est réussie pour la détection des intrusions à partir des données non étiquetées.

Les réseaux de neurones ont été utilisés à grande échelle comme des composants dans des systèmes hybrides. Pan et *al.* [022] combine les réseaux de neurones avec l'algorithme de décision C.45. Zhang et *al.* [021] combine le prétraitement statistique et la classification des réseaux de neurones pour construire un système hiérarchique de détection d'intrusions. Taniguchi et *al.* [020] combine les réseaux de neurones feed-forward basés sur l'apprentissage supervisé, avec des modèles gaussiens et réseaux Bayesiens. Endler [019] a examiné l'analyse statistique de probabilités et les réseaux de neurones, ou le premier a été utilisé pour la détection d'anomalies, et le deuxième pour la détection de signature.

Ludovic Mé et *al.* [069] propose l'utilisation d'une carte de Kohonen pour la détection des anomalies dans un système informatique, le principe est de découper le fichier d'audit en sous chaînes de longueur constante qui sont appelées des patterns d'activité. Chaque pattern correspond à un comportement élémentaire de l'utilisateur qui l'a généré. A l'aide de la carte de Kohonen il détermine l'ensemble des comportements élémentaires types de chaque utilisateur du système. À cette fin, une base d'apprentissage, constituée par des patterns d'activité résultant de l'observation de l'utilisateur pendant une période significative de son activité, est utilisée. La carte de comportement de l'utilisateur (CCU) est obtenue en comptant le nombre de fois qu'à été élu chaque neurone de

sortie lorsque l'on présente, l'apprentissage terminé, toutes les entrées de la base d'exemples. Lorsque le réseau à été entraîné, on lui présente des patterns d'activité nouvellement acquis. Le niveau d'activation de chaque neurone donnée en fonction de sa position permet de construire la carte de détection de l'utilisateur (CDU). Si l'excitation des neurones de la CDU est faible, le pattern n'est pas connu. Il est donc anormal au sens de l'approche comportementale.

VI.3. Algorithmes génétiques :

Ludovic Mé [047] a proposé une approche assez nouvelle d'IDS de type détection par scénario. Son travail a consisté en une reformulation de la détection de signatures en un problème d'optimisation dont la résolution est assurée par un algorithme génétique.

Sinclair et *al.* [072] a suggéré que les algorithmes génétiques peuvent être utilisés pour évoluer des règles simples du trafic réseau, qui peuvent ensuite être classé dans une base de règles et utilisés pour filtrer le trafic réseau. Li [073] utilise aussi les AGs pour évoluer des règles *if…then* afin de différencier entre les connections normales et anormales, basé sur les informations temporel et spatial liées aux connexions réseau. Pillai *et al.* [074] aussi adopte un AG pour générer des règles pour des connexions spécifiques basées sur les champs TCP/IP.

Balajinath et Raghavan [075] présentent un système de détection d'intrusions (GBID) à base d'algorithmes génétiques, qui utilise les AGs pour apprendre le comportement d'un utilisateur réseau en relation avec les commandes introduites.

Gonzalez *et al*. [076] utilise une approche évolutive, connue sous Evolving Fuzzy Rules (EFR), pour générer des signatures floues pour détecter les intrusions. Cela a été testé sur plusieurs ensembles de données et il offre des avantages tels qu'une séparation claire entre les anomalies et les activités réseau normales. Abadeh et *al.* [078] propose aussi un algorithme évolutif d'apprentissage basé sur une règle floue, en employant dans ce cas Particle Swarm Optimisation (PSO), une recherche heuristique basée sur la population, pour générer un ensemble de règles qui forment alors la base d'un IDS. L'algorithme évolutif proposé est basé sur l'approche de Michigan [077].

Conclusion :

Dans ce chapitre nous avons présenté les méthodes les plus connues dans le domaine de la classification non supervisée et nous avons vu comment elles sont utilisées plus particulièrement pour la détection des intrusions réseaux. Les travaux existants sur la détection d'intrusions par la classification non supervisée des paquets TCP/IP de la base KDD'99 sont très encourageants,

Dans ce qui suit nous présenterons deux architectures basées sur deux méthodes de la classification non supervisée appliquées sur les données de la base KDD'99 afin de construire deux systèmes de détection d'intrusions comportementaux.

I Présentation de la base KDD'99 :

L'évaluation des systèmes de détection d'intrusions est une problématique d'actualité très active. Le nombre important d'intervenants impliqués dans cette discipline (laboratoires de recherche, institutions gouvernementales et non gouvernementales, universités, développeurs du monde commercial, etc.) et la sensibilité du sujet, rendent indispensable la standardisation d'une méthodologie d'évaluation. D'une part, pour rendre compte des performances des systèmes commercialisés et déployables, d'autre part, pour offrir des moyens aux chercheurs travaillant sur ce problème afin d'élaborer et d'évaluer de nouvelles approches. C'est dans cette optique que l'agence américaine DARPA a initié un programme d'une série d'évaluations annuelles dont la finalité est l'élaboration d'une méthodologie d'évaluation standard. Après avoir engagé pour cela les laboratoires Lincoln du MIT2, la première campagne d'évaluation dans ce programme, DARPA'98, eut lieu en 1998 et permit de construire un premier corpus d'évaluation. Avec des objectifs limités et un nombre modeste de participants, cette campagne a suscité un grand intérêt de la part des différentes communautés de détection d'intrusions. Une autre base de données, synthétisée de DARPA'98 et utilisée pour les besoins de la compétition mondiale KDD'99, est actuellement largement utilisée aussi bien pour le développement que l'évaluation des systèmes de détection d'intrusions, notamment où les approches sont basées sur des techniques de fouille de données.

Les données de la base KDD'99 sont orientées détection d'intrusions, elles représentent des lignes de TCP/IP dump où chaque ligne est une connexion caractérisée par 41 attributs (détaillés dans le tableau TAB III.1), tel que la durée de la connexion, le type du protocole, etc. En tenant compte des valeurs de ses attributs, chaque connexion dans KDD'99 est considérée comme étant une connexion normale ou bien une attaque. Cette base de données, disponible au site de l'UCI1 [086],

Attributs basiques
A1 durée de la connexion (nb de secondes) A2 type du protocole, ex. tcp, udp, etc. A3 service réseau (destination) ex. http, telnet A4 statut de la connexion (normal ou erreur) A5 nb de données (en octets) de la source vers la destination A6 nb de données (en octets) de la destination vers la source A7 1 si la connexion est de/vers le même hôte/port; 0 autrement A8 nb de fraguements "erronnés" A9 nb de paquets urgents
Attributs relatifs au contenu
A10 nb d'indicateurs "hot" A11 nb d'essais login ratés A12 1 si succés du login ; 0 autrement A13 nb de conditions de "compromis" A14 1 si la racine shell est obtenu; 0 autrement A15 1 si la commande on a la commande "racine su" ; 0 autrement A16 nb d'accés à la "racine" A17 nb de créations d'opérations de _chiers A18 nb de shell prompts A19 nb d'opérations sur les _chiers de contrôle d'accès A20 nb de commandes outbound dans une session ftp A21 1 si le login appartient à la liste "hot" ; 0 autrement A22 1 si le login est login "invité" ; 0 autrement
Attributs basés sur le temps utilisant des fenêtres de temps de deux secondes
A23 nb de connex. pour le même hôte A24 nb de connex. pour le même service A25 % de connex. pour le même hôte ayant l'erreur "SYN" A26 % de connex. pour le même service ayant l'erreur "SYN" A27 % de connex. pour le même hôte ayant l'erreur "REJ" A28 % de connex. pour le même service ayant l'erreur "REJ" A29 % de connex. pour le même hôte utilisant le même service A30 % de connex. pour le même hôte utilisant di_érents services A31 % de connex. pour le même service utilisant di_érents hosts
Attributs basés sur le temps utilisant des fenêtres de temps de 100 connex.
A32 nb de connex. pour le même hôte A33 nb de connex. pour le même hôte utilisant le même service A34 % de connex. pour le même hôte utilisant le même service A35 % de connex. pour le même hôte utilisant di_érents services A36 % de connex. pour le même hôte ayant le port src A37 % de connex. pour le même hôte et le même service utilisant di_érents hosts A38 % de connex. pour le même hôte ayant l'erreur "SYN" A39 % de connex. pour le même hôte et le même service ayant l'erreur "SYN" A40 % de connex. pour le même hôte ayant l'erreur "REJ" A41 % de connex. pour le même hôte et le même service ayant l'erreur "REJ"

TAB III.1 Liste des attributs des connexions de la base KDD'99.

II Description et répartition des données dans la base KDD'99.

On trouve dans KDD'99 deux types de données : données d'apprentissage et données de test. Les recommandations de la méthodologie d'évaluation de la campagne DARPA'98 stipulent que les modèles doivent obligatoirement être élaborés sur la base de données d'apprentissage. Les données de test doivent servir uniquement pour l'évaluation.

La totalité de la base d'apprentissage de KDD'99 comporte exactement 4898430 connexions. Une partie de cette base, souvent référencée comme 10% de KDD'99, est spécialement synthétisée de la totalité de la base d'apprentissage pour les algorithmes accusant un manque de ressources de calcul ou de stockage pour pouvoir utiliser la totalité de KDD'99.

Dans KDD'99, une connexion est classifiée soit comme connexion normale (avec l'étiquette *Normal*) soit comme une connexion faisant partie d'une attaque, auquel cas, elle porte le nom de cette attaque. Les connexions appartiennent à l'une des catégories suivantes : Normal, DoS, R2L, U2R et Porbe.

Le tableau suivant décrit les fichiers de la base KDD'99 :

Nom du fichier	**Contenu**	**Taille (après écompression)**
kddcup.data.gz	Totalité de la base de données d'apprentissage	708 Mo
kddcup.data_10_percent.gz	10% de la base de données d'apprentissage	71.5 Mo
corrected.gz	Données de test	45 Mo

TAB III.2 Description des fichiers de la base KDD'99

Les connexions sont regroupées dans la base KDD'99 comme suit :

	Données d'apprentissage du KDD'99				Données de test de KDD'99	
	10% de KDD'99		Totalité de KDD'99			
	Effectif	%	Effectif	%	Effectif	%
Normale	97277	19,6909%	972781	19,8590%	60593	19,4815%
Anormal	396742	80,3091%	3925649	80.1410%	250436	80,5185%
Total	494019	100%	4898430	100%	3110 29	100%

TAB V.3 Répartitions des connexions dans la base KDD'99.

III Métriques retenues pour l'évaluation :

Étant donné que pour l'ensemble des algorithmes ayant pris part à la compétition KDD'99, la détection d'intrusions est considérée du point de vue classification, les métriques retenues pour évaluer ces algorithmes sont empruntées du domaine de l'évaluation des classificateurs. Il s'agit notamment du PCC, précision, rappel et de matrice de confusion, Nous avons retenu ces métriques qui sont nécessaires à l'analyse et explication des résultats de différentes expérimentations des deux modèles sur KDD'99. Ces métriques sont définies comme suit :

- Le PCC : il représente le pourcentage global des connexions correctement classifiées.
- Le rappel : c'est une métrique qui renseigne sur la proportion de connexions d'une catégorie correctement classifiées par rapport à l'effectif réel total de cette catégorie.
- La précision : cette métrique, également relative à chaque catégorie, renseigne sur la probabilité qu'une prédiction d'une catégorie donnée soit correcte.

Pour mettre en évidence les différents types d'erreurs de classification, nous considérons la matrice de confusion de chaque évaluation.

Introduction

Comme nous avons présenté dans le chapitre II, l’algorithme ANTClass est un algorithme de classification non supervisée de données, ce qui signifie qu’on ne fournit qu'une base d'entrées et c'est le système qui doit déterminer ses sorties (les classes). Cet algorithme est une hybridation de deux algorithmes, l’algorithme des fourmis artificielles et l’algorithme des centres mobiles, le premier effectue une classification de données selon le principe inspiré du tri collectif du couvain d’une colonie de fourmis réelles, il fournit comme résultat, une partition de données qui sera soumise à l’algorithme des centres mobiles pour améliorer la disposition des objets dans les classes. Ce processus se répète jusqu’à la stabilisation des objets.

Afin d’atteindre notre objectif, qui est la classification des attaques réseaux pour la détection des intrusions, nous proposons une architecture appliquant cet algorithme sur la base KDD’99 afin de modéliser le comportement normal des utilisateurs, les objets à classifier, dans notre cas, sont les connexions TCP/IP normales de cette base, et les classes obtenues sont le profil normal des utilisateurs.

Sera présentée dans ce chapitre la conception de l’application suivie de différentes expérimentations auxquelles nous avons procédé pour évaluer ANTClass sur la base de données KDD’99.

I. L'idée de base :

L'objectif de notre étude est de construire un système de détection d'intrusion comportemental générant le minimum de fausses alertes avec un grand taux de détection, nous avons vu que cette approche de détection repose sur l'hypothèse qu'une attaque provoque une utilisation anormale des ressources ou manifeste un comportement étrange de la part de l'utilisateur, pour détecter ces anomalies le système doit être habile à distinguer entre un comportement normal est un comportement inhabituel, pour cela avant de passer à l'étape de la détection le système établit d'abord le profil normal des utilisateurs au quel il compare les nouvelles utilisations du système, c'est la phase de la classification non supervisée des connexions TCP/IP normales de la base KDD'99. Le profil construit est un ensemble de classes de telle sorte que les objets d'une même classe soient plus similaires entre eux qu'avec les objets des autres classes. L'algorithme ANTClass proposé par N.Monmarché dans sa thèse [001] offre une solution idéale au problème de la classification non supervisée de données, cet algorithme sera la base de notre système.

L'idée principale de la détection des déviations du comportement normal est : si une connexion appartient à une classe C_i ayant un centre de gravité G_i elle doit être moins éloignée de G_i que l'objet le plus éloigné du centre de la même classe C_i, d'une autre manière :

Soit $C_1, C_2, ...C_K$ l'ensemble des classes du profil normal construites par le système, et $G_1, G_2, ...G_K$ les centres de gravité de ces classes.

On détermine pour chaque classe C_i une valeur seuil S_i qui est la distance euclidienne entre le centre de gravité G_i de la classe et l'objet le plus éloigné de ce centre et qui appartient à la même classe C_i.

Soit une connexion P qui se distance de $D_1, D_2, ...D_K$ des centres de gravité $G_1, G_2, ...G_K$. Cette connexion est une attaque si : $\forall i \in \{1,2,...k\} \quad Di > Si$ sinon s'il existe au moins une classes C_i tel que $Di \leq Si$ alors la connexion appartient à la classe C_i.

Après avoir construire le profil normal, le système recueille de nouvelles connexions TCP/IP qui doit les classer comme des attaques ou des connexions normales selon ce principe.

II. Conception du système :

Pour former le profil normal, on doit constituer une base contenant uniquement les connexions étiquetées normales, pour cela on traite 10% de la base KDD'99 qui représente les données

d'apprentissage. Dans cette base on trouve 494019 connexions, 97277 connexions sont normales et les autres sont des attaques.

La base d'apprentissage est soumise à un classificateur ANTClass pour former le profil normal, le résultat obtenu est un ensemble de classes $C1, C2, ...Ck$, tel que chacune attire les objets (les connexions) les plus proche de son centre de gravité.

Pour évaluer les résultats obtenus, nous utilisons une base de test qui comporte 311029 connexions, où 250436 connexions sont considérées comme des attaques. Le test se fait en comparant à chaque fois une connexion de la base de test avec les classes $C1, C2, ...Ck$ du profil normal obtenu, par le calcul de la distance euclidienne entre cette connexion et les centres de gravité $(G_1, G_2, ...G_K)$ de l'ensemble des classes du profil normal, ensuite la comparer à un seuil. Ce dernier représente la distance euclidienne entre le centre de gravité d'une classe et l'objet le plus éloigné de son centre de gravité.

L'implémentation de cette application passe par trois phases : la phase d'apprentissage, la phase de test et la phase de validation, ces trois étapes sont étudiées dans cette section.

II.1. Phase d'apprentissage :

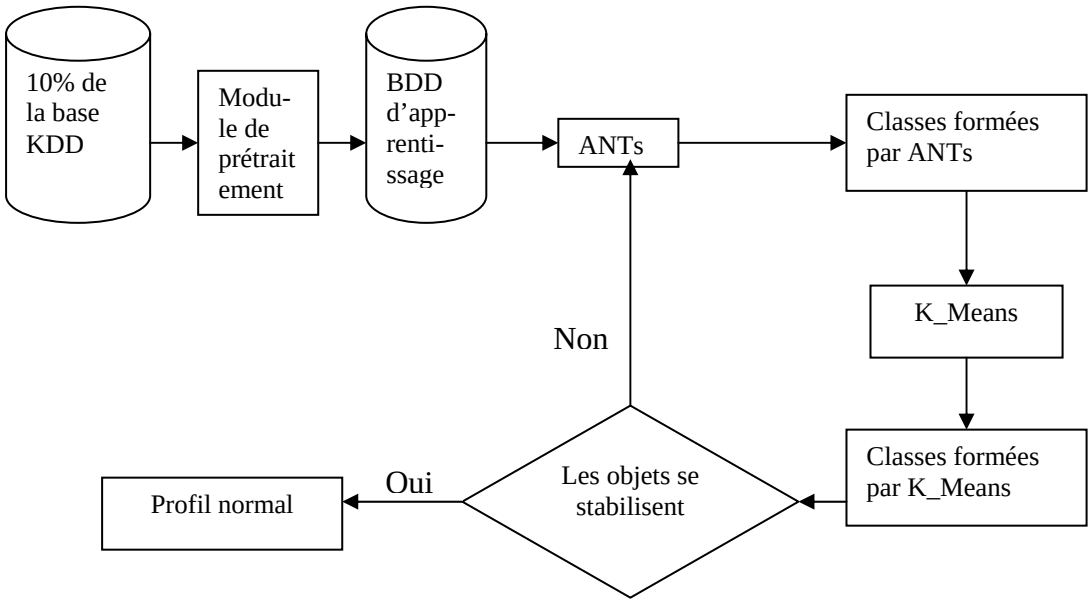

FIG IV.1 Schéma fonctionnel de la phase d'apprentissage.

II.1.1. Module de prétraitement des données :

L'architecture décrite par la figure IV.1 s'applique sur une base appelée base d'apprentissage, cette dernière est obtenue par le module de prétraitement des données qui doit :

- Lire les connexions à partir de la base 10% de la base KDD'99, cette dernière comporte des connexions normales et des attaques, chacune est caractérisée par 41attributs, parmi lesquels trois prennent des valeurs qualitatives (type de protocoles, statut de la connexion, service réseaux) et les autres ont des valeurs numériques.
- Extraire les connexions étiquetées normales : pour former le profil normal des utilisateurs le système doit donner des classes des connexions normales, pour ce faire le module de prétraitement extrait ces connexions de la base 10% de la base KDD'99 et les met dans une autre.
- Prendre seulement les attributs quantitatifs, car l'algorithme appliqué est basé sur des formules de calcule de la probabilité de ramassage et de dépôt, la distance entre objets, etc. ce module extrait seulement les attributs quantitatifs et élimine les autres.
- Eliminer les connexions redondantes : la distance entre deux objets est nulle signifie que la similarité=1, donc on a des connexions ayants les mêmes valeurs pour les mêmes attributs, dans ce cas ce module supprime l'une des connexions pour minimiser la taille de la base d'apprentissage et gagner du temps d'exécution.
- créer une autre base contenant que des connexions normales décrites seulement par des attributs quantitatifs et qui ne comprend pas de connexions redondantes.

II.1.2. L'algorithme ANTs :

La base obtenue par le module de prétraitement est ensuite obéissante au classificateur ANTs schématisé par l'organigramme présenté par la figure FIG IV.2.

L'ensemble des connexions à partitionnées $O = \{o_1, o_2, ...o_N\}$, avec N est le nombre de connexions normales de la base d'apprentissage, correspond chacun à un point d'un espace métrique de dimension *M*=38, donc cet ensemble devient un ensemble de vecteur X tel que : $X_i = (x_{i,1}, x_{i,2}, ...x_{i,38})$ avec $i \in \{1, N\}$

$$X = \begin{pmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_N \end{pmatrix} = \begin{array}{c} \begin{matrix} Att_1 & Att_2 & & & Att_{38} \end{matrix} \\ \begin{pmatrix} x_{1,1} & x_{1,2} & . & . & . & x_{1,38} \\ x_{2,1} & x_{2,2} & . & . & . & x_{2,38} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_{N,1} & x_{N,2} & . & . & . & x_{N,38} \end{pmatrix} \end{array}$$

Fig IV.2 Matrice représentative des objets

La technique la plus simple pour le positionnement des objets sur un plan à deux dimensions est la stratégie aléatoire, elle génère deux nombres aléatoires pour chaque objet qui vont représenter ses cordonnées.

La taille L de la grille dépend du nombre d'objets à partitionner, elle est déterminée par : $L = \left| \sqrt{2N} \right|$, dans notre cas la base d'apprentissage comporte 97277 connexions normales, donc $L = \left| \sqrt{2 * 97277} \right| = 441$

Les fourmis sont de même placées d'une manière aléatoire sur la grille, de telle sorte qu'une case ne comporte qu'une et une seule fourmi, le nombre de fourmis sera fixé initialement à une valeur aléatoire, puis soit on ajoute ou on exclue quelques fourmis, selon les expérimentations qui seront faites.

Les fourmis se déplacent d'une case à une autre avec une vitesse Vai , qui peut être variable d'une fourmi à une autre.

Si une fourmi a_i se trouve sur une case contenant un objet ou un tas d'objets T_j et qu'elle transporte un objet o_i (ou un tas d'objets), on calcule la probabilité de déposer $P_d(o_i, T_j)$ selon l'équation (II.12), si la $P_d(o_i, T_j)$ =1 elle dépose l'objet transporté sinon elle se déplace vers une autre case.

Si une fourmi a_i se trouve sur une case contenant un objet et qu'elle ne transporte aucun autre, elle le ramasse.

Si une fourmi a_i se trouve sur une case contenant deux objets qui se sont très similaire elle ne ramasse aucun, sinon elle prend l'un des deux.

Si une fourmi se trouve sur une case contenant un tas d'objets, on calcule la probabilité de ramasser $P_p(T_j)$ selon l'équation (II.11), si la $P_p(T_j)=1$ elle ramasse les objets les plus éloignés du centre de gravité du tas jusqu'à l'atteinte de sa capacité ou le tas soit vide.

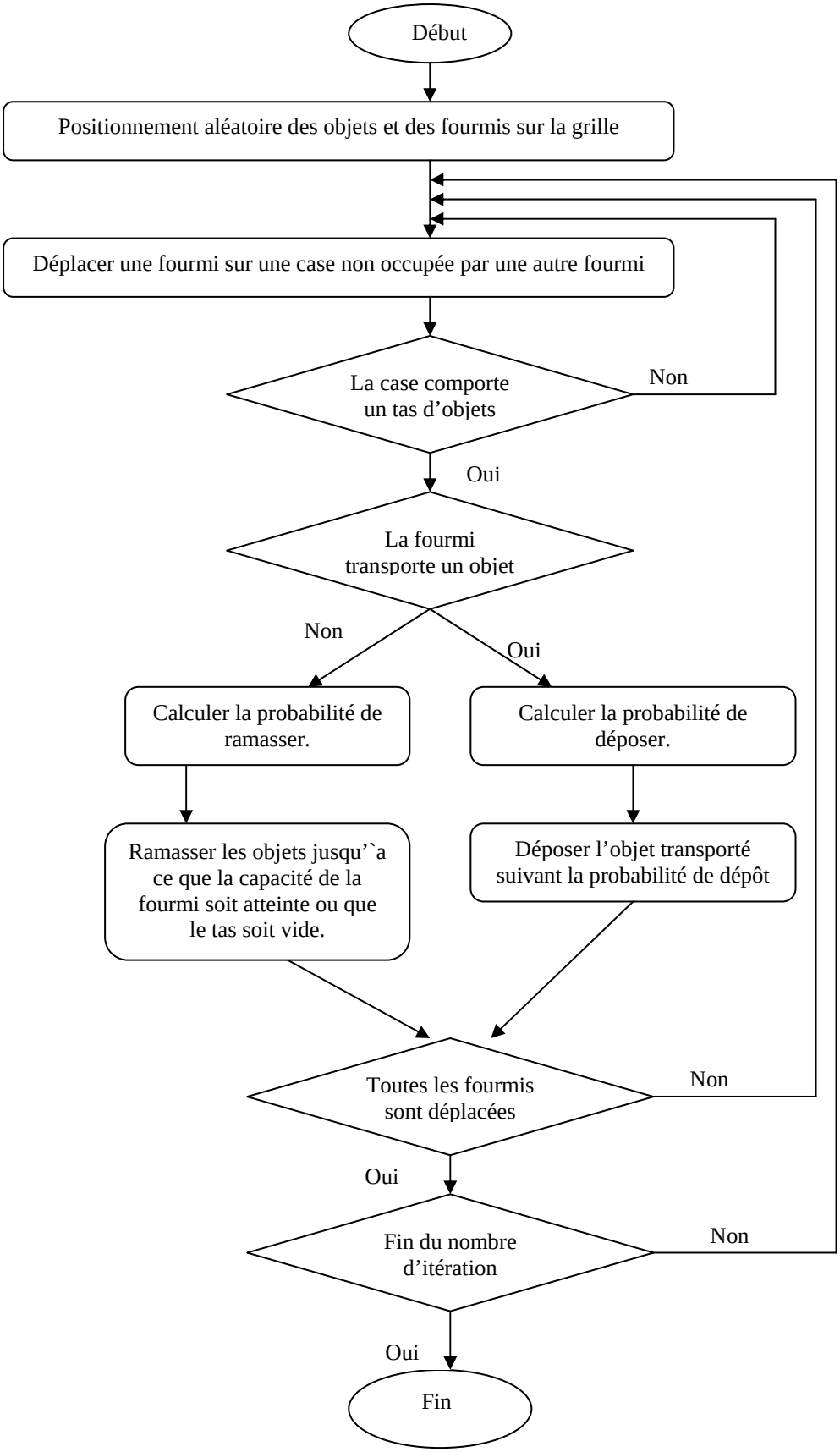

FIG IV.3 Modélisation algorithmique de ANTs.

Après l'exécution de l'algorithme ANTs on obtient une partition des connexions normales formée de *n* classes, qui n'est pas obligatoirement optimale, pour corriger les défaillances de cet algorithme, on remis les *n* classes à un autre classificateur basé sur le principe des centres mobiles appelé K_Means.

II.1.3. L'algorithme K_Means :

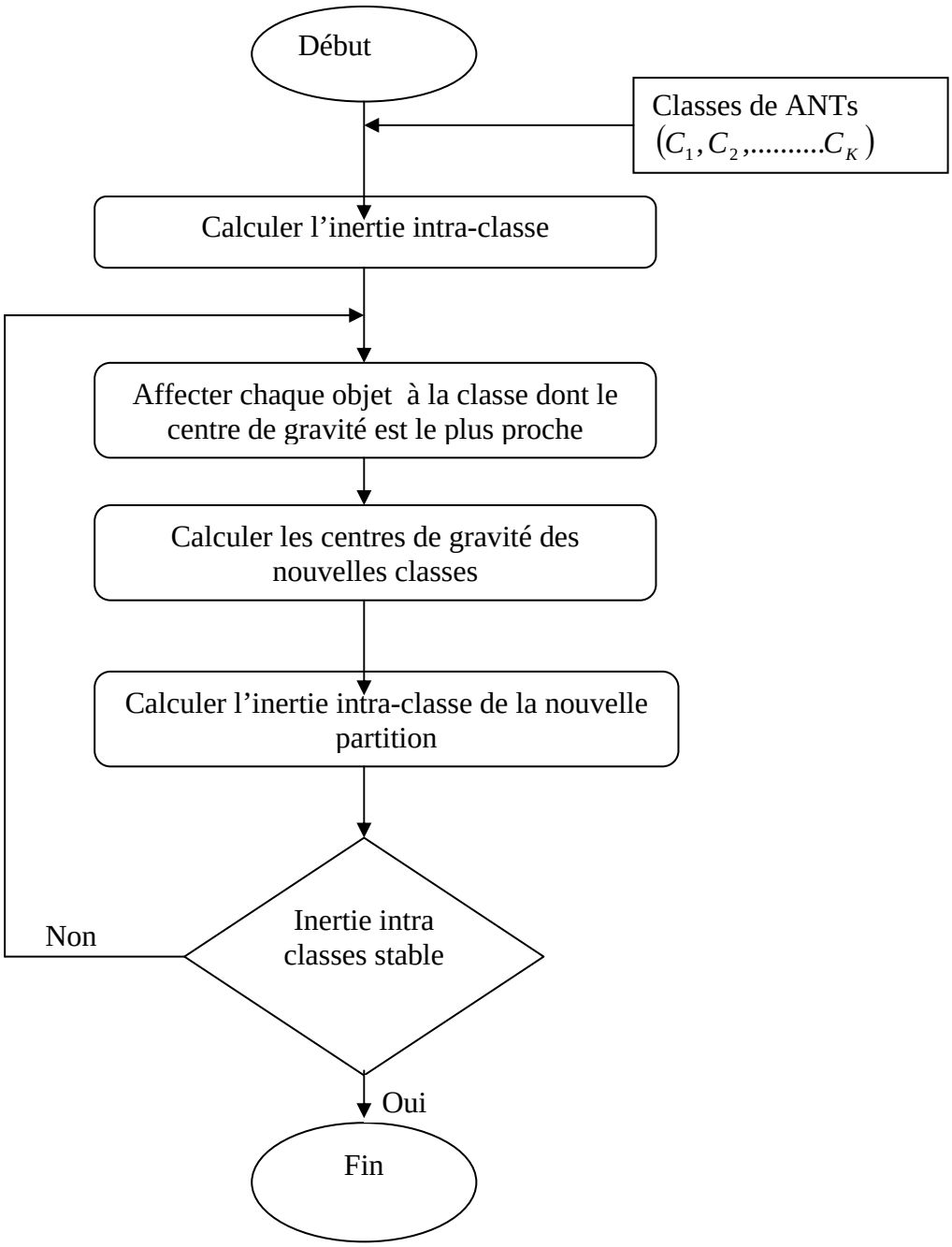

Fig IV.4 Modélisation algorithmique de K_Means.

Pour calculer l'inertie intra-classe on donne quelques notions nécessaires :

- La distance euclidienne entre deux objets : Une distance étant définie, elle est assimilable à une mesure de dissimilarité. Etant donné que nous nous intéressons à un espace métrique, l'indice de dissimilarité le plus communément utilisé est la métrique de Minkowski.

Soient les deux objets o_i et o_j de l'ensemble O, la distance euclidienne entre ces deux objets est donnée par :

$$d_2(o_i, o_j) = d\ (X_i, X_j) = \left(\sum_{k=1}^{M} \omega_k \left| X_{ik} - X_{jk} \right|^2 \right)^{\frac{1}{2}} \qquad \text{(IV.1)}$$

ω_k est un facteur de pondération que nous considèrerons égal à 1 par la suite.

- Centre de gravité d'une classe : Le centre de gravité G_i de la classe C_i est donné par :

$$G_i = \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} X_j^{(i)} \qquad \text{(IV.2)}$$

Où $X_j^{(i)}$ est le *j*-ième point de la classe *Ci*

- L'erreur quadratique sur une classe : l'erreur quadratique ε_i^2 sur la classe C_i est donnée par :

$$\varepsilon_i^2 = \sum_{j=1}^{|C_i|} d^2(X_j^{(i)}, G_i) \qquad \text{(IV.3)}$$

- L'inertie intra-classe de la partition : est donnée par

$$\tau_w = E_q^2 = \sum_{j=1}^{k} \varepsilon_j^2 \qquad \text{(IV.4)}$$

L'inertie intra-classe est un critère de mesure de la stabilité des objets dans une partition, elle est calculée par la somme de toutes les erreurs quadratiques sur toutes les classes, à une itération donnée de K_Means et lorsque tous les objets sont arrangés dans les classes les plus appropriées, les distances entre les objets et le centre de gravité de la classe dont ils appartiennent se stabilisent, ce qui provoque la stabilisation des erreurs quadratiques.

Après l'exécution de cet algorithme on obtient une partition plus uniforme, le nombre de classes peut être réduit car il peut exister des classes qui n'attirent aucun objet et seront vides, donc elles disparaissent. Cette partition peut être améliorée jusqu'à ce que les objets ne changent plus de

classes et ceci par l'appellation de ANTs puis K_Means et ainsi de suite, la partition obtenue par ANTClass représente le profil normal des utilisateurs.

II.2. La phase de test :

En vue de tester la cohérence de l'application avec les données d'apprentissage, on va faire un test sur l'application après implémentation et obtention du profil normal. Pour ce faire on doit entrer au système les même connexions qu'il a appris, puis on calcule le taux de détection qui désigne le nombre de connexions correctement classifiées par rapport au nombre total de connexions. La figure FIG IV.5 résume le processus de test.

Si le nombre de connexions normales trouvé par le système est proche de la totalité des connexions de la base d'apprentissage on conclu que le système est cohérent avec les données qu'il a appris.

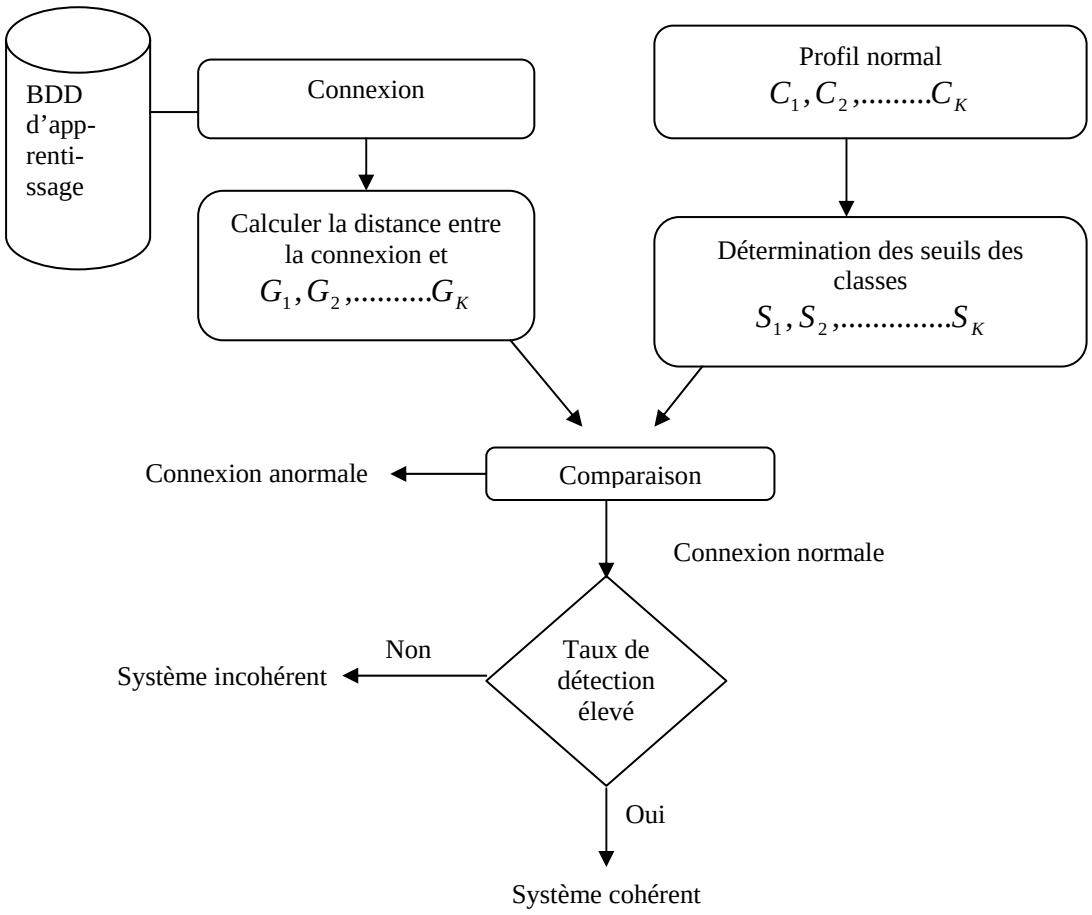

Fig IV.5 Schéma fonctionnel de la phase de test.

II.3. La phase de validation :

La phase de validation consiste à évaluer les performances du système proposé à l'aide d'une base de test contenant des connexions normales et des attaques non étiquetées. Nous rappelons que la détection d'anomalies se fait par la comparaison des connexions de test aux centres de gravité $G_1, G_2, ...G_K$ de différentes classes $C_1, C_2, ...C_K$ construites lors de la phase d'apprentissage, en calculant la distance euclidienne entre cette connexion et les centres de gravité $G_1, G_2, ...G_K$ des classes $C_1, C_2, ...C_K$, ensuite la comparer aux seuils $S_1, S_2, ...S_K$.

Selon cette comparaison le système classifie cette connexion comme une attaque ou qu'elle appartient à l'une des classes du profil normal.

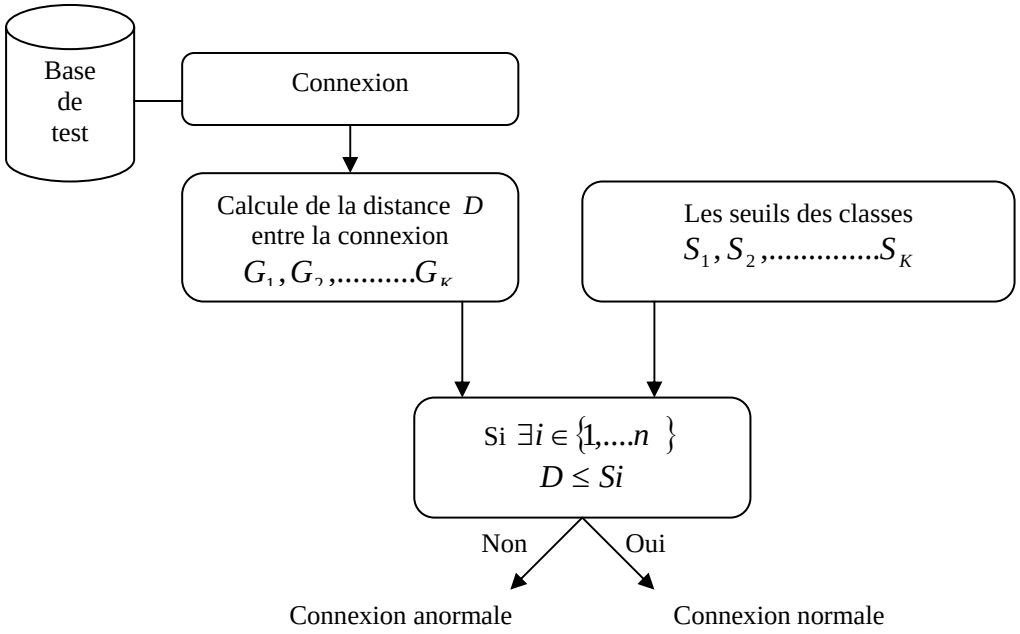

FIG IV.6 Schéma fonctionnel de la phase de validation.

III Environnement de développement :

Le modèle proposé à été développé entièrement avec JBuilder X version entreprise, qui est un langage de programmation orienté objet développé par Sun MicroSystems. Il est fortement inspiré des langages C et C++. Il est doté, en standard, de bibliothèques de classes très riches comprenant la gestion des interfaces graphiques (fenêtres, boites de dialogue, contrôles, menus, graphisme), la programmation multi-threads (multitâches), la gestion des exceptions, les accès aux fichiers et au

réseau…L'utilisation de ces bibliothèques facilitent grandement la tâche du programmeur lors de la construction d'applications complexes. Cette propriété de ce langage nous a poussé de le choisir pour l'implémentation de notre système. La plateforme utilisée est Windows XP professionnel.

La machine sur laquelle l'architecture proposée a été développée et testée est un Pentium4 de 480 Mo de mémoire RAM et 3.00 GHz de microprocesseur.

IV. Implémentation :

IV.1 Prétraitement des données :

Pour appliquer ANTClass sur la base KDD'99, on doit utiliser deux types de données, les données d'apprentissage et les données de test.

- Les premières sont stockées dans une base que nous appelons *KDD_normale*, car elle ne comporte que des connexions normales. Cette base est obtenue par un prétraitement effectué sur les données de la base 10% de la base KDD'99 par le module de filtrage. Après le filtrage des connexions nous obtenons 97277 connexions normales et 396742 attaques. Elle est ensuite traitée de façons à éliminer les connexions normales redondantes pour obtenir une base de 86874 connexions normales, elle est enregistrée dans un fichier texte appelé *kddnormal.txt* tel que chaque ligne représente une connexion. Cependant, les données de test sont enregistrées dans uns autre fichier que nous appelons *attaques.txt* sans élimination des attaques redondantes.
- Une autre base est aussi nécessaire pour la validation des résultats, c'est la base de test, elle est de même enregistrée dans un fichier texte que nous appelons *test_data.txt* après élimination des attributs qualitatifs.

```
/* creation du fichier KDDnormal contennant que des connexions normales */

 static void Filtrage () throws IOException
 { String nomfichoriginal;
   String x,y="normal.";
   FileWriter f = new FileWriter("kddnormal.txt");
   PrintWriter sortie = new PrintWriter(f);
  BufferedReader entree= new BufferedReader (new FileReader ("10%_KKD.txt"));
  while (true)
  {  String lignelue=entree.readLine();
     String ligne="";
     if (lignelue ==null) break;
     StringTokenizer tok= new StringTokenizer (lignelue, ",");
     for (int i=0; i<=41; i++)
     {  x=tok.nextToken(); //la lecture d'une donnée
        if (i!=1 & i!=2 & i!=3  &i!=41){
        ligne=ligne+x+",";}
        if (x.equals(y)==true && (i==41))
            sortie.println(ligne);
     }
     sortie.println(ligne);
   }
  entree.close();
  sortie.close();
}
```

FIG IV.7 Module de prétraitement des données.

IV.2. Implémentation de l'algorithme ANTs :

IV.2.1. implémentation de la grille de ANTs :

Comme nous avons présenté, l'algorithme ANTs est un algorithme de classification non supervisée basé sur le principe des fourmis artificielle qui se déplacent sur une grille à deux dimensions, une case de cette grille peut comporter une fourmi, un objet, un tas d'objets ou encore une fourmi et un tas d'objets, comme elle peut être vide, la cardinalité d'un tas varie d'une case à une autre et se modifie à chaque itération de ANTs, pour ces raisons la structure la plus adéquate pour implémenter une case de la grille est la liste chaînée, tel que :

- Les objets sont représentés par des entiers positifs ou nul.
- Les fourmis sont représentées par des entiers négatifs.

Donc la grille est une matrice carrée de listes, sa taille est déterminée par le nombre de connexions à partitionner.

La base d'apprentissage comporte 86874 connexions donc la taille de la grille est $L \times L$ avec $L = \left| \sqrt{2N} \right| = 417$

IV.2.2. Implémentation d'une fourmi :

Sur la grille de ANTs, les agents fourmis se déplacent, ramassent et déposent des objets chacune indépendamment des autres, donc chaque fourmi peut être implémentée par un processus autonome qui s'exécute en parallèle avec les autres processus implémentant les autres fourmis, le langage choisi pour l'implémentation *JBuilder X* est un langage bien adapté à cet objectif, la notion de la programmation multi Thread offerte par ce langage nous permet d'implémenter les agents fourmis soit par héritage de la classe *Thread* de sa bibliothèque soit en implémentant la classes *Runnable*, nous appelons la classe représentant un agent fourmi par *Fourmi*, Nous définissons ses méthodes *se_déplacer(), rammaser()* et *déposer()* qui permettent aux fourmis d'effectuer leurs tâches de la classification, ces méthodes sont définies et appelées par la même classe *Fourmi.*

Le segment de programme permettant de définir ces processus est donné par la figure suivante (les méthodes *ramasser ()* et *déposer ()* voir annexe D).

```
/* implémentation d'une fourmi*/

class Fourmi implements Runnable

{   static int ligne,colonne,vitesse,capacité,num;
    String nomProcessus;
    static double dist_moyenne ;
    static double min,Pd,Pp;
    Thread th;
    static Liste tas;
    static float gravité_classe[]=new float[38];
    static float X[]=new float [38];

    /** constructeur permettant de nommer le processus */
    public Fourmi(String s,int numéro,int v, int c, int lig, int col,Liste L)
     {        nomProcessus = s;
              num=numéro;
              vitesse=v; capacité=c; ligne=lig;colonne=col;  tas=L;        }

    public void start() {
       if (th == null) {
          th = new Thread(this,nomProcessus);
          th.start();
       }    }

    public void stop() {
       if (th != null) {
          th.stop();
          th = null;   }    }

public void run()
 {  for (int i=1;i<nombre_itération ;i++) //nombre de déplacement de  chaque fourmi

    { try {
       Fourmi.sedéplace();
       if (Principal.occupéparobjet(ligne, colonne) == true) //si la case comporte un objet
{  gravité_classe=Principal.centre_de_gravité(Principal.Grille[ligne][colonne].suivant);
     if (tas != null)                                    //la fourmi transporte au moin un objet
        Fourmi.déposer();
      else                                               // La fourmi ne transporte aucun objet
      Fourmi.ramasser();
     }    }
catch (IOException err)
{System.out.println("** erreur **");}
} }}
```

FIG IV.8 Implémentation d'une fourmi.

Après avoir définir les fourmis, on doit lancer leurs exécutions parallèles par appelle de la méthode *start()* comme suit :

```
static void Ants()
{  int nombre_fourmis=0;
     for (int s=0;s<DIM1;s++)
       for (int ss=0;ss<DIM2;ss++)
       {if (Grille[s][ss]!=null)
         if (Grille[s][ss].contenu<0)
          nombrefourmis++;
       }
 Fourmis=new int [nombrefourmis][3];
  nombre_fourmis=0;
    for (int s=0;s<DIM1;s++)
       for (int ss=0;ss<DIM2;ss++)
       {if (Grille[s][ss]!=null)
          if (Grille[s][ss].contenu<0)
           { Fourmis[nombre_fourmis][0]=Grille[s][ss].contenu;// num de la fourmi
            Fourmis[nombre_fourmis][1]=s; //la ligne ou se trouve la fourmi
             Fourmis[nombre_fourmis][2]=ss; // la colonne ou se trouve la fourmi
            nombre_fourmis++;}
       }
System.out.println("le nombre de fourmi est :"+ nombrefourmis);

     for ( i=0;i<nombrefourmis;i++)
 { System.out.println("donner la vitesse de la fourmi:");
  int v=Clavier.lireInt();
  new Fourmi("F",Fourmis[i][0],1,100,Fourmis[i][1],Fourmis[i][2],tas_transporté).start();
 }
}
```

FIG IV.9 Implémentation de l'algorithme ANTs

IV.3. Implémentation de l'algorithme K_Means :

L'algorithme K_Means est une simple méthode de la classification non supervisée, se repose sur une boucle *Tant que* conditionnée par la stabilisation de l'inertie intra-classe, sa partition initiale est celle fournie par l'algorithme ANTs, à chaque itération on calcule l'inertie intra-classe en appellent la méthode *iniertie_intraclasse(),* on affecte chaque connexion à la classe dont son centre de gravité est le plus proche, puis on retrouve les centres de gravité de la nouvelle partition et on recalcule l'inertie intra-classe, la figure suivante donne le listing de la méthode implémentant K_Means :

```
static void K_means() throws IOException
{     double a;
      float objet[] = new float[38];
      double inertie;
      int stable;
       do
       { inertie = Principal.inertie_intraclasse();
         stable = 0;
        Principal.les_centres_de_gravité();
        for (int i = 0; i < N; i++)
         { objet = Principal.tab[i];
           int num_classe1 = Principal.numéro_classe(i);
          double min = Principal.distance_entre_2_objets(objet, Principal.G[num_classe1]);
          int num_classe2 = num_classe1;
          for (int j = 0; j < K; j++)
         { a = Principal.distance_entre_2_objets(objet, Principal.G[j]);
            if (min > a) {
              min = a;
              num_classe2 = j;  }
           }
          if (num_classe1 != num_classe2) {
            Principal.suppression_objet_de_tas(i);
            Principal.affectation(i, indice[num_classe2][0],indice[num_classe2][1]);
           }
           else stable++;
         }
         System.out.println("le nombre de classes est " + Principal.K);
         System.out.println("le nombre d'objets stables est: " + stable);
        }
      while (inertie!=inertie_intraclasse() );
}
```

FIG IV.10 Implémentation de l'algorithme K_Means

V. Résultats et test de résultats :

Nous rapportons dans ce qui suit les résultats des expérimentations de ANTClass sur différentes bases, la première comporte 500 connexions, la deuxième comporte 2000 connexions, la troisième comporte 40000 et la dernière base est la base 10% de KDD'99, dans toute ces expérimentations les paramètres K1 et K2 sont fixés à1. Ici, l'IDS à base de ANTClass est construit et testé sur les mêmes données en vue d'évaluer à quel point le système construit est-il cohérent avec les données d'apprentissage. La figure suivante montre l'interface graphique du système qui permet de l'initialiser et d'effectuer ces différentes expérimentations.

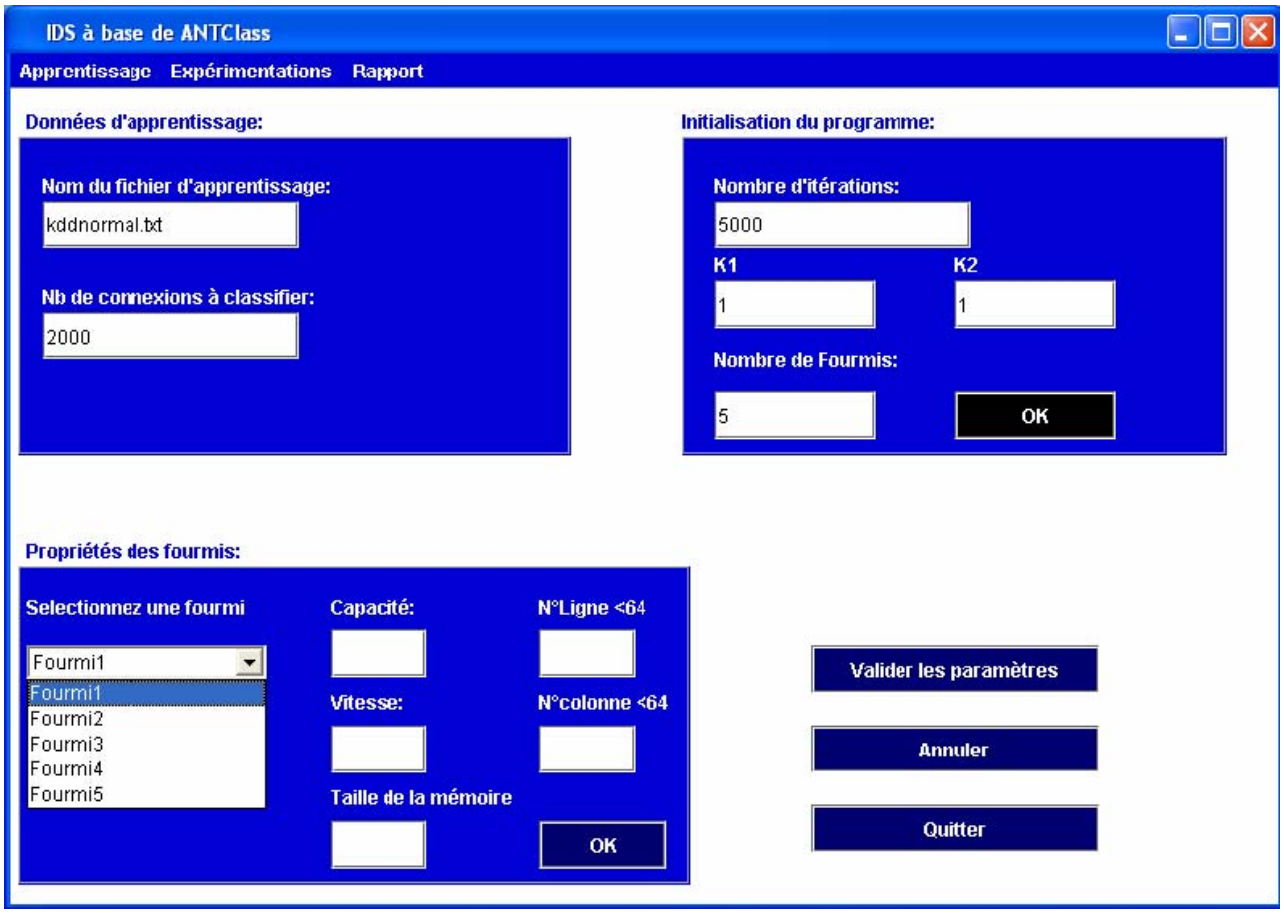

FIG IV.11 Interface graphique du système.

a) Expérimentation du modèle sur un échantillon de 500 connexions :

Pour partitionner 500 connexions la grille doit être de taille 32×32, le nombre de fourmis a été fixé à 5 fourmis chacune se déplace 10000 fois à chaque itération de ANTCLass. Le nombre de classes obtenues est 103 classes.

Les résultats de test des données d'apprentissage sont résumés dans le tableau suivant :

Nom du fichier	Effectif	Nombre de connexions classées normales	nombre de connexions classées attaques
Echantillons d'apprentissage	500	500	00
KDD_normale	97277	30566	66711
KDD_attaques	396742	841	395901

TabIV.1. Résultats du test des données d'apprentissage (base d'apprentissage de 500connexions).

La matrice de confusion est comme suit :

→	**Normale**	**Attaque**
Normale	31.42%	68.58%
Attaques	0.21%	99,78%
	PCC=86,32%	

TAB I.V.2 Matrice de confusion de ANTClass sur les données d'apprentissage (Base d'apprentissage de 500connexions)

b) Expérimentation du modèle sur un échantillon de 2000 connexions :

Pour partitionner 2000 connexions la grille doit être de taille 64×64, le nombre de fourmis a été fixé à 10 fourmis chacune se déplace 20000 fois à chaque itération de ANTCLass.

Le nombre de classes obtenues est 336 classes après 5 itérations de ANTClass, le tableau suivant montre l'évolution du nombre de classes en fonction des itérations de ANTs et de K_Means.

itération	1	2	3	4	5	6	7	8	9	10
Nombre de classes	893	536	421	389	359	348	340	338	338	336

TAB IV.3. Evolution du nombre de classes de ANTClass.

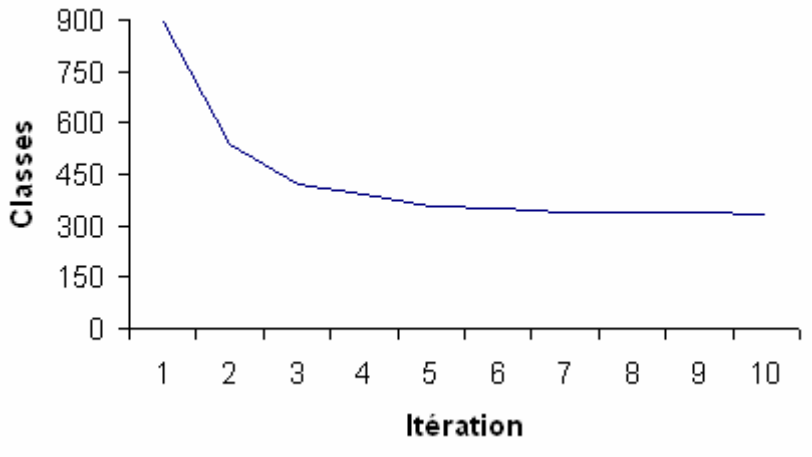

FIG IV.12 Evolution du nombre de classes en fonction des itérations de ANTCLass.

Nous remarquons bien, d'après le tableau, que chacun des deux algorithmes Ants et KMeans améliore la partition générée par l'autre tour à tour jusqu'à la stabilisation de tous les objets. De plus, la courbe de variation de nombre de classes montre bien que l'algorithme ANTClass converge vers la partition finale dés les premières itérations.

Les résultats de test des données d'apprentissage sont résumés dans le tableau suivant :

Nom du fichier	Effectif	Nombre de connexions classées normales	nombre de connexions classées attaques
Echantillons d'apprentissage	2000	2000	00
KDD_normale	97277	48060	49217
KDD_attaques	396742	1140	395602

TAB IV.4. Résultats du test des données d'apprentissage
(Base d'apprentissage de 2000connexions).

La matrice de confusion est comme suit :

→	**Normale**	**Attaque**
Normale	49,40%	50,59%
Attaques	0.28%	99,71%
	PCC=89,80%	

TAB IV.5 Matrice de confusion de ANTClass sur les données d'apprentissage (Base d'apprentissage de 2000connexions)

c) Expérimentation du modèle sur un échantillon de 40000 connexions :

Pour expérimenter le système sur 40000connexions nous avons fixé ses paramètres comme suit :

La taille de la grille= 283 ×283.

Le nombre de fourmi=20 fourmis, chacune se déplace 25000 fois.

Nombre de classes obtenu = 2292 classes.

Les résultats de test des données d'apprentissage sont comme suit :

Nom du fichier	Effectif	Nombre de connexions classées normales	nombre de connexions classées attaques
Echantillons d'apprentissage	40000	40000	00
KDD_normale	97277	72948	24329
KDD_attaques	396742	3957	392785

TAB IV.6 Résultats du test des données d'apprentissage (Base d'apprentissage de 40000connexions)

.

La matrice de confusion correspondante à ces résultats est comme suit :

→	**Normale**	**Attaque**
Normale	75,52%	24,47%
Attaques	0.99%	99,00%
	PCC=94,27%	

TAB IV.7 Matrice de confusion de ANTClass sur les données d'apprentissage (Base d'apprentissage de 40000connexions)

d) Expérimentation du modèle sur la base 10% de KDD'99 :

Pour expérimenter le système sur 10% de la base KDD'99 nous avons fixé ses paramètres comme suit :

La taille de la grille= 417 ×417.

Le nombre de fourmi=25 fourmis, chacune se déplace 25000 fois.

Nombre de classes obtenu = 11659 classes.

Les résultats de test des données d'apprentissage sont comme suit :

Nom du fichier	Effectif	Nombre de connexions classées normales	nombre de connexions classées attaques
Echantillons d'apprentissage	86874	86874	00
KDD_normale	97277	97277	00
KDD_attaques	396742	3338	393404

TAB IV.8 Résultats du test des données d'apprentissage (Base d'apprentissage 10% de KDD)

.

La matrice de confusion correspondante à ces résultats est comme suit :

→	**Normale**	**Attaque**
Normale	100%	00%
Attaques	0.84%	99,16%
	PCC=99,32%	

TAB IV.9 Matrice de confusion de ANTClass sur les données d'apprentissage (Base d'apprentissage 10% de KDD)

Les résultats des matrices ci-dessus confirment que le modèle construit est très compatible avec les données d'apprentissage. En effet, dans tous les cas les connexions normales constituants la base d'apprentissage sont bien classifiées lors de la phase de test. Nous remarquons aussi que le taux de faux positifs se réduit en incrémentant le nombre de connexions d'apprentissage tel que pour les bases de 500, 2000 et 40000 connexions le taux de faux positif est respectivement égal à 0.40, 0.33 et 0.20 puis une valeur nulle est obtenue lorsque le système a appris toute la base. De plus un PCC de l'ordre de 99,32% obtenu lors de la dernière expérimentation sur les données d'apprentissage témoigne de la cohérence du modèle et du fait qu'il rend vraiment compte des données d'apprentissage.

On constate que les connexions normales apprises par le système sont bien classifiées, cependant le système n'a pas pu détecté quelque attaques.

VI. Validation des résultats :

Dans ce qui suit nous rapportons les résultats obtenus sur les données de test dans les cas vus précédemment, le fichier utilisé pour le test est *kdd_test_data*.

a) 1ier cas :

Le tableau suivant présente les résultats trouvés par le système dont l'apprentissage a été fais sur un échantillon de 500 connexions en testant le fichier *kdd_test_data.*

Nom du fichier de test	Effectif (connexions)	Nombre de connexions normales	Nombre d'attaques	Nombre de connexions classées normales	Nombre de connexions classées attaques
Kdd_test_data	311029	60593	250436	19687	291342

TAB IV.10 Résultats du test des données de test (Base d'apprentissage de 500connexions)

La matrice de confusion correspondante à ce tableau est la suivante :

→	**Normale**	**Attaque**
Normale	32,49%	67,51 %
Attaques	0,00%	100%

PCC=86,84%, Précision=0.59, Rappel=1

TAB IV.11 Matrice de confusion de ANTClass sur les données de test (Base d'apprentissage de 500connexions)

b) 2ième cas :

Le tableau suivant présente les résultats trouvés par le système dont l'apprentissage a été fais sur un échantillon de 2000 connexions en testant le fichier *kdd_test_data.*

Nom du fichier de test	Effectif (connexions)	Nombre de connexions normales	Nombre d'attaques	Nombre de connexions classées normales	Nombre de connexions classées attaques
Kdd_test_data	311029	60593	250436	34157	276872

TAB IV.12 Résultats du test des données de test (Base d'apprentissage de 2000connexions)

.

La matrice de confusion correspondante à ce tableau est la suivante :

→	**Normale**	**Attaque**
Normale	56,37%	43,63 %
Attaques	0,00%	100%

PCC=91,50%, Précision=0.69, Rappel=1

TAB IV.13 Matrice de confusion de ANTClass sur les données de test.
(Base d'apprentissage de 500connexions)

c) 3ième cas :

Le tableau suivant présente les résultats trouvés par le système dont l'apprentissage a été fais sur un échantillon de 40000 connexions en testant le fichier *kdd_test_data.*

Nom du fichier de test	Effectif (connexions)	Nombre de connexions normales	Nombre d'attaques	Nombre de connexions classées normales	Nombre de connexions classées attaques
Kdd_test_data	311029	60593	250436	36853	274176

TAB IV.14 Résultats du test des données de test (Base d'apprentissage de 40000connexions)

La matrice de confusion correspondante à ces résultats est la suivante :

→	**Normale**	**Attaque**
Normale	60.82%	39.18 %
Attaques	0,00%	100%

PCC=92,36%, Précision=0.71, Rappel=1

TAB IV.15 Matrice de confusion de ANTClass sur les données de test.
(Base d'apprentissage de 500connexions)

d) 4ième cas :

Le tableau suivant présente les résultats trouvés par le système dont l'apprentissage a été fais sur la totalité de la base 10% de KDD'99 en testant le fichier *kdd_test_data.*

Nom du fichier de test	Contenu	Nombre de connexions normales	Nombre d'attaques	Nombre de connexions classées normales	Nombre de connexions classées attaques
Kdd_test_data	311029	60593	250436	57647	253382

TAB IV.16 Résultats du test des données de test (Base d'apprentissage 10% de KDD)

La matrice de confusion correspondante à ces résultats est la suivante :

→	**Normale**	**Attaque**
Normale	95.13%	4.86 %
Attaques	0,00%	100%

PCC=99.05%, Précision=0.95, Rappel=1

TAB IV.17 Matrice de confusion de ANTClass sur les données de test.
(Base d'apprentissage de 500connexions)

Les matrices de confusion du système sur les données de test, dans ces différentes expérimentations montrent que :

- Les attaques sont bien détectées par le système avec un taux de détection de l'ordre de 100% dans toutes les expérimentations.
- Le taux de faux positif qui renseigne sur le taux de fausses alertes généré par le système se réduit en agrandissant la base d'apprentissage (0.40, 0.30, 0.28 puis 0.04 respectivement pour le 1^{ier}, $2^{ième}$,$3^{ième}$ et le $4^{ième}$ cas).
- A chaque amélioration de la base d'apprentissage, le système améliore l'exactitude de la détection des intrusions, tel que la précision progresse de 0.59 à 0.95 en passant par 0.69 puis 0.71. Cependant le rappel qui traduit le taux d'intrusions correctement détectées par rapport au nombre total d'intrusions existantes reste toujours stable et égale à 1.

VII. Analyse et explication des résultats :

Ces résultats peuvent s'expliquer par :

- Le principe de classification de ANTClass : le test sur les données d'apprentissage a montré que l'algorithme ANTClass est très cohérent avec ces données, ceci peut être expliqué par sons principe de classification qui se base sur l'hybridation de ANTs et K_Means, Les agents fourmis utilisés par ANTs classifient les objets selon les probabilités de ramassage et de dépôt, tel qu'un objet n'est ramassé si et seulement si sa probabilité de ramassage soit égale à un, et

n'est déposé que si sa probabilité de dépôt soit aussi égale à un, et comme ces deux mesures dépendent de la distance euclidienne entre deux objets donc les classes formées comportent les connexions les plus similaires entre elles. Puis l'algorithme K_Means intervient pour améliorer la distribution des connexions dans les classes, et ceci toujours par le calcul de la distance euclidienne entre les connexions et les centres de gravité des classes formées par ANTs.

- Le taux de fausses alertes généré par le système lors de la phase de validation (4.86%) peut être expliqué par la non stabilisation des connexions lors de la phase d'apprentissage. Dans cette expérimentation le système a appris toute la base KDD'99 et il n'a pas aboutis à la partition finale car ceci nécessite une longue durée, donc on a employé une partition proche de la partition optimale.
- Les expérimentations ont montrées que le taux de détection a été amélioré en augmentant la taille de la base d'apprentissage, ce qui signifie que l'ajout de nouvelles connexions normales rapproche le profil établit au profil réel des utilisateurs.
- Limite de l'algorithme ANTclass : cet algorithme est basé sur des formules mathématiques qui ne peuvent être appliquées que sur des données numériques, pour cela l'étape de l'élimination des attributs qualitatifs de la base d'apprentissage été une étape indispensable, cette réduction des attributs de la base d'apprentissage pourrait affaiblir sensiblement les performances du système.
- L'apparition dans les données de test de valeurs nouvelles très rares (pour les attributs symboliques) ou extrêmes (pour les attributs continus) (Voir Annexe C).

Conclusion :

Nous avons présenté dans ce chapitre un modèle de détection d’intrusions réseaux selon l’approche comportementale, l’idée sur laquelle à été conçu est de modéliser le comportement normal des utilisateurs à l’aide d’une méthode de classification non supervisée des données qui est l’algorithme ANTClass. Le développement et le test du système proposé ont été faits en utilisant les connexions de la base KDD’99.

Les motivations de ce travail sont la construction d’un IDS comportemental non seulement détecte toute utilisation malveillante du système mais aussi génère le minimum de fausses alertes.

Les expérimentations ont montré qu’un tel système répond bien aux objectifs soulignés au départ et les résultats obtenus sont très prometteurs. Pour cela nous désirons de finaliser ce travail en appliquant ce modèle sur toute la base de données KDD’99.

Le chapitre suivant sera un autre modèle d’un IDS basé sur les réseaux de neurones artificiels.

Introduction :

Les réseaux de neurones sont souvent utilisés en classification de données notamment les cartes auto_organisatrices de Kohonen que nous avons présenté dans le chapitre II, ces cartes ont l'habilité de partitionner un ensemble de données en un ensemble de classes chacune comporte les objets les plus similaires entre eux qu'avec les objets appartenants aux autres classes sans connaître auparavant leurs classes originales.

Dans ce chapitre nous présentons une architecture d'un réseau de neurones qui nous permet de classifier des connexions TCP/IP de la base KDD'99 dans le but de construire un système de détection d'intrusions selon l'approche comportementale. L'implémentation de ce système sera ensuite détaillée et suivie des résultats obtenus.

I. Type de RNA choisi :

Compte tenu des différents types de réseaux neuronaux existants exposés au chapitre II, de notre choix de pouvoir partitionner la base KDD'99 nous décidons de nous attacher plus particulièrement à la carte auto organisatrice de Kohonen. Ce type de réseaux semble mieux adapté à la classification non supervisée des connexions normales car, contrairement aux autres modèles des réseaux neuronaux, ne nécessite pas de connaître le nombre de classes à obtenir préalablement.

II. Idée de base :

A l'aide de la carte auto organisatrice de Kohonen, nous souhaitons déterminer le profil normal des utilisateurs, à cette fin on utilise une base d'apprentissage comportant seulement des connexions normales.

Les réseaux de type auto-organisateur de Kohonen sont des réseaux compétitifs et dynamiques, dans le sens où ils ont tendance à élire un neurone vainqueur et à le favoriser, d'une part, et à créer ou détruire des neurones, d'autre part.

Le modèle de carte topologique proposé par Kohonen est un procédé d'auto-organisation qui va projeter les données initiales sur un espace discret et régulier de faible dimension (en général 1, 2 ou 3D). Les espaces utilisés sont des treillis réguliers dont chacun des noeuds est occupé par un neurone. La notion de voisinage entre les neurones découle alors directement de la structure et définit une topologie de la carte.

Grâce au procédé d'auto-organisation, la topologie qui lie les données initiales est conservée au niveau des réponses proposées par le réseau [091][087]. À la manière des réseaux à compétition, les cartes topologiques de Kohonen sont constituées de deux couches. La première, appelée couche d'entrée, est composée de N neurones sert uniquement à la présentation des objets à classer et les états de tous ses neurones sont forcés aux valeurs des données d'entrée, dans notre cas ces objets sont les connexion TCP/IP normale de la base KDD'99. La couche de sortie, ou couche compétitive, est composée d'un nombre M de neurones régulièrement répartis sur une carte bidimensionnelle.

À la présentation d'une entrée, un neurone sur la carte est sélectionné. Il correspond le plus possible à cette entrée de minimisation de la distance euclidienne.

Suivant ce principe et après apprentissage de toutes les connexions normales, nous obtenons le profil normal du système représenté par une carte d'activation des neurones gagnants et leurs voisins, tel que les neurones ayants le plus haut niveau d'activation correspond aux centres de gravité des classes normales du profil.

La fonction d'activation des neurones est définie en fonction de la distance euclidienne entre deux vecteurs (vecteur d'entrée et vecteur de sortie), tel que une distance minimale donne une valeur maximale de cette fonction.

Le principe de la détection des anomalies est : si une attaque est présentée sur la couche d'entrée aucun neurone de la couche de sortie n'est activé, cependant la présentation d'une connexion normale active forcement une sortie de la carte d'activation et qui correspond à la classe dont elle appartient.

III. Conception du système:

Nous avons choisi d'utiliser une carte de Kohonen, dont l'algorithme d'apprentissage est compétitif, nous présentons dans ce paragraphe les étapes de la conception d'une architecture appliquant un tel algorithme sur la base KDD'99 afin de construire un système de détection d'intrusions comportemental.

III.1. Prétraitement et normalisation des données :

Tout processus de classification commence par une étape d'acquisition des observations qui consiste à déterminer les attributs caractérisant aux mieux les objets à classifier.

Dans le cadre de l'approche ici proposée, l'échantillon d'observations est constitué des connexions TCP/IP extraites de la base 10% de la base KDD'99. Les caractéristiques descriptives de chaque connexion définissent les attributs des observations dans un espace à 41 dimensions. Nous constatons que ces attributs sont de deux types, des attributs quantitatifs et d'autres qualitatifs, nous intéressons seulement aux attributs prenant des occurrences quantitatives. Dans ce but nous reprenons le même module de prétraitement conçu pour le système de détection à base de ANTclass, nous rappelons que ce module est chargé de :

- Extraire les connexions normales de la base 10% de KDD'99.
- Eliminer les attributs qualitatifs.
- Eliminer les connexions redondantes.

Un autre traitement des données est nécessaire particulièrement pour les réseaux de neurones qui est le codage ou la normalisation des entrées, ces dernières doivent être codées entre 0 et 1 afin de faciliter la convergence, les poids s'ajustent, en effet, plus rapidement, lorsqu'ils traitent des valeurs ayant la même échelle de grandeur.

Pour obtenir des valeurs des attributs dans l'intervalle [0,1], nous utilisons la normalisation suivante :

Soit un attribut ATT_i ayant des valeurs dans l'intervalle $[Min_i, Max_i]$, la normalisation consiste à trouver pour chaque valeur de cet intervalle sa correspondante dans l'intervalle [0,1], pour ce faire nous appliquons la fonction suivante :

$$\text{Norm} : [Min_i, Max_i] \rightarrow [0,1]$$

$$\text{Val} \rightarrow \frac{Val - Min_i}{Max_i - Min_i}$$

Pour $Val = Min_i$ nous obtenons la limite inférieure de l'intervalle « 0 »

Pour $Val = Max_i$ nous obtenons la limite supérieure de l'intervalle « 1 »

Ces opérations de prétraitement et de codage serviront donc à alléger et à optimiser le processus de classification en réduisant la taille des données, ces données sont stockées dans une base d'apprentissage sous forme d'un fichier texte que nous appelons *kdd_normalisée.txt.*

III.2. La couche d'entrée :

Cette couche permet d'entrer au système les valeurs des attributs descriptifs des connexions (les exemples) qui doit les apprendre, dans notre cas les exemples sont les connexions de la base *kdd_normalisée* caractérisées par 38 attributs numériques, donc la couche d'entrée comporte autant de neurones qu'il y'a d'attributs des connexions de la base d'exemples, elle est représentée par un vecteur unidimensionnelle (vecteur E dans les figures FIG V.1) tel que chaque neurone de cette couche est connecté à tous les neurones de la couche de sortie.

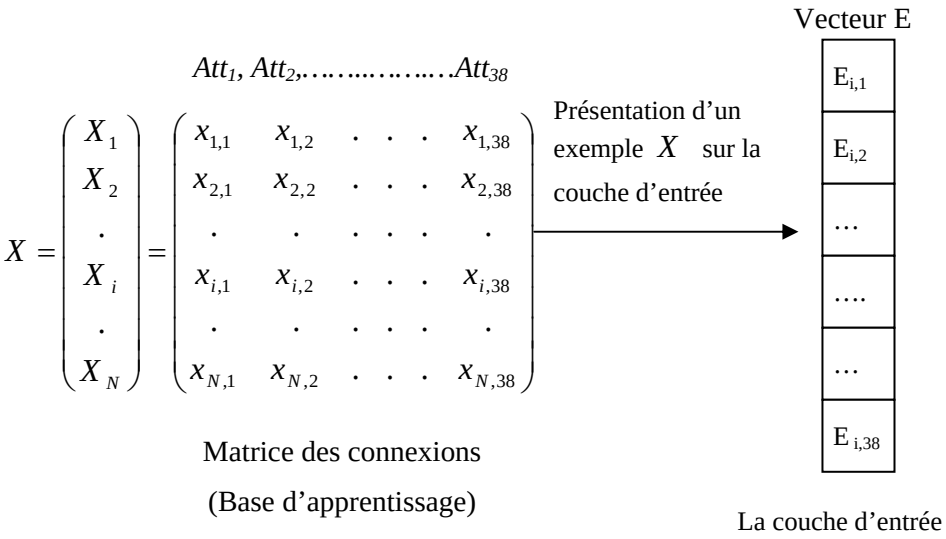

FIG V.1 La base d'exemples et la couche d'entrée

III.3. La couche de sortie :

La couche de sortie du réseau de Kohonen est appelée aussi carte auto organisatrice de kohonen, elle peut être de différentes formes : vecteur unidimensionnel, forme rectangulaire (figure a) ou hexagonale (figure b). Nous choisissons pour notre application la forme rectangulaire.

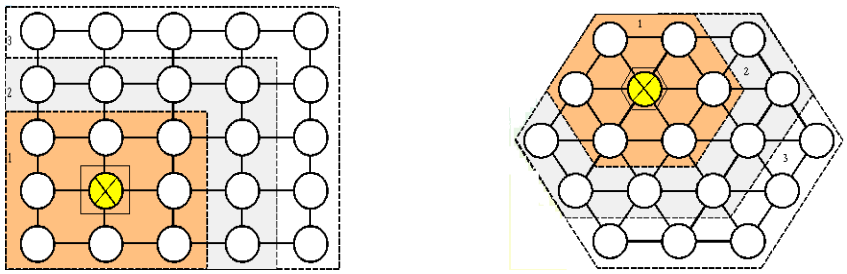

Fig a. carte rectangulaire Fig b. carte hexagonale

FIG V.2 Formes possible de la carte de Kohonen.

Le nombre de neurones de cette couche doit être fixé d'une manière à assurer qu'il dépasse le nombre de classes du profil normal, puisque il s'agit d'un apprentissage non supervisé le nombre de classes à obtenir n'est pas connu à priori et donc le nombre de neurones sera fixé après quelques essayes d'apprentissage, si la carte initiale est insuffisante on rajoute des lignes et des colonnes jusqu'à l'obtention d'une taille adéquate.

Les neurones de la couche d'entrée sont connectés à ceux de la couche de sortie. Chaque connexion d'un neurone d'entrée E vers un neurone de sortie K a le vecteur poids $W_{K,E}$, ainsi chaque neurone de sortie K a le vecteur poids $W_K = [W_{K,1}, W_{K,2}, \ldots\ldots W_{K,38}]$ (voir la figure V.3)

Chaque neurone de la couche de sortie est donc caractérisé par sa position relative sur la carte et par son vecteur poids dans l'espace de représentation des connexions. La figure FIG V.3 montre la structure du réseau de Kohonen utilisé par notre application.

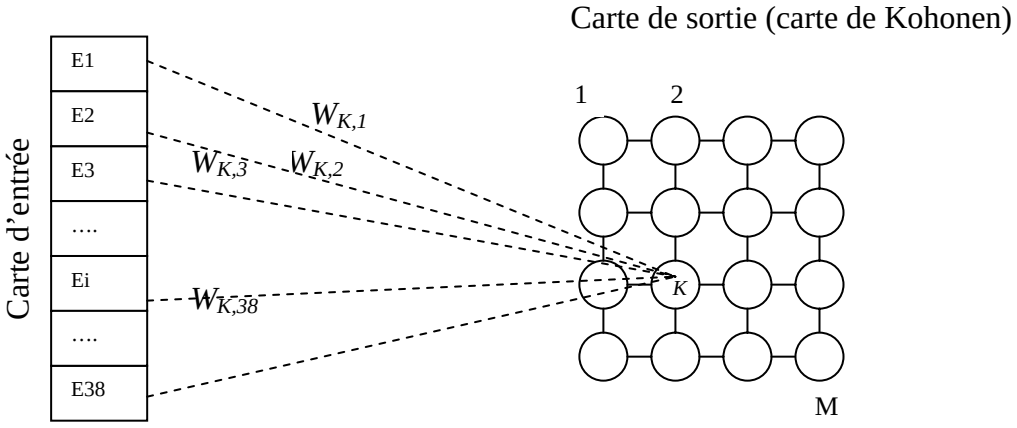

FIG V.3 Structure du réseau de Kohonen utilisé.

III.3. Fonction de transfert et fonction d'activation :

Un réseau de neurones est ainsi constitué de cellules (ou neurones), connectée entre elles par des liaisons affectées de poids. Ces liaisons permettent à chaque cellule de disposer d'un canal pour envoyer et recevoir des signaux en provenance d'autres cellules du réseau. Chacune de ces connexions reçoit un poids (une pondération), qui détermine sont impact sur les cellules qu'elle connecte. Chaque cellule dispose ainsi d'une entrée, qui lui permet de recevoir de l'information d'autres cellules, mais aussi de ce que l'on appelle une *fonction d'activation*.

Le principe de sélection du neurone gagnant par le modèle de Kohonen se base sur la distance euclidienne, qui est utilisée comme fonction de transfert, entre l'exemple présenté E et tous les neurones de sortie, tel que le neurone vainqueur K est celui qui est le plus proche de l'entrée présentée et qui vérifie l'inégalité suivante :

$$K: \quad \arg\min_i \quad \|W_K - E\| \leq \|W_j - E\| \qquad \forall j \in \{1, N\} \qquad \text{(V.1)}$$

La distance euclidienne est également utilisée pour calculer l'activation de chaque neurone de sortie en définissant la fonction d'activation de la manière suivante :

$$Act_K = \frac{A}{B + C\|W_K - p\|} \qquad \text{(V.2)}$$

A, B et C sont des constantes quelconques, leurs valeurs seront retenues après des essayes d'entraînement du réseau.

Les niveaux d'activation des neurones gagnants (correspondants aux centres de gravité des classes) et de leurs voisins sont stockés dans une carte rectangulaire (matrice) de la même taille que la couche de sortie, cette carte sera construite au fur et à mesure de l'entraînement du réseau et elle sera utilisée comme référence lors de la phase de la détection d'anomalies.

III.4. Algorithme d'apprentissage :

La phase d'apprentissage du système proposé se base principalement sur l'apprentissage du réseau de Kohonen, il est réalisé directement à partir de données numériques et se concrétise du point de vue interne par une adaptation automatique des seuls éléments susceptibles de varier dans un réseau de neurones : les poids synaptiques. Cet apprentissage est de type compétitif et qui est en général de favoriser le neurone vainqueur et ses voisins et ce en les rapprochant du vecteur d'entrée.

Soit la base d'exemples $X = \{X_1, X_2, ...X_N\}$ induite du module de prétraitement et normalisation des entrées et qui ne comporte que des connexions TCP/IP normales ordonnées de 1 à *N*, tel que chaque élément X_i est une connexion caractérisée par 38 attributs quantitatifs.

Le système proposé s'entraîne selon les étapes suivantes et qui sont décrites par la figure FIG V.4 :

- Définir une carte de sortie et initialiser les poids synaptiques de ses neurones à des valeurs aléatoires entre 0 et 1.
- définir une carte d'activation de la même taille que la couche de sortie.
- initialiser la carte d'activation à zéro.
- pour toute connexion X_i avec $1 \le i \le N$ faire :

  • Charger la connexion X_i de la base d'exemple dans un vecteur de 38 dimensions qui est la couche d'entrée.

• Trouver le neurone gagnant en calculant la distance euclidienne entre la connexion présentée et tous les neurones de la couche de sortie, ce neurone est celui qui est le plus proche de la connexion X_i.

• Calculer le niveau d'activation du neurone élu *K* en appliquant la fonction d'activation définie par l'équation V.2 .Si le niveau d'activation du neurone élu à l'itération t est inférieur au niveau d'activation du même neurone à l'itération t-1 alors mettre à jour les poids synaptiques de ses neurones voisins selon l'équation :

$$Wj(t) = Wj(t-1) + \alpha(t-1).V(k, j, t-1).(X_i - W_j(t-1)) \qquad \text{(V.3)}$$

La fonction de voisinage $V(k, j, t)$ et le taux d'apprentissage $\alpha(t)$ sont définis respectivement par les équations (II.19) et (II.18)

Ainsi leurs niveaux d'activation doivent être modifiés sur la carte d'activation.

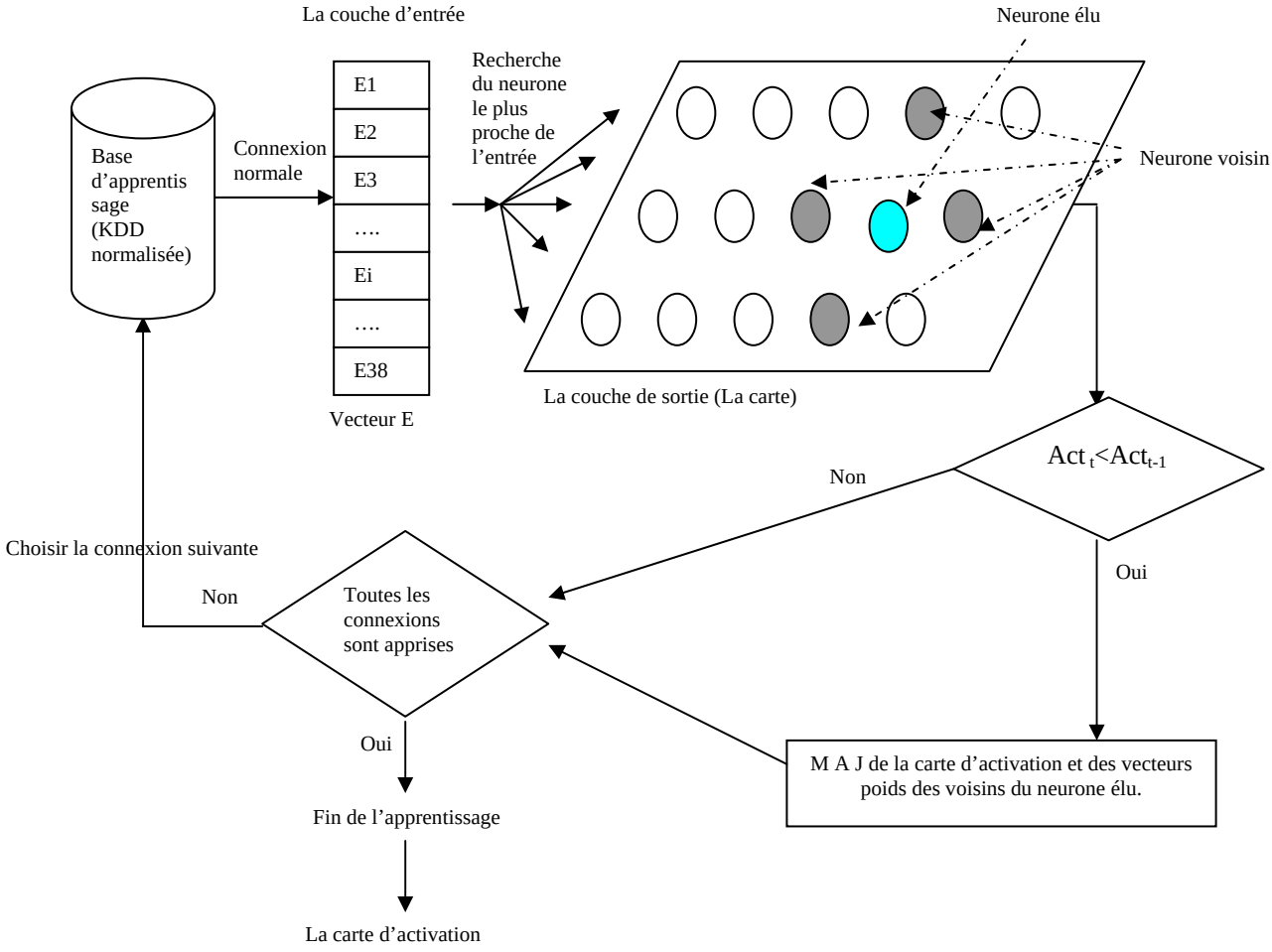

FIG V.4 Apprentissage du système à base de la carte de Kohonen

La carte d'activation obtenue comporte les niveaux minimaux d'activation des neurones élus et de leurs voisins, les régions des neurones exhibés sur la carte de Kohonen représente une partitions de la base d'apprentissage, tel que les neurones ayants le plus haut niveau d'activation correspondants aux centres de gravité des classes contrairement à leurs voisins qu'ils ont un niveau plus bas. Ces niveaux seront utilisés pour la détection des connexions non apprises par le système, tel qu'un niveau inférieur à ceux de la carte résulte de la présentation d'une connexion non apprise lors de l'apprentissage.

Algorithme d'apprentissage

1. Initialiser les vecteurs poids $W_K(0)$ avec de petites valeurs aléatoires entre 0 et 1.
2. Initialiser la carte d'activation à zéro.
3. Fixer le taux d'apprentissage initial $\alpha(0)$, la variance initiale σ_i et la variance finale σ_f.
3. Pour t=1 à N faire :

(a) Charger la connexion X_i de la base d'apprentissage.

(b) Déterminer le neurone gagnant *K* à l'aide de l'équation :

$$K: \quad \arg\min_i \quad \|W_K - X_i\| \le \|W_j - X_i\| \qquad \forall j \in \{1, N\}$$

(c) Calculer le niveau d'activation du neurone élu *K* à l'aide de la fonction :

$$Act_K = \frac{A}{B + C\|W_K - X_i\|}$$

(d) Si $Act_k(t) < Act_K(t-1)$ alors

Mettre à jour les poids des voisins à l'aide de l'équation:

$Wj(t) = Wj(t-1) + \alpha(t-1).V(k, j, t-1).(X_i - W_j(t-1))$ Où j est un neurone voisin de *K*

Mettre à jour les niveaux d'activation du neurone élu et de ses voisins.

(e) Fin pour

IV. Implémentation :

Ce modèle a été développé sur le même environnement de développement de ANTClass décrit au chapitre IV.

IV.1. Implémentation de la couche d'entrée :

La couche d'entrée d'un réseau de neurone lui permet de recevoir les données à apprendre, dans notre étude ces données sont des connexions TCP/IP stockées dans une base que nous avons appelé *kdd_normalisée.* Ces connexion sont caractérisées par 38 attributs ce qui nécessite lors de

l'implémentation un vecteur unidimensionnelle de 38 éléments, le module permettant de charger une connexion de la base d'apprentissage sur la couche d'entrée est listé par la figure suivante :

```
static double[ ] charger_une_connexion(int num)throws IOException
{ double connexion[ ]= new double [38];
 BufferedReader entree= new BufferedReader (new FileReader ("kdd_normalisée.txt"));
   int j=0;
   boolean repeter=true;
   while (repeter==true)
   {String lignelue=entree.readLine();
     if (j ==num)
     { repeter=false;
       StringTokenizer tok= new StringTokenizer (lignelue, ",");
       for (int k=0;k<38;k++) connexion[k]= Float.parseFloat(tok.nextToken());
      entree.close();
     }
     else j++;
   }
  return connexion;
}
```

FIG V.5 Listing du module de chargement des connexions de la base d'apprentissage.

La création et le chargement du vecteur d'entrée se fait comme suit :

```
double [ ] entrée=new double[38]; //création de la couche d'entrée
for ( T=0;T<N;T++)   // N est la cardinalité de la base d'apprentissage
   {entrée =charger_une_connexion(T) ; //Chargement de la connexion N° T
    // la Suite de l'algorithme
   }
```

FIG V.6 création et chargement du vecteur d'entrée

IV.2. Implémentation de la carte de Kohonen :

Comme nous avons vu, cette carte est bidimensionnelle, la structure la plus adéquate pour l'implémenter est une matrice de $l \times c$ avec l nombre de ligne et c nombre de colonne tel que $l \times c = M$ (M est le nombre de neurones de sortie), dans notre implémentation nous avons choisi d'utiliser une matrice carrée.

Nous avons choisi ainsi d'implémenter les liens entre les neurones de sorties et les neurones d'entrée, chacun par un vecteur, les poids caractérisant ces liens sont initialisés aléatoirement à des petites valeurs entre 0 et 1. Nous combinons ces deus structures pour obtenir une matrice de M vecteurs.

Le fragment du programme dans la figure suivante implémente cette structure.

```
static double [ ][ ][ ] carte= new double [L][L][38];  //la couche de sortie

/* initialisation des poids synaptiques */
 for (int i=0;i<L;i++)
   { for (int j=0;j<L;j++)
     {for (int p=0;p<38;p++)
       { carte[i][j][p]=Math.random();
     }}}
```

FIG V.7 Création de la carte de Kohonen et initialisation des poids

V. Résultats et test des résultats :

Plusieurs tests ont été effectués en jouant sur les différents paramètres ayant une incidence sur le résultat final : le taux d'apprentissage initial, la variance initiale et la taille de la carte de sortie. Ces tests ont permis de déterminer les valeurs optimales des paramètres du réseau en fixant un intervalle dans lequel sont prélevées celles qui seront utilisées avec l'échantillonnage choisi. D'après les expérimentations faites, nous remarquons que la taille de la carte de sortie s'accroît en agrandissant la taille de la base d'exemples. Nous avons pris un échantillon de 500 connexions partitionné à l'aide d'une carte de 225 neurones (15×15). Puis une expérimentation a été faites en prenant une base d'apprentissage de 2000 connexions ceci nécessite une carte de sortie de 10000 neurones (100×100). Un échantillon de 3000 connexions nécessite une carte de sortie de 22500 neurones (150×150). Et pour une base plus grande on doit agrandir la taille de la carte de sortie ce qui rend l'apprentissage et le test trop lents. Les paramètres ont été fixé pour toutes les expérimentations comme suit :

- Taux d'apprentissage initiale= 2.5.
- La variance initiale et la variance finale sont fixées respectivement à 1.5 et 0.5.
- Le paramètre $K_\alpha = 1$.
- les constantes de la fonction d'activation A=10, B=1, C=1.

- le voisinage d'un neurone est fixé à 2.

L'interface graphique du système développé permettant d'introduire ces paramètres et de lire les résultats est présentée par la figure suivante :

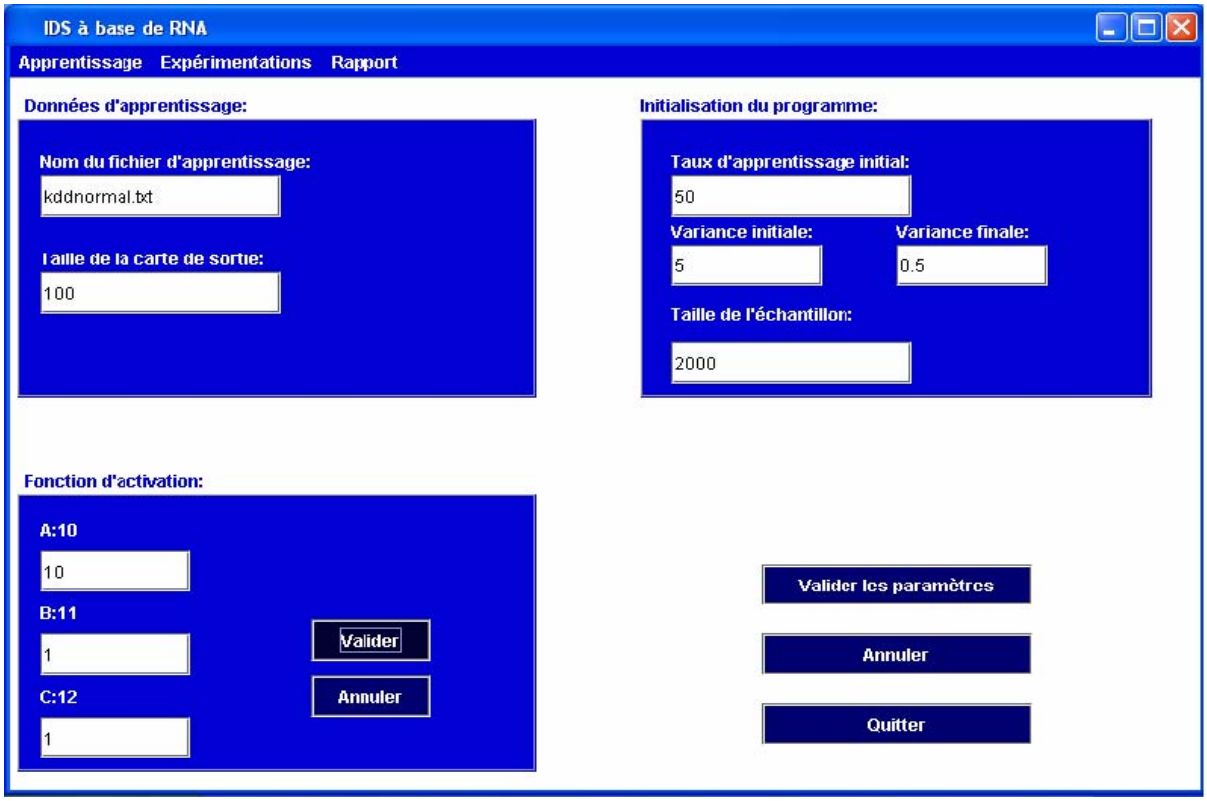

FIG V.8 Interface graphique du système.

a) Expérimentation du modèle sur un échantillon de 500 connexions :

Nous avons pris un échantillon de 500 connexions de la base *kdd_normalsée*, la carte de Kohonen obtenue est visualisée par la figure V.9 :

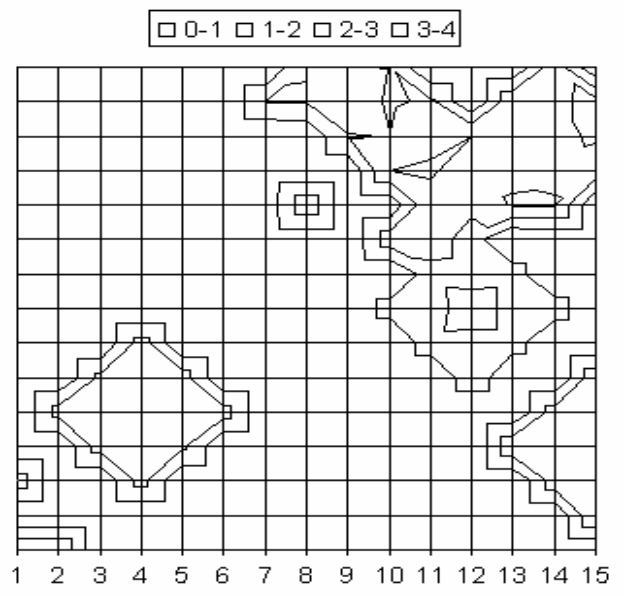

Fig. a

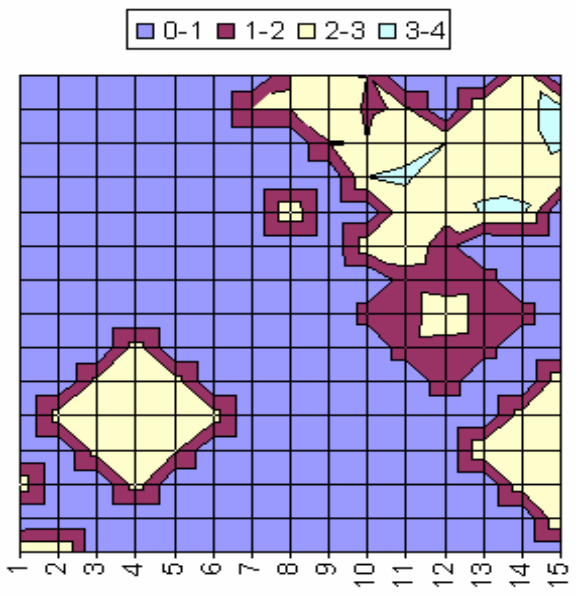

Fig. b

FIG V.9 Vue de la carte de Kohonen après apprentissage

Les deux figures donnent une vue de la carte de Kohonen après apprentissage de 500 connexions normales, les régions colorées dans la figure b montre les différents niveaux d'exhibition des neurones élus et leurs voisins. Comme la légende montre, la carte est divisée en 4 régions, la plus marquante est celle où la couleur est en bleu foncé qui sont les neurones non actives ou les neurones dont le niveau d'activation est faible (entre 0 et 1), les neurones colorés en bleu clair au centre des régions exhibées sont les neurones les plus exhibés tel que leurs niveau d'activation dépasse 3.

La carte d'activation raccordée à cette carte de neurone est listée sous forme d'un tableau (TAB V.1)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	2,88	3	2.57	0	0	0	0	0	0	0	0	0	0	0	0
2	2.72	2.71	0	0	0	0	0	0	0	0	0	0	0	0	3
3	2,58	0	0	2,4	0	0	0	0	0	0	0	0	0	2,63	2,66
4	0	0	2,49	3	2,17	0	0	0	0	0	0	0	2,84	2,75	2,83
5	0	2,41	2,58	2,87	2,52	2,42	0	0	0	0	0	0	0	2,51	2,8
6	0	0	2,29	2,46	2,49	0	0	0	0	0	0	1,65	0	0	2,71
7	0	0	0	2,42	0	0	0	0	0	0	1,62	1,55	1,47	0	0
8		0	0	0	0	0	0	0	0	1,41	1,45	2,76	1,5	1,47	0
9	0	0	0	0	0	0	0	0,15	0	0	1,5	1,36	1,48	0	0
10	0	0	0	0	0	0	0,18	0,18	0,17	2,56	2,71	1,38	0	0	0
11	0	0	0	0	0	0,18	0,2	2,74	0,17	0,16	2,86	2,36	3,07	3,11	0
12	0	0	0	0	0	0	0,18	0,16	0,11	3	3,05	2,78	2,86	2,7	2,57
13	0	0	0	0	0	0	0	0,11	1,94	2,02	2,89	3	2,7	2,77	3,12
14	0	0	0	0	0	0	2,02	2,03	2,87	1,84	2,17	0	2,88	2,74	3,32
15	0	0	0	0	0	0	0	1,98	2,03	1,97	0	0	0	2,7	0
16															

TAB V.1 Carte d'activation obtenue

Les cellules égales à zéro de la carte d'activation représentent les neurones non actives cependant les autres donnes les niveaux d'activation des neurones correspondants. La projection de ces valeurs sur une carte bidimensionnelle donne le résultat suivant :

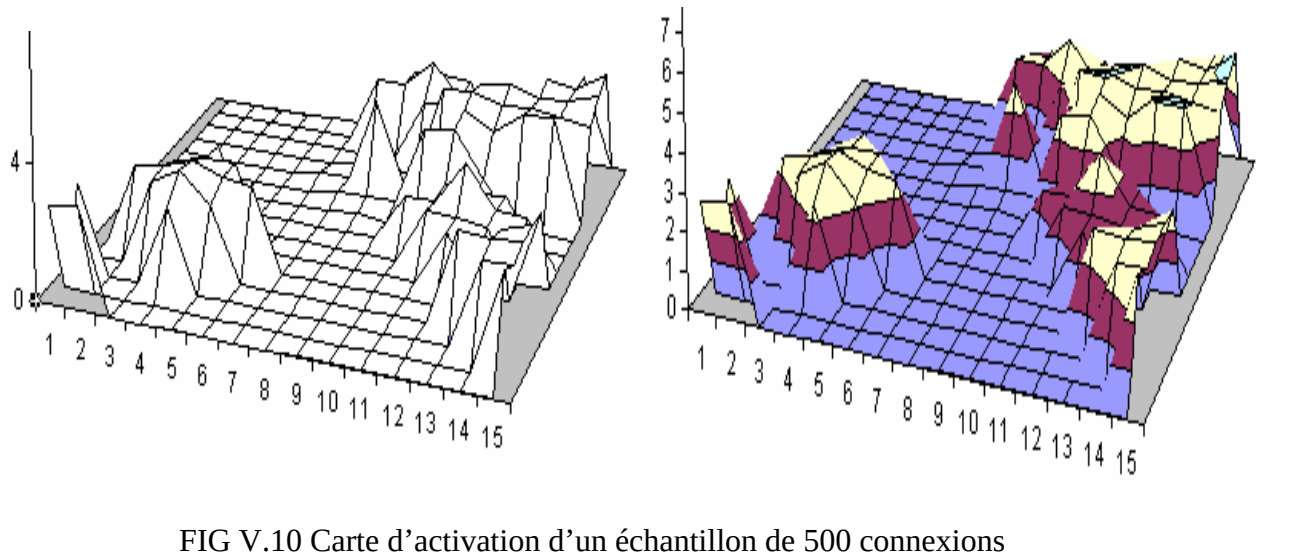

FIG V.10 Carte d'activation d'un échantillon de 500 connexions

Sur la figure FIG V.11, nous présentons la détection d'une connexion normale par le système, cette connexion exhibe le neurone dont les coordonnées sont 11 et 8, avec un niveau d'activation égale à 3.63 qui dépasse le niveau minimale d'activation de ce neurone qui est égale à 2.74, cette différence est largement suffisante pour décider que l'entrée présenté est une connexion normale.

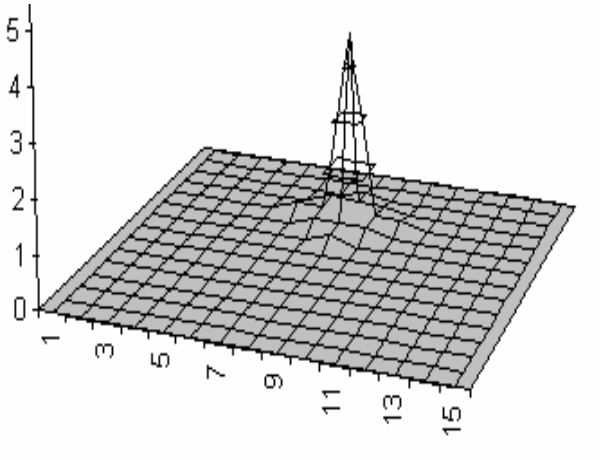

FIG V.11 Exemple de la détection d'une connexion normale.

Les figures suivantes montrent deux cartes d'activations lorsque les entrées n'appartiennent à aucune classe du profil normal formé par le système, dans la figure Fig. a la connexion Attaque 1 présenté à l'entrée du système n'active aucun neurone de la carte de sortie contrairement à la connexion Attaque 2 de la figure Fig. b qui active l'un des neurones de l'une des classes du profil normal (ces coordonnés sont (12,9)) par un niveau égale à 0.003, mais ce niveau est négligeable en comparaison avec le niveau d'activation de ce neurone (0.11) . Par conséquent dans les deux cas les connexions sont considérées comme des attaques.

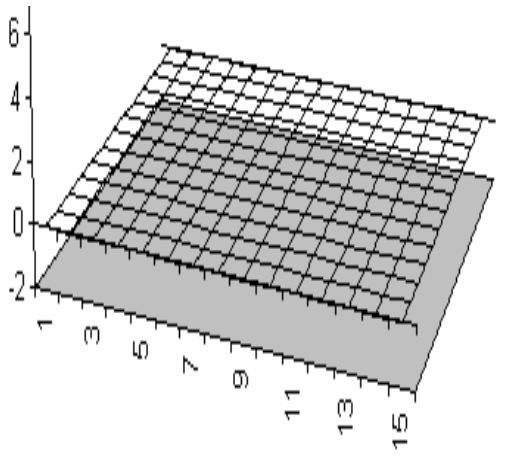

Fig. a Attaque 1

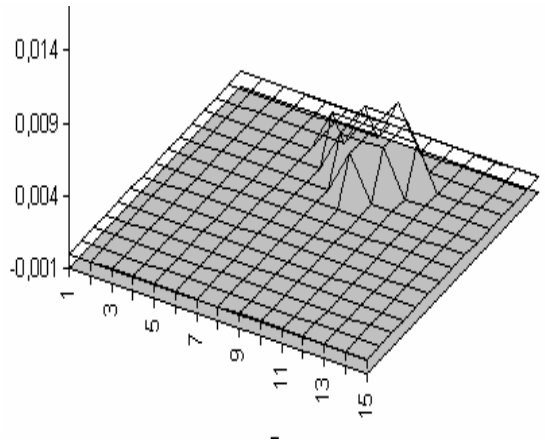

Fig. b Attaque 2

FIG V.12 Exemple de la détection de deux attaques.

L'expérimentation du système proposé sur les données d'apprentissage a donné les résultats suivants :

Nom du fichier	Effectif (connexions)	Nombre de connexions classées normales	Nombre de connexions classées attaques
Echantillons d'apprentissage	500	500	00
KDD_normalisée	97277	27199	70078
KDD_attaques	396742	2703	394039

TAB V.2 Résultats du test des données d'apprentissage (Base d'apprentissage de 500 connexions)

La matrice de confusion correspondante est présentée par le tableau suivant :

→	**Normale**	**Attaque**
Normale	27.96%	72.03%
Attaques	0.68%	99.32%
	PCC=85.26%	

TAB V.3 Matrice de confusion de la carte de Kohonen sur les données d'apprentissage (Base d'apprentissage de 500 connexions)

b) Expérimentation du modèle sur un échantillon de 2000 connexions :

Pour cette expérimentation nous avons pris un échantillon de 2000 connexions de la base *kdd_normalisée*:

L'expérimentation du modèle proposé sur les données d'apprentissage a donné les résultats suivants :

Nom du fichier	Effectif (connexions)	Nombre de connexions classées normales	Nombre de connexions classées attaques
Echantillons d'apprentissage	2000	2000	00
KDD_normalisée	97277	81579	15698
KDD_attaques	396742	3835	392907

TAB V.4 Résultats du test des données d'apprentissage (Base d'apprentissage de 2000 connexions)

La matrice de confusion correspondante est présentée par le tableau suivant :

→	**Normale**	**Attaque**
Normale	83,86%	16,13%
Attaques	0.96%	99,03%
	PCC=96,04%	

TAB V.5 Matrice de confusion de la carte de Kohonen sur les données d'apprentissage (Base d'apprentissage de 2000 connexions)

C) Expérimentation du modèle sur un échantillon de 3000 connexions :

Nous avons pris, dans ce cas, un échantillon de 3000 connexions de la base *kdd_normalisée*, L'expérimentation du système proposé sur les données d'apprentissage a donné les résultats suivants :

Nom du fichier	Effectif (connexions)	Nombre de connexions classées normales	Nombre de connexions classées attaques
Echantillons d'apprentissage	3000	3000	00
KDD_normalisée	97277	83555	13722
KDD_attaques	396742	4839	391903

TAB V.6 Résultats du test des données d'apprentissage (Base d'apprentissage de 3000 connexions)

La matrice de confusion correspondante est présentée par le tableau suivant :

→	**Normale**	**Attaque**
Normale	85,89%	14,10%
Attaques	1,22 %	98,78 %
	PCC=99,02%	

TAB V.7 Matrice de confusion de la carte de Kohonen sur les données d'apprentissage (Base d'apprentissage de 3000 connexions)

L'évaluation de ce système sur les données d'apprentissage montre que le modèle proposé est très cohérent avec les données d'apprentissage. En effet, nous avons obtenu un PCC de l'ordre de 99,02%. L'ajout de nouvelles connexions à la base d'apprentissage à amélioré la classification des connexions normales mais ceci est accompagné par une élévation du taux de faux négatifs de 0.68% à 1.22%.

VI. Validation :

L'évaluation du système se fait en introduisons au système les données de test qui sont stockées dans la base de test et qui comporte 311029 connexions parmi les quelles 60593 sont des connexions normales et les autres sont des attaques, les résultats obtenus sont représentés ci-après :

1ier cas :

Le tableau suivant présente les résultats trouvés par le système dont l'apprentissage a été fais sur un échantillon de 500 connexions :

Nom du fichier de test	Effectif (connexions)	Nombre de connexions normales	Nombre d'attaques	Nombre de connexions classées normales	Nombre de connexions classées attaques
Kdd_test_data	311029	60593	250436	53825	257204

TAB V.8 Résultats du test sur les données de test (Base d'apprentissage de 500 connexions)

La matrice de confusion est donnée par le tableau suivant :

→	**Normale**	**Attaque**
Normale	88.83%	11.17%
Attaques	0.00%	100%

PCC=97.82%, Précision=0.89, Rappel=1

TAB V.9 Matrice de confusion de la carte de Kohonen sur les données de test (Base d'apprentissage de 500 connexions)

2ième cas :

Le tableau suivant présente les résultats trouvés par le système dont l'apprentissage a été fais sur un échantillon de 2000

Nom du fichier de test	Effectif (connexions)	Nombre de connexions normales	Nombre d'attaques	Nombre de connexions classées normales	Nombre de connexions classées attaques
Kdd_test_data	311029	60593	250436	78002	233027

TAB V.10 Résultats du test sur les données de test (Base d'apprentissage de 2000 connexions)

La matrice de confusion est donnée par le tableau suivant :

→	**Normale**	**Attaque**
Normale	100%	0,00%
Attaques	6,95%	93,04%

PCC=94.40%, Précision=1, Rappel=0.93

TAB V.11 Matrice de confusion de la carte de Kohonen sur les données de test.
(Base d'apprentissage de 2000 connexions)

3ième cas :

Le tableau suivant présente les résultats trouvés par le système dont l'apprentissage a été fais sur un échantillon de 3000

Nom du fichier de test	Effectif (connexions)	Nombre de connexions normales	Nombre d'attaques	Nombre de connexions classées normales	Nombre de connexions classées attaques
Kdd_test_data	311029	60593	250436	106016	205013

TAB V.12 Résultats du test sur les données de test(Base d'apprentissage de 3000 connexions)

La matrice de confusion est donnée par le tableau suivant :

→	**Normale**	**Attaque**
Normale	100%	0,00%
Attaques	18,13%	81,86%

PCC=85,39%, Précision=1, Rappel=0.81

TAB V.13 Matrice de confusion de la carte de Kohonen sur les données de test.
(Base d'apprentissage de 500 connexions)

La matrice de confusion du système sur les données de test montre que :

- Les connexions normales sont bien classifiées par le système avec un taux de détection de l'ordre de 100% lorsque le système a appris plus de 2000 connexions.
- Les attaques sont bien détectées lors de la première expérimentation (taux de détection=100%), cependant l'ajout de nouvelles connexions diminue la détection des attaques.
- La précision de détection, qui renseigne sur l'exactitude des intrusions signalées, est amélioré en agrandissant la base d'apprentissage tel que dans le premier cas était de l'ordre de 0.89 et dans les deux autres cas est égale à 1.
- Le rappel, qui traduit le taux d'intrusions correctement détectées par rapport au nombre total d'intrusions existantes, est diminué à chaque expérimentation ainsi que le PCC.
- Un taux de faux négatif remarquable de l'ordre de 18.13%.

VIII. Analyse et explication des résultats :

Ces résultats peuvent trouvés leurs explications dans :

-Le principe d'apprentissage de la carte de Kohonen et l'emploi de la carte d'activation : ce type de carte est caractérisé par un apprentissage compétitif entre les neurones de sortie, chaque neurone attire les connexions les plus proche de son vecteur poids, puis une mise à jour de ses voisins permet de rapprocher leurs vecteurs poids au poids du neurone gagnant, cette mise à jour

dépend de l'entrée présenté, si elle est plus éloignée du neurone vainqueur que la dernière entrée exhibant le même neurone, les voisins sont alors modifiés ainsi que les niveaux d'activations qui seront affaiblis car la distance entre l'entrée présentée et le neurone gagnant a été élevée, sinon cette entrée n'influe ni sur les vecteurs poids et par conséquent ni sur les niveaux d'activations. Cet affaiblissement des niveaux d'activation à chaque présentation d'une entrée plus éloignée rend les neurones très sensibles aux connexions présentées sur la couche d'entrée. Cet apprentissage explique le taux de détection élevé généré par le système, tel que, par le test de la base *KDD_normale* qui comporte 97277 connexions normales le système a trouvé 27199 connexions normales (correspond à 27.96%) lorsqu'il a appris 500 connexions et 81579 connexions (correspond à 83,86%) malgré qu'il a appris seulement 2000 connexions puis 83555 connexions (correspond à 85,89%) par l'apprentissage de 3000 connexions normales. Ce principe a permis au système de classer correctement toutes les connexions normales de la base de test mais il a généré un taux remarquable de faux négatif, ce dernier a été étendu de 0.00% à 18,13% en agrandissant la base d'exemple de 500 à 3000 connexions.

- L'initialisation de l'algorithme : le choix des valeurs des paramètres de l'algorithme n'est pas fixé à l'aide des règles ou des lois bien déterminés, tous dépend des résultats des expérimentations.

IX. Confrontation de ANTCLass et la carte de Kohonen :

Nous résumons et comparons ici les performances de ANTCLass et la carte auto organisatrice de Kohonen selon les perspectives que nous avons considérées plus haut. De l'étude expérimentale de ces deux algorithmes sur KDD'99, il ressort que :

- ✓ Les deux classificateurs construits sont cohérents et compatibles avec les données d'apprentissage.
- ✓ Le test des deux systèmes sur les données d'apprentissage a permis de confirmer que les attaques détectées sont vraiment des attaques car les connexions de la base d'apprentissage sont étiquetées contrairement à la base de test ou les connexions ne sont pas étiquetées.
- ✓ L'expérimentation de ANTCLass sur des bases de tailles déférentes a révélé que le partitionnement des connexions normales de la base KDD'99 par cet algorithme rapproche le profil établit au profil normal des utilisateurs, tel que chaque ajout de nouvelles connexions normales à la base d'apprentissage améliore le profil établit par le système. le taux de faux négatif, qui est dans toutes les expérimentations nul, il est accompagné par une réduction du taux de fausse alertes en agrandissant la base d'apprentissage (67.51%,43.63%, 39.18%,

4.86% pour les base 500, 2000,40000 et 86874 connexions) et donc un accroissement du taux de vrais négatifs (32.49%,56.37%, 60.82% puis 95.13%). cette réduction de fausses alertes résulte un progrès de la précision de détection (0.40, 0.69, 0.71 puis 0.95). Ces résultats nous encourage d'appliquer ANTCLass sur toute la base KDD'99 pour atteindre le meilleure compromis taux de fausses alertes/ taux de vrais négatifs sous-jacent, contrairement au système construit à base de la carte de Kohonen qui n'a pas pu aboutir à cet arrangement.

- ✓ Le système construit à base d'une carte de Kohonen de grande dimension ne garantit pas la convergence vers le profil normal des utilisateurs et génère un grand taux de faux négatif c'est-à-dire il peut raté un grand nombre d'attaque ce qui le rend inefficace dans la pratique.
- ✓ Les expérimentations ont montré que ANTCLass nécessite l'apprentissage de toute la base d'exemples pour établir le profil le plus proche du profil normal des utilisateurs, ce profil est l'ensemble des classes constituées par l'algorithme, et comme la détection se base sur des seuils qui sont les distances maximales entre les centres de gravité des classes et les objets les plus éloignés de ces centres, la présence de toutes les connexions normales est importante, par contre pour la carte de Kohonen il suffit de présenter la connexion la plus éloigné du centre de gravité d'une classe pour trouver le niveau minimal d'activation du neurone de sortie, cette caractéristique donne à tous les deux techniques des avantages et des inconvénients :
  - ➢ Pour ANTCLass :
    - o La détermination de la taille de la base d'apprentissage détermine automatiquement la taille de la grille sur laquelle se déplacent les fourmis.
    - o Le partitionnement de toute la base d'apprentissage garantit la présence de tous les types de connexions possibles.

  Cependant :
    - o Une grande base nécessite une grille volumineuse et donc un temps d'apprentissage et de test lents.
    - o La mise à jour du profil est nécessaire à chaque apparition de nouvelles connexions normales sur le système.
  - ➢ Pour la carte de Kohonen :
    - o Un échantillon réduisant la base d'apprentissage est suffisant pour entraîner le système.
    - o La réduction de la base d'apprentissage facilite l'implémentation et accélère l'apprentissage et le test.

Cependant :

- Une grande base d'exemple comme la base 10% de la base KDD'99(494019 connexions) nécessite une carte de sortie de grandes dimensions ce qui complique l'implémentation, étend le temps d'exécution et d'après les expérimentations les résultats ne sont pas affirmés.
- L'algorithme d'apprentissage de la carte de Kohonen est un algorithme paramétré (taille de la carte de sortie, taux d'apprentissage initial, variance initiale, variance finale), la détermination des valeurs optimales de ces paramètres n'est pas évidente.

De cette comparaison nous concluons que l'algorithme ANTCLass est meilleur que les réseaux de neurone pour établir un profil proche du comportement normal des utilisateurs et donc pour construire un système de détection performant qui détecte le maximum des attaques en générant le minimum de fausses alertes.

Conclusion :

Nous avons présenté dans ce chapitre un modèle de détection d’intrusions réseaux selon l’approche comportementale, l’idée sur laquelle à été conçu est de modéliser le comportement normal des utilisateurs à l’aide d’une méthode de classification de données qui est les réseaux de Kohonen. Le développement et le test du modèle proposé ont été faits en utilisant les connexions de la base KDD’99.

Les motivations de ce travail sont la construction d’un IDS comportemental non seulement détecte toute utilisation malveillante du système mais aussi génère le minimum de fausses alertes.

Les expérimentations, ont montré que le système modélise bien le comportement normal des utilisateurs lorsque la base d’exemple comporte un petit nombre de connexions et donc l’utilisation d’une petite carte, mais il présente un échec lorsque la carte utilisée est de grande dimensions.

La comparaison des deux algorithmes montre que chacun à des avantages par rapport à l’autre,cependant l’algorithme ANTCLass répond bien aux objectifs soulignés au départ.

Conclusion générale

Au terme de ce travail, il y a lieu d'abord de faire le point sur les résultats que nous avons obtenus par rapport à nos objectifs de départ. Il y a lieu de revenir également sur les principales contributions de notre travail et les plus importantes conclusions auxquelles nous avons abouties au cours de ce projet. Enfin, nous allons entrevoir certaines perspectives et pistes pouvant être explorées toujours en rapport avec la problématique dont il est question dans ce mémoire.

Notre objectif premier, faut-il le rappeler, était d'appliquer certaines méthodes d'intelligence artificielle sur la base de données KDD' 99, dans le but de développer des systèmes de détection d'intrusions comportementaux détectant le maximum des attaques en générant le minimum de fausses alertes.

Nous avons proposé deux architectures se basant sur deux méthodes différentes, l'une est basée sur l'algorithme de fourmis artificielles ANTCLass et l'autre sur la carte auto organisatrice de Kohonen. L'idée de base était de partitionner les connexions de la base de données KDD'99 pour modéliser le comportement normals des utilisateurs, puis la détection des attaques se fait en comparant les connexions arrivant sur le système informatique au profil établit.

Les principales métriques qui présentent un intérêt pour évaluer les systèmes proposés et développés durant ce projet sont le taux de détection, taux de faux négatif et le taux de faux positifs, en d'autres termes les attaques signalées, les attaques ratées et les fausses alertes générées par le système. Un système de détection d'intrusion performant est le système qui donne le meilleur arrangement de ces trois facteurs c'est-à-dire qui peut détecter toute utilisation malveillante du système en générant le minimum de fausses alertes. Selon ces métriques nous avons expérimenté nos systèmes sur des bases d'apprentissage de tailles déférentes, prises de la base 10% de la base KDD'99.

ANTClass, un classificateur très simple à construire et à utiliser, s'est avéré très efficace notamment lorsque la quantité des données d'apprentissage est importante mais cela nécessite un long temps pour l'apprentissage. En effet, sur des échantillons de KDD'99, ANTCLass a réalisé des performances prometteuses et encourageantes de l'appliquer sur toutes les données de cette base. Les expérimentations ont montré que cet algorithme converge vers un meilleur arrangement entre le taux de détection et le taux de faux positifs et il répond bien aux objectifs de ce travail. Il

semble bien qu'un système de détection d'intrusions fondé sur cet algorithme peut être exploité dans la pratique mais une mise à jour du profil établit est indispensable à chaque apparition de nouvelles connexions sur le système informatique.

La carte auto organisatrice de Kohonen modélise l'espace d'exemples d'apprentissage dans un autre espace, appelé carte de sortie, d'une façon compétitive entre les neurones de cette carte. En effet une grande base d'apprentissage, tel que la base KDD'99, requiert des cartes de sortie de grandes dimensions ce qui complique leur implémentation ainsi leur utilisation. Le principe de compétition accroît la sensibilité des neurones de sortie aux données présentées sur la couche d'entrées. Ce principe affaiblisse les performances de cet technique car, par expérimentation, elle a présenté un grand taux d'erreur de détection notamment le grand nombre des attaques ratées (un grand taux de faux négatifs). Mais cela n'exclue pas entièrement la carte de Kohonen du domaine de la détection d'intrusions car elle a donnée un bon résultat sur quelques échantillons de KDD'99 et cela peut être exploité dans d'autres travaux.

KDD'99 est une base de données riche d'abord par rapport au nombre de connexions qu'elle contient, ce qui la prédispose pour l'apprentissage et le test des techniques d'apprentissage automatiques et de classification Aussi bien dans le cadre de l'approche comportementale que par scénarios. Il est cependant dans KDD'99 de sérieux problèmes qui mettent les techniques d'apprentissage automatique et de classification dans leur grande majorité en situation critique en ce sens que KDD'99 se caractérise par une différence significative entre les distributions des données d'apprentissage et de test. Aussi, KDD'99 est décrite par un espace d'attributs et de classe de grande dimension et mixte, ce qui requiert énormément de ressources de calcul et de stockage.

Avant de mettre un point final à cette conclusion, en guise de perspectives, nous tenons à mettre l'accent sur certains aspects qui méritent, à notre point de vue, d'être explorés toujours en ce qui concerne la détection d'intrusions. Ces pistes se résument essentiellement aux points suivants :

- ✓ Finaliser le test de ANTCLass sur toute la base de données KDD'99 et comparer ses résultats aux résultats de l'algorithme gagnant de cette compétition.
- ✓ Pour soutenir les performances du modèle qui se base sur ANTCLass, nous proposons d'ajouter un autre module qui donne le type ou bien la catégorie des attaques détectées, et cela par la classification des attaques de la base 10% de KDD'99.

- ✓ L’hybridation avec d’autres méthodes fondées sur l’approche de détection par scénario.
- ✓ La coopération avec d’autres méthodes capables de traiter les attributs qualitatifs des connexions TCP/IP de la base KDD’99.

ANNEXE A

Variantes de l'algorithme K_Means.

1. H-Means et KH-Means:

H_Means est la première variante des K-francs, dans cette approche les partitions initiales sont crées de la même façon que k-Means. Ensuite, durant la phase de convergence, tous les objets sont affectés au cluster dont le centre est le plus proche, à la différence de l'approche K-Means où les centres sont recalculés après chaque réaffectation d'un seul objet.

Il apparaît donc que H-Means converge plus vite vers un optimum local que K-Means, mais K-Means obtiennent de meilleurs résultats. Pour améliorer les résultats une hybridation entre les deux approches est possible, en réalisant successivement une exécution de H-Means puis une exécution des K-Means. Cette approche hybride est appelée KH-Means.

2. K-Harmonic Means :

La méthode K-Harmonic Means a été introduite dans [011] et est différente de l'approche des K-Means sur plusieurs points :

- Cette méthode minimise un critère différent de l'erreur quadratique, qui correspond à la distance harmonique de chaque objet aux centres de toutes les classes.
- Un objet a une probabilité d'appartenir à un cluster (approche non exclusive).
- Dans la méthode des k-Heans chaque objet à un poids identique, ce qui n'est pas vérifié dans K-Harmonic Means.

3. L'algorithme SBKM :

SBKM [010] signifie « symmetry –based K-Means » a été développé dans le but de pallier certains limitations de l'algorithme K_Means original, la distance Euclidienne utilisée dans l'algorithme K-Means permet d'obtenir des cluster sous forme hypersphérique, alors qu'il est possible d'obtenir d'autres formes en employant d'autres distances. Par exemple, la mesure de Mahalanobis qui permet de découvrir des clusters de formes hyperelliptiques, cette mesure s'exprime comme suit entre un objet o_i et le centre d'un cluster c_j :

Mahalanobis $(o_i , c_j) = (o_i — c_j)^{T} \sum^{-1} (o_i — c_j)$

Avec $\sum^{-1}$ l'inverse de la matrice de covariance des objets du cluster concerné g_j

Les auteurs proposent donc une nouvelle mesure de distance appelée « point symmetry distance » (équation suivante) qui associe chaque objet au cluster qui possède un objet qui lui est le plus symétrique possible. Cette mesure de distance d_s est définie entre un objet o_i appartenant à un ensemble g_j de N_j objets ayant pour centre c_j comme suit :

$$ds(oi, c_j j) = \min_{o_k \in [1, N_j],\, o_k \neq oi} \frac{|| (oi - c_j) + (o_k - c_j) ||}{|| (oi - cj) ||\_ + || (o_k - c_j) ||}$$

Cette mesure de distance permet de créer des clusters symétriques par rapport à leurs centres et non obligatoirement hypersphérique. Cet algorithme donne des clusters qui se chevauchent lorsqu'il est utilisé conjointement à la variante K-Means (algorithme II.1).

Algorithme A1.1 SBKM **[062]**

(1) Initialisation Choisir aléatoirement k objets parmi les n disponibles comme centres des premiers clusters.
(2) Classification « vulgaire » Exécuter les K-Means jusqu'à ce que le critère d'arret choisi soit rencontré.
(3) Classification « fine »
(4) Pour x =1 à n faire
(5) Trouver le centre k* le plus proche en tèrme de distances symétrique :
K*← argmin $_{k\in[1,k]}$ d_s (x,c_k)
(6) si ds $(x,k^*) \leq \Theta$ alors
(7) Affecter x au cluster k*
(8) sinon
(9) K*← argmin $_{k\in[1,k]}$ $d_{\text{Euclidienne}}$ (x,c_k)
(10) Affecter x au cluster k*
(11) finsi
(12) fin pour
(13) Mettre à jour les centres des clusters : $c_k \leftarrow (1/n_k) \sum_{i \in sk} x_i$ avec s_k l'ensemble des n_k objets affectés au centre c_k.
(14) Retourner à la ligne (4) tant qu'il y'a des changements d'affectation d'objets à des clusters ou tant qu'un nombre d'itération maximal n'a pas été ateint.

ANNEXE B

Réseaux de neurones artificiels.

1. Structure d'interconnexion des RNAs:

Un réseau de neurone se compose de multiples neurones interconnectés, les structures qui peuvent être utilisées sont très variées. La structure la plus utilisée est la structure de réseau à couches. On distingue quatre types de connexion :

- **Réseau multicouche :** C'est un réseau à trois couches, couche d'entrée, couche cachée et une couche de sortie oŭ, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement :

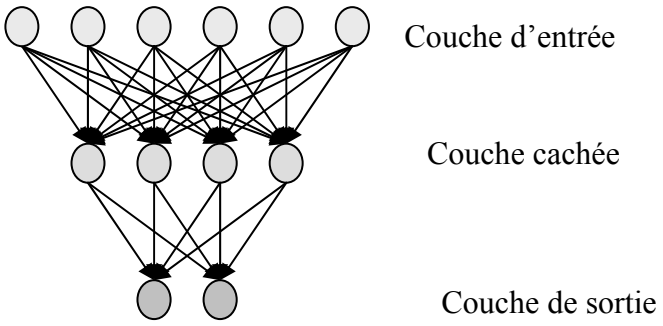

FIG A1.1 Réseau multicouche.

- **Réseau à connexions locales :** Il s'agit d'une structure multicouche, mais chaque neurone entretien des relations avec un nombre réduit et localisé de neurones de la couche avale. Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique.

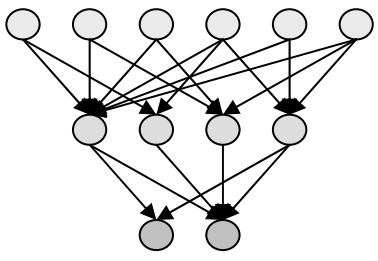

FIG A1.2 Réseau à connexion locale.

- **Réseau à connexion récurrente :** Les connexions récurrentes ramènent l'information en arrière par rapport au sens de la propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales .

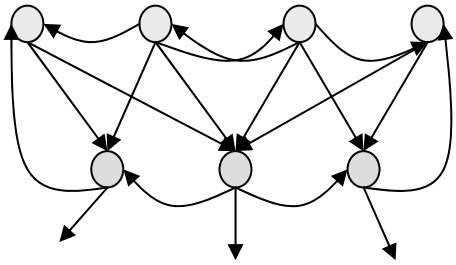

FIG A1.3 Réseau à connexions récurrentes.

- **Réseau à connexion complète :** C'est la structure d'interconnexion la plus générale dont chaque neurone est connecté à tous les neurones du réseau (et à lui-même).

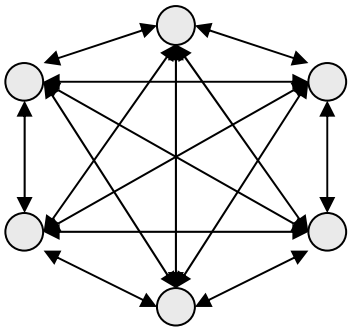

FIG A1.4 Réseau à connexion complète.

2. Mise en oeuvre des RNAs :

Les réseaux de neurones ont la capacité d'apprendre à partir d'exemples de manière assez analogue à celle des humains basée sur leur expérience [050]. La mise en oeuvre d'un réseau de neurones passe par trois étapes :

- **Codage des entrées** [050]

  Les réseaux de neurones fonctionnent mieux lorsque toutes les valeurs d'entrées se situent entre 0 et 1. Ceci demande une conversion de toutes les valeurs, continues ou énumératives, pour définir de nouvelles valeurs de cet intervalle. Pour normaliser les valeurs continues, la limite inférieure de l'intervalle est soustraite de la valeur à normaliser, et le résultat est divisé par la taille de l'intervalle. Par exemple, pour

normaliser la valeur ‘200’ qui se trouve dans l’intervalle [100][800], on fait l’opération suivante :

(200-100)/(800-100)=0,1428. Si nous ne connaissons pas la taille de l’intervalle, une estimation approximative est utilisée. Pour les valeurs énumératives, des fractions entre 0 et 1 sont affectées. Par exemple, on a une donnée qui prend les valeurs A, B ou C, on donne à A la valeur 0, à B la valeur 0,5 et à C la valeur 1.

La sortie est également un nombre entre 0 et 1, il faut donc dénormaliser la valeur.

- **Choix de l’architecture et réglage des paramètres**

  Dans cette phase, on choisit l’architecture adéquate et on fixe le nombre de couches et le nombre de neurones pour chaque couche après un certain nombre d’exemples et d’expériences.

- **Apprentissage**

  On essaye de régler les valeurs des poids des connexions avec des essais itératifs jusqu’à l’obtention des valeurs désirés.

3. Modèles des réseaux neuronaux :

- **Les réseaux à fonction radiale :**

Ce sont les réseaux que l’on nomme aussi RBF (pour « Radial Basic Functions »). L’architecture est la même que pour les PMC cependant, les fonctions de base utilisées ici sont des fonctions Gaussiennes. Les RBF seront donc employés pouyr résoudre le même type que les problèmes résolus par les PMC à savoir, en classification et en approximation de fonctions, particulièrement. L’apprentissage le plus utilisé pour les RBF est le mode hybride et les règles sont soit, la règle de correction de l’erreur, soit la règle d’apprentissage par compétition.

- **Les ARTs :** [089]

Les réseaux ART « Adaptatice Resonance Theory » sont des réseaux à apprentissage par compétition. L’étape de construction du réseau est une phase d’apprentissage qui consiste à opérer de multiples itérations avec des échantillons de départ pour apprendre les différents voisinages des couches cachées c’est-à-dire identifier les classe des données d’entrée [088].

Des tests vérifient qu’un voisinage existant est modifié seulement si l’individu courant en entrée est suffisamment similaire à l’individu moyen de ce voisinage. Si le vecteur d’entrée passe le test, le nœud le plus proche du voisinage est activé, ses poids sont mis à jour pour réduire le décalage, et l’individu moyen pour ce voisinage est mis à jour. Sinon le nœud le plus proche à l’extérieur du

voisinage est activé et est utilisé comme noyau de nouveau voisinage (création de classe). Le rapprochement entre un voisinage et un individu courant se fait grâce à une distance euclidienne.

Il existe deux principaux types de réseaux ART : les ART-1 pour des entrées binaires et les ART-2 pour des entrées continues. Le mode d'apprentissage des ART peut être supervisé ou non.

4. Règles d'apprentissage :

- **Règle de correction d'erreur :**

Cette règle s'inscrit dans le paradigme d'apprentissage supervisé. Si on considère Y la sortie calculée par le réseau et D la sortie désirée, le principe de cette règle est d'utiliser l'erreur (D-Y) afin de modifier les connexions et de diminuer ainsi l'erreur globale du système. Le réseau va donc s'adapter jusqu'à ce que Y soit égal à D.

- **Règle de Boltzman :**

Les réseaux de Boltzman sont des réseaux symétriques récurrents et possèdent deux sous groupes de cellules, le premier étant relié à l'environnement (cellules dites visibles) et le second ne l'étant pas (cellules dite cachées). Cette règle d'apprentissage est dite stochastique (qui relève partiellement du hasard) est consiste à ajuster les poids des connexions, de telle sorte que l'état des cellules visibles satisfasse une distribution probabiliste souhaitée.

- **La règle de Hebb :**

C'est la méthode d'apprentissage la plus ancienne (1949), elle est inspirée de la biologie. Le principe est de renforcer les connexions entre deux neurones lorsque ceux-ci sont actifs simultanément. Cette règle peut être classée comme apprentissage non supervisé, ou supervisé car on sait calculer directement les poids correspondant à l'apprentissage d'un certains nombres d'exemples.

- **Règle d'apprentissage par compétition :**

La particularité de cette règle est que l'apprentissage ne concerne qu'un seul neurone. Le principe est de regrouper les données en catégories. Les patrons similaires vont donc être rangés dans une même classe, en se basant sur les corrélations des données, est seront représentés par un seul neurone, on parle de « winner-take-all ».

Dans un réseau à compétition simple, chaque neurone de sortie est connecté aux neurone de la couche d'entrée, aux autres cellules de la couche de sortie (connexion inhibitrices) et à elle-même (connexion excitatrices) [015]. La sortie va donc dépendre de la compétition entre les connexions inhibitrices et excitatrices

Mode d'apprentissage	Règle d'apprentissage	Architecture	Algorithme	Tache
Supervisé	Corrélation d'erreur	Perceptron simple ou multicouches	Rétro propagation, adaline ,madaline	Classification, approximation de fonctions, prédiction, contrôle.
	Boltzman	Récurrente	Apprentissage de Boltzman,	Classification.
	Hebb	Multicouches non bouclées.	Analyse de descri_ minants linéaires	Analyse de données, classification.
	Par compétition	A compétition	LVQ	Catégorisation au sein d'une classe, compression de données
		ART	ARTMap	Classification, catégorisation au sein d'une classe.
Non supervisé	Correction d'erreur	Multicouches non bouclés	Projection de Sammon	Analyse de données.
	Hebb	A compétition	Analyse en composantes principales	Analyse de données, compression de données
	Par compétition	Carte de Kohonen	VQ	Catégorisation, compression de données.
		ART	SOM	Catégorisation, analyse de données
			ART-1, ART-2	Catégorisation.
Hybride	Correction d'erreur et par compétition	RBF	RBF	Classification, approximation de fonctions, prédiction, contrôle.

TAB A1.1Résumé des réseaux de neurones.

ANNEXE C
Aperçus des fichiers de la base KDD'99

1. Base d'apprentissage 10% de KDD :

Aperçus du fichier de la base d'apprentissage KDD'99 avant le prétraitement au format *txt* .

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0067)http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz -->
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META content="MSHTML 6.00.2900.5626" name=GENERATOR></HEAD>
<BODY><PRE>0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,
0,tcp,http,SF,239,486,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,19,19,1.00,0.00,0.05,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,235,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,29,29,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,219,1337,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,39,39,1.00,0.00,0.03,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,49,49,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,217,2032,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,59,59,1.00,0.00,0.02,0.00,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,212,1940,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,2,0.00,0.00,0.00,0.00,1.00,0.00,1.00,1,69,1.00,0.00,1.00,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,159,4087,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.00,0.00,0.00,0.00,1.00,0.00,0.00,11,79,1.00,0.00,0.09,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,210,151,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,89,1.00,0.00,0.12,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,212,786,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,8,99,1.00,0.00,0.12,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,210,624,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,18,18,0.00,0.00,0.00,0.00,1.00,0.00,0.00,18,108,1.00,0.00,0.06,0.05,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,177,1985,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,28,119,1.00,0.00,0.04,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,222,773,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,11,0.00,0.00,0.00,0.00,1.00,0.00,0.00,38,129,1.00,0.00,0.03,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,256,1169,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00,1.00,0.00,0.00,4,139,1.00,0.00,0.25,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,241,259,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,14,149,1.00,0.00,0.07,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,260,1837,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,11,0.00,0.00,0.00,0.00,1.00,0.00,0.00,24,159,1.00,0.00,0.04,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,241,261,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,34,169,1.00,0.00,0.03,0.04,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,257,818,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,12,12,0.00,0.00,0.00,0.00,1.00,0.00,0.00,44,179,1.00,0.00,0.02,0.03,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,233,255,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,8,0.00,0.00,0.00,0.00,1.00,0.00,0.25,54,189,1.00,0.00,0.02,0.03,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,233,504,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,7,7,0.00,0.00,0.00,0.00,1.00,0.00,0.00,64,199,1.00,0.00,0.02,0.03,0.00,0.00,0.00,0.00,normal.
0,tcp,http,SF,256,1273,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,17,17,0.00,0.00,0.00,0.00,1.00,0.00,0.00,74,209,1.00,0.00,0.01,0.03,0.00,0.00,0.00,0.00,normal.

2. Base d'apprentissage kddnormal :

Cette base est obtenue par le prétraitement de 10% de KDD'99 (élimination des attributs qualitatifs et des connexions redandantes)

```
0.0,181.0,5450.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,8.0,8.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,9.0,9.0,1.0,0.0,0.11,0.0,0.0,0.0,0.0,0.0,
0.0,239.0,486.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,8.0,8.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,19.0,19.0,1.0,0.0,0.05,0.0,0.0,0.0,0.0,0.0,
0.0,235.0,1337.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,8.0,8.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,29.0,29.0,1.0,0.0,0.03,0.0,0.0,0.0,0.0,0.0,
0.0,219.0,1337.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,6.0,6.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,39.0,39.0,1.0,0.0,0.03,0.0,0.0,0.0,0.0,0.0,
0.0,217.0,2032.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,6.0,6.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,49.0,49.0,1.0,0.0,0.02,0.0,0.0,0.0,0.0,0.0,
0.0,217.0,2032.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,6.0,6.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,59.0,59.0,1.0,0.0,0.02,0.0,0.0,0.0,0.0,0.0,
0.0,212.0,1940.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,2.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,1.0,69.0,1.0,0.0,1.0,0.04,0.0,0.0,0.0,0.0,
0.0,159.0,4087.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,5.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,11.0,79.0,1.0,0.0,0.09,0.04,0.0,0.0,0.0,0.0,
0.0,210.0,151.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,8.0,8.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,8.0,89.0,1.0,0.0,0.12,0.04,0.0,0.0,0.0,0.0,
0.0,212.0,786.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,8.0,8.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,8.0,99.0,1.0,0.0,0.12,0.05,0.0,0.0,0.0,0.0,
0.0,210.0,624.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,18.0,18.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,18.0,109.0,1.0,0.0,0.06,0.05,0.0,0.0,0.0,0.0,
0.0,177.0,1985.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,28.0,119.0,1.0,0.0,0.04,0.04,0.0,0.0,0.0,0.0,
0.0,222.0,773.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,11.0,11.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,38.0,129.0,1.0,0.0,0.03,0.04,0.0,0.0,0.0,0.0,
0.0,256.0,1169.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,4.0,4.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,4.0,139.0,1.0,0.0,0.25,0.04,0.0,0.0,0.0,0.0,
0.0,241.0,259.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,14.0,149.0,1.0,0.0,0.07,0.04,0.0,0.0,0.0,0.0,
0.0,260.0,1837.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,11.0,11.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,24.0,159.0,1.0,0.0,0.04,0.04,0.0,0.0,0.0,0.0,
0.0,241.0,261.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,2.0,2.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,34.0,169.0,1.0,0.0,0.03,0.04,0.0,0.0,0.0,0.0,
0.0,257.0,818.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,12.0,12.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,44.0,179.0,1.0,0.0,0.02,0.03,0.0,0.0,0.0,0.0,
0.0,233.0,255.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,2.0,8.0,0.0,0.0,0.0,0.0,1.0,0.0,0.25,54.0,189.0,1.0,0.0,0.02,0.03,0.0,0.0,0.0,0.0,
0.0,233.0,504.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,7.0,7.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,64.0,199.0,1.0,0.0,0.02,0.03,0.0,0.0,0.0,0.0,
0.0,256.0,1273.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,17.0,17.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,74.0,209.0,1.0,0.0,0.01,0.03,0.0,0.0,0.0,0.0,
0.0,234.0,255.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,5.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,84.0,219.0,1.0,0.0,0.01,0.03,0.0,0.0,0.0,0.0,
0.0,241.0,259.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,12.0,12.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,94.0,229.0,1.0,0.0,0.01,0.03,0.0,0.0,0.0,0.0,
0.0,239.0,968.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,3.0,3.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,3.0,239.0,1.0,0.0,0.33,0.03,0.0,0.0,0.0,0.0,
0.0,245.0,1919.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,13.0,13.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,13.0,249.0,1.0,0.0,0.08,0.03,0.0,0.0,0.0,0.0,
0.0,248.0,2129.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,23.0,23.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,23.0,255.0,1.0,0.0,0.04,0.03,0.0,0.0,0.0,0.0,
```

3. Base de test :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0070)http://kdd.ics.uci.edu/databases/kddcup99/kddcup.testdata.unlabeled.gz -->
<HTML><HEAD>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META content="MSHTML 6.00.2900.5626" name=GENERATOR></HEAD>
<BODY><PRE>0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,1,1.00,0.00,1.00,0.00,0.00,0.00,
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.01,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.01,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.01,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.01,0.00,0.00,0.00,0.00,0.00
0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,252,0.99,0.01,0.01,0.00,0.00,0.00,0.00,0.00
```

4. Exemples des valeurs extrêmes de la base de test :

0,315,1791,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,7,7,0.00,0.00,0.00,0.00,1.00,0.00,0.00,7,253,1.00,0.00,0.14,0.04,0.00,0.00,0.00,0.00,
0,280,13680,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,17,0.00,0.00,0.00,0.00,1.00,0.00,0.12,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0,312,1600,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,255,1.00,0.00,1.00,0.02,0.00,0.00,0.00,0.00,
0,308,261,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,11,11,0.00,0.00,0.00,0.00,1.00,0.00,0.00,11,255,1.00,0.00,0.09,0.02,0.00,0.00,0.00,0.00,
0,308,6511,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,21,21,0.00,0.00,0.00,0.00,1.00,0.00,0.00,21,255,1.00,0.00,0.05,0.02,0.00,0.00,0.00,0.00,
0,315,1009,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,6,6,0.00,0.00,0.00,0.00,1.00,0.00,0.00,6,255,1.00,0.00,0.17,0.02,0.00,0.00,0.00,0.00,
317**,62825648**,90476,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,152,1,0.01,0.05,0.01,0.00,0.00,0.00,0.00,0.0
0,297,714,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,4,4,0.00,0.00,0.00,0.00,1.00,0.00,0.00,94,255,1.00,0.00,0.01,0.04,0.00,0.00,0.00,0.00,
0,302,8505,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,13,13,0.00,0.00,0.00,0.00,1.00,0.00,0.00,103,255,1.00,0.00,0.01,0.04,0.00,0.00,0.00,0.00,
0,105,147,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,253,0.99,0.01,0.00,0.00,0.00,0.00,0.00,0.00,
0,308,4448,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,24,253,1.00,0.00,0.04,0.04,0.00,0.00,0.00,0.00,
0,318,2449,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,10,13,0.00,0.00,0.00,0.00,1.00,0.00,0.15,10,253,1.00,0.00,0.10,0.01,0.00,0.00,0.00,0.00,
0,238,9204,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0,324,1118,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,23,253,1.00,0.00,0.04,0.01,0.00,0.00,0.00,0.00,
0,308,1565,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,2,16,0.00,0.00,0.00,0.00,1.00,0.00,0.12,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,

0,257,3814,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,17,17,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,
0,740,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,171,34,0.20,0.03,0.20,0.00,0.00,0.00,0.00,0.00,
0,235,2237,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,7,7,0.00,0.00,0.00,0.00,1.00,0.00,0.00,25,255,1.00,0.00,0.04,0.02,0.00,0.00,0.00,0.00,
0,201,4841,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,5,0.00,0.00,0.00,0.00,1.00,0.00,0.60,182,255,1.00,0.00,0.01,0.04,0.00,0.00,0.00,0.00,
0,7012,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,5,0.00,0.00,0.00,0.00,1.00,0.00,0.00,180,43,0.24,0.03,0.24,0.00,0.00,0.00,0.00,0.00,
0,8173,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,15,15,0.00,0.00,0.00,0.00,1.00,0.00,0.00,190,53,0.28,0.03,0.28,0.00,0.00,0.00,0.00,0.00,
0,305,33229,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,2,4,0.00,0.00,0.00,0.00,1.00,0.00,0.75,2,255,1.00,0.00,0.50,0.05,0.00,0.00,0.00,0.00,
0,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,
8,319,**5203179,**0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
0,301,1450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,2,4,0.00,0.00,0.00,0.00,1.00,0.00,0.75,9,255,1.00,0.00,0.11,0.04,0.00,0.00,0.00,0.00,
0,297,625,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,5,13,0.00,0.00,0.00,0.00,1.00,0.00,0.31,10,252,1.00,0.00,0.10,0.06,0.00,0.00,0.00,0.00,
0,204,260,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,5,5,0.00,0.00,0.00,0.00,1.00,0.00,0.00,25,255,1.00,0.00,0.04,0.07,0.00,0.01,0.00,0.00,
0,209,846,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,15,15,0.00,0.00,0.00,0.00,1.00,0.00,0.00,35,255,1.00,0.00,0.03,0.06,0.00,0.01,0.00,0.00,
0,320,65034,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,5,22,0.00,0.00,0.00,0.00,1.00,0.00,0.09,202,255,1.00,0.00,0.00,0.01,0.00,0.00,0.00,0.00,
0,214,36065,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,3,255,1.00,0.00,0.33,0.03,0.00,0.00,0.00,0.00,
0,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,

ANNEXE D
Implémentation des méthodes de ANTCLass.

1. Ramassage d'un objet ou un tas d'objets :

```
static synchronized void ramasser() throws IOException
  { System.out.println("le tas trouvé par la fourmi "+ -num);
     Liste.afficher(Principal.Grille[ligne][colonne]);
   if (Principal.Grille[ligne][colonne].suivant.suivant==null) //le tas contient un seul objet
    {Pp=1;
      Principal.stabilité=false;
      tas = new Liste(Principal.Grille[ligne][colonne].suivant.contenu, tas);
     Principal.Grille[ligne][colonne]=Liste.supprimer
    (Principal.Grille[ligne][colonne].suivant.contenu,Principal.Grille[ligne][colonne]);  }
 else
{if (Principal.Grille[ligne][colonne].suivant.suivant.suivant==null) // le tas d'objets contient 2
                                                                        objet
{
min=Principal.dist_moyenne_entre_g_Tj(Principal.Grille[ligne][colonne].suivant)/dist_moyenne;
Pp=Math.min(Math.pow(min,1),1);
 if (Pp>=0.9)
{Principal.stabilité=false;
tas = new Liste(Principal.Grille[ligne][colonne].suivant.contenu, tas);
Principal.Grille[ligne][colonne]=Liste.supprimer(
                Principal.Grille[ligne][colonne].suivant.contenu,Principal.Grille[ligne][colonne]);
    } }
 else
{  double a=Principal.dist_moyenne_entre_g_Tj
                    (Principal.Grille[ligne][colonne].suivant)+Math.pow(10,-5);
   float[] X=new float [38];
   X=Principal.centre_de_gravité(Principal.Grille[ligne][colonne].suivant);

  double b=Principal.dist_max_entre_g_Tj(Principal.Grille[ligne][colonne].suivant,X)+
```

```
                                Math.pow(10,-5);
   Pp=1-min;
   if (Pp>=0.9)
   {Principal.stabilité=false;
     float [ ] objet=new float [38];
    int o;
   objet=Principal.chargement_connexion_dans_tableau(
                              Principal.Grille[ligne][colonne].suivant.contenu,"kddnormal.txt");
   double min=Principal.distance_entre_2_objets(X,objet);
   o=Principal.Grille[ligne][colonne].suivant.contenu;
   Liste courant=Principal.Grille[ligne][colonne].suivant.suivant;
   while(courant!=null)
  { double mmin;
mmin=Principal.distance_entre_2_objets(X,Principal.chargement_connexion_dans_tableau
                                    (courant.contenu,"kddnormal.txt"));
if (min>mmin) {min=mmin;o=courant.contenu;}
  courant=courant.suivant;   }
System.out.println("l'objet "+o+"   est le plus éloigné");
  tas = new Liste(o, tas);
Principal.Grille[ligne][colonne]=Liste.supprimer(o,Principal.Grille[ligne][colonne]);
                        while((Principal.Grille[ligne][colonne].suivant!=null)& (Pp>=0.9) )
                         Fourmi.ramasser();


        }   }      }
  System.out.print("Pp="+Pp +";");
}
```

2. Dépôt d'un objet ou un tas d'objets :

```
static synchronized void déposer ()  throws IOException
        {    if (tas.suivant != null) //la fourmi transporte plus qu'un objet
                {X = Principal.centre_de_gravité(tas);}
else // La fourmi transporte un seul objet
```

```
   X=Principal.tab[tas.contenu];
    double distance = Principal.distance_entre_2_objets(X, gravité_classe);
    // La distance maximale ente les objets de la classe et son centre de gravité
double dist_max = Principal.dist_max_entre_g_Tj(Principal.Grille[ligne][colonne].suivant,X);
if (distance <= dist_max) Pd
 else {
                    min = Math.pow(distance / dist_moyenne,1); //k2=2
                    Pd=1-(0.9* Math.min(1,min));}
                     System.out.println("*** Pd="+ Pd);
                     if (Pd==1)
                     { System.out.println("la fourmi "+ -num +"dépose selon la Pd=" + Pd);
while (tas!=null)
{Principal.Grille[ligne][colonne]=Principal.Grille[ligne][colonne].ajouter
            (tas.contenu,Principal.Grille[ligne][colonne]);
  tas=tas.supprimer(tas.contenu,tas);}
 }}
```

Bibliographie :

[001] : N. MONMARCHE, Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation, thèse de doctorat, Université de Tours, décembre 2000.

[002]: L. Portnoy, E. Eskin, S. Stolfo. Intrusion Detection With Unlabeled Data Using Clustering. In Proceedings of the ACM Workshop on Data Mining Allied to Security, 2001.

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

[004]: J.Anderson. Computer security threat monitoring and surveillance, 1980.

[005]: D E. Denning. An intrusion-detection model. IEEE transactions on software engeneering, SE-13 :222–232, 1987.

[006]: V.Estivill-castro and J-Yang. Fast and robust general purpose clustering algorithms. In pacific Rim international conference on artificial intelligence, 208-218, 2000.

[007] : L. Mé, Un complément à l'approche formelle : la détection d'intrusions.

[008]: J.Heer and E.Chi. Mining the structure of user activity using cluster stability. In proceedings of the workshop on web analytics, SIAM conference on Data Mining, April 2002.

[009]: P.Hansen and N.Mladenovic. J-Means: A new local search heuristic for minimum sum-of-squares clustering. Pattern recognition, 34(2):405-413, 2001.

[010]: M. Su and C. Chou. A modified version of the K-Means algorithm with a distance based on cluster symmetry. IEEE Transactions on pattern analysis and machine intelligence, 23(6): 674-680, 2001.

[011]: B. Zhang and M. Msu. K-Harmonic Means-a data clustering algorithm. Technical report HPL 1999-124, Hewlett-Packard Labs, 1999.

[012]: Bertrand Le Saux. Classification non exclusive et personnalisation par apprentissage: application à la navigation dans les bases d'images. 2003.

[013]: Y. Chiou and L. lan. Genetic clustering algorithms. European journal of operational research, (135): 413-427, 2001.

[014]: K. Sastry, H. A. Abbass, and D. E. Goldberg. Sub-Structural Niching in Non–Stationary Environments. In *Australian Artificial Intelligence Conference*,pages 873–885, 2005.

[015]: Jian_kang wu, Neural networks and simulation methods, editeur CRC Press, décembre 1993.

[016]: R. Greiner and R. Isukapalli. Learning to select useful landmarks. IEEE transactions an systems, Man, and Cybernetics-part B, special issue on learning autonomous robots, 1996.

[017]: W. S. Hu and H. Zhan. A Study of a Clustering Based Intrusion Detection Algorithm. Postgraduate Journal, Wuhan University, 21(3), 2004.

[018]: J. Oldmeadow, S. Ravinutala, and C. Leckie. Adaptive Clustering for Network Intrusion Detection. In Proceedings of the 3rd International Pacific Asia Conference on Knowledge Discovery and Data Mining, 2004.

[019]: D. Endler. Intrusion Detection: Applying Machine Learning to Solaris Audit Data. In Proceedings of the Annual Computer Security Applications conference, pages 267–279, 1998.

[020]: M. Taniguchi, M. Haft, J. Hollmn, and V.Tresp. Fraud Detection in Communications Networks Using Neural and Probabilistic Methods. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pages 1241–1244, Seattle, USA, May 1998.

[021]: X. Zhuang, T. Zhang, and S. Pande. HIDE: an infrastructure for efficiently protecting information leakage on the address bus. In ASPLOS-XI: Proceedings of the 11th international conference on Architectural support for programming languages and operating systems, pages 72–84, New York, NY,USA, 2004. ACM Press. 2463–2467. Xi'an, 2003.

[022]: Z.S. Pan, S.C. Chen, G. Bao Hu, and D.Q. Zhang. Hybrid Neural Network and C4.5 forMisuse Detection. In Machine Learning and Cybernetics.

[023]: E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. In Data Mining for Security Applications. Kluwer, 2002.

[024] : T. ABBES, Classification du trafic et optimisation des règles de filtrage pour la détection d'intrusions,Thèse de doctorat, Université Henri Poincaré – Nancy 1,Décembre 2004.

[025]: A. J. Hoglund and K. Hatonen. Computer Network User Behaviour Visualisation Using Self Organising Maps. In Proceedings of the 8th International Conference on Artificial Neural Networks, pages 899–904, 1998.

[026]: T. F Lunt. A Survey of Intrusion Detection Techniques. Computers & Security,12(4):405_ 418, 1993.

[027]: http://swatch.sourceforge.net/

[028]: http://www.tripwire.com/

[029]: http://www.cisco.com;

[030]: http://www.nfr.net;

[031]: http://www.snort.org;

[032]: http://all.net/dtk/dtk.html;

[033]: http://www.iss.net/products_services/enterprise_protection/rsnetwork/sensor.php

[034]: http://www.netiq.com/products/sm/default.asp

[035] : Guillaume Desgeorge, La sécurité des réseaux, 2000.

[036]: A. Sundaram. An introduction to intrusion detection. Crossroads, 2(4) :3_7, 1996.

[037]: H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. Comput. Networks, 31(9):805_822, 1999.

[038]: H. Debar, M. Dacier, and A. Wespi. A revised taxonomy for intrusion-detection systems. Annales des tlécommunications, 55(7_8) :361_378, 2000.

[039]: S. Axelsson. Intrusion detection systems : A survey and taxonomy. Technical Report 99-15,Chalmers Univ, 2000.

[040]: R. Bace and P. Mell. Intrusion detection systems. Special Publication 8000631, National Institute of Sta, 2001.

[041]: R. W. Shirey, IETF RFC 2828, Internet Security Glossary, 2000.
URL http://www.itef-editor.org

[042]: N. Dagorn, Détection et prévention d'intrusion : présentation et limites.

[043] : C. Bontemps, principes mathématiques et utilisation des algorithmes génétiques, 1995.

[044]: J.H Holland, Adaptation in Natural and Artificial Systems, University of Michigan press, 1975.

[045]: D. Goldberg, Genetic Algorithms, Addison Wesley editor, 1989.

[046]: F.Z.KETTAF, présentation des algorithmes génétiques, université FRANCOIS RABELAIS, rapport interne n°146.

[047] : L. Mé. Audit de Sécurité par algorithmes génétiques, PhD thesis, Université de Rennes 1,1994.

[048]: Y. Chen, A. Abraham, and J. Yang. Feature Selection and Intrusion Detection Using Hybrid Flexible Neural Tree. In International Syumposium on Neural Networks (ISNN 03), pages 439–444, Chongqing, China, 2005.

[049]: CAMILLE SERRUYS, Classification automatique des tumeurs noires de la peau par des techniques numériques d'analyse d'images fondées sur des méthodes d'apprentissage par l'exemple : Aide au dépistage des mélanomes. Thèse de doctorat, Université Paris V, décembre 2002.

[050]: J. Michael. A. Berry, Gordon Linoff, « Data Mining, Techniques appliqué au marketing, à la vente et aux services clients », Edition INTEREDITIONS Paris 1997.

[051]: K. L. Fox, R. R. Henning, J. H. Reed, and R. Simonian. A Neural Network Approach Towards Intrusion Detection. In Proceedings of the 13th National Computer Security Conference, pages 125_134, Washington, DC, Octobre 1990.

[052]: H. Debar, M. Becker, and D. Siboni. A Neural Network Component for an Intrusion Detection System. In Proceedings of the 1992 IEEE Computer Society Symposium on Reserach in Security and Privacy,IEEE Service Center, Piscataway,NJ, 1992.

[053]: L. J. Ryan and R. Miikkulainen. Intrusion detection with neural networks. In Advances in Neural Information Processing Systems, volume 10, pages 943_949, Cambridge,1998. MA : MIT Press.

[054]: K. Tan. The application of neural networks to unix computer security. In Proceedings of the IEEE International Conference on Neural Networks, volume 1, 1995.

[055]: A. K. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior pro_les for intrusion detection. In Proceedings of the Workshop on Intrusion Detection and Network Monitoring, pages 51_62. USENIX Association, 1999.

[056]: P. Lichodzijewski, A. Zincir-Heywood, and M. I. Heywood.Host-based intrusion detection using self-organizing maps. In The IEEE World Congress on Computational Intelligence, International Joint Conference on Neural Networks, IJCNN 02, 2002.

[057]: A. Zincir-Heywood P. Lichodzijewski and M. I. Heywood. Dynamic intrusion detection using self-organizing maps. In The 14th Annual Canadian Information Technology Security Symposium (CITSS), 2002.

[058]: M. Ramadas, S. Ostermann, and B. Tjaden. Detecting anomalous network traffic with self organizing maps. In Proceedings of the 6th International Workshop on the Recent Advances in Intrusion Detection (RAID'2003), LNCSv. 2820, pages 36_54, Novembre 2003.

[059]: T. Lunt. Ides: An intelligent system for detecting Intruders. In Computer Security,Threats and Countermeasures. Novembre 1990.

[060]: H. Debar. Application des réseaux de neurones à la détection d'intrusions sur les systèmes informatiques. PhD thesis, Université de Paris 6, 1993.

[061]: http://www..sylbarth.com/nn.php

[062] : N. Labroche . Modélisation du système de reconnaissance chimique des fourmis pour le problème de la classification non_supervisés : application à la mesure d'audience sur Intenet. Décembre 2003.

[063]: W. Jing-Xin, W. Zhi-Ying, and D. Kui. A Network Intrusion Detection System Based on Artificial Neural Networks. In Proceedings of the 3rd ACM International Conference on Information Security, volume 85, pages 166–170, Shanghai, China, 2004.

[064]: J. Cannady. Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks. In Proceedings of the 23rd National Information Systems Secuity Conference, 2000.

[065]: A. Ghosh and A. Schwartzbard. A Study in Using Neural Networks for Anomaly and Misuse Detection. In Proceedings of the 8th USENIX Security Symposium, 1999.

[066] : E. Leon, O. Nasaoui, and J. Gomez. Anomaly Detection Based on Unsupervised Niche Clustering with Application to Network Intrusion Detection. In *Proceedings of the congress of evolutionary Computation*, 2004.

[067]: Jain, A. and Dubes, R. Algorithms for Clustering Data. Prentice Hall Advanced Reference Series.1988

[068]: K. Leung and C. Leckie. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In Proceedings of the 28th Australasian Computer Science Conference, pages 333–342, Newcastle, Australia, 2004.

[069]: L. Mé. Véronique Alanou. Jorg abraham. Utilisation de carte de Kohonen pour détecter des intrusions dans un système informatique:une pré_étude.

[070]: C. Elkan, Results of the KDD'99 Classifier Learning, In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA 1(2), pp.63-64, 2000.

[071]: J. R. Quinlan, C4.5 : Programs for Machine Learning, Morgan Kaufmann Pub., San Mateo,CA, 1993.

[072]: C. Sinclair, L. Pierce, and S. Matzner. An Application of Machine Learning to Network Intrusion Detection. In Proceedings of the Annual Computer Security Applications Conference, pages 371–377, Phoenix, 1999.

[073]: W. Li. Using Genetic Algorithms for Network Intrusion Detection. In Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference, Kansas City, USA, May 2004.

[074]: M.M. Pillai, J.H.P. Eloff, and H.S.Venter. An Approach to Implementing a network Intrusion Detection System using Genetic Algorithms. In Proceedings of the Annual south African Institute of Computer Scientists and Information Technologists, Stellenbosch, SA, 2004.

[075]: B. Balajinath and S. V. Raghavan. Intrusion Detection Through Learning Behaviour Model. International Journal of Computer Communications, 24(12):1202–1212, July 2001.

[076]: F.A. Gonzlez, J. Gmez, M. Kaniganti, and D. Dasgupta. An Evolutionary Approach to Generate Fuzzy Anomaly Signatures. In IEEE Information Assurance workshop, pages 251–259, West Point, NY, 2003

[077]: W. H. Au, K. C. C. Chan, and X. Yao. A Novel Evolutionary Data Mining Algorithm with Application to Churn Prediction. IEEE Transactions on Evolutionary Computation, 7(6), 2003.

[078]: M. S. Abadeh, J. Habibi, and S. Aliari. Using a Particle Swarm Optimization Approach for Evolutionary Fuzzy Rule Learning: A Case Study of Intrusion Detection. In Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU), Paris, France, July 2–7,2006.

[079]: N. Friedman, D. Geiger & M. Goldszmidt, Bayesian Network Classifiers, In Machine Learning, vol. 29, pp. 131-161,1997.

[080]: S. Benferhat, N. Ben Amor & Z. Elouedi, Naïve Bayes vs Decision Trees in Intrusion Detection Systems, In Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus, pp.420-424, 2004.

[081] : F. Kettaf. Contributions des algorithmes évolutionnaires au partitionnement des données. PhD thesis, Universite de Tours.1997

[082]: Cucchiara, R. Analysis and comparison of different genetic models for the clustering problem in image analysis . In Albrecht, R., Reeves, C., and Steele, N., editors, International Conference on Artificial Neural Networks and Genetic Algorithms, pages 423–427. Springer-Verlag.1993

[083]: Falkenauer, E. A new representation and operators for genetic algorithms applied to grouping problems . Evolutionary Computation, 2(2) :123–144.1994

[084]: D. Jones and M. Beltrano. Solving partitioning problems with genetic algorithms . In (Belew and Booker, 1991), pages 442–449.

[085]: von Laszewski, G. Intelligent Structural Operators for the k-Way Graph Partioning Problem . In (Belew and Booker, 1991), pages 45–52.

[086] : UCI KDD Archive, KDD 1999 data set, 1999.
URL http: //kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[087] : S. Thiria, Y. Lechevalier, O. Gascuel, et S Canu Statistique et méthodes neuronales. Éditions Dunod, Paris, 336 p.1997

[088]: Carpenter G.A, Distributed learning, recorgnition and prediction by ART and ARTMAP neural networks. Neural networks, vol(10) n°8, 1473_1494, 1997.

[089] : N Turenne, Apprentissage statistique pour l'extraction de concepts à partir de textes : application au filtrage d'informations textuelles, Thèse de doctorat, Université Louis-Pasteur Strasbourg, Novembre 2000.

[090]: P. Lingras, Classifying highways: hierarchical grouping vs Kohonen neural networks, technical report, Algoma university college SoltMarys (Canada), 1994.

[091]: T. Kohonen , Self organisation and associative memory, ed. Springer Verlag, 1989.

[092]: D. G. Yang, C. Y. Hu, and Y. H. Chen. A Framework of Cooperating Intrusion Detection Based on Clustering Analysis and Expert System. In Proceedings of the International Conference on Information Security, Shanghai, China,November 2004.

[093]: J. Marin, D. Ragsdale, and J. sirdu. A Hybrid Approach to Profile Creation and Intrusion Detection. In Proceedings of the DARPA Information Survivability Conference and Exposition, pages 69–76, Los Alamitos, USA, 2001.

[094]: H. S. Javitz and A. Vadles. The NIDES Statistical Component: Description and Justification. SRI International Computer Science Laboratory Technical Report CSL-93-12, 1993.

[095]: V. Paxson, Bro: A System for Detecting Network Intruders in Real-Time, In Proceedings of 7th USENIX Security Symposium, 1998.

[096]: S. M. Bellovin, Packets Found on an Internet, In Computer Communications Review, 23(3)pp. 26-31, 1993.
URL http://www.research.att.com/~smb/papers/packets.ps

[097]: E. Soroush, M. S. Abadeh, and J. Habibi. Intrusion Detection Using a Boosting Ant Colony Based Data Miner. In Proceedings of the 11th International CSI Computer Conference, pages 563–566, 2006.

[098]: O. Depren, M. Topallar, E. Anarim, and M.K. Ciliz. An Intelligent Intrusion Detection System for Anomaly and Misuse Detection in Computer Networks. Expert systems with Applications, 29:713–722, 2005.

[099]: J. Z. Lei and A. A. Ghorbani. Network Intrusion Detection Using an Improved Competitive learning neural Network. In Conference on Communication networks and Services Research, pages 190–197, Fredericton, Canada, 2004.