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

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Systèmes d'informations.

Sujet :

Utilisation des colonies de fourmis artificielles pour la résolution du problème de classification.

Présenté par : - FERHAT Taha.
- HENTABLI M^ed Amine.

Promoteur : M^er KOUDIL Mouloud.
Co-Promotrice : M^lle BENATCHBA.

Organisme d'accueil : Institut National d'Informatique - Oued Smar-Alger.

Soutenue le: 23/10/2007, devant le jury composé de :

M^me ABED	**Présidente**
M^mc OUAHRANI	**Examinatrice**
M^er HADJ YAHIA	**Examinateur**

- 2006/2007-

Remerciements

Nous exprimons nos vifs remerciements à notre promoteur, Monsieur KOUDIL, pour ses remarques pertinentes, sa gentillesse et sa simplicité. Sa méthode de travail nous a poussée à donner le meilleur de nous même.

Nous adressons nos vifs remerciements à notre Co-promotrice, Mademoiselle BENATCHBA, pour sa disponibilité et son suivi fructueux. Ses conseils constants nous ont été une aide inestimable et ont largement contribué à notre formation.

Nous tenons à remercier les membres du jury pour l'honneur qu'ils nous ont fait de bien vouloir juger notre travail.

Enfin, nos remerciements à toute personne ayant contribué de prés ou de loin pour la réalisation de ce modeste travail.

Dédicaces

Je dédie ce modeste travail à tous ceux que j'aime ...

A la famille REZIG et à la famille FERHAT,
Entre vous, j'ai grandi entre de bonnes mains,
Mes succès sont à vous.

A tous mes amis de l'université de BLIDA
Avec vous j'ai passé des moments agréables,

A tous mes amis de BOUFARIK,
Avec vous, j'ai goûté l'amitié,

A mon grand père et ma grande mère,
A mon frère OUASSIM, à qui je dois beaucoup,
A ma mère, à qui je dois TOUT,
...que Dieu vous garde pour moi.

Taha.

Je dédie ce modeste travail à tous ceux que j'aime . . . sans vous, ma vie n'aura aucun sens.

A mon père et ma mère, à qui je dois TOUT,
A ma sœur SOUMAYA, qui est toujours dans mon cœur,
A mes frères ABDERRAHMANE, CHOUAIB, à qui je dois beaucoup,
. . . que Dieu vous garde pour moi.

A la famille HENTABLI, la famille NAIMI,
Entre vous, j'ai grandi entre de bonnes mains,
Mes succès sont à vous.

A tous mes amis de l'Université de BLIDA,
Avec qui j'ai passé des moments agréables,
Leurs souvenirs sont éternels,

A mes chères amis de BLIDA,
Avec vous, j'ai passé des moments inoubliables,
Merci pour tout.

Amine.

Résumé

Aujourd'hui, des milliards de données sont collectées chaque jour dans le monde. Néanmoins, si l'essor de l'informatique a rendu possible le stockage de volumes de données toujours plus importants, les entreprises étaient jusqu'alors incapables de transformer leurs données en connaissance directement utilisable.

Pour générer ces informations, un ensemble d'architectures, de démarches et d'outils ont été regroupés sous le terme de **Data Mining**. Le Data Mining consiste essentiellement à extraire de l'information de gigantesques bases de données de la manière la plus automatisée possible.

La classification de données est identifiée comme une des problématiques majeures en extraction des connaissances à partir de données. Dans ce domaine, les fourmis suggèrent des heuristiques très intéressantes pour les informaticiens.

Notre objectif est d'implémenter une nouvelle méthode de classification non supervisée basée sur les fourmis artificielles et l'intégrer dans un environnement complet que nous devons mettre en oeuvre.

Mots clés:

Data Minig, Data Warehouse, classification non-supervisée, fourmis artificielles,

Abstract

Today, billions of data are collected each day all around the world. Nevertheless, even though computer science has made it possible to store huge volumes of data, companies were still unable to transform their data into knowledge directly usable.

To generate this information, a whole of architectures, steps and tools were gathered under the term of Date Mining. Data Mining primarily consisted in extracting gigantic data bases from the available information, using the most automated way.

Clustering is identified as one of the major problems in extracting knowledge from data. In this field, the ants suggest very interesting heuristics for the data processing specialists.

Our objective is to implement a new method based on the artificial ants to solve unsupervised clustering problem and integrate it in an environment that we have to implement.

Keywords:

Data Mining, Data Warehouse, clustering, artificial ants.

Sommaire

Liste des figures

Liste des tableaux

Introduction Générale

L'accroissement de la concurrence, l'individualisation des consommateurs et la brièveté du cycle de vie des produits obligent les entreprises à non plus simplement réagir au marché mais à l'anticiper. Elles doivent également cibler au mieux leur clientèle afin de répondre à ses attentes. La connaissance de son métier, des schémas de comportement de ses clients, de ses fournisseurs est essentielle à la survie de l'entreprise, car elle lui permet d'anticiper sur l'avenir.

Aujourd'hui, les entreprises ont à leur disposition une masse de données importante. En effet, les faibles coûts des machines en terme de stockage et de puissance ont encouragé les sociétés à accumuler toujours plus d'informations. Cependant, alors que la quantité de données à traiter augmente énormément le volume d'informations fournies aux utilisateurs n'augmente lui que très peu. Ces réservoirs de connaissance doivent être explorés afin d'en comprendre le sens et de déceler les relations entre données, des modèles expliquant leur comportement ainsi le rendement de l'entreprise sera plus élevé que la concurrence.

Ce rendement est, souvent, le résultat d'une meilleure technologie de l'information fournissant la base pour des décisions judicieuses et pertinentes. Les systèmes d'information se sont développés pour contribuer à améliorer la productivité des traitements. Ils ont, dans un premier temps, été conçus essentiellement pour collecter des données et y appliquer des traitements de masse dans un souci d'automatisation des tâches répétitives. Depuis deux décennies environ, l'attention des entreprises s'est progressivement détournée des systèmes opérationnels pour se porter sur des systèmes décisionnels, qui contribuent véritablement à la différenciation stratégique de l'entreprise.

Des outils de décision apparaissent alors regroupés sous le terme "*Data Mining*" permettant d'extraire de l'information pertinente afin de prendre les bonnes décisions.

Dans ce problème de fouille de données, une des activités concerne la classification qui est un problème de partitionnement et qui est reconnu NP-Difficile. Par conséquent, il ne peut être résolu par des approches exhaustives dès que la taille du problème à résoudre devient importante. Le recours à des méthodes heuristiques devient alors le seul recours. Cependant, ce genre de méthode ne permet pas de

trouver avec certitude la solution optimale au problème. C'est pourquoi de nombreuses méthodes ont vu le jour pour tenter de résoudre le problème qui nous intéresse. Les méthodes inspirées de la biologie ont de plus en plus de succès car elles ont donné des résultats intéressants dans de nombreux cas. Elles sont utilisées avec succès dans la classification depuis quelques années. Cependant, tout comme l'ensemble des heuristiques, aucune ne donne de bons résultats dans toutes les instances du problème. Cela signifie qu'elles peuvent se révéler très intéressantes dans certains exemples de problèmes de classification, et beaucoup moins pour d'autres. C'est pourquoi il nous a été demandé de développer et de mettre en œuvre une nouvelle méthode inspirée du monde des vivants, et en particulier des colonies de fourmis. Nous proposons donc un algorithme de résolution du problème de classification inspiré d'un certain type de fourmis naturelles. L'approche développée est intégrée dans un environnement qui permet à l'utilisateur de tester la méthode sur des benchmarks supervisés à travers un certain nombre d'outils graphiques et statistiques que nous avons mis en œuvre dans notre environnement.

Ce document est composé de cinq chapitres. Le premier chapitre portera sur la définition, les tâches, les techniques du Data Mining. Le deuxième chapitre sera dédié à la classification, sa définition, ses types et ses méthodes de résolution. Dans le troisième, nous abordons les fourmis artificielles et les différents algorithmes de classification par colonie de fourmis. La conception et la mise en œuvre feront l'objet du quatrième chapitre. Enfin, nous présenterons dans le chapitre cinq quelques tests et résultats.

Chapitre I: Le Data Mining

1. Introduction

Le Data Mining est apparu au début des années 90. Cette émergence n'est pas le fruit du hasard, mais le résultat de la combinaison de nombreux facteurs technologiques, économiques et socio-politiques.

On peut voir le Data Mining comme une nécessité imposée par le besoin des entreprises de valoriser les données qu'elles accumulent dans leurs bases de données. En effet, le développement des capacités de stockage et des vitesses de transmission des réseaux ont conduit les utilisateurs à accumuler de plus en plus de données. Certains experts estiment que le volume des données double tous les ans [WIK 07].

2. Processus d'extraction des données

On utilise le terme *knowledge discovery in databases* (KDD) pour définir le processus d'extraction de la connaissance à partir des données et cela depuis 1989 [POL 00]. Le KDD est le fruit de travaux menés par des chercheurs en statistiques, en intelligence artificielle, en reconnaissance de formes, en bases de données et en visualisation [BIA 01].

2.1. Définition

Plusieurs définitions ont été attribuées au KDD, citons parmi elles:

- Extraction à partir de grands volumes de données de nouvelles unités de connaissances, significatives et réutilisables. Ces unités de connaissances sont sous plusieurs formes, par exemple: classes d'individus, motifs (ensemble de propriétés), règles d'associations et dépendances fonctionnelles [NAP 06].

- Automatisation du processus de découverte de connaissances pertinentes, nouvelles et utiles dans les très grandes quantités d'information [BIA 01].

- La découverte de nouvelles corrélations, tendances et modèles par le tamisage d'un large volume de données [LEN XX].

- Le traitement automatique de grandes quantités de données brutes (raw data), en extrayant les connaissances les plus significatives et en les représentant sous formes appropriées à l'utilisateur [PER 96].

- La recherche de connaissances utiles cachées parmi la vaste quantité de données dans une grande base de données [PER 96].

2.2. Les étapes du processus d'extraction de données KDD

La principales étapes du KDD peuvent être résumées dans ce qui suit.

2.2.1. Formulation du problème

Dans cette phase, on doit définir avec précision les objectifs et les résultats attendus, et ce qu'on souhaite faire avec les connaissances obtenues. On doit également connaître la nature des personnes qui utiliseront ces connaissances afin de choisir la technique la mieux adaptée pour résoudre ce problème.

2.2.2. Acquisition des données

Dans cette phase, on sélectionne les données généralement situées dans des bases de données, des entrepôts de données (Data Warehouses) ou sur le web et qui ont un rapport avec le problème posé. Cette phase nécessite dans la plupart des cas la présence d'un expert pour déterminer les attributs les plus significatifs.

2.2.3. Prétraitement des données

Dans la plupart des cas, les données acquises ne sont pas toutes exploitables par des techniques de fouille de données. En effet, la plupart des techniques utilisées ne traitent que des tableaux de données numériques rangées sous forme lignes/colonnes. Certaines méthodes sont plus contraignantes que d'autres. Elles peuvent par exemple exiger des données binaires. Ainsi, les données acquises peuvent être de types différents. On peut y trouver des textes de longueurs variables, des images, des enregistrements quantitatifs ou des séquences vidéo [ZIG 03].

2.2.3.1 La sélection des lignes/colonnes

Cette sélection s'applique sur les données qui sont déjà sous forme tabulaire. Par la suite, on essaie de définir un filtre qui permet de sélectionner un sous-ensemble de lignes ou de colonnes. Le but est de réduire le nombre de données ou bien de sélectionner des lignes ou colonnes les plus pertinentes. Les méthodes utilisées relèvent généralement des méthodes statistiques d'échantillonnage. Cette sélection peut aussi s'effectuer selon des conditions fixées par l'utilisateur (exemple: il ne veut

garder que les attributs dont la moyenne est supérieure ou inférieure à un seuil donné). La sélection des attributs est en train de devenir l'un des sujets importants de la recherche en Data Mining [ZIG 03].

2.2.3.2 *Le nettoyage des données*

On a recours à cette opérations afin d'éliminer les données incohérentes (erreurs de saisie, champs nuls, valeurs aberrantes...). Pour cela, on effectue différents traitements.

- Traitement des données aberrantes: ce sont les données qui ont des valeurs anormales par rapport à leurs natures (leurs types). Pour les traiter, il faut d'abord repérer ces dernières au moyen d'une règle préétablie. Par exemple: toutes les données numériques dont la valeur sur un attribut donné s'écarte de la valeur moyenne (plus deux fois l'écart type) pourraient être considérées comme des données probablement aberrantes et qu'il conviendrait de traiter [ZIG 03].

- Traitement des données manquantes: ce sont les champs qui n'ont aucune valeur. On peut procéder de plusieurs manières pour régler ce genre de problèmes:

  * *Exclure les enregistrements incomplets* [SEK 06]: On élimine les enregistrements ayant une valeur manquante. L'inconvénient est que l'on réduit la base de données.

  * *Remplacer les données manquantes* [SEK 06]: On peut remplacer les champs manquants par la valeur la plus fréquente, par la moyenne des valeurs prises par ce champ, par l'estimation ou bien par une méthode d'induction (régression, réseau de neurones et graphes d'induction).

2.2.3.3 *La transformation des attributs*

Il s'agit de transformer un attribut A par exemple en un attribut A' qui serait plus approprié pour l'étude et pour qu'il soit exploitable par les techniques du Data Mining. Pour cela, plusieurs techniques sont utilisées comme la discrétisation qui transforme les attributs continus en découpant le domaine des valeurs de ces attributs en intervalles, afin d'obtenir des attributs qualitatifs. Il est également possible de centrer par rapport à la moyenne et réduire par l'écart type les valeurs des variables continues [ZIG 03].

2.2.4. Fouille de données

La fouille de données concerne le Data Mining dans son sens restreint et est au cœur du processus d'ECD. Elle consiste à extraire les connaissances utiles à partir d'un large volume de données et à les représenter sous forme synthétique. La fouille de données fait appel à plusieurs méthodes issues de la statistique, de l'apprentissage automatique, de la reconnaissance de formes ou de la visualisation [ZIG 03].

2.2.5. Validation des résultats

On essaie dans cette phase d'évaluer et de valider les modèles extraits car ces derniers ne peuvent pas être utilisés directement en toute fiabilité. Il est alors nécessaire de leur faire passer une série de tests, afin de vérifier leurs performances et leur justesse [ZIG 03]. On choisit la technique de validation selon la nature du problème. Il existe deux types de techniques de validation: la validation statistique et la validation par expertise.

On utilise la validation par expertise pour les problèmes de clustering et d'association. La validation est du ressort de l'expert qui estime la pertinence des clusters formés ou des règles d'association constituées [MAZ 04].

La validation statistique dont le but est d'évaluer la capacité de généralisation du modèle. Elle est utilisée pour les problèmes de classification et d'estimation. On peut estimer la qualité d'ajustement du modèle sur l'échantillon d'apprentissage. Un taux d'erreur est calculé sur cet échantillon d'apprentissage. Afin d'appréhender au mieux le taux d'erreur réel, on se sert des techniques suivantes: la validation croisée ou le bootstrap [MAZ 04].

- validation croisée: La validation croisée ou cross validation consiste à répartir l'échantillon d'apprentissage aléatoirement en K paquets d'effectifs identiques. Par la suite, on calcule le taux d'erreur de chaque paquet. Ainsi, le taux d'erreur en validation croisée équivaut à la moyenne arithmétique des taux d'erreurs partiels. La validation croisée est conceptuellement simple, efficace et largement utilisée pour estimer l'erreur. Cependant, elle implique un surplus de calculs [ZIG 03].

- Le bootstrap: Il a été conçu par Bradly Efron (1982). Dans cette méthode, l'échantillon est tiré aléatoirement et avec remise. Par analogie à la validation croisée, l'erreur du modèle est estimée par l'erreur moyenne réalisée sur les différents échantillons de validation [ZIG 03].

2.2.6. Intégration du résultat

On implante les résultats dans les systèmes informatiques ou dans le processus de l'entreprise. C'est l'étape de la transition du domaine des études au domaine opérationnel [MAZ 04].

3. Définition du Data Mining

Le Data Mining est apparu au début des années 1990. Plusieurs définitions lui ont été attribuées. Dans ce qui suit, nous en citons quelques-unes:

- De manière générale, on peut le définir comme l'extraction d'informations ou de connaissances originales, auparavant inconnues, potentiellement utiles à partir de gros volumes de données (d'après Frawley et Piateski-Shapiro) [DAT 06].

- Le "Data Mining" (littéralement: "forage de données", ou "exploitation stratégique des données") peut se définir comme "l'exploration et l'analyse de grandes quantités de données, afin de découvrir des formes et des règles significatives, en utilisant des moyens automatiques ou semi-automatiques" (M. Berry et G. Linoff) [CEL 04].

- Il s'agit du processus de sélection, d'exploration, d'extraction et de modélisation de grandes bases de données, afin de découvrir les relations entre des données jusqu'alors inconnues.

- Le Data Mining est un terme générique englobant toute une famille d'outils facilitant l'analyse des données contenues au sein d'une base de type Data Warehouse. Le Data Mining présente l'avantage de trouver des corrélations informelles entre les données. Il permet de mieux comprendre les liens entre des phénomènes en apparence distincts et d'anticiper des tendances encore peu discernables [PIL 07].

- Un processus non trivial d'extraction de modèles valides, nouveaux, potentiellement utiles et compréhensibles à partir de données [PIL 07].

- Le terme "Data Mining" signifie littéralement "forage de données". Son but est de pouvoir extraire un élément: la connaissance. Ces concepts s'appuient sur le constat qu'il existe au sein de chaque entreprise des informations cachées dans le gisement de données. Ils permettent, grâce à un certain nombre de techniques spécifiques, de faire apparaître des connaissances [NAK 98].

4. Utilité du Data-Mining

Aujourd'hui, les bases de données deviennent de plus en plus volumineuses puisque leurs tailles atteignent des volumes de plusieurs tera-octets (1 tera-octet = 10^{12} octets) [ZIG 03]. Les grandes compagnies collectent annuellement plusieurs tera-octets de données relatives aux consommations de leurs clients. Entre ces compagnies, la concurrence devient de plus en plus rude. La connaissance des clients et de leurs comportements permet de mieux anticiper ceux qui risquent de passer chez un concurrent et de mieux adapter les opérations commerciales pour tenter de les garder.

L'une des grandes difficultés est de savoir comment extraire ce profil dans un si grand amas de données. Le Data Mining offre entre autres, les moyens d'analyse pour chercher s'il existe un profil comportemental typique des clients qui changent de fournisseurs. L'entreprise pourra ainsi repérer plus facilement parmi ses clients, lesquels ont le profil pour partir vers la concurrence, afin de tenter par des actions commerciales de les garder [ZIG 03].

Les domaines d'utilisation du Data Mining ne se limitent pas qu'aux opérateurs téléphoniques. Il s'étend sur presque tous les types d'entreprises; par exemple: les assurances, les laboratoires pharmaceutiques, les banques etc.

5. Différentes tâches du Data-Mining

Une entreprise a plusieurs besoins et rencontre plusieurs problèmes. Le Data Mining est un outil très efficace car une multitude de problèmes d'ordre intellectuel,

économique ou commercial peuvent être regroupés dans leur formalisation [HEL 04] dans l'une des tâches suivantes:

- La classification,
- L'estimation,
- La prédiction,
- Les règles d'association,
- La segmentation.

Une définition de chaque tâche s'impose afin de lever toutes les ambiguïtés qui peuvent apparaître entre ces termes.

5.1. La classification

La classification se fait naturellement depuis déjà bien longtemps pour comprendre et communiquer notre vision du monde (par exemple les espèces animales, minérales ou végétales) [HEL 04].

La classification consiste à créer des classes (c'est-à-dire des sous-ensembles) de données similaires entre elles et différentes des données d'une autre classe. Autrement dit, l'intersection des classes entre elles doit toujours être vide. La classification définit donc les grands types de regroupement et de distinction. On parle de méta typologie (types de types) [LIA 06].

Son but est de rechercher un ensemble de prédicats caractérisant un ensemble d'objets et qui peut être appliqué à des objets inconnus pour prévoir leurs classes d'appartenance [PON XX]. Elle permet une vision générale de l'ensemble (de la clientèle, par exemple: une banque peut vouloir classer ses clients pour savoir si elle doit accorder des crédits ou non) [LIA 06].

Les techniques les plus appropriées à la classification sont [HEL 04]:

- Les arbres de décision,
- Le raisonnement basé sur la mémoire,
- Eventuellement, l'analyse des liens.

5.2. L'estimation

L'estimation est obtenue par une ou plusieurs fonctions combinant les données en entrée. Le résultat d'une estimation permet de procéder aux classifications grâce à un barème. Exemple: estimer le revenu d'un ménage selon divers critères (type de véhicule et nombre, profession ou catégorie socioprofessionnelle, type d'habitation, etc...) [HEL 04].

Son But est de permettre l'estimation de valeurs inconnues [LIA 06]. Elle permet aussi d'ordonner les résultats pour ne retenir, si on le désire, que les n meilleures valeurs [HEL 04].

Les techniques les plus appropriées à l'estimation [LIA 06]:

- Analyse statistique classique: régression linéaire simple, corrélation, régression multiple, intervalle de confiance et estimation de points.
- Réseaux de neurones.

5.3. La prédiction

La prédiction ressemble à la classification et à l'estimation, mais les enregistrements sont classés selon un certain comportement futur prédit ou à une valeur future estimée [FIL 02]. Donc, la différence qui existe avec la classification et l'estimation est l'échelle temporaire. Tout comme les tâches précédentes, elle s'appuie sur le passé et le présent, mais son résultat se situe dans un futur généralement précisé. La seule méthode pour mesurer la qualité de la prédiction est l'attente [HEL 04].

Son but est, tout comme l'estimation, de permettre l'estimation de valeurs inconnues [LIA 06].

Les techniques les plus appropriées sont [HEL 04]:

- L'analyse du panier de la ménagère,
- Le raisonnement basé sur la mémoire,
- Les arbres de décision,
- Les réseaux de neurones.

5.4. Les règles d'association

La méthode consiste à trouver les variables qui vont ensembles [HEL 04]. C'est une sorte de classification générale. Les règles d'association sont de la forme:

Si antécédent, alors conséquence [LIA 06].

Son but est de mieux connaître les comportements [LIA 06].

Les techniques les plus appropriées sont: [HEL 04, LIA 06]

- Algorithme à priori,
- Algorithme du GRI (induction des règles généralisées),
- Analyse du panier de la ménagère.

5.5. La segmentation

La segmentation permet de découper à posteriori une population hétérogène en classes homogènes [FES 02]. Contrairement à la classification, les sous populations ne sont pas préétablies [HEL 04].

Cette tâche ressemble à l'estimation ou la prédiction, sauf que ce n'est pas un montant etc... à estimer ou prévoir; mais un ensemble de comportements [FAD 07].

Son but: Permettre l'estimation des valeurs inconnues [LIA 06].

Exemples:

- En fonction de critères d'achat d'une voiture, faire une segmentation des acheteurs.
- En fonction des notes obtenues dans différentes matières, faire une segmentation des étudiants.

Les techniques les plus appropriées:

- Analyse des clusters [HEL 04],
- Graphique et nuages de points [LIA 06],
- Méthodes des K plus proches voisins [LIA 06].

6. Conclusion

Le Data Mining correspond à l'ensemble des techniques et des méthodes qui, à partir de données, permettent d'obtenir des connaissances exploitables. Son utilité est avérée dès lors que l'entreprise possède un grand nombre d'informations stockées sous forme de bases de données. Cependant, il ne constitue pas une solution miracle car il ne résout pas tous les types de problèmes de l'entreprise. En plus, ses méthodes et ses techniques ne sont pas toujours simples à mettre en œuvre car elles nécessitent parfois une certaine expertise. Toutefois, le Data Mining

reste une solution intéressante pour extraire de la connaissance à partir des données.

Chapitre II: La classification

1. Introduction

Le Data Mining, comme nous l'avons dans le chapitre 1, peut résoudre un certain nombre de problèmes de différents types qui peuvent se regrouper dans leur formalisation sous l'une de ses tâches qui sont: l'estimation, la prédiction, la segmentation, etc...

Dans ce chapitre, nous allons nous intéresser à une tâche plus précisément : celle de la classification, car de nombreuses applications de fouille de données sollicitent l'utilisation d'un algorithme de classification; par exemple pour réduire le nombre de données à des classes plus facilement interprétables.

2. Définition de la classification

Classifier est le processus qui permet de rassembler des objets dans des sous ensembles tout en conservant un sens dans le contexte d'un problème particulier. Les objets appartenant au même sous ensemble présentent la plus grande similarité et deux objets de sous groupes différents doivent être les moins similaires possible (ou les plus dissimilaires possibles).

Aujourd'hui, les problèmes de classification sont de plus en plus fréquents dans la pratique, comme pour la reconnaissance de l'écriture manuscrite, la reconnaissance de la parole ou encore l'interprétation de photos aériennes ou médicales. Pour chacune de ces problématiques, il s'agit de détecter certaines caractéristiques pour en extraire des informations exploitables par la suite.

Par exemple, pour la reconnaissance de l'écriture manuscrite : pour un mot donné, il s'agit de classer les formes de chacune des lettres le constituant dans une des 26 classes formant l'alphabet latin. Une fois cette phase de classification effectuée, pour chacune des formes composant le mot on dispose de la lettre correspondante [MON 00].

Dans les bases de données, les objets à traiter sont généralement des enregistrements et l'opération de classification est matérialisée par la mise à jour d'un champ de l'enregistrement par un code de la classe qui lui correspond [MAZ 04].

3. Différents types de classification [MON 00]

Plusieurs types de classification existent comme le montre la Figure 1 suivante:

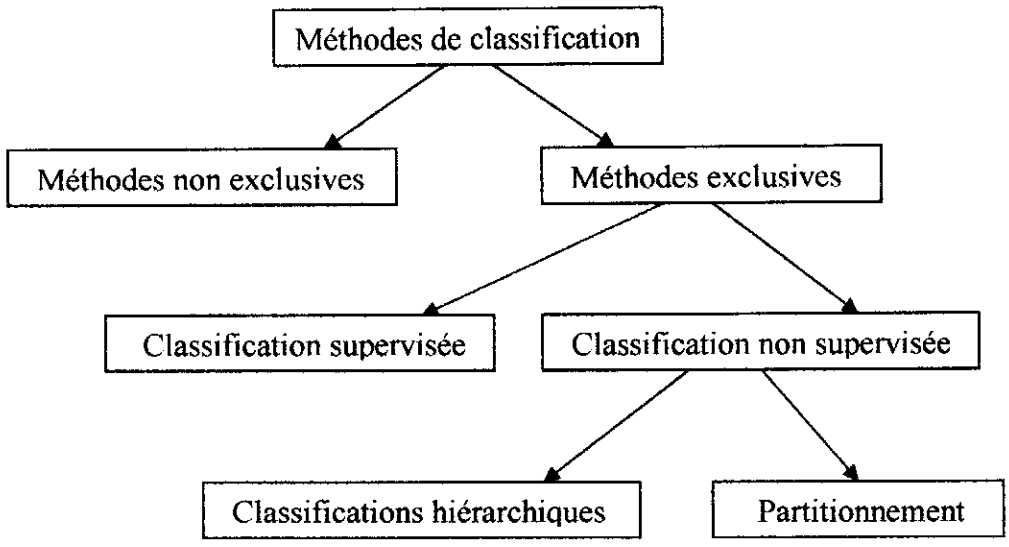


Figure 1. Les méthodes de classification [MON 00]

3.1. La 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.

3.2. La classification supervisée / non supervisée

Dans le cas de la classification non supervisée, aucune information sur la classe d'un objet n'est préalablement fournie à la méthode (on dit que les objets ne sont pas étiquetés). Par contre, pour la classification supervisée, les classes sont connues au préalable.

L'objectif de la classification non supervisée est opposé au cas supervisé. Dans le premier cas, il s'agit de rassembler les objets en groupes non connus auparavant, alors que dans le deuxième cas, il s'agit d'utiliser les groupes initialement connus pour pouvoir classer les objets dans l'un d'eux.

3.3. La classification hiérarchique / partitionnement

Une méthode de classification hiérarchique construit une séquence de partitions imbriquées, alors que le partitionnement ne construit qu'une seule partition des

données; c'est-à-dire qu'il s'agit d'un problème d'optimisation dans lequel on cherche à minimiser un critère particulier qui est propre à chaque méthode de résolution et dépend du type de données considéré. La valeur de ce critère est minimale quand la partition trouvée est acceptable au sens de la fonction d'évaluation.

4. Différentes méthodes de résolution du problème de classification

4.1. Les méthodes de classification et de prédiction

Dans ces méthodes, on utilise un ensemble d'objets dont on connaît le résultat de la classification: c'est l'ensemble d'apprentissage. Cet échantillon sert à prévoir le classement des nouveaux objets. Parmi les méthodes de cette classe, citons :

4.1.1. Le raisonnement à base de cas (RBC)

Les fondements du RCB proviennent de travaux en science cognitive menés par Roger Schank et son équipe de recherche durant les années 1980 [LAM 01]. Aamodt & Plaza ont défini "Le Case-Based Reasonning" (CBR) comme étant une approche capable d'utiliser la connaissance spécifique de problèmes (cas) s'étant déjà produits dans le passé pour répondre à de nouveaux problèmes. Un nouveau problème est résolu en cherchant un cas passé similaire et en réutilisant sa solution dans le problème présent [NOB 04]. Typiquement, un cas contient au moins deux parties: une description de la situation représentant un "problème" et une "solution" utilisée pour remédier à cette situation [LAM 01].

Au début de la dernière décennie, le domaine a connu un regain de popularité. Depuis le milieu des années 90, le CBR s'est révélé une précieuse technique pour la mise en oeuvre d'applications commerciales. Actuellement, le CBR est l'une des techniques de l'intelligence artificielle les plus largement répandues [LAM 01].

4.1.1.1 Principe [LAM 01]

Un système CBR est une combinaison de processus et de connaissances qui permettent de préserver et d'exploiter les expériences passées. Un modèle générique a été proposé par Aamodt et Plaza pour décrire les différentes étapes du processus de résolution CBR; ce modèle est décrit dans la Figure 2.

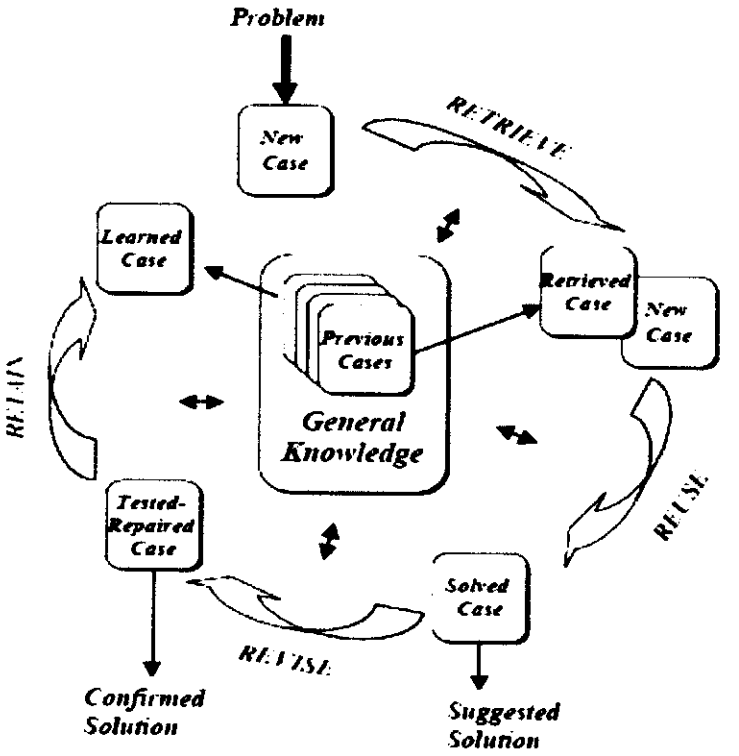


Figure 2. Modèle de processus CBR de Aamodt et Plaza [LAM 01].

D'après cette figure, le raisonnement à base de cas se compose de trois processus principaux:
La recherche ("retrieval"), l'adaptation ("reuse") et la maintenance ("retain"). A présent, nous allons définir chaque processus.

4.1.1.2 La recherche ("retrieval")

Cette phase permet de déterminer les cas de la base qui sont les plus similaires au problème à résoudre. La procédure de recherche de similarité est implantée par une approche basée sur les plus proches voisins ("k-nearest-neighbors") ou sur la construction par induction d'une structure de recherche.

4.1.1.3 L'adaptation ("reuse")

Après la sélection des cas dans la phase de recherche, l'adaptation consiste à modifier et à réutiliser les solutions de ces cas pour résoudre son problème courant. En général, on retrouve deux approches pour l'adaptation de cas:

- l'adaptation structurelle: on obtient une nouvelle solution en modifiant des solutions précédentes et les réorientant afin de satisfaire le nouveau problème.

- l'adaptation dérivationnelle: pour chaque cas passé, on garde une trace indiquant comment la solution a été générée. Pour un nouveau problème, une nouvelle solution est générée en appliquant une de ces suites d'étapes.

Peu de systèmes CBR font de l'adaptation complètement automatique. Pour la plupart des systèmes, une intervention humaine est nécessaire pour compléter une solution partielle ou tout simplement pour générer une solution entièrement à partir d'exemples

4.1.1.4 La maintenance ("retain")

C'est la phase d'intégration de la nouvelle solution dans la base de cas et de la modification du contenu et de la structure du système CBR. Une stratégie simple pour aborder cette phase est d'insérer tout nouveau cas dans la base. Mais d'autres stratégies visent à apporter des modifications à la structuration de la base de cas (ex: indexation) pour en faciliter l'exploitation. On peut également altérer le cas en modifiant les attributs et leur importance relative. Cet aspect de recherche est actuellement l'un des plus actifs dans le domaine du CBR.

4.1.1.5 Critique de la méthode

A –Points forts:

- Clarté des résultats: bien que la méthode ne produise pas de règles explicites, la liste des plus proches voisins peut expliquer comment le RBC arrive à un résultat spécifique.
- S'applique à tout type de données: le RBC est une technique très générale qui ne dépend pas de la représentation des données. La méthode peut s'appliquer dès qu'il est possible de définir une distance sur les champs. Le RBC a souvent été appliqué à des données qui ne sont pas si faciles à traiter par les autres techniques, comme par exemple les images et les données audio.
- capable de travailler sur de nombreux champs: les performances du RBC dépendent plus de la taille de l'ensemble d'apprentissage que du nombre d'attributs dans les enregistrements.
- facile à mettre en oeuvre et à comprendre.

B –Points Faibles:

- Distance et nombre de voisins: les performances de la méthode dépendent du choix de la distance et du nombre de voisins.
- Nécessite un grand volume de données pour être performant; cela entraîne un grand temps de calcul et un grand espace mémoire pour pouvoir stocker toutes les données.

4.1.2. Les arbres de décision

La construction des arbres de décision à partir de données est une discipline déjà ancienne. Morgan et Sonquist (1963) sont les premiers à avoir utilisé les arbres dans un processus de prédiction et d'explication (AID – Automatic Interaction Detection) [RAK 05]. Un arbre de décision est un outil d'aide à la décision et à l'exploration de données.

Il permet de modéliser simplement, graphiquement et rapidement un phénomène mesuré plus ou moins complexe [WIK 07].

Un arbre de décision se compose: [NAK 98]

- D'un noeud racine par lequel entrent les enregistrements,
- De questions,
- De réponses qui conditionnent la question suivante,
- De noeuds feuilles qui correspondent à un classement.

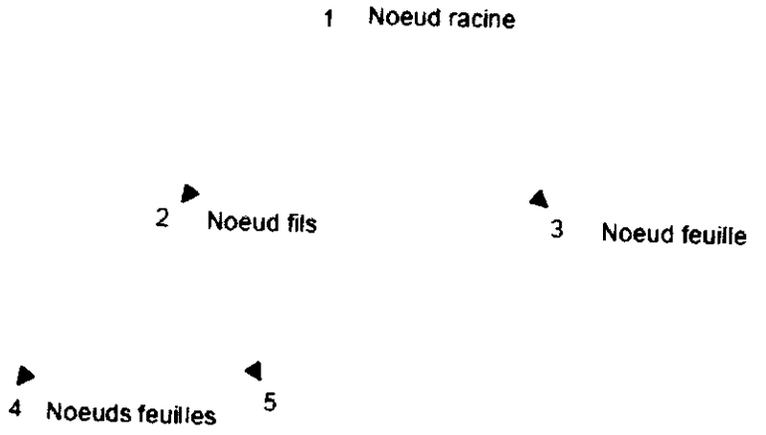


Figure 3. Structure d'un arbre de décision [NAK 98].

Le jeu de questions et réponses est itératif jusqu'à ce que l'enregistrement arrive à un nœud feuille. Afin de déterminer quelle variable doit être affectée à chaque

noeud, la technique applique un algorithme sur chacun des paramètres et conserve le plus représentatif d'un découpage.

4.1.2.1 Principe [GRA XX]

L'idée centrale de l'algorithme est de diviser récursivement les exemples de l'ensemble d'apprentissage par des tests définis à l'aide des attributs jusqu'à ce que l'on obtienne des sous-ensembles d'exemples ne contenant que des exemples appartenant tous à une même classe.
Dans toutes les méthodes, on trouve les trois opérateurs suivants:

A. **Décider si un noeud est terminal**. C'est-à-dire décider si un noeud doit être étiqueté comme une feuille. Par exemple: tous les exemples sont dans la même classe, il y a moins d'un certain nombre d'erreurs,...
B. **Sélectionner un test à associer à un noeud**. Par exemple: aléatoirement, utiliser des critères statistiques,...
C. **Affecter une classe à une feuille**. On attribue la classe majoritaire sauf dans le cas où l'on utilise des fonctions de coût ou risque.

Les méthodes diffèrent par les choix effectués pour les différents opérateurs. C'est-à-dire sur le choix d'un test (par exemple, utilisation du gain et de la fonction entropie) et le critère d'arrêt (quand arrêter la croissance de l'arbre, quand décider si un noeud est terminal). Le schéma général des algorithmes est le suivant:

Algorithme d'apprentissage générique

```
Entrée: langage de description; échantillon S
Début
        Initialiser l'arbre à vide; la racine est le noeud courant
        Répéter
        Décider si le noeud courant est terminal
                Si le noeud est terminal Alors
                        Affecter une classe
                Sinon
                        Sélectionner un test et créer le sous arbre
                Fin Si
        Passer au noeud suivant non exploré s'il en existe
        Jusqu'à obtenir un arbre de décision
Fin
```

Figure 4. Le principe de l'algorithme des arbres de décision

Il existe une variété d'algorithmes pour construire un arbre de décision, citons l'exemple de CART (Classification And Regression Trees) et CHAID (Chi-Squared Automatic Interaction Detection) et l'algorithme C4. 5.

4.1.2.2 Critique de la méthode [NAK 98, TUF 02]

A -Points forts:

- Simplicité d'utilisation: L'utilisation des produits conçus sur la technique des arbres de décision est très simple, car elle est très visuelle et très intuitive; ce qui la rend abordable pour les utilisateurs.
- Bonne lisibilité: La clarté du modèle résultant permet une validation rapide, contrairement à certaines techniques où il est impossible d'expliquer le résultat.
- Bonne adaptation aux données: Les arbres de décisions permettent de manipuler des variables continues, discontinues, catégoriques et énumératives.

B- Points faibles:

- Mauvaise performance: Les arbres de décisions deviennent peu performants lorsqu'il y a beaucoup de classes. En effet, ils risquent de devenir trop détaillés, ce qui leur fait perdre de leur lisibilité ou encore d'aboutir à de mauvais classements.
- Coût de l'apprentissage: Le fait de devoir calculer pour chaque noeud, le meilleur critère lors du premier passage puis ensuite lors de l'élagage, alourdit les calculs.

4.1.3. Les réseaux de neurones

De façon générale, on situe le début des réseaux de neurones artificiels en 1943 avec les travaux de McCulloch et Pitts; les réseaux de neurones sont un modèle mathématique dérivé d'une analyse de la réalité biologique [USH XX]. Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires (neurone artificiel) fonctionnant en parallèle [CLE XX].
Un neurone est une fonction algébrique non linéaire, paramétrée et à valeurs bornées [DRE 02]

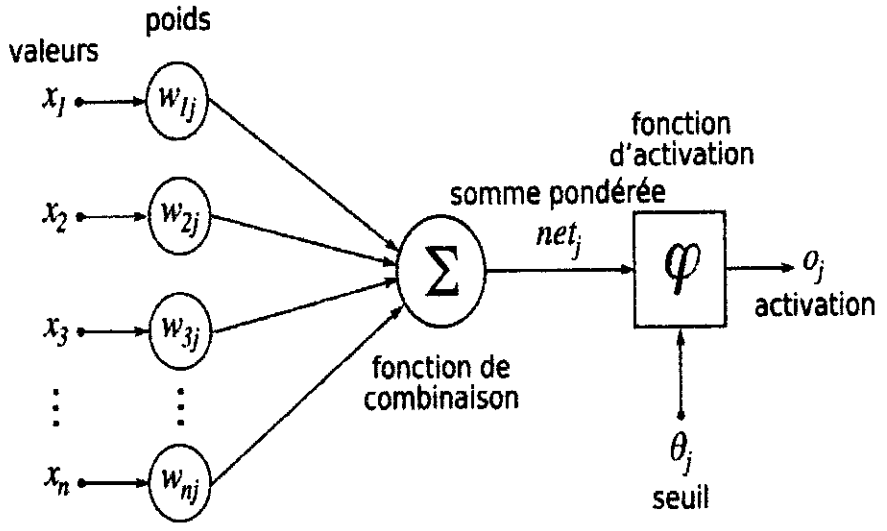


Figure 5. Structure d'un neurone artificiel [WIK 07]

Donc, un neurone artificiel est composé de: [PHI XX]

- Un ensemble de valeurs d'entrée $\{X_1, X_2, \ldots, X_n\}$ discrètes {0, 1} ou réelles [-1, 1].
- Un ensemble de poids réels $\{W_1, W_2, \ldots, W_n\}$ entre les neurones.
- Une fonction de sommation (on l'appelle aussi fonction de combinaison) $\sum$ qui calcule la somme pondérée (sous-entendu pondérée par les poids) des entrées:

  $$(X_1 \times W_1) + (X_2 \times W_2) + \ldots + (X_n \times W_n)$$

- Une fonction d'activation φ qui calcule l'activité / l'état du neurone à partir de cette somme: $\varphi(x)$

Un réseau de neurones se compose de trois couches comme le montre la figure ci-dessous:

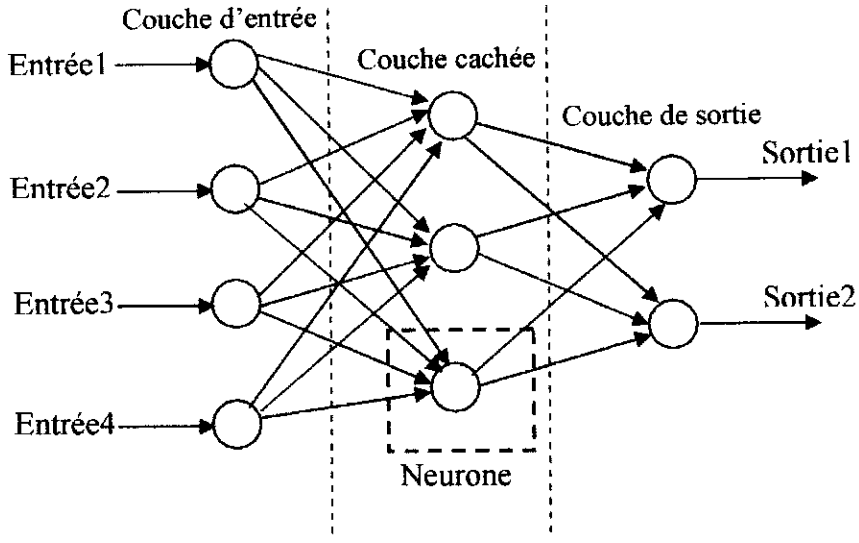


Figure 6. Représentation d'un réseau de neurones [PHI XX].

4.1.3.1 Principe [MAZ 04]

La mise en œuvre d'un réseau de neurones passe par trois étapes:

1- Codage du problème

Il est souvent nécessaire de normaliser les entrées dans l'intervalle [0, 1] pour un meilleur fonctionnement; plusieurs codages sont envisageables:

- on peut coder de façon binaire,
- ou bien coder les nombres sur n bits; par exemple coder sur 3 bits donc le nombre 2 sera codé sous la forme 010
- ou bien, si on a par exemple les valeurs 1, 2, 3, 4, 5 qui peuvent être codées par: 0, 0. 25, 0. 5, 0. 75, 1.

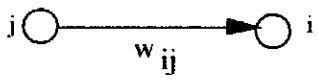
Le nombre de sorties dans un réseau de neurones dépend du nombre d'entrées et de la tâche. Pour un problème de classification, on peut créer une sortie pour chaque classe par exemple. Pour un problème d'estimation, le nombre de sorties est égal au nombre d'entrées (chaque sortie sert à estimer une des variables normalisées).

2-Choix de l'architecture et réglage des paramètres

Le choix de l'architecture est de définir les nombres de couches et les nombres de neurones pour chaque couche. Ce choix se fait par essais successifs ou par expérience; il faut choisir une architecture assez riche et pas trop complexe en même temps pour le bon fonctionnement du réseau.

3- Apprentissage [CLE XX]

L'apprentissage est la mise à jour itérative des poids des connexions jusqu'à l'obtention du comportement désiré. Pour cela, il existe plusieurs règles pour la modification des poids des connexions; on peut citer:



- Loi de Hebb: $\Delta w_{ij}=Ra_ia_j$

Si deux unités connectées sont actives simultanément, le poids de leur connexion est augmenté ou diminué. R est une constante positive qui représente la force d'apprentissage (learning rate)

- Loi de Widrow-Hoff (delta rule): $\Delta w_{ij}=R\ (d_i - a_i)a_j$

a_i activation produite par le réseau.

d_i réponse désirée par l'expert humain.

Par exemple, si la sortie est inférieure à la réponse désirée, il va falloir augmenter le poids de la connexion, à condition que l'unité j soit excitatrice (égale à 1), dans l'hypothèse d'unités booléennes {0,1}.

4.1.3.2 Critique de la méthode [VAN 05, TUF 02]

A- Points forts:

- Taux d'erreurs généralement bon,
- Aptitude à modéliser des problèmes très variés: segmentation, classification, prédiction, domaine médical, reconnaissance de la parole etc...,
- Robustesse (bruit et données manquantes),
- Classification rapide (réseau étant construit),
- Possibilité de combinaison avec d'autres méthodes (ex: arbre de décision pour sélection d'attributs).

B – Points faibles:

- Apprentissage très long: il faut parcourir un grand nombre de fois tous les exemples de l'échantillon d'apprentissage avant de converger. Donc, le temps d'apprentissage peut être long,

- Plusieurs paramètres (architecture et coefficients synaptiques ...): il n'est pas facile de choisir l'architecture du réseau et de régler les paramètres d'apprentissage,
- Evolutivité dans le temps (phase d'apprentissage): si les données évoluent avec le temps, il est nécessaire de relancer une phase d'apprentissage pour s'adapter à cette évolution,
- Impossibilité de traiter un grand nombre de variables.

4.1.4. Les règles d'association

Une règle d'association est une relation d'implication X à Y entre deux ensembles d'articles X et Y [KOU 03]. Cette règle indique que les transactions qui contiennent les articles de l'ensemble X ont tendance à contenir les articles de l'ensemble Y.
X est appelée *condition* ou *prémisse*.
Y est appelée *résultat* ou *conclusion*.

4.1.4.1 Principe [ABD 03]

L'extraction des règles d'association peut être décomposée en quatre étapes. La Figure 7 résume ce processus:

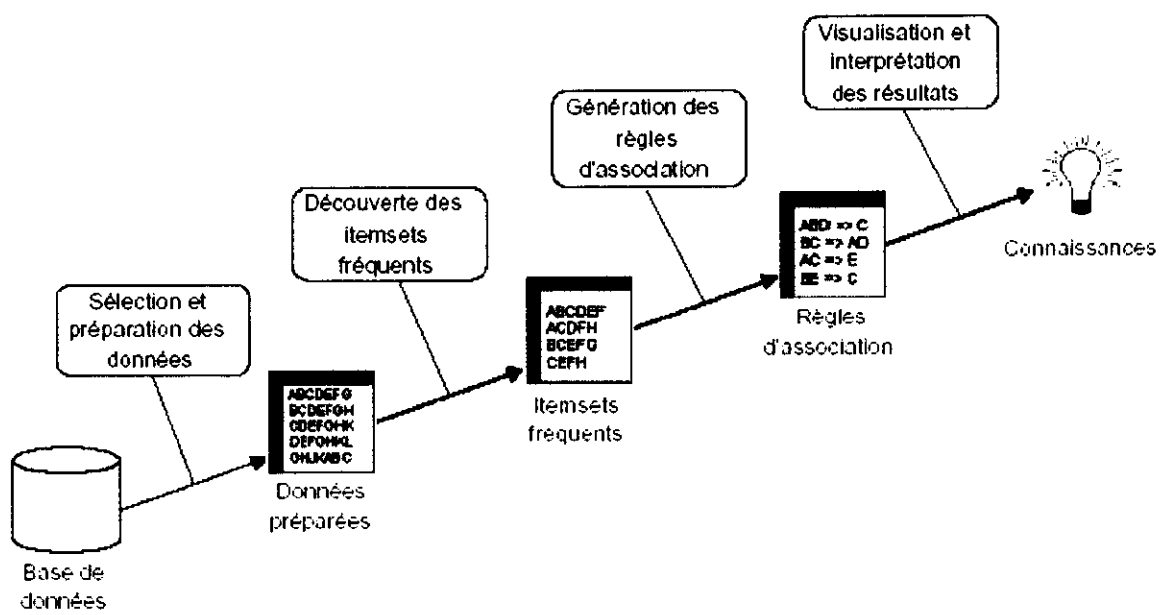


Figure 7. Les étapes d'extraction de règles d'association [ABD 03].

A- Sélection et préparation des données:

Cette étape permet de préparer les données. Elle est constituée de deux phases:

- La sélection des données de la base qui permettront d'extraire les informations intéressant l'utilisateur. Ainsi, la taille des données traitées est réduite, ce qui assure une meilleure efficacité de l'extraction.
- La transformation de ces données en un contexte d'extraction (il s'agit d'un triplet constitué d'un ensemble d'objets, d'un ensemble d'itemsets[1] et d'une relation binaire entre les deux). La transformation des données sélectionnées en données binaires améliore l'efficacité de l'extraction et la pertinence des règles d'association extraites.

B- Découverte des itemsets fréquents:

C'est l'étape la plus coûteuse en temps d'exécution, car le nombre d'itemsets fréquents dépend exponentiellement du nombre d'items[2] manipulés (pour n items, on a 2 puissance n itemsets potentiellement fréquents).

C- Génération des règles d'association

A partir de l'ensemble des itemsets fréquents pour un seuil minimal de support *minsup*, la génération des règles d'association pour un seuil de confiance minconf est un problème qui dépend exponentiellement de la taille de l'ensemble des itemsets fréquents.

D- Visualisation et interprétation des règles d'association

Elle met entre les mains de l'utilisateur un ensemble de déductions fiables qui peuvent l'aider à prendre des décisions.

Il faut que l'outil de visualisation prenne en compte la priorité des règles les unes par rapport aux autres, ainsi que les critères définis par l'utilisateur. De plus, il doit présenter les règles sous une forme claire et compréhensible.

4.1.4.2 Critique de la méthode [VAN 05]

[1] Pour désigner un ensemble d'éléments ou d'articles.

[2] Pour désigner un élément ou un article.

A- Points forts:

- Résultats clairs: règles faciles à interpréter,
- Simplicité de la méthode: les calculs sont élémentaires et la méthode est facile à implémenter,
- Aucune hypothèse préalable (Apprentissage non supervisé),
- Introduction du temps: méthode facile à adapter aux séries temporelles. Exemple:"un client ayant acheté le produit A est susceptible d'acheter le produit B dans un an".

B- Points faibles:

- Coût de la méthode: méthode coûteuse en temps,
- Qualité des règles: production d'un nombre important de règles triviales ou inutiles,
- Articles rares: méthode non efficace pour les articles rares.

4.1.5. L'algorithme des centres mobiles (K-Means) [MON 00]

K-means est une technique qui utilise l'erreur quadratique comme critère d'évaluation d'une partition. Dans un premier temps, les objets sont regroupés autour de K centres arbitraires $c_1 \ldots c_k$ de la manière suivante: la classe c_i associée au centre c_i est constituée de l'ensemble des points plus proches de c_i que de tout autre centre.

Géométriquement, cela revient à partager l'espace des points en K zones définies par les plans médiateurs des segments $[c_i, c_j]$. La Figure 8 donne l'exemple d'une partition associée à trois centres dans le plan.

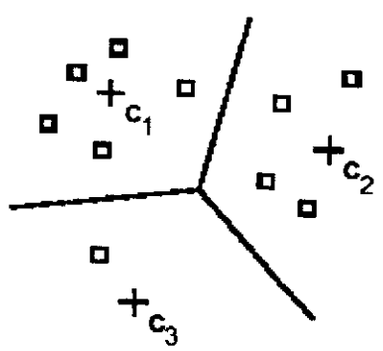


Figure 8. Exemple de partition obtenue par les centres mobiles.

Les centres de gravité $g_1,\ldots, g_k$ sont ensuite calculés à partir des classes qui viennent d'être formées. On recommence l'opération en prenant comme centre de classe les centres de gravité trouvés et ainsi de suite jusqu'à ce que les objets ne changent plus de classe. L'algorithme suivant résume toutes ces opérations:

K-MEANS (partition P de K classes)
Tant que l'inertie intra-classe ne s'est pas stabilisée **Faire**
Générer une nouvelle partition P' en affectant chaque objet à la classe dont le centre est le plus proche
Calculer les centres de gravité des classes de la nouvelle partition P'
P= P'
Fin Tant que
Retourner p

Figure 9. Le principe de l'algorithme des centres mobiles (K-means).

Nous ne nous attarderons pas plus sur cette méthode car elle est bien décrite dans la littérature et il est aisé de trouver de nombreuses définitions et détails sur son principe dans de nombreux ouvrages.

4.2. Les méthodes statistiques

Les méthodes statistiques sont basées sur des concepts mathématiques et statistiques qui nécessitent un taux de calcul important. Parmi ces méthodes on peut citer:

4.2.1. La méthode hiérarchique [SAU 03, SEK 06]

La méthode de classification hiérarchique génère une séquence de partitions imbriquées les unes dans les autres. L'agglomération (méthode ascendante) est la plus connue. Elle débute avec un grand nombre de groupes (classes) et à chaque itération, elle fusionne les groupes jugés assez proches jusqu'à un nombre final de classes.

Algorithme (méthode hiérarchique: l'agglomération)

Définir une classe par objet.
Calculer la matrice de proximité entre toutes les classes.
Tant Que le critère d'arrêt est non vérifié (Nombres de classes est 1)
Fusionner chaque deux classes suffisamment proches.
Mettre à jour la matrice de proximité.
Fin tant que

Figure 10. Algorithme général de la méthode d'agglomération.

L'algorithme d'agglomération ne fournit pas une partition en *Q* classes de *N* objets, mais une hiérarchie de partitions se présentant sous forme d'un arbre appelé dendogramme.
Il existe plusieurs variantes de cet algorithme; car on peut utiliser plusieurs critères d'arrêt et on a aussi la possibilité de choisir entre les différentes stratégies de regroupement.
Dans l'approche du saut minimal (*single-link*), la distance entre deux classes est évaluée par la distance minimale entre deux objets de chacune des classes. Pour le saut maximal (*complete-link*), c'est la distance maximale entre les paires d'objets de chacune des classes qui est prise en compte. Il existe également une autre approche qui évalue les distances moyennes entre les classes.
L'avantage de ces méthodes est qu'elles ne requièrent pas de spécifier à priori le nombre de groupes à retrouver; l'utilisateur peut analyser le dendogramme et fixer le seuil de manière à avoir la partition qui lui semble la plus appropriée. Mais, l'inconvénient de ces méthodes est le manque de lisibilité sur les grands volumes de données.

4.2.2. La régression logistique [SER 02]

C'est un modèle statistique linéaire particulièrement bien adapté pour les problèmes de classification à deux classes. Si l'on cherche à prédire une variable qualitative à deux classes, on peut appliquer une combinaison linéaire des variables prédictives à

une fonction continue (la fonction *logit)* traduisant la probabilité d'apparition d'une des deux classes de la variable à prédire.

$$Logit[\ probabilité(y_i=1)]=a_0+a_1 x_{1i}+a_2 x_{2i}+\ldots+a_{pi}+e_i$$

Le processus de classification se fera donc par la recherche du modèle séparant au mieux les deux classes dans l'espace des variables prédictives. Cette méthode présente plusieurs avantages qui sont:

- Simple et facile à mettre en oeuvre,
- Rapide à l'exécution (apprentissage et test),
- Intéressante comme outil de première approche,
- Utilisable sur un petit nombre de données, s'il y a peu de paramètres.

Il est important de souligner que cette procédure n'est pas applicable dans le cas de problèmes non linéairement séparables, situation qui nécessite l'utilisation de modèles non linéaires.

4.3. Les méthodes biomimétiques

Les méthodes biomimétiques s'inspirent de phénomènes observés dans la nature et se sont modélisées dans des applications informatiques pour la résolution de plusieurs catégories de problèmes, comme ceux de classification par exemple.

4.3.1. Les approches évolutives classiques [AZZ 04]

Ces méthodes utilisent des principes globalement communs, car ils se sont tous inspirés des mêmes principes du neo-darwinisme: utilisation d'une population, évaluation des individus par une fonction, sélection des meilleurs et génération d'une nouvelle population avec des opérateurs de croisement et de mutation.

4.3.1.1 Les algorithmes génétiques

Les algorithmes génétiques sont inspirés de la théorie darwinienne. Cette théorie repose sur deux postulats simples:

- " Dans chaque environnement, seules les espèces les mieux adaptées perdurent au cours du temps, les autres étant condamnées à disparaître ".
- "Au sein de chaque espèce, le renouvellement des populations est essentiellement dû aux meilleurs individus de l'espèce "[AMO 06].

L'utilisation des algorithmes génétiques dans la résolution de problèmes est le fruit des travaux de John Holland de l'Université du Michigan en 1960. Le premier

aboutissement de ces recherches a été la publication en 1975 de 'Adaptation in Natural and Artificial System ' [WIK 07].

Un algorithme génétique recherche le ou les extrema d'une fonction définie sur un espace de données [REC 07]. Pour l'utiliser, on doit disposer des cinq éléments suivants:

a- Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques.

b- Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global.

c- Une fonction à optimiser. Celle-ci retourne une valeur appelée *fitness* ou fonction d'évaluation de l'individu.

d- Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population. L'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états.

e- Des paramètres de dimensionnement: taille de la population, nombre total de générations ou critère d'arrêt et probabilités d'application des opérateurs de croisement et de mutation.

La Figure 11 montre le Principe général des algorithmes génétiques:

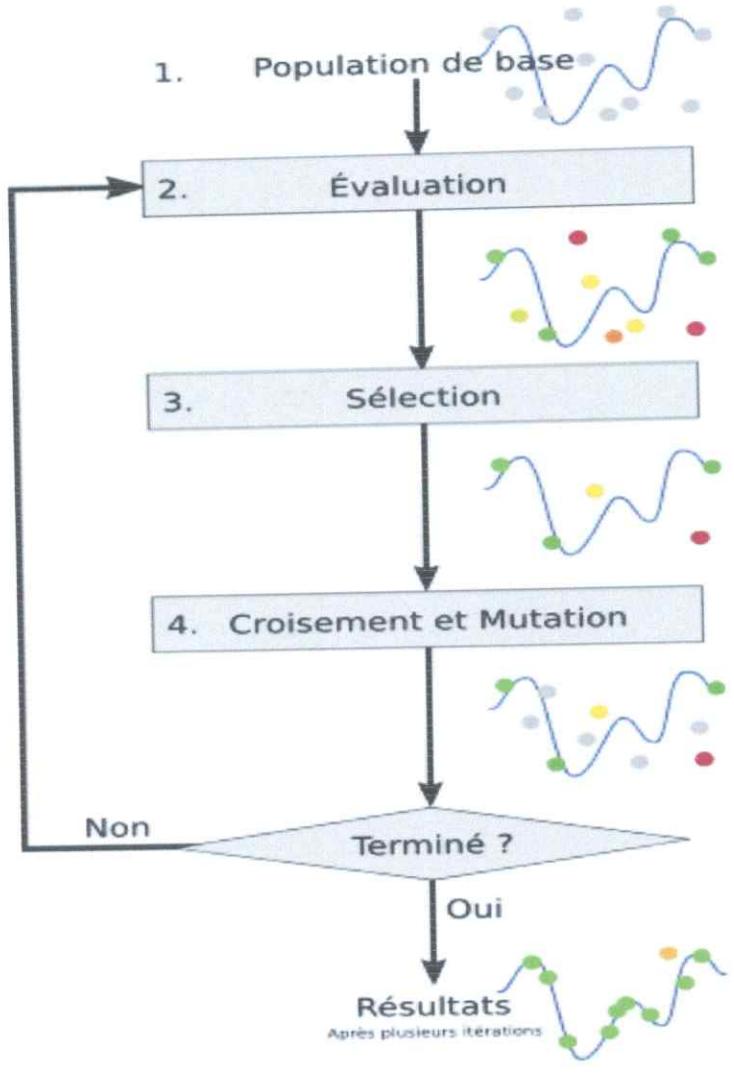


Figure 11. Principe général des algorithmes génétiques [WIK 07].

On commence par générer une population d'individus de façon aléatoire. Pour passer d'une génération k à la génération k+1, les trois opérations suivantes sont répétées pour tous les éléments de la population k. Des couples de parents P_1 et P_2 sont sélectionnés en fonction de leurs adaptations. L'opérateur de croisement leur est appliqué avec une probabilité P_c (généralement autour de 0. 6) et génère des couples d'enfants C_1 et C_2. D'autres éléments P sont sélectionnés en fonction de leur adaptation.

L'opérateur de mutation leur est appliqué avec la probabilité P_m (P_m est généralement très inférieur à P_c) et génère des individus mutés P'. Le niveau d'adaptation des enfants ($C1$, $C2$) et des individus mutés P' est ensuite évalué avant insertion dans la nouvelle population. Différents critères d'arrêt de l'algorithme peuvent être choisis:

- Le nombre de générations que l'on souhaite exécuter peut être fixé à priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.
- L'algorithme peut être arrêté lorsque la population n'évolue plus ou pas assez rapidement.

4.3.1.2 La programmation génétique [CHI 03]

La programmation génétique est une méthode basée sur la sélection naturelle proposée pour la première fois par John Koza. Il s'est inspiré des concepts des algorithmes génétiques; c'est pour cela que certains disent que la programmation génétique est une branche des algorithmes génétiques. Elle présente quelques modifications par rapport aux algorithmes génétiques. Elle permet la variation de la taille des chromosomes et change les techniques de croisement. Par exemple: un croisement avec un seul père peut donner deux fils différents. Or, ce n'est pas le cas dans les algorithmes génétiques.

Algorithme

(1) **Générer** une population initiale.
(2) **Exécuter** chaque programme et lui assigner une valeur qui indique son aptitude à résoudre le problème posé.
(3) **Créer** une nouvelle population de programmes en utilisant les opérations génétiques.
(4) **Répéter** les étapes (2) et (3) jusqu'à atteindre l'optimum.

Figure 12. Algorithme de la programmation génétique.

4.3.2. L'intelligence en essaim

Les études éthologistes ont pu identifier chez certains animaux et insectes des comportements auto-organisés sans contrôle global, menant à l'émergence de tâches qualifiées de complexes.

En effet, ces insectes sociaux sont capables de créer des communautés géantes et réalisent collectivement des tâches extraordinaires telles que le fourragement chez les fourmis, la construction de véritables cathédrales de terre chez les termites et les déplacements collectifs chez les oiseaux et les bancs de poissons. Ce comportement

global complexe émerge à partir des comportements locaux et simples de chaque élément du groupe.

Cette capacité à passer de comportements individuels simples à des comportements collectifs complexes est appelée "intelligence en essaim" ("swarm intelligence") ou «intelligence collective » dans le domaine de l'intelligence artificielle distribuée [OUA 06].

En 1998, les principes de l'intelligence en essaim ont été appliqués pour la première fois à un problème de classification. Les agents représentent chacun une donnée. Ils sont représentés sur un environnement 2D ou 3D. Chaque agent est caractérisé par:

- La position: ce sont les coordonnées de l'agent dans l'environnement,
- Le vecteur vitesse: il nous donne le sens de déplacement et la vitesse de chaque individu,
- Les règles comportementales: elles gèrent les déplacements des individus et sont communes à tous les individus.

La proximité en terme de position et de vitesse dans l'espace de déplacement des individus matérialise la similarité des données dans leur espace multidimensionnel. Un agent réagit aux autres agents présents dans son voisinage en tenant compte de la similarité des données. Un agent se déplacera plutôt vers des données qui lui sont similaires. Cette règle comportementale permet donc de former des groupes de données similaires [AZZ 04].

Algorithme de l'intelligence en essaim

Lire les n données d'entrée $c_1,\ldots. c_n$ (données à classer).
Placer initialement les n insectes de façon aléatoire dans l'environnement 2D.
Calculer la distance idéale entre chaque couple d'insectes.
Tant Que Itération < NbItération **faire**
  Calculer un déplacement pour chacun des insectes en fonction d'une règle locale.
Fin Tant Que
Construire éventuellement des classes à partir des regroupements d'insectes et donner ces classes en sortie.

Figure 13. Algorithme décrivant le principe de fonctionnement de l'intelligence en essaim.

4.3.3. Les systèmes immunitaires

Un système immunitaire artificiel *(SIA)* est une catégorie d'algorithmes d'optimisation inspirée par les principes et le fonctionnement du système immunitaire naturel *(SIN)* des vertébrés. Ces algorithmes exploitent typiquement les caractéristiques du système immunitaire pour ce qui est de l'apprentissage et de la mémorisation, comme moyens de résolution de problèmes. Ils sont liés à l'intelligence artificielle et sont très proches des travaux sur les algorithmes génétiques [WIK 07]. Ils utilisent les principes suivants: des agents (lymphocytes) qui génèrent des anticorps vont apprendre à reconnaître le soi, du non-soi (les antigènes). Pour cela, ces agents doivent d'abord être engendrés en utilisant un principe de composition de briques élémentaires. Ensuite, ils subissent un test de sélection (dit de sélection négative): les agents rejetant le soi sont éliminés et les autres qui vont rejeter le non-soi sont gardés. Chaque fois qu'il y a reconnaissance d'un antigène par un anticorps, la présence des lymphocytes générant ces anticorps est favorisée par un processus de sélection par clonage et par la disparition des lymphocytes non stimulés par les antigènes. Ce clonage donne donc lieu à des interactions entre les lymphocytes et peut mettre en oeuvre des mutations.

Certains lymphocytes, lorsqu'ils sont souvent utilisés, prennent un rôle d'élément de mémorisation à long terme. Ces systèmes disposent de propriétés complexes car ils sont capables de générer des solutions et de les sélectionner en fonction de leur efficacité selon des heuristiques originales [AZZ 04].

Principe des systèmes immunitaires [AZZ 04]

En ce qui concerne la classification, les principes des systèmes immunitaires sont à un niveau général les suivants: les données d1,...,dn représentent les antigènes. Ces antigènes sont présentés itérativement au système jusqu'à l'obtention d'une condition d'arrêt. On suppose que les données sont numériques et donc qu'un antigène est un vecteur de dimension n. A chaque itération, l'antigène présenté va activer des anticorps (assimilés dans cette modélisation à des lymphocytes - B). Un anticorps est également représenté par un vecteur de dimension n. Les anticorps suffisamment proches de l'antigène (au sens de la distance euclidienne) vont subir des clonages avec mutation (interaction anticorps/antigènes) afin d'amplifier et d'affiner la réponse du système. Egalement, ces anticorps vont subir une sélection (interaction anticorps/anticorps): ceux qui sont trop proches les uns des autres seront

diminués en nombre. Après ces itérations, le système converge en plaçant des anticorps (qui agissent comme des détecteurs) de manière judicieuse et en nombre adapté aux données.

Algorithme des systèmes immunitaires

1. A chaque étape d'itération, faire:
   1.1. Pour chaque antigène *i*, faire:
      1.1.1. *Déterminer* son affinité à toutes les cellules de réseau (anticorps) selon une distance métrique dans l'espace des formes, d_{ij},
      1.1.2. *Sélectionner* les *n* cellules d'affinité la plus élevée.
      1.1.3. *Reproduire* les *n* cellules choisies. Le nombre de la progéniture de chaque cellule Nc est proportionnel à son affinité: plus l'affinité est haute, plus la taille du clonage est grande.
      1.1.4. *Augmenter* l'affinité de ces cellules Nc à l'antigène *i* en réduisant la distance entre elles.
      1.1.5. *Calculer* l'affinité de ces cellules améliorées avec de l'antigène *i*.
      1.1.6. *Re-sélectionner* ζ% des cellules (dont l'affinité la plus élevée sur les cellules améliorées) et les mettre dans la matrice partielle Mp des cellules de mémoire;;;;;;
      1.1.7. *Eliminer* les cellules dont l'affinité est inférieure au seuil □d (seuil d'affinité),
      1.1.8. *Calculer* l'affinité du réseau cellule-cellule, s_{ij}.
      1.1.9. *Eliminer* les cellules de s_{ij} dont l'affinité est inférieure au seuil σ d (seuil d'affinité), une autre réduction de la matrice M_p (suppression clonale).
      1.1.10. *Concaténer* la matrice originale de cellules de réseau avec la matrice partielle des cellules de mémoire (C <- [C,M_p]).
   1.2. *Déterminer les affinités entières d'inter- cellule du réseau et éliminez les cellules dont l'affinité de l'un des deux est inférieure au seuil* σ (suppression de réseau).
   1.3. *Remplacer* r% des plus mauvais individus par des nouveaux individus généré aléatoirement.
2. Tester le critère d'arrêt.

Figure 14. Algorithme des systèmes immunitaires.

5. Conclusion

Dans ce chapitre, nous avons défini la classification, ses types et ses différentes méthodes.

Les problèmes de classification sont des problèmes *Np_difficiles* qui ne peuvent être résolus par des approches exhaustives dès que la taille du problème à résoudre devient importante. Le recours à des méthodes heuristiques devient alors la seule possibilité. Cependant, ce genre de méthode ne permet pas de trouver avec certitude la solution optimale au problème. C'est pourquoi de nombreuses méthodes ont vu le jour pour tenter de résoudre le problème qui nous intéresse. Les méthodes

inspirées de la biologie ont de plus en plus de succès car elles ont donné des résultats intéressants dans de nombreux cas. Elles sont utilisées avec succès dans la classification depuis quelques années. Les fourmis artificielles sont une de ces méthodes qui connaissent un grand succès.

Dans le chapitre suivant, nous allons nous focaliser sur les fourmis artificielles et citer les principales approches inspirées de ces insectes pour la résolution du problème de classification non supervisée.

Chapitre III: La classification par la méthode des fourmis artificielles

1. Introduction

Dans la pratique, on trouve souvent des problèmes liés à la classification. Par exemple, regrouper un ensemble de données en un nombre de classes pour mieux les exploiter après. Pour cela, il a fallu concevoir des algorithmes de classification.
Les fourmis ont rapidement suscité l'intérêt des chercheurs car ce sont des créatures qui peuvent par leur comportement et leurs organisations résoudre une multitude de problèmes quotidiens et complexes.
Le travail des chercheurs sur les fourmis ont donné naissance à plusieurs algorithmes basés sur le comportement des fourmis sous le nom « algorithmes de fourmis artificielles ». Ces algorithmes ont rapidement eu une grande popularité due d'une part à la facilité de mise en oeuvre et d'autre part à la complexité des problèmes qu'elles peuvent résoudre.
Dans ce chapitre, nous allons exposer certains de ces algorithmes, tels que: les algorithmes de Lumer et Faieta, et les algorithmes AntClass, AntClust, AntTree et AntPart

2. Le comportement des fourmis réelles [WIK 07, OUA 06]

Les fourmis sont des insectes sociaux (insectes vivants s'organisant en colonies et démontrant une intelligence collective leur permettant de retirer un bénéfice de leur instinct grégaire) formant des colonies appelées fourmilières, parfois extrêmement complexes et contenant de quelques dizaines à plusieurs millions d'individus. Certaines espèces forment des « colonies de colonies » (ou super colonies).
Une estimation du nombre de fourmis vivant aujourd'hui sur terre à un instant donné est d'environ 10 millions de milliards d'individus. Chaque individu ne pèse que de 1 à 10 milligrammes, mais leur masse cumulée dépasserait celle de l'humanité. Environ 12 000 espèces de fourmis sont répertoriées en 2005, mais on en découvre régulièrement (essentiellement en zone tropicale et dans la canopée). Mais, seules 400 espèces sont connues en Europe, alors qu'on peut en compter jusqu'à 40 espèces différentes sur un seul mètre carré de forêt tropicale en Malaisie (668 espèces comptées sur 4 hectares à Bornéo, 43 espèces sur un seul arbre de la forêt péruvienne amazonienne; soit presque autant que pour toute la Finlande ou les îles Britanniques). Les activités des communautés de fourmis sont caractérisées par un

certain degré de division du travail, souligné par une différenciation fonctionnelle et anatomique des individus. Une fourmilière peut abriter de 50 mille à plus de 1 million d'individus bien différenciés tant au niveau physique qu'au niveau des comportements et des tâches à accomplir. On les sélectionne en castes:

Les reines: dans une fourmilière on trouve une ou plusieurs reines. Les reines sont nettement plus grosses que les autres fourmis et peuvent vivre jusqu'à dix ou quinze ans. Leur rôle se résume essentiellement à pondre des œufs. Elles sont donc les fondatrices de nouvelles colonies.

Les ouvrières: elles forment la majorité des habitants de la cité et se chargent de la défense et de l'entretien de la colonie, qui comprend la construction des galeries, les soins apportés aux jeunes, la quête de la nourriture, etc...

Les soldats: Ils sont plus massifs que les ouvrières et possèdent souvent de grosses mandibules. Leur rôle est de défendre la fourmilière et de transporter des charges lourdes. Mais certains, comme chez les fourmis "Grand Galop" *Camponotus maculatus*, participent aux soins des larves et donnent à manger aux fourmis qui le demandent. Chez cette espèce, qui est la plus grosse fourmi vivant à La Réunion, on peut distinguer des formes intermédiaires entre la petite ouvrière grêle et le puissant soldat.

Les jeunes sexués: ce sont des fourmis femelles et mâles. Ils sont facilement reconnaissables par leur plus grande taille, par la présence de deux paires d'ailes membraneuses sur le thorax et par 3 ocelles disposés en triangle sur le dessus de la tête. Les femelles sont plus grosses que les mâles. Ce sont les futurs rois et reines qui iront fonder de nouvelles colonies.

Le couvain: Il est constitué par les oeufs, les larves et les nymphes. Au bout de quelques jours, les oeufs donnent naissance à des larves qui, bien nourries par les ouvrières pendant 15 jours à 3 semaines, se transforment en nymphes. Pendant la nymphose, la larve ne se nourrit plus.

Chez la plupart des espèces, la colonie a une organisation sociale complexe et est capable d'accomplir des tâches difficiles (exploiter au mieux une source de nourriture, par exemple). Cette organisation apparaît grâce aux nombreuses interactions entre fourmis et n'est pas dirigée -- contrairement à une idée répandue -- par la reine. On parle alors d'intelligence collective pour décrire la manière dont ce comportement collectif complexe apparaît grâce à des règles individuelles relativement simples. Cela est impossible sans la présence d'une communication

entre ces dernières. Pour cela, les fourmis sont munies de plusieurs mécanismes de communication, dont le plus répandu est la communication chimique à travers un produit chimique volatile appelé « phéromones » émis par diverses glandes, parfois dans une substance lipophile qui recouvre naturellement tout le corps de la fourmi. Comme d'autres insectes, les fourmis sentent avec leurs antennes. Celles-ci sont assez mobiles et leur permettent d'identifier aussi bien la direction que l'intensité des odeurs.

3. Les fourmis artificielles [OUA 06]

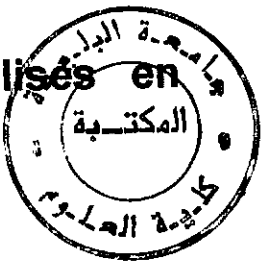
Les fourmis sont devenues une source d'inspiration pour la conception de méthodes de résolution de problèmes complexes. De plus, cette source d'inspiration n'est pas unique étant donné que les fourmis sont dotées d'une grande diversité de caractéristiques disjointes et de comportements collectifs variés. Une nouvelle classe d'algorithmes est alors apparue sous le nom « algorithmes de fourmis artificielles ». Deux comportements collectifs ont été principalement étudiés chez les fourmis: l'optimisation de chemin et le tri des cadavres.

Le premier comportement appelé aussi fourragement permet aux fourmis de retrouver le plus court chemin entre leur nid et une source de nourriture grâce à un système de marquage de phéromones. Ce comportement naturel a été modélisé et transposé à la résolution de nombreux problèmes d'optimisation combinatoires sous le nom d'une nouvelle métaheuristique « optimisation par les colonies de fourmis ou OCF ».

Le deuxième comportement collectif des fourmis concerne la capacité de certaines espèces de fourmis à organiser collectivement des cimetières composés de cadavres empilés les uns sur les autres. Là aussi, les chercheurs ont exploité ce comportement pour fournir des algorithmes de classification pour lequel l'informatique classique n'a pas donné de solution satisfaisante.

Une fourmi artificielle est une entité simple dotée d'un comportement similaire ou étendu à celui de la fourmi réelle. Ce comportement doit être élémentaire, restreint et donc facile à programmer. A l'intérieur d'une colonie, les fourmis sont concurrentes et asynchrones. Elles coopèrent inconsciemment ensemble pour la résolution du problème considéré. Les fourmis artificielles communiquent entre elles indirectement par stigmergie via des modifications de leur environnement (par exemple par dépôt

de traces de phéromone artificielle) qui représente la mémoire collective de la colonie. Elles ont été de plus enrichies de contraintes et de comportements qu'on ne trouve pas chez les fourmis réelles mais qui sont spécifiques aux problèmes qu'elles résolvent.



4. Les différents algorithmes de Fourmis utilisés en classification

4.1. Algorithme de Lumer et Faieta (LF) [LUM 94]

Lumer et Faieta ont proposé un algorithme utilisant 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 et sont plongés dans un espace direct de dimension moindre (typiquement de dimension 2). Cet espace discret s'apparente alors à une grille G dont chaque case peut contenir un objet. Les agents se déplacent sur G et perçoivent une région R de s×s cases dans leur voisinage. La Figure 15 donne un exemple de grille avec une fourmi (représenté par X) et son périmètre de détection (en trait épais). Les objets sont représentés par des carrés dont l'intérieur (invisible pour la fourmi) représente la classe d'origine.

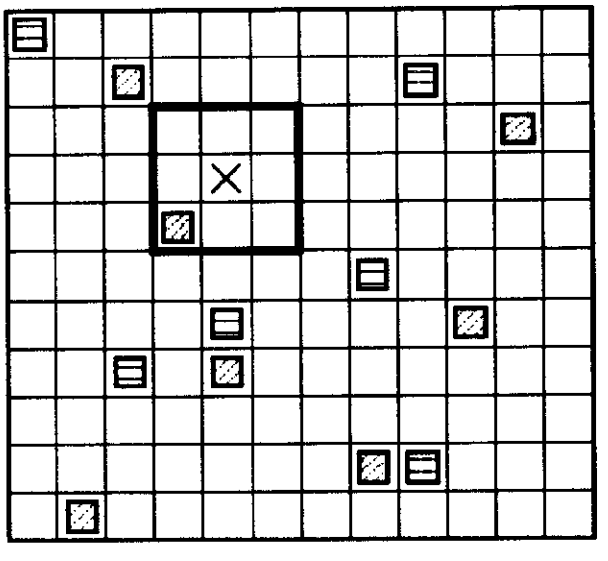


Figure 15. Un exemple de la grille dans l'algorithme LF [MON 01].

La formule p_p représente respectivement la probabilité pour qu'une fourmi ramasse un objet si elle ne transporte aucun objet, et la formule p_d représente la probabilité pour qu'elle dépose un objet.

$$p_p(o_i) = \left(\frac{k_1}{k_1 + f(o_i)}\right)^2$$

$$p_d(o_i) = \begin{cases} 2f(o_i) & \text{si } f(o_i) < k_2 \\ 1 & \text{si } f(o_i) > k_2 s \end{cases}$$

La fonction de densité locale est calculée de la manière suivante:

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum_{o_j \in R_s(r(o_i))} 1 - \frac{d(o_i,o_j)}{\alpha} & \text{si } f > 0 \\ 0 & \text{sinon} \end{cases}$$

f (o_i) dépend de l'objet considéré o_i et de sa position sur la grille r' (o_i). f(oi) est alors une mesure de la similarité moyenne de l'objet o_i avec les objets o_j présents dans son voisinage.

α est un facteur d'échelle déterminant dans quelle mesure la dissimilarité entre deux objets est prise en compte. L'algorithme donne les étapes de la méthode en utilisant A fourmis {α_1,.....α_A}. Les paramètres ont les valeurs suivantes:

$k_1 = 0.1$, $k_2 = 0.15$, $s = 3$, $\alpha = 0.5$ et $T_{max} = 10^6$

LF ()
(1) Placer aléatoirement les N objets o_1..... o_N sur la grille G
(2) Pour T=1 à T_{max} **faire**
(3) Pour tout α_j {α_1..... α_A} **faire**
(4) Si la fourmi α_j ne transporte pas d'objet et $r(o_i)=r(\alpha j)$ **alors**
(5) Calculer $f(o_i)$ et $P_p(o_i)$
(6) la fourmi α_j ramasse l'objet o_i suivant la probabilité $P_p(o_i)$
(7) Si non
(8) Si la fourmi α_j transporte l'objet o_i et la case $r(\alpha_j)$ est vide **alors**
(9) Calculer $f(o_i)$ et $P_d(o_i)$
(10) la fourmi α_j dépose l'objet o_i sur la case $r(\alpha_j)$ avec une probabilité $P_d(o_i)$
(11) **Fin si**
(12) **Fin si**
(13) déplacer la fourmi α_j sur une case voisine non occupée par une autre fourmi
(14) **Fin pour**
(15) **Fin pour**
(16) **Retourner** l'emplacement des objets sur la grille.

Figure 16. Le principe de l'algorithme de Lumer et Faieta.

La Figure 17 donne un résultat possible de l'exécution de l'algorithme LF sur la grille de la Figure 15. On peut remarquer que le nombre de groupes d'objets (3) ne correspond pas obligatoirement au nombre original de classes (2).

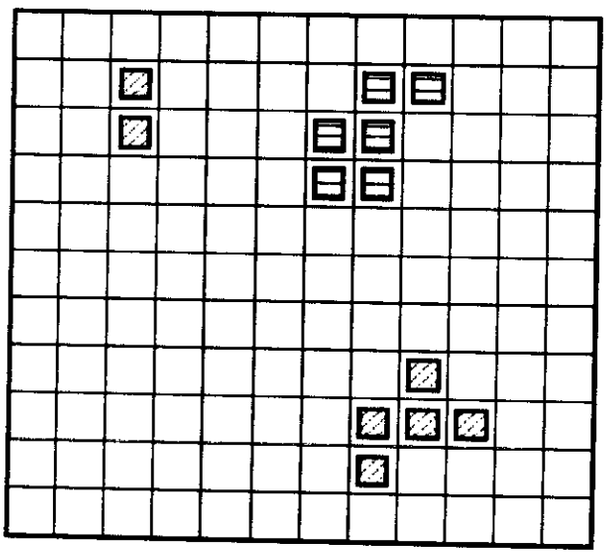


Figure 17. Résultat possible de l'exécution de l'algorithme LF sur la grille de la Figure 15 [MON 01].

Des tests ont été effectués sur des données artificielles: 200 points sont générés dans un espace à deux dimensions ([0,1] × [0,1]) suivant quatre lois gaussiennes de moyenne variable et d'écart type fixe. La grille utilisée comporte 100×100 cases et 10 fourmis sont utilisées. Les résultats obtenus montrent que 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 $v_{max} = 6$ 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.

Cet algorithme a été expérimenté sur des bases de données de taille importante et les résultats, tant du point de vue de la classification que de l'intérêt de la représentation planaire se sont montrés prometteurs.

Les principales critiques concernent le temps de calcul relativement important ($Tmax=10^6$). De plus, l'interprétation des résultats devient difficile quand la taille de la base augmente puisque la frontière entre deux groupes d'objets peut être réduite à une case vide alors que ces deux groupes peuvent représenter deux groupes d'objets très distincts.

4.2. Algorithme AntClass [MON 00, OUA 06]

En se basant sur les travaux de Lumer et Faieta, Monmarché proposa un nouvel algorithme de classification AntClass. Cet algorithme utilise une colonie de fourmis artificielles qui se déplacent sur une grille toroïdale carrée afin d'éviter les effets de bords. La taille de la grille est calculée automatiquement en fonction de la taille des objets à classer. Chaque fourmi a la capacité de transporter plusieurs objets à la fois et de placer un tas d'objets sur une seule case de la grille. Cette façon de procéder permet une extraction plus facile des classes que dans le modèle de Lumer et Faieta. De plus, AntClass est une hybridation de l'algorithme stochastique de fourmis et de l'algorithme déterministe de classification « le K-means » (présenté dans le chapitre précédent). Afin d'accélérer la convergence vers une partition stable, AntClass comporte deux exécutions successives d'un motif constitué d'une application de l'algorithme de fourmis suivi de l'algorithme K-means. Les fourmis génèrent une partition initiale de bonne qualité qui sera par la suite raffinée par K-means. AntClass a été testé sur des bases de données réelles et les résultats obtenus sont significatifs.

A l'initialisation, les A fourmis $\{a1,..., aA\}$ 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 ai se déplace aléatoirement sur la grille à une vitesse $v(ai)$.

Concrètement, si $(x(ai), y(ai))$ sont les coordonnées de la fourmi ai sur G, après un déplacement, la nouvelle position sera dans l'intervalle $([x(ai)-v(ai), x(ai)+v(ai)], [y(ai)-v(ai), y(ai)+v(ai)])$ 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.

Dans ce cas, elle a une chance sur deux de tourner de 45 degrés à gauche ou à droite.

Chaque fourmi dispose d'une capacité de transport *c(ai)*. Par la suite, nous n'étudierons que deux valeurs pour ce paramètre: une capacité *c(ai)* = 1 et infinie (*c(ai)* = ∞).

Afin de déterminer les règles que les fourmis vont utiliser sur la grille pour manipuler les objets, il faut leur donner certaines mesures: de la dispersion d'un tas ou de la dissimilarité entre deux objets par exemple. Pour cela, on définit les distances suivantes:

– la distance maximale entre deux objets de l'ensemble *O:*

$$d^*(O) = \max_{(i,j)\in\{1,\ldots,N\}^2} \{d(\mathbf{x}_i, \mathbf{x}_j)\}$$

– la distance moyenne entre deux objets de l'ensemble *O:*

$$\bar{d}(O) = \frac{2}{N(N-1)} \sum_{(i,j)\in\{1,\ldots,N\}^2, i<j} d(\mathbf{x}_i, \mathbf{x}_j)$$

– la distance maximale entre les objets d'un tas *Tj* et son centre de gravité **g***j:*

$$d_g^*(T_j) = \max_{\mathbf{x}_i\in T_j} \{d(\mathbf{x}_i, \mathbf{g}_j)\}$$

- la distance moyenne entre les objets d'un tas *Tj* et son centre de gravité **g***j:*

$$\bar{d}_g(T_j) = \frac{1}{|T_j|} \sum_{\mathbf{x}_i\in T_j} d(\mathbf{x}_i, \mathbf{g}_j)$$

Ramassage d'objets

Si la fourmi a_i 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:

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

Où ε est une petite valeur positive 10^{-5}, k_1 est un paramètre réel positif permettant de contrôler la forme de la densité de $p_p(T_j)$ quand $|T_j|>2$ et $|T_j|$ le nombre d'objets dans le tas.

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:

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

Où k_2 est un paramètre réel positif permettant de contrôler la forme de la densité P_d (o_i,T_j) quand $d(o_i, g_j) > d_g^{*}(T_j)$.

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

Patience des fourmis

S'il y a trop de fourmis par rapport au nombre de tas ou d'objets, on peut tomber sur le problème suivant: tous les tas (dans le cas d'une grande capacité de transport des

fourmis) ou tous les objets sont transportés, ce qui n'offre plus de possibilité aux fourmis de déposer ce qu'elles transportent. Dans le cas où les fourmis ont une capacité de transport égale à 1, il suffit de s'assurer que le nombre de fourmis est nettement inférieur au nombre d'objets. Par contre, quand les fourmis ont une capacité de transport supérieure, le problème peut réapparaître. La solution la plus immédiate est de doter les fourmis d'une certaine patience, qui peut être individuelle et notée *p*(*ai*). Quand la fourmi a effectué plus de *p*(*ai*) déplacements sans avoir réussi 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. Par la suite, cette patience sera utilisée en particulier quand la capacité *c*(*ai*) d'une fourmi est supérieure à1.

Mémoire des fourmis

Pour Lumer et Faieta, quand un objet *oi* est ramassé par une fourmi *ai*, il est comparé aux *m* (*ai*) mémoires de la fourmi (où *m*(*ai*) représente la taille de la mémoire de la fourmi *ai*). Elle se dirige ensuite vers l'emplacement de l'objet le plus similaire à *oi*.

Dans Antclass, le même principe est utilisé en remplaçant la comparaison des objets sur la distance les séparant par la distance entre le centre de gravité du tas transporté par la fourmi et les tas qu'elle a mémorisés;puisque nos fourmis peuvent transporter plusieurs objets. Dans le cas où la fourmi ne transporte qu'un seul objet, il se confond avec le centre de gravité du tas qu'il forme à lui tout seul. La fourmi gère sa mémoire 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.

Algorithme

AntClass se compose de deux parties: Ants et K-Means.

Algorithme Ants: regroupement des objets par les fourmis

ANTS(Grille G)

(1) pour t =1 à T **faire**

(2) pour k = 1 à A **faire**

(3) Déplacer la fourmi α_k sur une case non occupée par une autre fourmi

(4) Si il y a un tas d'objets T_j sur la même case que α_k **alors**

(5) Si la fourmi α_k transporte un objet o_i [un tas d'objets T_i] **alors**

(6) Déposer l'objet o_i [le tas T_i] transporté par la fourmi sur le tas T_j suivant la probabilité $p_d(o_i, T_j)$ [$p_d(T_i, T_j)$]

(7) Sinon

(8) /* La fourmi ne transporte pas d'objet */

Ramasser l'objet o_i le plus dissimilaire du tas T_j [jusqu'à ce que la capacité c (α_k) de la fourmi soit atteinte ou que le tas soit vide] selon la probabilité $p_p(T_j)$.

(9) Fin si

(10) Fin si

(11) Fin pour

(12) Fin pour

(11) Retourner la grille G.

Figure 18. Le principe de l'algorithme Ants.

Afin de pouvoir évaluer la qualité du partitionnement obtenu, la mesure d'erreur suivante est utilisée. Si, pour chaque objet *oi* on connait sa classe d'origine *c*(*oi*) et la classe obtenue par AntClass *c'*(*oi*), l'erreur de classification *Ec* est calculée de la façon suivante:

$$E_c = \frac{2}{N(N-1)} \sum_{(i,j)\in\{1,\ldots,N\}^2, i<j} \varepsilon_{ij}$$

Où

$$\varepsilon_{ij} = \begin{cases} 0 & \text{si } (c(o_i) = c(o_j) \wedge c'(o_i) = c'(o_j)) \vee (c(o_i) \neq c(o_j) \wedge c'(o_i) \neq c'(o_j)) \\ 1 & \text{sinon} \end{cases}$$

Cette erreur considère tous les couples d'objets possibles et augmente à chaque fois que deux objets n'ont pas été classés ensembles par AntClass, alors qu'ils faisaient partie de la même classe à l'origine et réciproquement.

Algorithme K-MEANS

L'algorithme des centres mobiles permet d'améliorer la partition découverte par les fourmis.

Considérons qu'une partition sous la forme de K classes ($c_1,\ldots,c_k$) a été obtenue.

Le centre de gravité g_i de la classe c_i est donnée par:

$$g_i = \frac{1}{|c_i|}\sum_{j=1}^{|Ci|} x_j^{(i)}$$

Où $x_j^{(i)}$ est $j^{ème}$ point de la classe c_i. $|c_i|$ le nombre d'éléments de la classe c_i.

L'erreur quadratique ε_i^2 sur la classe c_i est donnée par:

$$\varepsilon_i^2 = \sum_{j=1}^{|Ci|} d^2\left(x_j^{(i)}, g_i\right)$$

L'inertie intra-classe est donnée par:

$$I_w = \sum_{j=1}^{k} \varepsilon_j^2$$

Algorithme

K-MEANS (partition P de K classes)
TQ l'inertie intra-classe ne s'est pas stabilisée **Faire**
Générer une nouvelle partition P' en affectant chaque objet à la classe dont le centre est le plus proche
Calculer les centres de gravité des classes de la nouvelle partition P'
P:= P'
FTQ
Retourner p

Figure 19. Le principe de l'algorithme K-Means.

Algorithme AntClass de classification non supervisée par des fourmis et les centres mobiles.

ANTCLASS ()
(1) Soit P_0 la partition initiale formée de N classes.
(2) **Pour** t=1 à $T_{AntClass}$ **faire**
(3) Initialiser la gille G à partir de la partition P_{t-1} (un tas par classe)
(4) $G^1 \leftarrow$ANTS(G)
(5) Construire la partition P^1 associée à la grille G^1
(6) $P_t \leftarrow$K-MEANS(P^1)
(7) **Fin pour**
(8) **Retourner** la partition $T_{AntClass}$

Figure 20. Le principe de l'algorithme AntClass.

4.3. Algorithme AntTree [AZZ 03]

Cet algorithme est inspiré du principe d'auto assemblage des fourmis qui se fixent progressivement à un support fixe puis les autres fourmis se fixent successivement aux fourmis déjà fixées. AntTree reprend ce principe en utilisant un graphe (arbre) qui est construit par les fourmis (chacune d'elles représente une donnée à classer). L'assemblage entre les fourmis signifie qu'il y a une similarité entre elles (entre les données).

A partir de ce comportement qui est l'auto assemblage, AntTree s'est fondé sur les propriétés suivantes:

- ✓ la structure est construite à partir d'un point de départ sur un support fixe,
- ✓ les fourmis peuvent se déplacer sur la structure en cours de construction,
- ✓ les fourmis peuvent atteindre tous les points de la structure et peuvent se fixer n'importe où sur la structure,
- ✓ les fourmis qui composent la structure peuvent être bloquées sans aucune possibilité de déplacement,
- ✓ un certain nombre de fourmis sont reliées à la structure par un lien qu'elles maintiennent elles mêmes; donc elles peuvent se détacher de la structure,
- ✓ la structure peut croître et décroître (des fourmis qui se décrochent de la structure).

Principe général

Le principe de l'algorithme AntTree est le suivant: chaque fourmi f_i représente un noeud de l'arbre à assembler, c'est-à-dire une donnée à classer v_i partant d'un point de départ- le support- matérialisé par un noeud fictif f_0. Les fourmis vont progressivement se fixer sur ce point initial, puis successivement sur les fourmis fixées à ce point initial et ainsi de suite, jusqu'à ce que toutes les fourmis soient rattachées à la structure.

Tous ces déplacements et ces accrochages dépendent de la valeur retournée par la fonction de similarité *Sim (i,j)* entre les données et du voisinage local de la fourmi en déplacement. Lors de la construction de la structure, chaque fourmi f_i est soit en déplacement sur le graphe, soit connectée à celui-ci. Dans le premier cas, elle est libre de se déplacer sur l'une des voisines de la fourmi sur laquelle elle se trouve (ou sur le support). Dans le deuxième cas f_i ne pourra plus se dégager. De plus, nous allons considérer le fait que chaque fourmi a un seul lien sortant vers d'autres fourmis et ne peut avoir plus de L_{max} liens connectés à elle, venant d'autres fourmis (arbre ayant au plus L_{max} fils par nœud). Initialement, toutes les fourmis sont placées sur le support f_0. Elles vont chacune disposer d'un seuil de similarité et d'un seuil de dissimilarité qui sont initialisés respectivement à 1 et à 0. Une fourmi f_i va se connecter sous un noeud existant de l'arbre (fourmi f_{pos}) si elle est suffisamment similaire à ce nœud, mais également suffisamment dissimilaire aux fils de ce noeud. f_i formera donc une sous classe de f_{pos} qui sera la plus dissimilaire possible aux

autres sous classes de f_{pos} (éventuellement déjà existantes). Sinon, f_i se déplacera aléatoirement dans l'arbre à la recherche d'un autre emplacement pour se fixer. Au fur et à mesure que la fourmi f_i échoue dans ses tentatives d'accrochage à la structure, elle est rendue plus tolérante afin d'augmenter ses chances de se connecter à la prochaine itération la concernant: son seuil de similarité est diminué et son seuil de dissimilarité est augmenté. Le cas particulier du support f_0 est traité comme suit: une fourmi se connecte au support si elle est suffisamment dissimilaire aux autres fourmis déjà connectées directement sur f_0. Cela signifie que l'on vient de construire une nouvelle classe au plus haut niveau de l'arbre et que cette classe doit se distinguer le plus possible des autres déjà créées. L'algorithme se termine lorsque toutes les fourmis sont connectées. Les sous arbres apparaissant au premier niveau de l'arbre, juste sous le support, seront interprétés comme étant des classes différentes. Les propriétés de l'arbre peuvent être analysées de manière visuelle et interactive (par exemple, l'erreur de classification diminue au fur et à mesure que l'on descend dans l'arbre).

Algorithme

Initialement toutes les fourmis sont sur le support et leurs seuils de similarité et dissimilarité sont respectivement initialisés à 1 et 0.
TQ il existe une fourmi f_i non connecté **Faire**
  Si sa position est le support **Alors** Cas support
  Sinon Cas fourmi
FTQ

Figure 21. Le principe de l'algorithme AntTree.

Cas du support

```
Si aucune fourmi n'est connectée au support f0
Alors connecter fi à f0
  /* une nouvelle classe */
Sinon /* f+ la fourmi connectée au support la plus similaire à fi */
        Si Sim(fi, f+) >= Ssim(fi)
        Alors Déplacer fi vers f+  /* fi est  suffisamment similaire à f+ */
        Sinon
                Si Sim (fi, f+) < Sdissim (fi)
                Alors /* fi est suffisamment dissimilaire à f+ */
                        Si place libre sur le support
                        Alors Connecter fi au support f0 /*une nouvelle classe*/
                        Sinon Diminuer Ssim (fi) et déplacer fi vers f+
                        Fsi
                Sinon Dminuer Ssim (fi) et augmenter Sdissim (fi) de fi
                /* rendre fi plus tolérante */
                Fsi
        Fsi
Fsi
```

Figure 22. Le principe du cas du support.

Cas de la fourmi

```
/* fpos la fourmi sur laquelle fi est en déplacement */
Si Sim (fi,fpos) >= Ssim (fi) Alors
   Si Sim (fi,f+)  <Sdissim(fi) Alors
      /* f+ la fourmi voisine de fpos la plus similaire à fi */
Si place libre sur fpos Alors connecter fi à fpos
Sinon Déplacer fi aléatoirement vers fk
      /* fk = 1…n, tous les éléments (fourmis ou support) connectés à fpos */
 Fsi
Sinon
Diminuer Sdissim (fi) et augmenter Ssim (fi) et déplacer fi aléatoirement vers fk
  Fsi
Sinon Déplacer fi aléatoirement vers fk
Fsi
```

Figure 23. Le principe du cas de la fourmi.

4.4. Algorithme AntClust [LAB 06]

L'algorithme AntClust s'inspire du système de reconnaissance chimique chez les fourmis. Celui-ci est connu sous le nom de "*fermeture coloniale*". Il se base sur l'apprentissage et le partage d'une odeur coloniale commune à toutes les fourmis d'un même nid. Dans cet algorithme, une fourmi artificielle est associée à un objet à classer. Chaque fourmi artificielle est capable d'apprendre une odeur coloniale grâce à un processus de rencontres aléatoires et un ensemble de règles comportementales locales. Ainsi, les fourmis d'une même colonie se partagent une odeur similaire.

Principe de fonctionnement

Dans cette méthode, chaque donnée de l'espace de départ est représentée par une fourmi artificielle et plus précisément par son génome. Tout au long des rencontres qu'elle va effectuer, la fourmi va tenter d'accorder son label et son template à son génome pour trouver la colonie qui lui ressemble le plus. Nous définissons donc les paramètres suivants pour une fourmi *i:*

- Le label $label_i$ indique le nid d'appartenance de la fourmi. Au départ, les fourmis n'appartiennent à aucun nid et donc $label_i$=0. Cette valeur évolue jusqu'à ce que la fourmi trouve le nid qui lui est le plus adéquat.
- Le template est à la fois défini par le génome $Génome_i$ de la fourmi (i. e: une donnée de l'espace de départ) et par un seuil d'acceptation noté $Template_i$. Celui-ci, fait l'objet d'un apprentissage à l'initialisation des fourmis artificielles et d'une mise à jour continue pendant la classification. Le calcul de $Template_i$ s'appuie sur l'estimation par la fourmi *i* des similarités maximales et moyennes observées lors de rencontres avec d'autres fourmis.
- L'estimateur M_i indique la réussite des rencontres de la fourmi *i*. Au départ M_i=0, puisque la fourmi *i* n'a pas encore réalisée de rencontres. M_i estime la taille du nid de la fourmi *i*, c'est-à-dire le nombre de fourmis ayant le même label que la fourmi *i*. M_i est augmenté quand la fourmi *i* rencontre des individus de son nid et est diminué dans le cas contraire.
- L'estimateur M_i^+ mesure l'intégration de la fourmi *i* dans son nid. Il est augmenté si la fourmi *i* et une autre de son nid se rencontrent et s'acceptent et est diminué sinon.

– L'âge A_i, qui au départ vaut 0, est utilisé dans les calculs de mise à jour du seuil d'acceptation $Template_i$.

- Acceptation entre fourmis: La résolution des rencontres est conditionnée par l'acceptation ou le rejet préalable des fourmis. Une fourmi accepte toutes les fourmis dont le génome est proche du sien, relativement à son seuil d'acceptation. Elle est définie comme suit:

$$Acceptation\ (i, j) = (Sim(i, j) > Template_i) \wedge (Sim(i, j) > Template_j)$$

Apprentissage du seuil d'acceptation $Template_i$:

Les fourmis artificielles estiment, au cours d'un nombre fixé de rencontres aléatoires, la similarité maximale et moyenne qu'il peut y avoir entre leur génome et celui des fourmis rencontrées. Une fourmi *i* définit alors son seuil initial d'acceptation de la manière suivante:

$$Template_i = \frac{\overline{Sim}(i,.) + \max(Sim(i,.))}{2} \qquad (1)$$

Les valeurs des similarités maximale et moyenne sont aussi remises à jour après chaque rencontre en fonction de l'âge de la fourmi. Ainsi, la valeur du *Template* est réapprise continuellement par chaque fourmi.

Pour chaque fourmi, AntClust simule une rencontre avec une autre fourmi choisie aléatoirement. L'issue de ces rencontres est déterminée par un ensemble de règles comportementales qui font évoluer les paramètres des fourmis:

***R1:* règle de création de nouveau nid.**

Si $(label_i = label_j = 0)$ et *Acceptation (i, j)* Alors créer un nouveau label $label_{new}$ et $label_i := label_{new}$, $label_j := label_{new}$.

Si *Acceptation* est faux, alors la règle ***R6*** s'applique.

***R2:* règle d'ajout d'une fourmi sans label à un nid existant.**

Si $(label_i = 0\ \&\ label_j \neq 0)$ et *Acceptation (i, j)* Alors $label_i := label_j$.

***R3*: règle de rencontre positive entre deux fourmis du même nid.**

Si $(label_i = label_j)\ \&\ (label_i \neq 0)\ \&\ (label_j \neq 0)$ et *Acceptation (i, j)* Alors Augmenter M_i, M_j, M_i^+, M_j^+.

Augmenter une variable *x* veut dire: $x := (1-\alpha)*x + \alpha$.

Diminuer une variable x veut dire: $x := (1-\alpha)*x$.

***R4:* règle de rencontre négative entre deux fourmis du même nid.**

Si $(label_i = label_j)$ & $(label_i \neq 0)$ et *Acceptation (i, j)=faux* Alors Augmenter M_i, M_j et diminuer M_i^+, M_j^+. La fourmi x $(x=i, x=j)$ qui est la moins intégrée dans son nid $(x|\ M_x^+ = Min\ (M_i^+, M_j^+))$ perd son label et n'a donc plus de nid $(Label_x:=0,\ M_x := M_x^+ := 0)$ pour essayer après de trouver un nid plus adéquat.

***R5:* règle de rencontre positive entre deux fourmis d'un nid différent.**

Si $(label_i \neq label_j)$ & *Acceptation (i,j)* Alors Diminuer M_i et M_j. La fourmi x qui a le plus petit M_x (la fourmi appartenant au nid le plus petit) change son label pour appartenir au nid de la fourmi rencontrée.

***R6*: règle par défaut.**

Si aucune autre règle ne s'applique, alors rien ne sa passe.

Algorithme: AntClust

ANTCLUST ()

(1) **Initialiser** les fourmis artificielles:
(2) $Genome_i \leftarrow i^{eme}$ objet des données à classer
(3) $Label_i \leftarrow 0$
(4) $Template_i$ est initialisé selon l'équation 1
(5) $M_i \leftarrow 0$, $M^+_i \leftarrow 0$, Ai$\leftarrow$ 0
(6) **Simuler** Nb_{ITER} itérations durant lesquelles chaque fourmi en rencontre une autre
Choisie aléatoirement
(7) **Supprimer** les nids avec moins de $P \times n$ $(P << 1)$ fourmis
(8) **Ré-affecter** chaque fourmi sans nid, au nid de la fourmi dont elle est la plus similaire.

Figure 24. Le principe de l'algorithme AntClust.

4.5. Algorithme AntPart [MAZ 04]

Cette heuristique est inspirée du comportement de fourragement (recherche de nourriture) d'une espèce de fourmis tropicale vivant au Mexique appelée *Pachycondyla Apicalis.* Ces fourmis ont un comportement et une technique de chasse simple. Les principales caractéristiques sont:

- la stratégie de recherche de proie est individuelle: chaque fourmi explore le voisinage du nid sans utiliser de marquage chimique mais en mémorisant des sites de chasse,
- La règle comportementale principale que suit une fourmi lorsqu'elle sort du nid pour chasser est de retourner systématiquement explorer le dernier site de chasse fructueux qu'elle a rencontré,
- Pour chasser, les fourmis les plus âgées s'éloignent plus du nid que les jeunes, qui ont tendance à rester dans le voisinage de celui-ci,
- Les fourmis de cette espèce étant incapables de reconstruire leur nid, celui-ci est régulièrement déménagé. Lors du déménagement, des comportements de recrutements apparaissent où une fourmi guide une de ses congénères au nouveau nid.

Principe général

Cet algorithme reproduit le comportement de ces fourmis (Pachycondyla Apicalis). Le Tableau 1 montre la façon dont a été modélisé ce comportement pour un problème de classification:

Fourmis réelles	Fourmis artificielles
Site de chasse	Classe
Rechercher une proie	Rechercher une classe pour la donnée à classer
Capture d'une proie	Trouver une classe à une donnée
Mémoriser les sites de chasses	Utilisation d'une mémoire pour chaque fourmi
Fourmis les plus âgées	Fourmis maîtres
Fourmis les plus jeunes	Fourmis ouvrières
Recrutement	Utilisation d'une mémoire commune pour toutes les fourmis.
S'éloigner du nid (pour les fourmis les plus âgées)	Améliorer la partition (pour les fourmis maîtres)

Tableau 1. Correspondance entre fourmis réelles et fourmis artificielles [MAZ 04].

Dans cet algorithme, il y'a une population de n fourmis $a_1, \ldots, a_n$ de l'espèce *Pachycondila Apicalis* et m données de type numérique ($n =< m$). Ces fourmis sont composées de n_1 fourmis maîtres et de n_2 fourmis ouvrières ($n = n_1 + n_2$). Les fourmis ouvrières sont positionnées au départ dans le nid où se trouvent les données à classer. Elles vont tenter de classer ces données en essayant de minimiser la similarité intra classe (minimiser la similarité des objets appartenant à des classes différentes) et maximiser la similarité interclasse (maximiser la similarité des objets appartenant à une même classe).

La similarité entre objets est à calculer à l'aide de fonctions qui retournent 1 si les deux objets sont identiques et 0 s'ils sont totalement différents.

De plus, chaque donnée D_i a un seuil de similarité *Ssim*(D_i) initialisé à 1 et un seuil de dissimilarité *Sdissim*(D_i) initialisé à 0 (les deux seuils sont mis à jour par la fourmi qui transporte cette donnée) permettant de définir si la donnée Di est suffisamment similaire ou suffisamment dissimilaire à une autre donnée.

Mémoire des fourmis ouvrières

Les fourmis ouvrières sont dotées de mémoire locale leur permettant de se rappeler l'emplacement des *m* objets qu'elles ont déposés dans les *s* sites qu'elles ont rencontrés. A chaque objet transporté, la fourmi compare ses caractéristiques aux caractéristiques des *m* objets de sa mémoire. Si elle trouve un objet suffisamment similaire à celui qu'elle transporte, elle se dirige vers le site de l'objet qu'elle a trouvé.

Mémoire de la colonie

La colonie est dotée d'une mémoire commune pour toutes les fourmis. Cette mémoire contient les sites que les fourmis ont visités et les objets qu'elles ont déposés (un site n'est présent qu'une seule fois dans la mémoire). Si une fourmi ouvrière ne trouve pas d'objets suffisamment similaires à celui qu'elle transporte dans sa mémoire locale, elle cherche dans la mémoire de la colonie l'objet le plus similaire à celui qu'elle transporte. Si ces deux objets sont suffisamment similaires, elle se dirige vers le site où se trouve cet objet.

Maintenant on peut donner le principe de l'algorithme AntPart:

Chaque fourmi ouvrière $f_i, i \in [1, n_1]$ quitte le nid et choisit aléatoirement une donnée D_i à transporter pour lui chercher une classe (un site). La fourmi f_i commence par visiter les sites qu'elle a rencontrés selon une stratégie choisie au préalable, en commençant la recherche dans sa mémoire locale. Si elle trouve un objet D_i^+ suffisamment similaire à D_i, elle classe l'objet D_i de telle sorte que D_i et D_i^+ appartiennent à la même classe. Si la fourmi f_i ne trouve pas d'objets suffisamment similaires à D_i dans sa mémoire, elle cherche dans la mémoire commune de la colonie l'objet D_i^+ le plus similaire à D_i.

Si D_i et D_i^+ sont suffisamment similaires, alors elle classe l'objet D_i de telle sorte que D_i et D_i^+ appartiennent à la même classe. Si D_i et D_i^+ sont suffisamment dissimilaires, alors une nouvelle classe est créée contenant uniquement la donnée D_i.

Si D_i et D_i^+ ne sont ni suffisamment similaires ni dissimilaires, la fourmi retourne au nid avec la donnée (sans la classer) et met à jour les seuils de similarité et

dissimilarité de D_i afin d'augmenter ses chances d'être classée à la prochaine itération.

Le rôle des fourmis maîtres est d'améliorer la partition obtenue par les fourmis ouvrières en essayant de reclasser les objets mal classés et en fusionnant les sites qui sont suffisamment similaires entre eux.

Pour reclasser les objets, une fourmi maître choisit aléatoirement un objet o_i le plus éloigné du centre de gravité G_{k1} du site s_k auquel cette donnée appartient. Elle cherche ensuite le centre de gravité G_{k2} d'un site s_{k2} (s_k différent s_{k2}) le plus proche à la donnée o_i. Si la distance entre o_i et G_{k2} est inférieure à la distance entre o_i et G_{k1}, alors la fourmi déplace l'objet o_i vers s_{k2}.

Algorithme AntPart

AntPart()

TQ le nid contient encore des objets **Faire**

Pour chaque fourmi f_i **Faire**

Sélectionner un objet D_i dans le nid

Chercher dans la mémoire de la fourmi f_i l'objet D_i^+ le plus similaire à D_i

Si (D_i et D_i^+ sont suffisamment similaires) **Alors**

Classer D_i dans la classe de D_i^+

Sinon

Chercher dans la mémoire de la colonie l'objet D_i^+ le plus similaire à Di

Si (D_i et D_i^+ sont suffisamment similaires) **Alors**

Classer D_i dans la classe de D_i^+

Sinon

Si (D_i et D_i^+ sont suffisamment dissimilaires) **Alors**

Créer une nouvelle classe qui contiendra l'objet D_i

Sinon

Rendre l'objet au nid et diminuer le seuil de similarité et augmenter le seuil de dissimilarité de D_i.

FSi

FSi

FSi

FPour

FTQ

Figure 25. Le principe de l'algorithme AntPart.

5. Conclusion

Dans ce chapitre, nous avons présenté les algorithmes à base de fourmis artificielles inspirés du comportement collectif des fourmis. Ces algorithmes sont intéressants pour plusieurs raisons: ce sont des systèmes auto-organisés et non centralisés qui utilisent une population de fourmis artificielles autonomes. Chaque fourmi possède un ensemble de règles comportementales qu'elle va suivre selon les situations dans lesquelles elle se trouve.

Cependant, tout comme l'ensemble des heuristiques, aucune ne donne jusqu'à présent de bons résultats dans toutes les instances du problème. Cela signifie qu'elles peuvent se révéler très intéressantes dans certains exemples de problèmes de classification, et beaucoup moins pour d'autres. C'est pourquoi il nous a été demandé de développer et de mettre en œuvre une nouvelle méthode inspirée colonies de fourmis. Nous allons donc proposer une nouvelle approche intitulée « ClusterAnts » qui sera détaillée dans le chapitre suivant.

Chapitre IV: Conception et mise en œuvre

1. Introduction

Dans ce chapitre, nous allons présenter la conception et la mise en œuvre de la méthode que nous avons proposée pour la résolution du problème de classification non supervisée. Cette approche a été intégrée dans un algorithme permettant de la tester et d'expérimenter différentes valeurs des principaux paramètres de la méthode afin de tenter de déterminer celles qui s'adaptent le mieux au problème en cours de résolution.

Dans ce qui suit, nous présentons la méthode que nous proposons, les différents modules de l'environnement que nous avons développé ainsi que ses différentes bibliothèques et classes.

2. Description du problème

Le problème peut se résumer comme suit: étant donné un ensemble d'objets, chaque objet est caractérisé par un nombre fini d'attributs. Notre but est d'arriver à affecter ces objets à des classes selon leur similitude. C'est-à-dire que deux objets appartiennent à la même classe si et seulement s'ils sont « suffisamment similaires », et deux classes distinctes doivent être « suffisamment dissimilaires ». Ce problème est un problème de classification non supervisée, puisqu'au début de l'algorithme, le nombre de classes à trouver est inconnu.

3. Architecture de l'environnement

Notre environnement a été développé sur la plateforme ***Windows,*** en langage ***C++ Builder Version 5.0.***

Le problème peut être divisé en plusieurs tâches, chaque tâche sera représentée séparément dans l'environnement. Notre environnement sera donc constitué de plusieurs modules où chaque module représente une tâche particulière comme l'illustre la Figure 26.

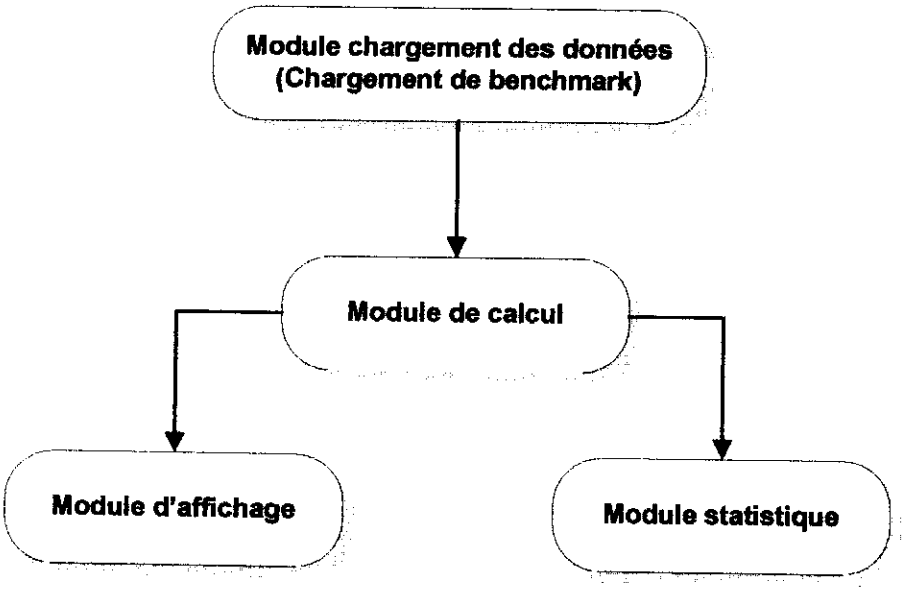


Figure 26. Architecture de l'environnement.

L'application est structurée en couches. Une couche contient un ou plusieurs modules cités ci-dessus, comme par exemple le module statistiques et le module d'affichage ils sont présents dans la couche1. Les différentes couches sont représentées dans la Figure 27.

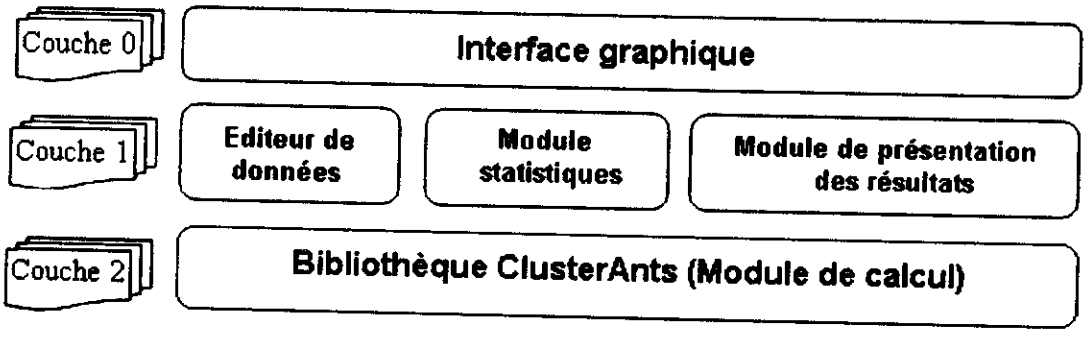


Figure 27. Structure générale de l'application.

Couche 0: Interface graphique

C'est l'interface graphique manipulée par l'utilisateur. L'application offre plusieurs fonctionnalités, à savoir:

- Chargement et visualisation des données.
- Choix des paramètres de ClusterAnts.

- Visualisation des résultats numériques et graphiques.
- Rapport détaillé de la méthode et des résultats obtenus.

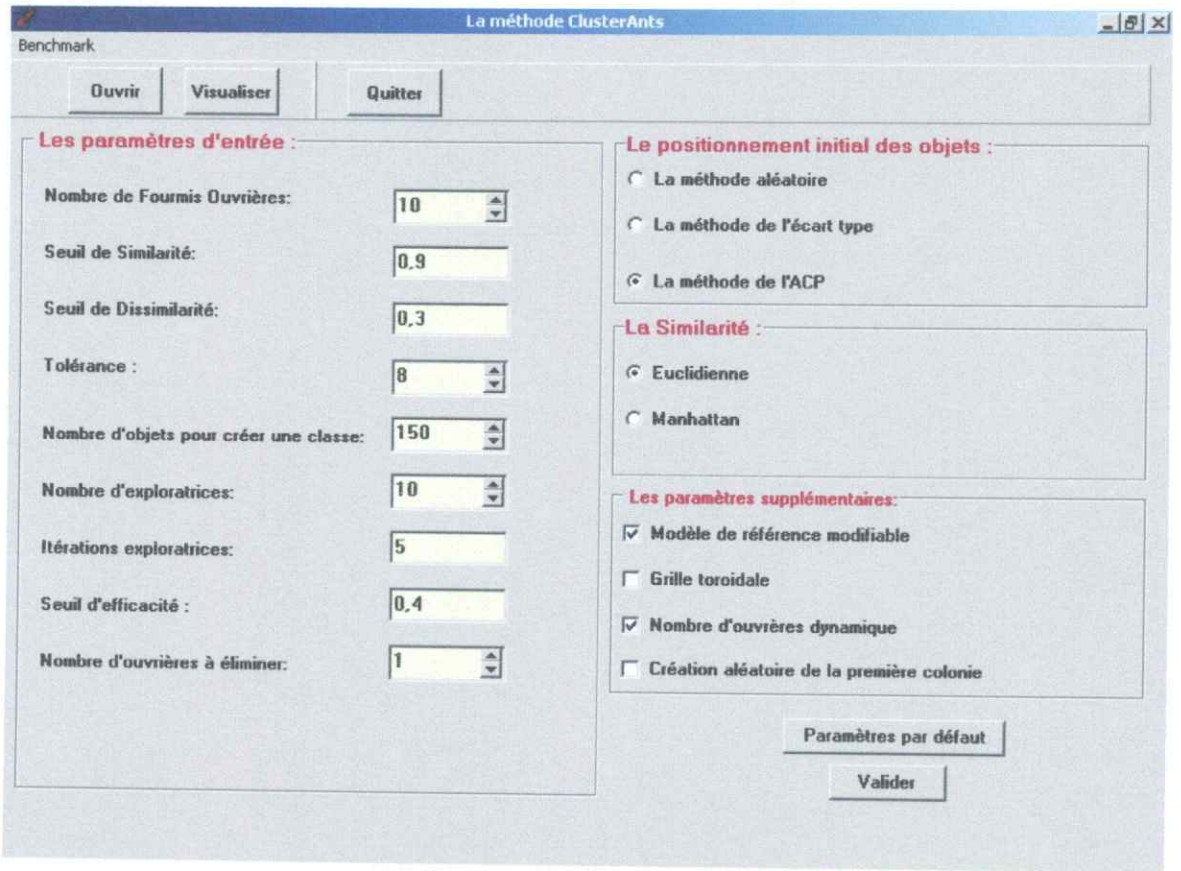


Figure 28. Interface graphique de l'application

Couche 1: Communication

Cette couche est un moyen de communication entre l'utilisateur et l'application. Elle sert de jonction entre la méthode de résolution et l'interface graphique qui assure la communication avec l'utilisateur.

Couche 2: Noyau

Dans cette couche, on trouve les différentes classes qui constituent l'application de l'algorithme de classification (ClusterAnts).

4. Module de chargement des données

Les données utilisées sont sous forme de fichiers texte, qui ont la forme suivante:

- La première ligne du fichier représente les noms des attributs des objets.
- Chaque ligne représente ensuite un objet à travers la liste des valeurs de ses attributs séparées par des blancs ou des virgules. Chaque objet est identifié par un ensemble de valeurs numériques réelles signées.

- Le benchmark peut être de type supervisé, ce qui signifie que la classification initiale est connue. La classe à laquelle appartient l'objet est alors représentée par le dernier attribut de chaque ligne.

Ce module de chargement a pour fonction de lire les données à partir des benchmarks et de les stocker dans une structure de données en mémoire afin qu'elles soient exploitables.

4.1. La mise en œuvre

Le module chargement des données (éditeur de données) appartient à la couche communication (couche1). L'éditeur de données que nous avons utilisé permet le chargement à partir d'un fichier texte.

Cet éditeur est composé d'une matrice dont les lignes représentent les objets et les colonnes représentent les attributs ou les caractéristiques de ces derniers. La dernière colonne représente le numéro de la classe à laquelle appartient chaque objet.

La figure suivante montre un exemple de données extraites d'un benchmark supervisé.

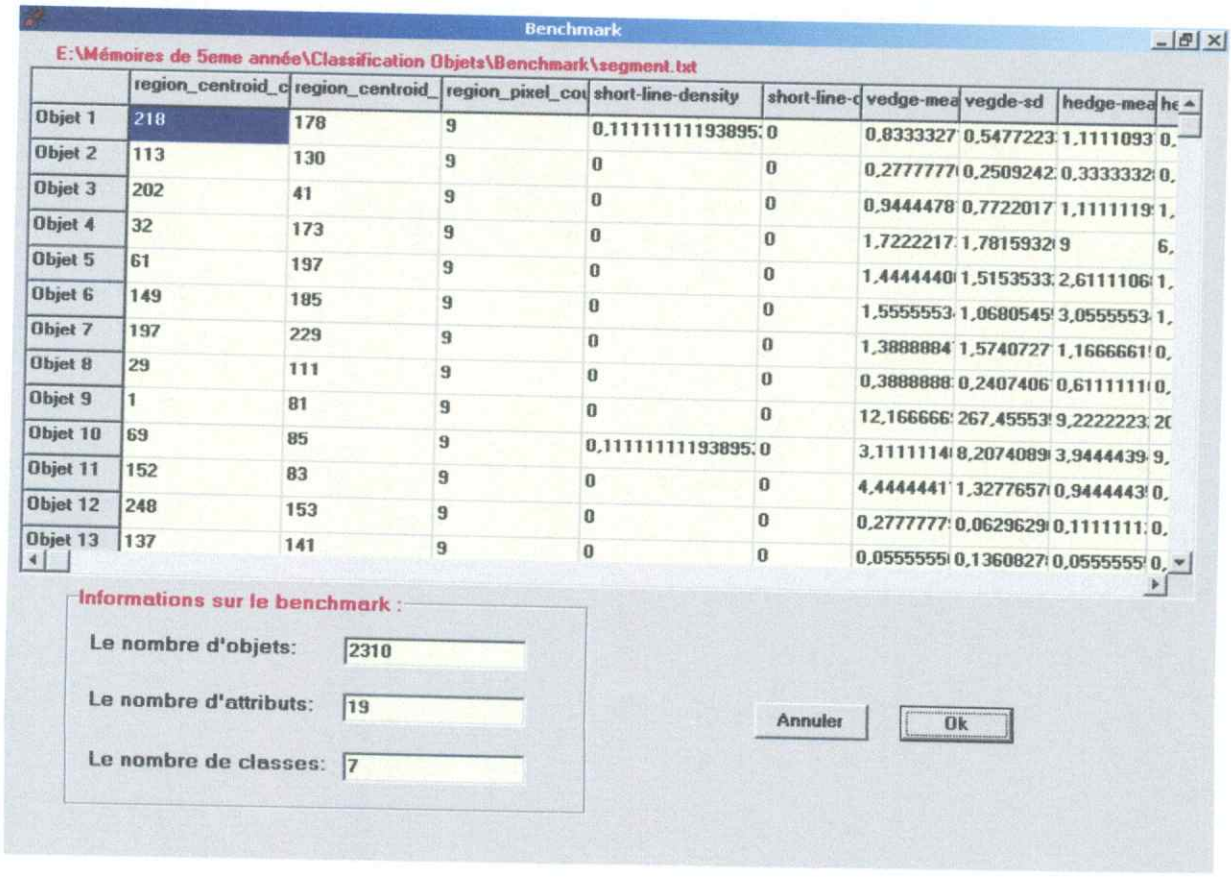


Figure 29. L'éditeur de données

5. Module de calcul

Après le chargement des données à partir d'un benchmark, la deuxième étape consiste à lancer la méthode pour la résolution du problème. La méthode de que nous avons mise en œuvre est une nouvelle méthode appelée ClusterAnts.

5.1. Inspiration biologique [DOR 99]

L'heuristique que nous avons développée s'inspire du comportement des fourmis dans les colonies pour la recherche de la nourriture. Chaque colonie a une reine unique qui détermine un modèle de référence pour cette dernière. Elle donne naissance à des fourmis ouvrières pour rechercher de la nourriture similaire au modèle de référence de la colonie. Ces dernières utilisent de la phéromone (odeur chimique unique) comme moyen de communication entre les fourmis et pour marquer le chemin entre la colonie et une zone de nourriture. On peut résumer le principe général dans les points suivants:

- La création d'une nouvelle reine donne naissance à une nouvelle colonie.
- Une colonie se compose de fourmis ouvrières, mercenaires et exploratrices.
- Lorsqu'une fourmi ouvrière trouve une nourriture similaire à son modèle de référence, elle le ramasse et laisse de la phéromone sur le chemin de retour à la colonie pour attirer les autres ouvrières de sa colonie vers cette zone de nourriture.
- Lorsqu'une zone est épuisée c-à-d qu'il n'y a plus de nourriture dans cette zone, l'ouvrière dépose de la phéromone négative pour signaler aux autres ouvrières de sa colonie que cette zone est sèche et que ce n'est pas la peine de chercher dedans.
- Si une ouvrière rencontre une nourriture différente de son modèle de référence, elle fait appel à une fourmi mercenaire qui va ramasser cette nourriture et faire le tour des autres colonies pour voir s'il y a une colonie qui peut recevoir cette nourriture.
- Les exploratrices partent à la recherche de nouveaux sites de chasses et informent les ouvrières dans le cas où des sites existent.

5.2. Modélisation algorithmique

Nous allons adapter les principes de cette communauté de fourmis au problème de classification non supervisée comme suit:

- Chaque classe est représentée par une colonie de fourmis différenciée par le type de nourriture qu'elle accepte et une phéromone qui la distingue des autres colonies.
- La recherche de la nourriture chez les fourmis ouvrières correspond à la recherche d'objets similaires à leur modèle de référence.
- Les objets sont répartis initialement sur un plan (grille) à deux dimensions.
- La reine est la référence de sa colonie, elle possède le modèle de référence de son espèce. La position de la reine sur le plan représente la position du nid.
- Les fourmis ouvrières engendrées par la reine correspondent dans notre approche à des fourmis qui se spécialisent dans la recherche d'objets similaires au modèle de référence de la colonie (classe).
- La phéromone est un chemin qui va guider et attirer les fourmis ouvrières de la même colonie vers un voisinage supposé être riche en objets recherchés.
- Les fourmis exploratrices interviennent périodiquement pour chercher de nouveaux endroits contenant des objets similaires aux objets de leur colonie.
- La fourmi mercenaire cherche dans l'ensemble des colonies celle qui peut recevoir l'objet non encore classé qu'elle transporte.
- De la phéromone négative est déposée sur les zones arides, indiquant aux ouvrières de même type la saturation d'un site (l'emplacement doit être abandonné).
- Chaque position est représentée par une case qui peut contenir à la fois plusieurs objets, une fourmi ainsi que des quantités de phéromones de chaque colonie.
- Périodiquement, on fait un balayage de toutes les cases de la grille pour diminuer les taux de phéromones (l'aspect évaporation de phéromone).
- Chaque déplacement d'objets vers un nid est l'équivalent d'un classement de cet objet.

5.3. Principe général

Soit un ensemble de N objets à classifier avec j attributs représentant chacun des objets. Au départ, il faut représenter ces objets sur un plan 2D. Il faut ensuite créer une première reine qui donne naissance au premier nid. La position de ce nid est celle du premier objet à classer, et qui sera considéré comme modèle de référence du nid.

La reine va créer à son tour un nombre P de fourmis ouvrières. Ce sont ces fourmis qui vont à la recherche des objets. Si une ouvrière rencontre un objet similaire au modèle de référence de sa classe; elle le ramène à son nid et laisse une trace (phéromone) sur le chemin de retour indiquant aux autres ouvrières de sa colonie la présence d'objets recherchés. Dans le cas où elle ne trouve pas d'objets dans cette région, elle dépose une phéromone négative informant que cette piste est à écarter. Si elle rencontre un objet non similaire à ceux qui caractérisent la colonie, elle fait appel à une fourmi mercenaire. La mercenaire fait alors le tour des autres colonies afin de tenter d'en trouver une qui convienne à cet objet. Si elle en trouve, elle classe l'objet sinon, deux cas de figure se présentent :

- S'il est suffisamment dissimilaire à toutes les colonies existantes, alors un nouveau nid est créé avec une nouvelle reine et l'objet sera attribué à ce nouveau nid.
- Dans le cas où la fourmi mercenaire trouve que l'objet n'est ni assez similaire ni assez dissimilaire à toutes les colonies visitées, elle le dépose dans une classe nommée « classe fictive ».

Périodiquement, un autre type de fourmis est lancé: « les fourmis exploratrices » pour essayer de découvrir de nouveaux sites de chasse. Ces fourmis diffèrent des ouvrières par le fait qu'elles se déplacent plus loin du nid et ont une plus grande autonomie. De la même manière que les ouvrières, elles déposent de la phéromone pour marquer les pistes vers les éventuels nouveaux sites de chasse trouvés. Après un certain temps, la phéromone disparaît (s'évapore).

5.3.1. Positionnement initial des objets sur le plan

Soit X un ensemble de vecteurs (objets) X_1, X_2, X_3, ..., X_n où chaque X_i est un vecteur de dimension *d*.

$$X = \begin{pmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_n \end{pmatrix} = \begin{pmatrix} & \text{Att}_1 & \text{Att}_2 & & \text{Att}_d \\ & x_{11} & x_{12} & \cdots & x_{1d} \\ & x_{21} & x_{22} & \cdots & x_{2d} \\ & . & . & \cdots & . \\ & . & . & \cdots & . \\ & . & . & \cdots & . \\ & x_{n1} & x_{n2} & \cdots & x_{nd} \end{pmatrix}$$

Figure 30. Matrice représentative des objets.

On positionne les objets sur un plan 2D sous forme d'une grille carrée. Cette dernière est toroïdale ou non (selon le choix de l'utilisateur) avec un nombre de cases par ligne égal à $L = \sqrt{2N}$ [MON 00].

Il existe plusieurs techniques pour le positionnement des objets sur un plan 2D: la méthode aléatoire, l'ACP (Analyse en Composantes Principales) et l'écart type.

- **La technique aléatoire:** est la plus simple pour le positionnement des objets sur un plan à deux dimensions, elle génère deux nombres aléatoires pour chaque objet qui vont représenter ses cordonnées.
- **La méthode de l'ACP (Analyse en Composantes Principales):** l'ACP est une méthode souvent utilisée pour la réduction de dimensions. Elle permet d'extraire des données d'un nuage de points multidimensionnels. Les objets sont identifiés par une liste d'attributs. Il s'agit d'extraire à partir de cette liste deux coordonnées qui vont expliquer cette dernière (garder le maximum d'informations de cette liste dans les deux coordonnées) pour avoir les objets les plus similaires dans des positions proches.

Les étapes à suivre pour effectuer une analyse en composantes principales pour réduire la dimension des X_i à m (pour notre cas m=2) sont:

- Calculer le vecteur moyen: $\mu = \frac{1}{n}\sum_{i=1}^{n} X_i$
- Calculer la matrice Z de dimension $n \times d$ comme suit: $Z_i = X_i - \mu$
- Calculer la dispersion: $S = Z^t Z$ avec Z^t la transposée de Z.
- Calculer les vecteurs propres $\{e_1, e_2, \ldots, em\}$ de dimension d correspondant aux m plus grandes valeurs propres de S.
- Soient $e_1, e_2, \ldots, e_m$ les colonnes de la matrice $E = [e_1, e_k, \ldots, e_m]$

- De là, on calcule les vecteurs Y_i qui représentent la meilleure approximation des X_i dans un sous-espace de dimension *m, (m = 2)*:

$$Y_i = Z_i\ E$$

*** Calcul des *m (m=2)* plus grandes valeurs propres et les vecteurs propres associés:**

Soit *A* une matrice $n \times n$ symétrique réelle. On note $\lambda_1, \lambda_2, ..., \lambda_n$ ses valeurs propres (supposées réelles) et $v_1, v_2, ..., v_n$ les vecteurs propres correspondants.

On suppose que les λ_i sont des modules (grandeurs) distincts et on les ordonne par modules décroissants:

$$|\lambda_1| > |\lambda_2| > .... > |\lambda_n|$$

Soit $x^{(0)}$ un vecteur de $\boldsymbol{R}^n$ et la suite $x^{(k)}$ définie par la relation de récurrence:

$$x^{(k+1)} = \frac{Ax^{(k)}}{\left\|Ax^{(k)}\right\|}$$ où $\|\ \|$ est la norme euclidienne sur $\boldsymbol{R}^n$.

1- Calculer v_1:

$\lim_{k \to +\infty} x^{(k)} = v_1$ (v_1 est le vecteur propre associé à la plus grande valeur propre λ_1)

2- calculer λ_1 :

Pour calculer λ_1 on peut simplement utiliser la définition: $Av_1 = \lambda_1 v_1$.

Il est cependant préférable d'utiliser la formule suivante: $\lambda_1 = \frac{v_1' A v_1}{v_1' v_1}$.

3- On forme la matrice $A_1 = A - \lambda_1 v_1 v_1'$ (v_1' vecteur ligne transposé du vecteur colonne v_1)

Les valeurs propres de A_1 sont $\lambda_2, \lambda_3, ..., \lambda_n, 0$. On calcule v_2 à partir de A_1 de la même façon qu'on a calculé v_1 à partir de *A*.

- **La méthode de l'écart type:** Soit X_i un objet de dimension *d*. Soient (z_1, z_2) les cordonnées de X_i sur la grille. Les variables z_1, z_2 seront calculées en utilisant l'écart type qui explique la dispersion entre les objets:

$$\text{Ecart - Type}(y_1, y_2, \ldots, y_n) = \sqrt{\frac{1}{n}(\sum_{i=1}^{n} y_i^2) - (\bar{y})^2}$$

Où $\bar{y}$ est la moyenne des objets $(y_1, y_2, \ldots, y_n)$.

Donc on aura la formule suivante:

$$Ecart - Type(Att_j) = \sqrt{\frac{1}{n}(\sum_{i=1}^{n} x_{ij}^2) - (\bar{x}_j)^2}$$

Maintenant z_1, z_2 prendront les valeurs des deux plus grands écarts-type des attributs.

5.3.2. Fonction de similarité

Soit $O = \{O_1, O_2, \ldots, O_n\}$ l'ensemble des *N* objets à classifier. Chaque objet contient *d* attributs: $O_i = (O_{i1}, O_{i2}, \ldots, O_{id})$ La fonction de similarité calcule la similarité entre deux objets O_i et O_j et renvoie une valeur comprise entre 0 et 1:

- Quand les deux objets sont identiques, la similarité entre eux est égale à 1.
- Quand les objets sont complètement dissimilaires, la similarité est égale à 0.

La formule générale de similarité est: $Sim(O_i, O_j) = 1 - (\frac{1}{d}\sum_{k=1}^{d} |O_{ik} - O_{jK}|^r)^{\frac{1}{r}}$ **(1)**

Afin que le calcul de similarité soit correct et significatif, il faut au départ normaliser les attributs des objets dans l'échelle [0,1]. Nous allons utiliser cette formule pour normaliser chaque valeur des attributs de chaque objet.

X_i normalisé= $X_i - X_{min} / X_{max} - X_{min}$ où X_{max}, X_{min} représentent respectivement la plus grande valeur et la petite valeur pour un attribut donné.

Pour le calcul de similarité nous allons utiliser deux formes connues de la formule générale à savoir:

a. Similarité de Manhattan

C'est (1) à condition de fixer la valeur de *r* à 1:

$$Sim(O_i, O_j) = 1 - \frac{1}{d}\sum_{k=1}^{d}|O_{ik} - O_{jK}|$$

b. Similarité euclidienne

C'est (1) en fixant le facteur *r* à 2:

$$Sim(O_i, O_j) = 1 - \sqrt{\frac{1}{d}\sum_{k=1}^{d}(O_{ik} - O_{jk})^2}$$

- **Les seuils de similarité et de dissimilarité**

Chaque donnée D (objet) est dotée d'un seuil de similarité *Sim* (D) et d'un seuil de dissimilarité *Dissim* (D) qui sont initialisés par l'utilisateur.

5.3.3. Les fourmis ouvrières

Ce sont des fourmis qui parcourent la grille dans le voisinage du nid, à la recherche d'objets similaire à l'objet de référence de leur nid. Si elles en trouvent, deux cas se présentent:

- l'objets est assez similaire au modèle de référence (la similarité entre l'objet et la référence est supérieur au seuil de similarité) alors elles le ramènent au nid et dépose de la phéromone positive sur le chemin de retour. Le compteur des objets classés est incrémenté.

- l'objet n'est pas suffisamment similaire à la référence, elles font alors appel à une fourmi mercenaire.

Dans le cas où l'ouvrière ne trouve pas d'objets dans une zone donnée, au bout d'un certain temps, elle dépose de la phéromone négative pour informer les autres fourmis qu'il s'agit d'une zone sèche et donc qu'il est inutile de continuer à y chercher de la nourriture.

5.3.3.1 Nombre de fourmis ouvrières

Initialement, P fourmis ouvrières sont créées (P est un paramètre qui peut être configuré par l'utilisateur). Il est modifié dynamiquement et peut diminuer selon le nombre d'objets trouvés et selon un certain seuil d'efficacité des fourmis qu'on fixe.
Ainsi, si le nombre d'objets trouvé par une colonie au bout d'un certain nombre d'itérations est inférieur à un seuil d'efficacité paramétrable par l'utilisateur, alors le

nombre d'ouvrières est diminué afin de ne pas consommer trop de temps de calcul sur une colonie qui se trouve sur un site pauvre en objets.

5.3.3.2 Déplacement des fourmis ouvrières

La fourmi se déplace sur une grille à deux dimensions avec un pas de déplacement qui est égal à 1 (elle passe d'une case à une case voisine). Pour passer d'une case à une autre, la fourmi vérifie d'abord la présence d'un objet dans les cases de son voisinage. Si un objet existe, elle se déplace directement vers cette case.
Si elle ne trouve pas d'objets dans son voisinage immédiat, elle recherche une trace de phéromone dans son voisinage immédiat. S'il en existe, elle choisit la case où il y a la quantité la plus importante de phéromone et elle se déplace vers cette dernière. Le même processus est alors répété.
S'il n'existe aucune piste de phéromone, alors une case non encore visitée est choisie aléatoirement dans le proche voisinage de la fourmi et celle-ci se déplace vers elle.

5.3.3.3 Mémoire de la fourmi ouvrière

Chaque fourmi ouvrière dispose d'une mémoire locale lui permettant de se rappeler le chemin parcouru depuis son nid. A chaque départ du nid, la fourmi réinitialise sa mémoire (la vide) pour sauvegarder les positions des cases à parcourir. Cette mémoire sert à indiquer le chemin de retour pour déposer de la phéromone dans le cas d'un classement d'objet, ainsi que les cases déjà visitées afin de ne pas les parcourir à nouveau.

5.3.4. Tolérance de la fourmi

C'est le nombre de déplacements maximums que peut faire une fourmi sans trouver un objet qui ressemble au modèle de référence de sa colonie. Si la tolérance est atteinte, cela signifie que la fourmi n'a pas trouvé d'objet identique à sa référence dans sa zone de chasse. Elle dépose alors de la phéromone négative sur les cases parcourues afin d'indiquer que le site de chasse est stérile, puis elle rentre au nid.

5.3.5. Les fourmis mercenaires

Les mercenaires sont appelées par les ouvrières lorsque celles-ci trouvent un objet qui est différent de leur référence. Elles ont pour rôle de parcourir les autres colonies

pour voir si l'une d'elle dispose d'une référence identique à l'objet trouvé. Pour cela, elles disposent d'une mémoire commune qui contient les positions de tous les nids créés jusque là.

Si la mercenaire trouve une colonie qui accepte l'objet qu'elle transporte (l'objet est similaire au modèle de référence de la colonie), l'objet est classé dans cette colonie.

S'il n'est similaire à aucune colonie et qu'il est suffisamment dissimilaire à toutes les colonies existantes, une nouvelle classe (colonie) est créée, et l'objet est déposé dedans.

Si l'objet n'est ni suffisamment similaire, ni suffisamment dissimilaire à aucune colonie, il est déposé dans une classe nommée « classe fictive » et qui regroupe les objets traités mais non encore classés. Après que la mercenaire ait déposé l'objet soit dans sa colonie soit dans la classe fictive, elle meurt.

5.3.6. Les fourmis exploratrices

Elles font un travail similaire à celui des ouvrières, sauf qu'elles explorent de nouveaux sites de chasse. Elles n'ont pas de tolérance et peuvent donc aller beaucoup plus loin que les ouvrières. De plus, leur trajectoire est différente de celle des ouvrières car elles ont des trajectoires rectilignes.

Dans le cas où une exploratrice trouve un objet similaire à son modèle de référence, elle le ramène au nid et dépose de la phéromone positive sur le chemin de retour.

De la même manière que l'ouvrière, lorsqu'une exploratrice trouve un objet différent à sa référence, elle fait appel à une fourmi mercenaire.

Dans le cas où elle ne trouve aucun objet au bout d'un certain nombre d'itérations, elle dépose de la phéromone négative pour signaler qu'il s'agit d'une zone vide.

5.3.7. Le dépôt de la phéromone

La phéromone est le moyen de communication entre les fourmis de la même colonie. Les fourmis s'échangent des messages sur:

- Une zone fertile en nourriture (objet à classer).
- Une zone où il n'existe aucune nourriture.

Dans notre approche, nous utilisons deux types de phéromone:

- La phéromone positive: elle est déposée par une ouvrière ou une exploratrice lorsqu'elle trouve un objet similaire et le ramène vers son nid. La phéromone est déposée sur toutes les cases se trouvant entre la position où l'objet a été trouvé et le nid (le chemin mémorisé par la fourmi).
- La phéromone négative: si la fourmi ouvrière ou une exploratrice ne trouve aucun objet au bout d'un certain nombre de déplacements dans une zone donnée, elle dépose de la phéromone négative dans son voisinage pour indiquer que cette zone est sèche.

5.3.8. Le traitement dans la classe fictive

Périodiquement, tous les objets, sont comparés avec ceux des différentes colonies (classes):

- Si l'objet traité est similaire à la référence d'une colonie, il est affecté à cette classe.
- Si aucune colonie ne correspond à cet objet, il est conservé dans la classe fictive.

Après avoir terminé le traitement sur la grille (il n'y a plus d'objets sur la grille), on procède à la même comparaison mais cette fois entre les objets de la classe fictive eux même. L'un des cas suivants peut se présenter:

- Des objets identiques sont trouvés dans la classe fictive elle même et leur nombre est supérieur ou égal au seuil de création d'une classe[1], alors on procède à la création d'une nouvelle classe contenant ces objets.
- S'il n'existe pas d'objets identiques dans la classe fictive ou que leur nombre est inférieur au seuil cité ci-dessus, tous les objets restants sont alors affectés aux classes qui leur sont les moins distantes.

5.4. Algorithme de l'approche proposée

Les étapes de l'algorithme ClusterAnts se résument dans l'organigramme suivant:

[1] Seuil de création d'une classe: nombre minimum d'objets identiques pour créer une nouvelle classe.

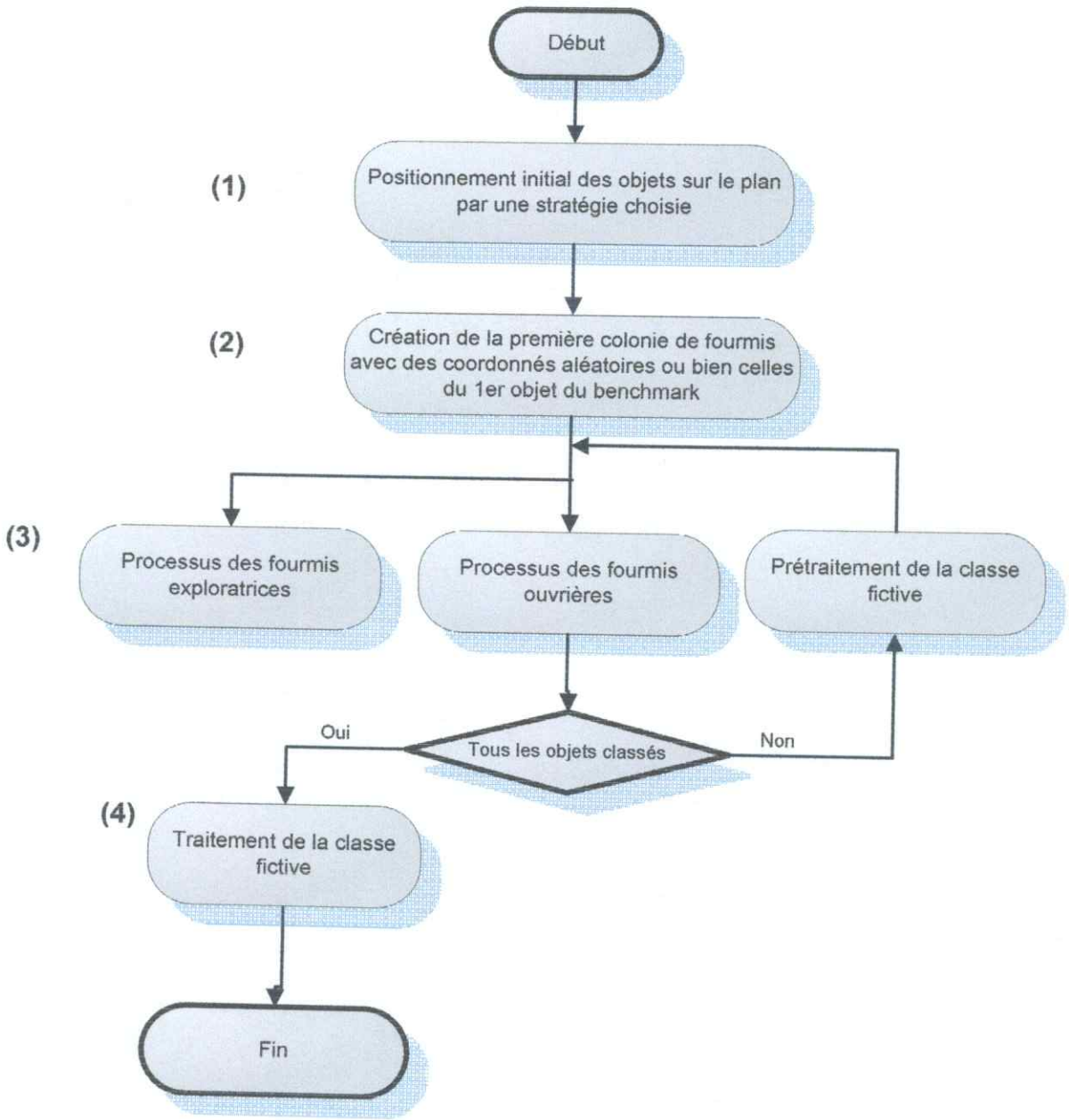


Figure 31. Organigramme de l'algorithme ClusterAnts.

Les étapes de l'algorithme ClusterAnts sont les suivantes

(1) la première étape est de positionner les objets sur un plan 2D selon la méthode choisie par l'utilisateur: la méthode aléatoire, l'ACP (Analyse en Composantes Principales), ou la méthode de l'écart type.

(2) On commence par la création de la première colonie, sa position est aléatoire ou bien celle du premier objet sur la liste (le benchmark).

(3) Cette étape regroupe le déroulement des différents processus de l'algorithme à savoir: le processus des fourmis ouvrières, processus des fourmis exploratrices

(Processus des fourmis mercenaires se fait à l'intérieur des processus des fourmis ouvrières et exploratrices) et le prétraitement de la classe fictive.
(4) Cette étape est dédiée au traitement de la classe fictive.

Dans ce qui suit nous allons détailler les processus cités ci-dessus.

a. Processus des fourmis ouvrières

Les étapes du processus des fourmis ouvrières se résument dans l'organigramme suivant:

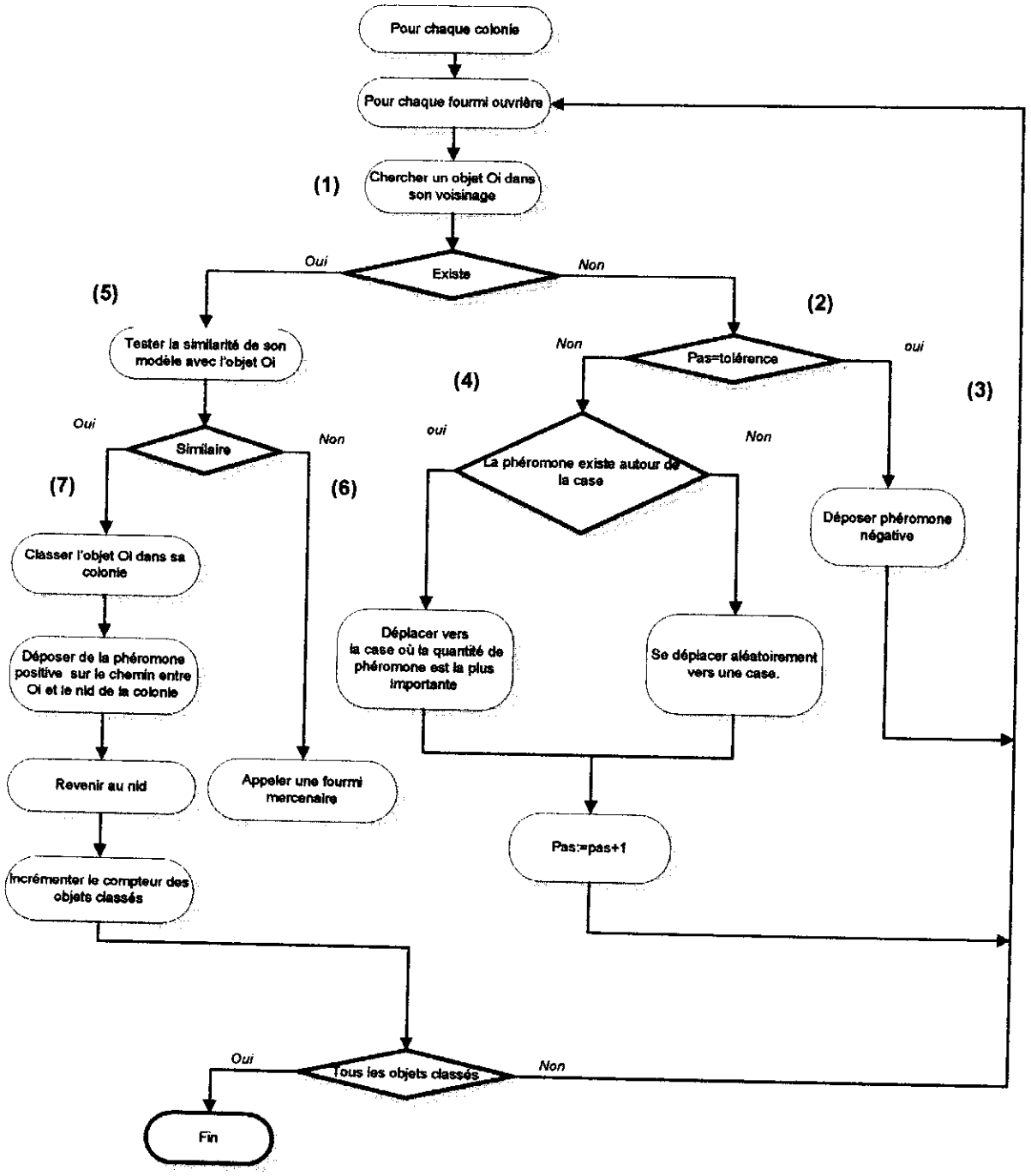


Figure 32. Organigramme du processus des fourmis ouvrières.

Etapes du processus des fourmis ouvrières

(1) Chaque fourmi ouvrière va vérifier l'existence d'un objet dans son voisinage (les huit cases qui l'entourent).

(2) Dans le cas où elle ne trouve pas d'objets dans son voisinage alors elle vérifie si le nombre de pas qu'elle a effectué n'est pas égale à sa tolérance.

(3) Si elle trouve que son nombre de pas est égal à sa tolérance, alors elle dépose de la phéromone négative pour signaler que cette zone ne contient pas d'objets similaires à celui de sa colonie.

(4) Si le nombre de pas est inférieur à sa tolérance, alors elle se déplace vers la case où il y a la plus importante quantité de phéromone positive, en incrémentant son nombre de pas. Dans le cas où il n' y a pas de la phéromone dans son voisinage. La fourmi se déplace alors au hasard vers l'une des cases qui l'entourent, tout en incrémentant son nombre de pas aussi.

(5) Si l'ouvrière trouve un objet dans son voisinage, alors elle teste sa similarité avec son modèle de référence.

(6) Si l'objet est non similaire à son modèle de référence, alors elle fait appel à une fourmi mercenaire.

(7) Si l'objet est similaire à son modèle de référence, alors elle le classe dans sa colonie, dépose de la phéromone positive sur le chemin de retour à son nid pour signaler aux autres ouvrières la présence d'objets dans cette zone et enfin incrémente le compteur d'objets classés.

b. Processus des fourmis mercenaires

Les étapes du processus des fourmis mercenaires se résument dans l'organigramme suivant:

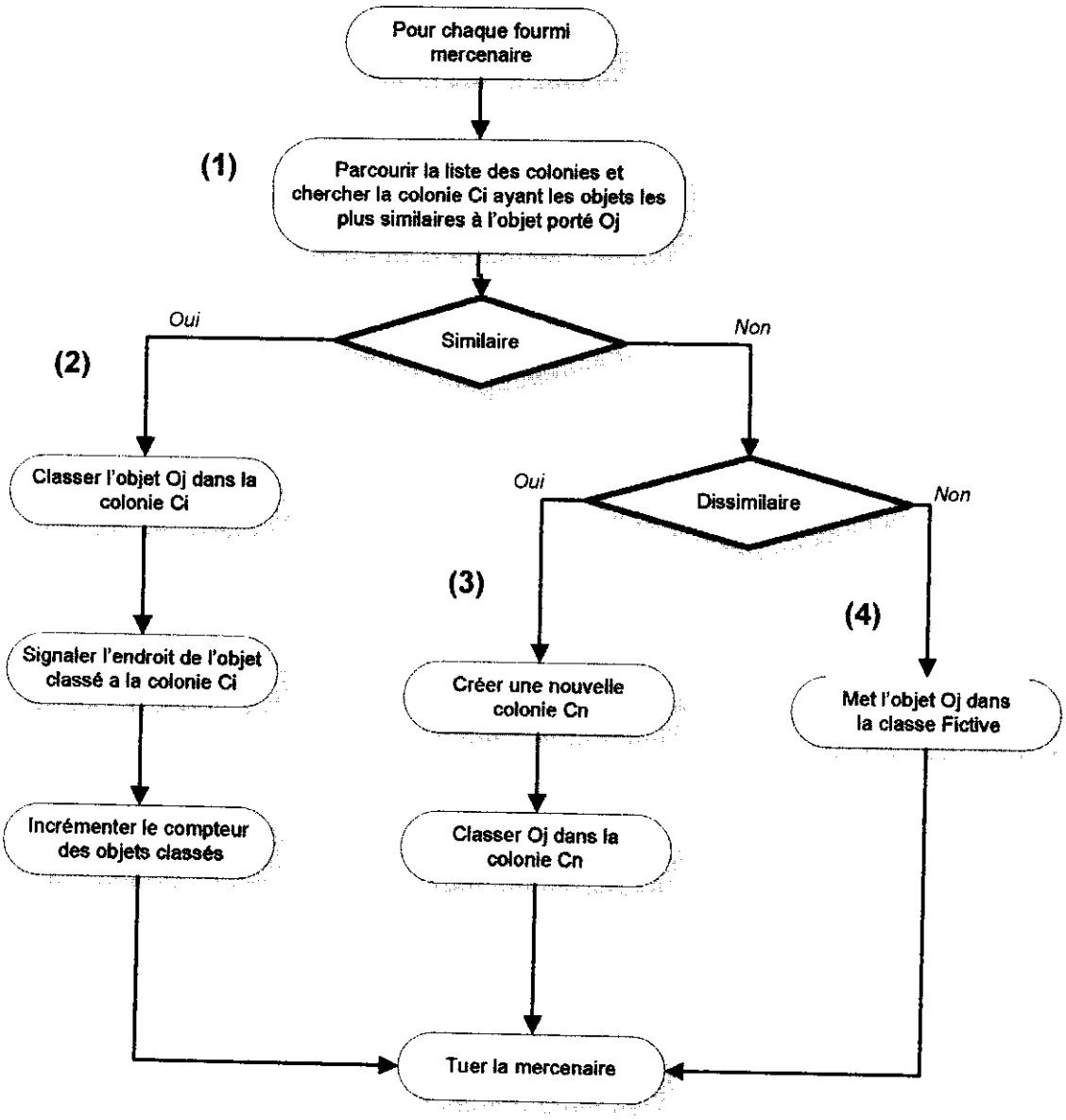


Figure 33. Organigramme du processus des fourmis mercenaires.

Etapes du processus des fourmis mercenaires

(1) Elle parcourt toutes les colonies pour trouver celle qui possède les objets similaires à l'objet traité.

(2) Si l'objet traité est similaire à la référence de la colonie courante, alors l'objet est classé dans la colonie, et de la phéromone est déposée sur le chemin menant de l'emplacement où se trouvait l'objet vers l'emplacement de la colonie. Enfin, le compteur d'objets classés est incrémenté.

(3) Si aucune colonie similaire n'est trouvée, et que l'objet est totalement dissimilaire de toutes les colonies existantes, alors une nouvelle colonie est créée et l'objet y est classé.

(4) Dans le cas où aucune colonie similaire n'est trouvée mais que l'objet n'est pas suffisamment dissimilaire de toutes les colonies (il est entre les seuils de similarité et de dissimilarité), l'objet est placé dans la classe fictive. A la fin du traitement, la fourmi mercenaire est tuée.

c. Processus des fourmis exploratrices

Les étapes du processus des fourmis exploratrices se résument dans l'organigramme suivant:

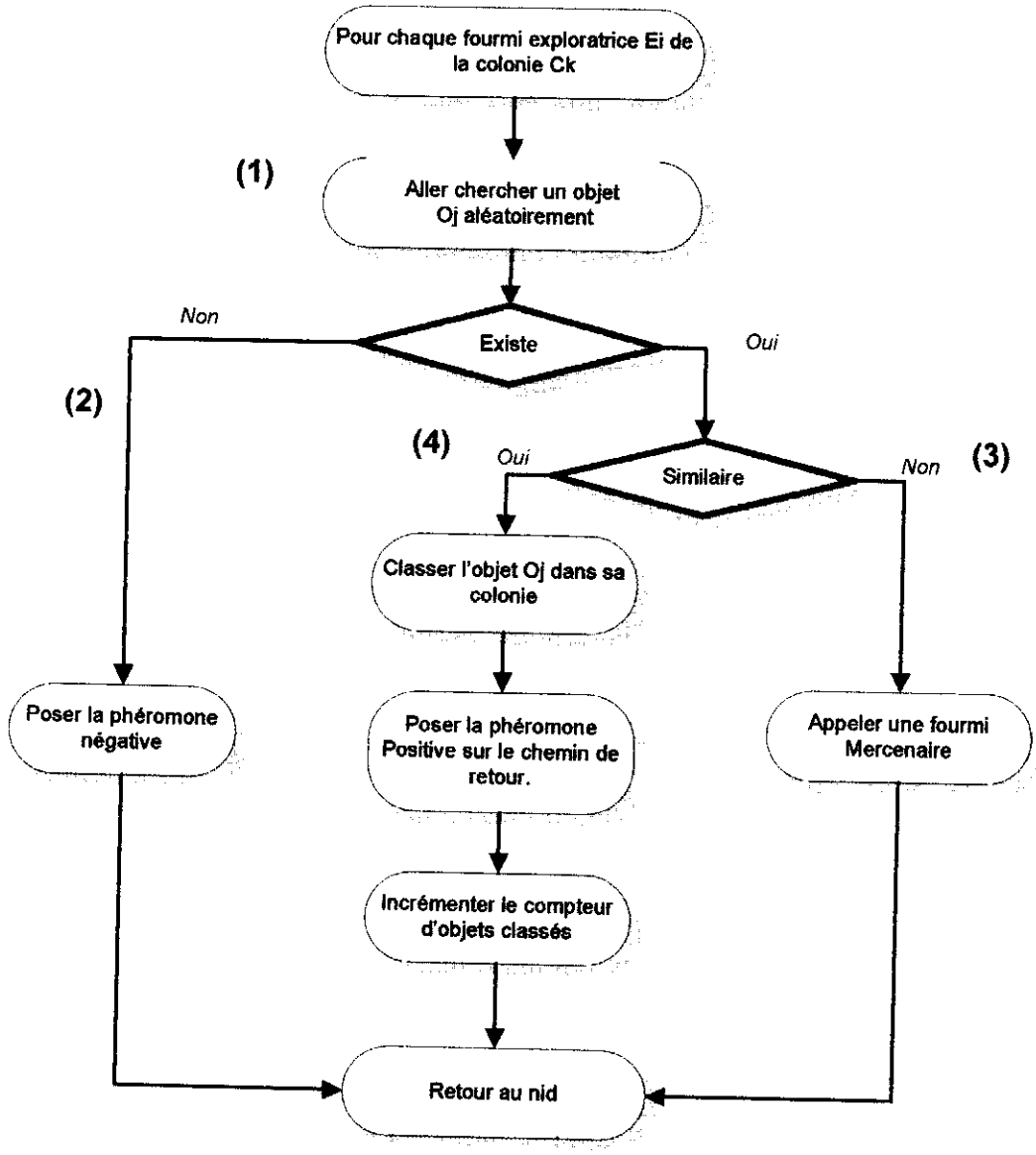


Figure 34. Organigramme du processus des fourmis exploratrices.

Etapes du processus des fourmis exploratrices

(1) L'exploratrice cherche dans l'espace de la grille des objets qui se trouvent dans des zones isolées ou non encore visitées.

(2) Si elle ne trouve pas d'objets alors elle dépose de la phéromone négative pour signaler aux fourmis ouvrières que c'est une zone stérile, et qu'il est inutile de continuer la recherche dans cette zone. Elle retourne alors au nid.

(3) Si elle trouve un objet mais qu'il n'est pas similaire à son modèle de référence, elle fait appel à une fourmi mercenaire, puis elle retourne au nid.

(4) L'objet qu'elle trouve un objet similaire à son modèle de référence, elle classe cet objet dans sa colonie, met de la phéromone positive sur le chemin de retour vers son nid et le compteur des objets classés est incrémenté.

d. Traitement de la classe fictive

La classe est traitée de la façon qui est résumée dans l'organigramme suivant:

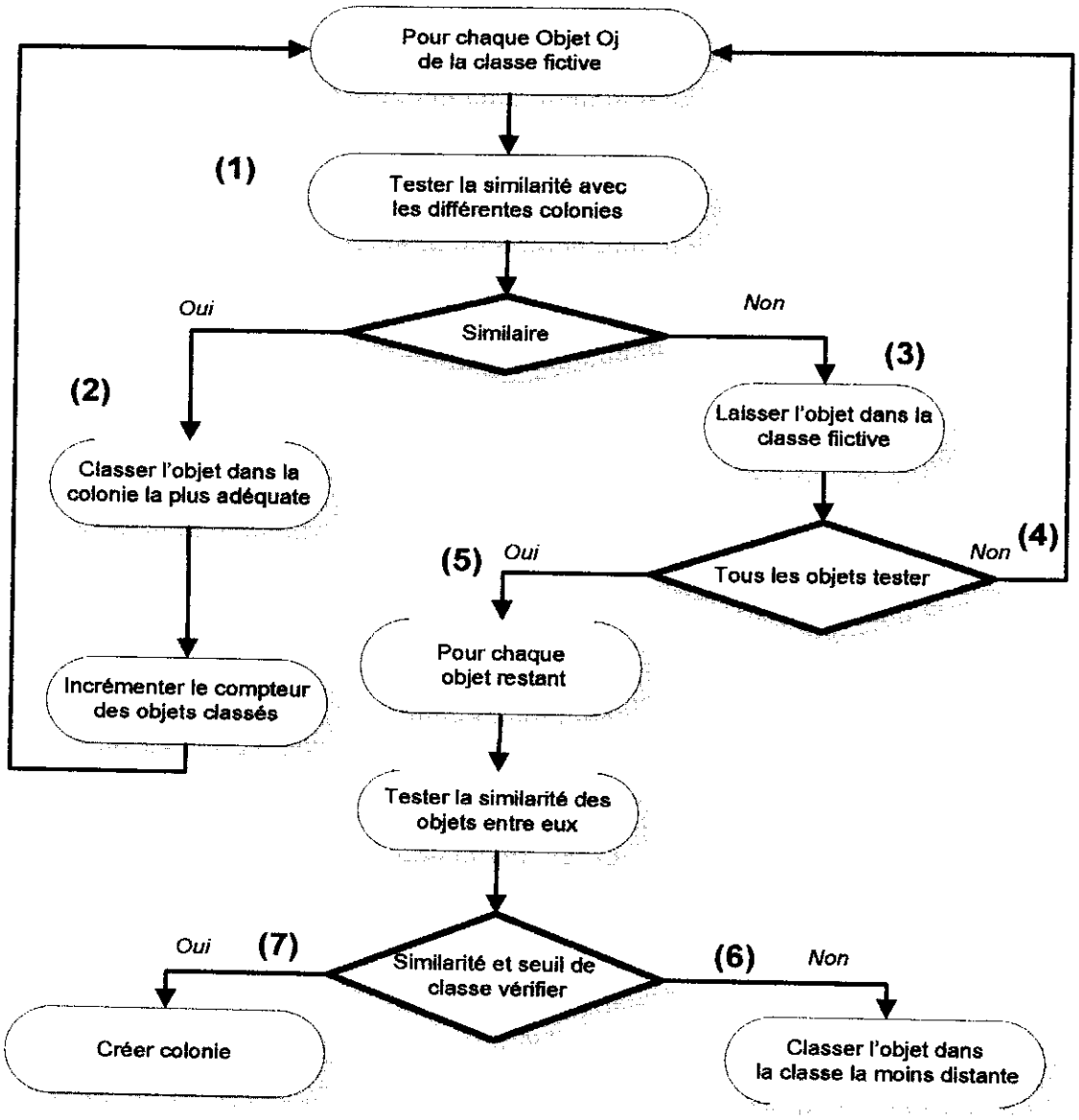


Figure 35. Organigramme du traitement de la classe fictive.

Etapes du traitement de la classe fictive

(1) On prend chaque objet de la classe fictive l'un après l'autre et on teste sa similarité avec les modèles des colonies existantes (c'est le prétraitement de la classe fictive qu'on a cité dans la Figure 35 qui représente les étapes de l'algorithme « ClusterAnts »).

(2) Si on trouve une similitude entre l'objet et le modèle d'une colonie, alors on le classe dans cette dernière et on incrémente le compteur des objets classés.

(3) On ne trouve pas de classe similaire à cet objet, alors on le remet dans la classe fictive.

(4) Si les objets du plan n'ont pas été tous classés alors on refait le même processus autrement dit on retourne à (1) et on refait le même traitement en prenant un autre objet de la classe fictive.

(5) Si tous les objets de la grille sont classés, donc il ne reste à classer que les objets de la classe fictive. On prend alors un à un les objets de la classe fictive et on les compare (tester la similitude) avec les autres objets de cette classe.

(6) Si l'objet pris n'a pas d'objet similaire dans la classe fictive ou que le nombre d'objets similaires n'atteint pas le seuil de création d'une nouvelle classe, alors on le classe dans la classe la moins distante qui existe déjà.

(7) Si on trouve un nombre N d'objets similaires à notre objet et que ce nombre est supérieur ou égal au seuil de création d'une nouvelle classe, alors on crée une nouvelle classe pour ces objets.

5.5. La mise en œuvre

Ce module est composé d'un ensemble de classes et structures nécessaires pour implémenter et mettre en œuvre l'algorithme que nous avons proposé.

- **Classe Plan**

Le plan est la structure qui identifie les positions, manipule les déplacements des objets ainsi que les déplacements des différents types de fourmis. Il est sous forme d'une grille à deux dimensions, qui à son tour est composée de plusieurs cases. Chaque case est une instance de la classe « CaseAP », cette classe comprend:

- Une liste d'objets (une case peut contenir plusieurs objets).
- Une liste de fourmis ouvrières.
- Une liste d'instances de la structure « phéromone», et chaque phéromone est composée d'un identifiant de sa colonie ainsi que du taux de phéromone qui a été déposé dans cette case.

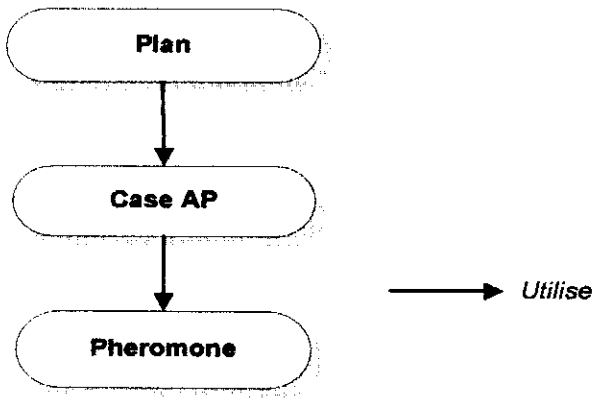


Figure 36. Diagramme de la classe Plan.

- **Classe fourmi**

La fourmi est l'agent principal du processus car c'est elle qui va à la recherche des objets. Elle est caractérisée par une position sur la grille et la colonie à laquelle elle appartient.

Afin de retourner à la colonie sans se perdre, la fourmi est dotée s'une mémoire qui garde la trace du chemin parcouru.

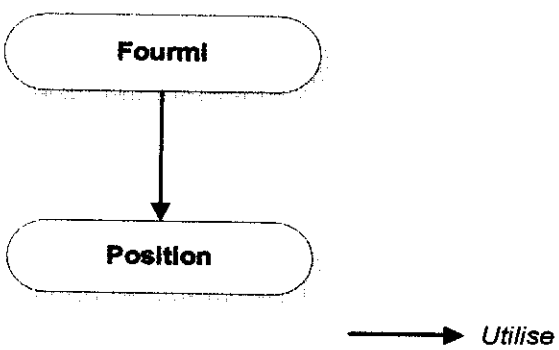


Figure 37. Diagramme de la classe Fourmi.

La fourmi peut alors faire plusieurs actions, à savoir:

- Changer de position sur la grille.
- Ramasser un objet.
- Classer un objet.
- Appeler une fourmi mercenaire.
- Déposer une quantité de phéromone positive sur un chemin.

- Déposer une quantité de phéromone négative dans une zone.

➢ **Classe Exploratrice**

L'exploratrice est qualifiée d'informatrice. Elle se charge de trouver de nouvelles zones de nourriture (objets).

Chaque fourmi exploratrice est caractérisée par:

- Une position sur la grille.
- L'objet qu'elle porte.

➢ **Classe ClusterAnts**

C'est la classe qui contient les procédures principales de la méthode *ClusterAnts* telles que :

TraitementOuvriere (c'est la procédure qui traite les ouvrières de chaque colonie) ; *TraitementExploratrices* (exécute les programmes des fourmis exploratrices crées dynamiquement) et le traitement de la classe fictive.

Elle fait appelle à toutes les classes décrites précédemment.

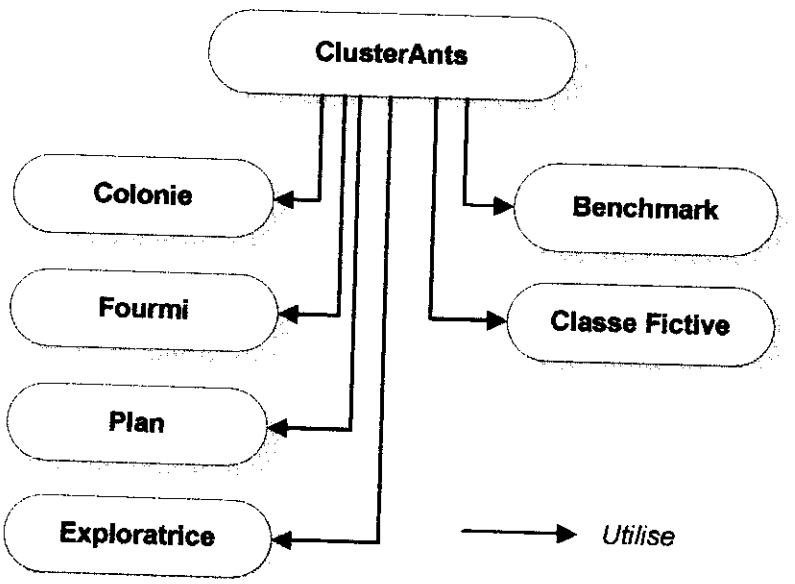


Figure 38. Diagramme de la classe ClusterAnts.

6. Module statistique

Ce module permet d'étudier les performances de l'algorithme par le calcul du taux d'erreurs. Pour cela, on utilise la mesure d'erreurs suivante:

Si, pour chaque objet o_i on connaît sa classe d'origine $c(o_i)$ et la classe obtenue par l'algorithme ClusterAnts est $c'(o_i)$, l'erreur de classification E_c est calculée de la façon suivante [MON 00]:

$$E_c = \frac{2}{N(N-1)} \sum_{(i,j)\in\{1,\ldots,N\}^2, i<j} \varepsilon_{ij}$$

Où:

$$\varepsilon_{ij} = \begin{cases} 0 & \text{Si } \left(c(o_i)=c(o_j) \wedge c'(o_i)=c'(o_j)\right) \vee \left(c(o_i)\neq c(o_j) \wedge c'(o_i)\neq c'(o_j)\right) \\ 1 & \text{Sinon} \end{cases}$$

Cette erreur considère tous les couples d'objets possibles et augmente à chaque fois que deux objets n'ont pas été classés ensemble par la méthode utilisée alors qu'ils faisaient partie de la même classe à l'origine (dans le benchmark supervisé) et réciproquement.

6.1. La mise en œuvre

Le module statistique appartient à la couche communication (couche1). Il offre un ensemble d'outils statistiques et graphiques pour l'étude et l'analyse des résultats obtenus après chaque exécution, ainsi que l'influences des différents paramètres et leurs effets sur l'évolution de l'algorithme (rapidité et convergence).

La figure suivante illustre un type d'affichage graphique qui représente le taux d'affectation des objets pour chaque classe ainsi que différents résultats finaux.

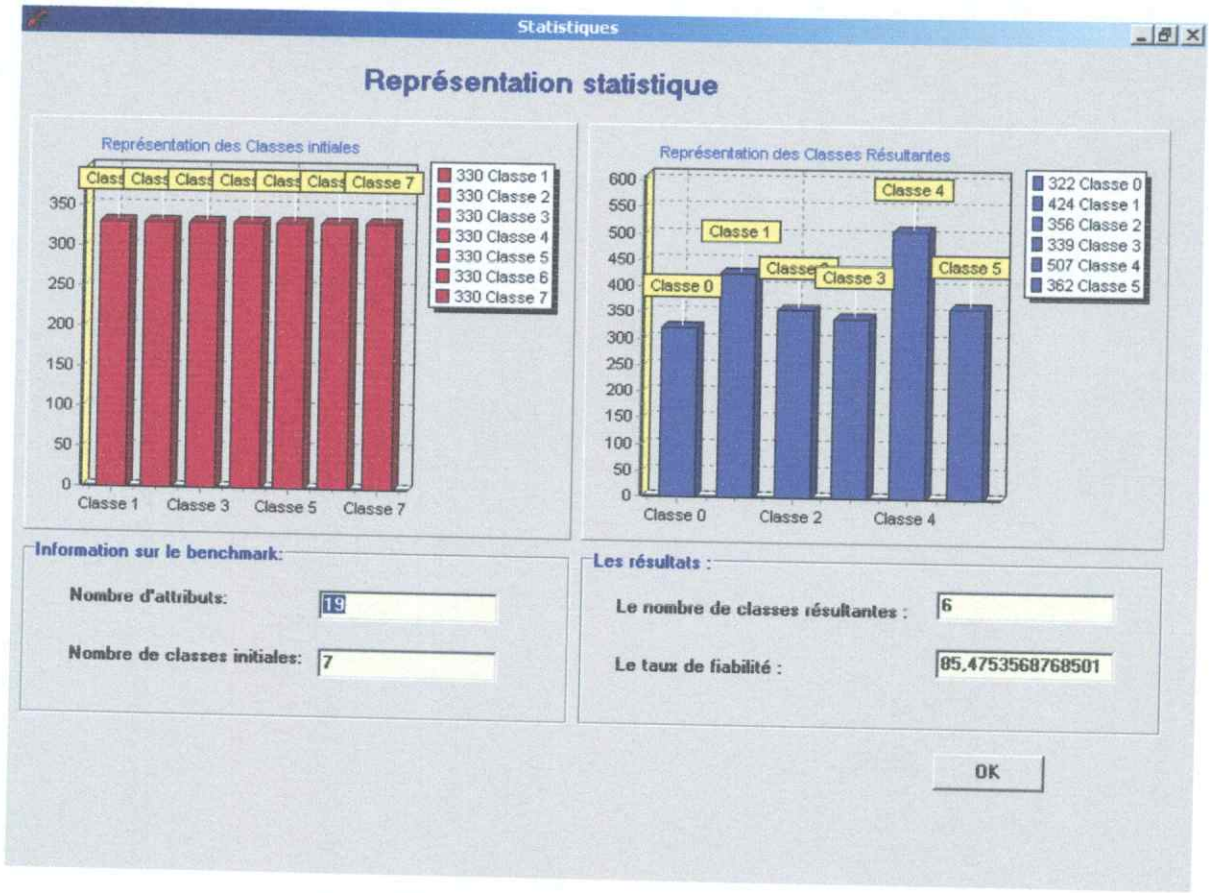


Figure 39. Représentation graphique des résultats

7. Module d'affichage

Afin de bien comprendre l'exécution de l'algorithme ClusterAnts, on a mis en place un affichage graphique qui va permettre de connaître l'emplacement des objets, les postions des nids et le déplacement des fourmis.

7.1. La mise en œuvre

Ce module est consacré au suivi des affectations des données pendant et à la fin de l'exécution d'une méthode. Il se compose de deux parties:

La Figure 40 représente la fenêtre qui indique les position des objets et leur affectation. Cette fenêtre est propre à la méthode *ClusterAnts*. La deuxième *(Fig 4. 6)* donne un rapport détaillé de l'exécution de l'algorithme sur un benchmark donné.

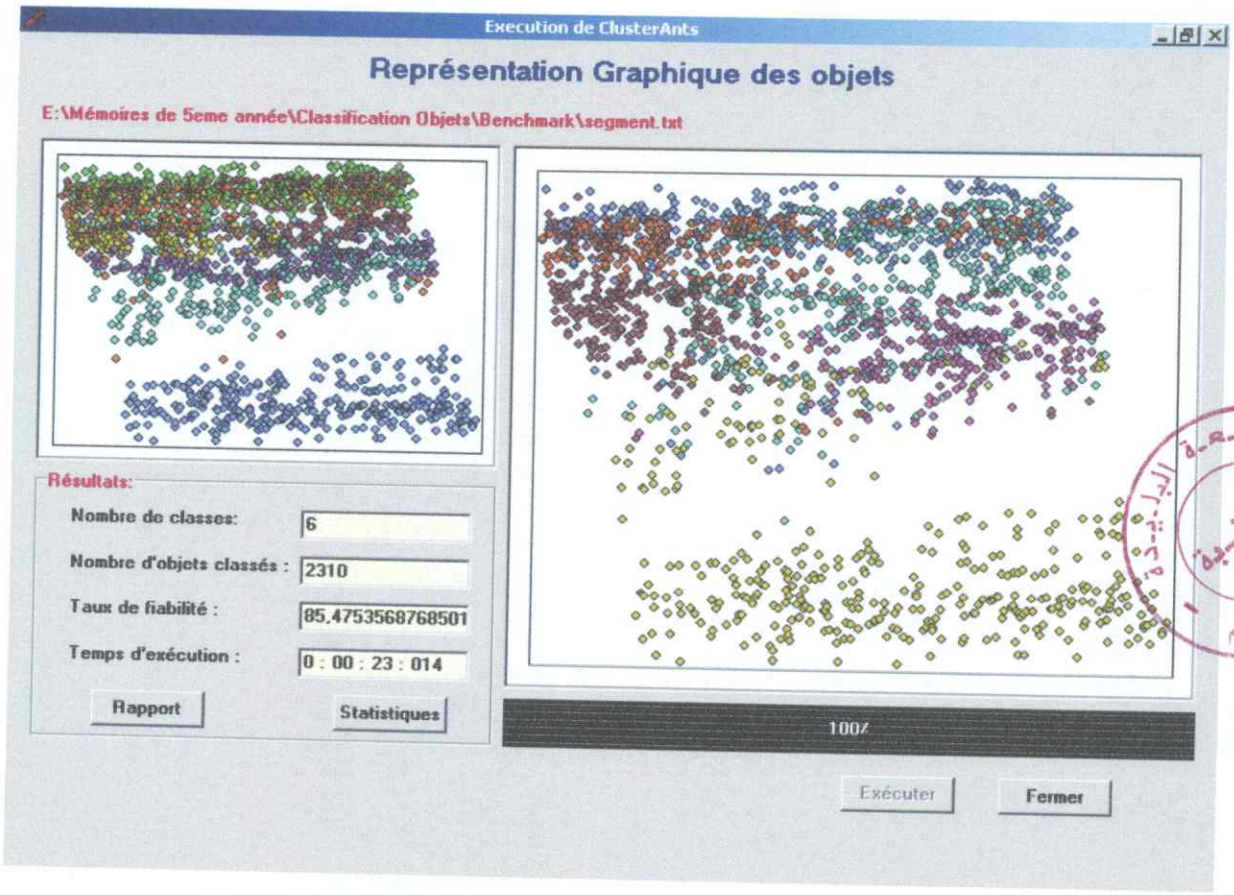


Figure 40. Représentation graphique des objets ainsi que des résultats.

Afin de voir le comportement de l'algorithme, un ensemble d'informations est retourné dans un rapport d'exécution illustré en Figure 41.

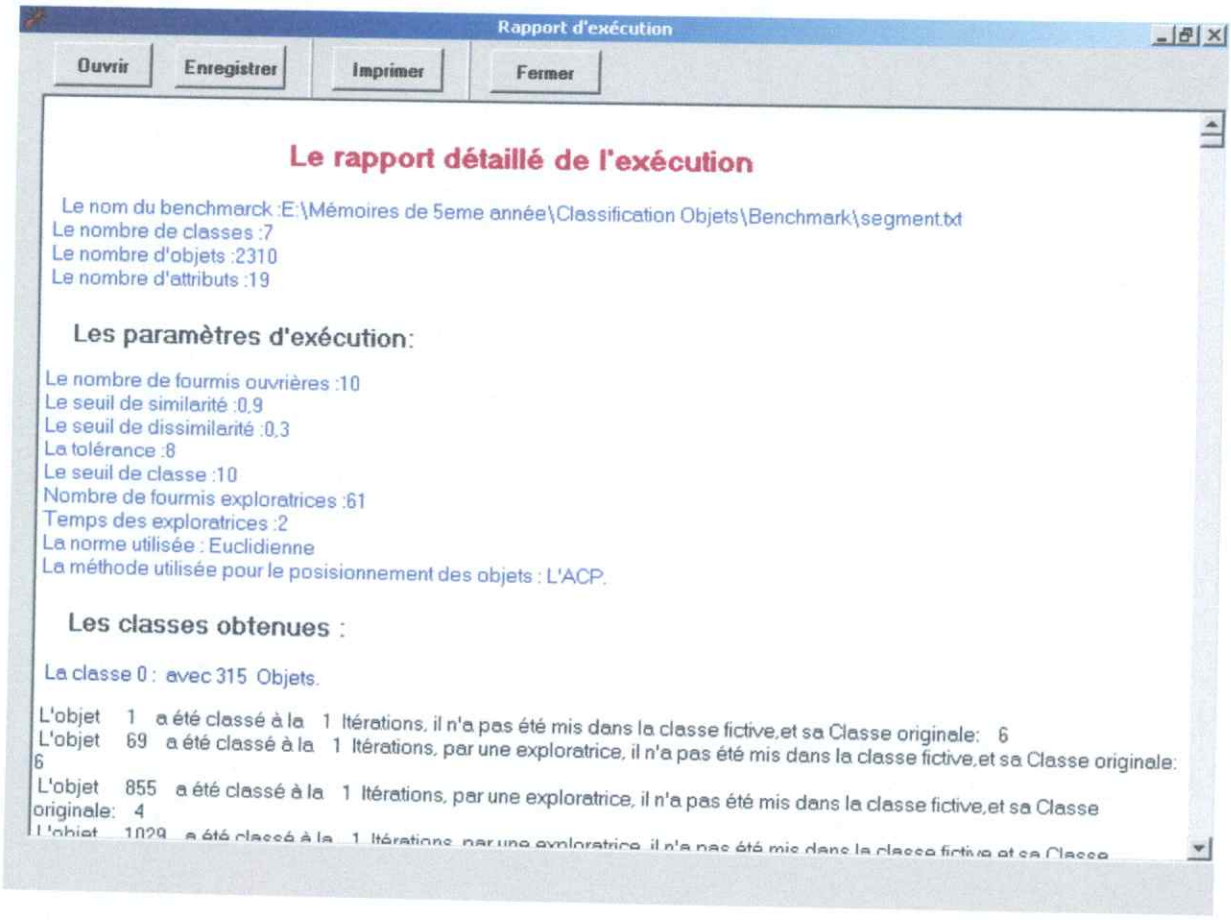


Figure 41. Rapport détaillé de l'exécution.

8. Conclusion

Dans ce chapitre, nous avons présenté la conception et la mise en œuvre de l'environnement que nous avons développé pour mettre en œuvre la méthode que nous avons proposée afin de tenter de résoudre le problème de classification non supervisée, ainsi que le schéma fonctionnel de notre application avec les différents modules qui la composent.

Afin de tester la méthode que nous avons proposée, nous allons procéder à une série de tests et tenter d'interpréter les résultats obtenus dans le prochain chapitre.

Chapitre V: Tests et résultats

1. Introduction

Dans ce chapitre, nous allons tester notre application sur trois benchmarks pour voir le comportement de l'algorithme. Tous les résultats présentés sont une moyenne de cinq essais.

Tous les tests ont été effectués sur un PC standard (Intel Pentium IV 1.7 GHz avec 624 Mo de RAM).

2. Jeux de données

Notre étude est effectuée sur des bases de données réelles issues du *Machine Learning Repository*. Ces bases de données sont supervisées (pour chaque objet on connaît la classe initiale) afin de pouvoir évaluer la qualité du partitionnement que nous obtenons.

Les caractéristiques principales des benchmarks utilisés sont résumées dans le Tableau 2.

Nom du Benchmark	Nombre d'exemples	Nombre d'attributs	Nombre de classes
Diabetes	768	8	2
Segment	2310	19	7
Soybean	48	35	4

Tableau 2. Caractéristiques des différents benchmarks.

3. Exécution de l'algorithme

Avant de procéder aux tests, nous allons montrer le positionnement initial des objets sur la grille avec la méthode de l'ACP. Les figures suivantes montrent le positionnement des objets des benchmarks pris ci-dessus comme jeux de données.

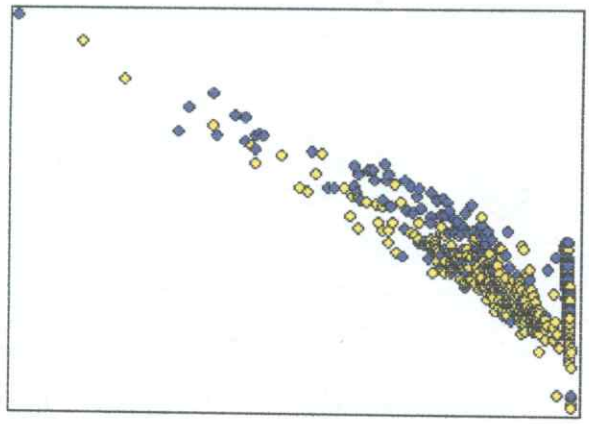


Figure 42. Positionnement des objets de Diabetes.

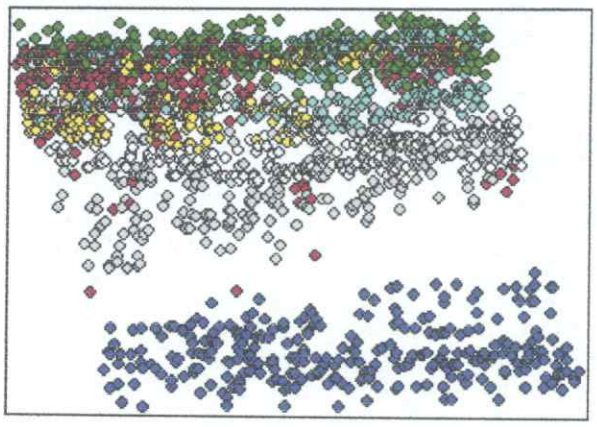


Figure 43. Positionnement des objets de Segment.

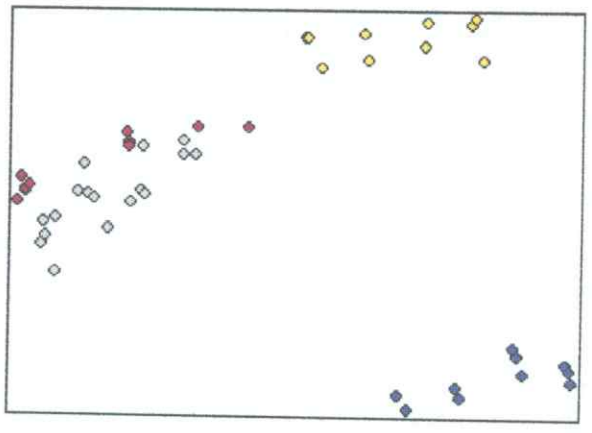


Figure 44. Positionnement des objets de Soybean.

Dans ce qui suit, nous allons présenter les résultats obtenus dans des tableaux. Nous allons ensuite interpréter les résultats.

Pour cela, nous avons utilisé les abréviations suivantes:

SSim: Seuil initial de similarité.

SDis: Seuil initiale de dissimilarité.

Nbf: Nombre de fourmis par colonie.

Tol: Tolérance des fourmis pour déposer une phéromone négative.

Nbc: Nombre d'objets minimum pour créer une classe depuis la classe fictive.

Nex: Nombre d'exploratrices.

Iex: Nombre d'itérations pour lancer les exploratrices.

Cal: Création de la première colonie à une position aléatoire.

Tor: Grille toroïdale.

Nbd: Nombre dynamique de fourmis ouvrières.

NC: Nombre de classes (nombre de colonies à la fin de l'exécution).

TF: Taux de fiabilité de la partition (TF = 1 – Ec ; Ec erreur de classification citée précédemment).

TExe: Temps d'Exécution de *ClusterAnts* sans prendre en compte le temps d'affichage.

Pour l'exécution de l'algorithme nous avons utilisé: la similarité l'Euclidienne, la première colonie est à la position du premier objet du benchmark, la grille est non toroïdale, le nombre de fourmis ouvrières est inchangeable et la tolérance **Tol** = 8.

3.1. Le benchmark « Segment »

Benchmark	**Seg I**	**Paramètres**						**Résultats**		
		SSim	SDis	Nbc	Nbf	Nex	Iex	NC	TF	TExe
Segment	Seg1	0.9	0.3	30	10	5	10	17	84,24	06:37:025
	Seg 2	0.9	0.3	100	10	5	10	10	83,12	06:42:964
	Seg 3	0.9	0.3	150	10	5	10	6	82,79	07:42:005

Tableau 3. Résultats d'exécution sur «Segment » en modifiant le Nbc.

Benchmark	**Seg II**	**Paramètres**						**Résultats**		
		SSim	SDis	Nbc	Nbf	Nex	Iex	NC	TF	TExe
Segment	Seg1	0.9	0.3	150	10	20	5	7	79,38	01:04:993
	Seg 2	0.9	0.3	150	10	40	5	6	81,77	00:34:977
	Seg 3	0.9	0.3	150	10	40	2	6	78,80	00:21:957

Tableau 4. Résultats d'exécution sur «Segment » en modifiant le Nex et le Iex.

Dans le tableau suivant nous allons utiliser la similarité de Manhattan.

Benchmark	**Seg III**	**Paramètres**						**Résultats**		
		SSim	SDis	Nbc	Nbf	Nex	Iex	NC	TF	TExe
Segment	Seg1	0.9	0.3	150	10	40	5	6	84,13	00:21:901
	Seg 2	0.94	0.3	150	10	40	5	7	80,15	00:37:047
	Seg 3	0.876	0.3	150	10	40	5	4	75,03	00:20:910

Tableau 5. Résultats d'exécution sur «Segment » en utilisant la similarité de Manhattan.

Dans le tableau suivant, les paramètres sont les suivants:

SSim: 0.9, **SDis**: 0.3, **Nbf**: 10, **Nbc:** 150, **Nex:** 40 et **Iex:** 5

Benchmark	**Seg IV**	**Paramètres supplémentaires**				**Résultats**		
		Tor	Cal	Tol	Ndd	NC	TF	TExe
Segment	Seg1	Oui	Non	8	Non	7	82,05	00:27:047
	Seg 2	Oui	Non	50	Non	7	81,92	00:28:073
	Seg 3	Non	Oui	8	Non	7	84,11	00:30:933
	Seg 4	Non	Non	8	Oui	7	79,33	00:33:995
	Seg 5	Oui	Oui	8	Oui	7	79,77	00:28:984

Tableau 6. Résultats d'exécution sur «Segment » en modifiant Tor, Cal, Tol et Ndd.

3.2. Le benchmark « Diabetes »

Dans « Dia3 » nous utilisons une grille toroïdale et la première colonie est créée aléatoirement.

Benchmark	**Dia**	**Paramètres**						**Résultats**		
		SSim	SDis	Nbc	Nbf	Nex	Iex	NC	TF	TExe
Diabetes	Dia 1	0.9	0.3	35	10	5	10	2	50,89	00:07:964
	Dia 2	0.9	0.3	35	10	40	5	2	51,23	00:02:945
	Dia 3	0.9	0.3	35	10	40	5	2	51,73	00:02:983

Tableau 7. Paramètres et résultats d'exécution sur «Diabetes ».

3.3. Le benchmark « Soybean »

Dans « Soy 2 » nous utilisons la similarité de Manhattan et dans « Soy 3 » nous utilisons la similarité de Manhattan, la première colonie est créée aléatoirement et la grille est toroïdale.

Benchmark	Soy	Paramètres						Résultats		
		SSim	SDis	Nbc	Nbf	Nex	Iex	NC	TF	TExe
Soybean	Soy 1	0.9	0.3	2	10	20	5	3	63,10	00:00:010
	Soy 2	0.9	0.3	6	10	20	5	4	85,83	00:00:009
	Soy 3	0.9	0.3	6	10	20	5	4	96,80	00:00:008

Tableau 8. Paramètres et résultats d'exécution sur «Soybean ».

3.4. Interprétation des résultats

D'après les résultats obtenus dans les tableaux précédents, on peut extraire les remarques suivantes:

- A travers *Tableau 3*, nous constatons que le nombre d'objets pour créer une classe depuis la classe fictive (seuil de classe) influe énormément sur le nombre de classes obtenues et cela en raison du fait que la plupart des objets sont placés dans la classe fictive (car leur similarité avec les modèles de référence des colonies se situe entre le seuil de similarité et le seuil de dissimilarité). En général, nous pouvons dire que le seuil du nombre d'objets pour créer une nouvelle classe qui semble le plus intéressant doit être proportionnel au nombre d'objets dans ce benchmark. Par exemple: dans le cas de « Segment » avec un seuil de classe égal à 150, les résultats obtenus sont acceptables.
- Nous remarquons que le nombre d'exploratrices ainsi que le nombre d'itérations pour lancer ces exploratrices influent considérablement sur le temps d'exécution. Avec 5 exploratrices et 10 itérations pour les lancer (*Tableau 3*), le temps d'exécution était de 6min37sec.Dans le tableau 4, lorsque nous avons augmenté le nombre d'exploratrices à 40 et diminué le nombre d'itérations pour leur lancement à 2, le temps d'exécution est devenu 21sec957msec. L'explication de l'influence des exploratrices est qu'elles vont beaucoup plus loin que les ouvrières qui sont limitées par la tolérance. Dans

le cas d'un grand benchmark, la grille peut être de très grande taille, les ouvrières mettant alors beaucoup de temps pour atteindre les dernières cases de la grille.

- D'après le *Tableau 5,* la fonction de similarité influe assez peu sur le résultat.
- A travers le *Tableau 6,* la nature de la grille, l'endroit de la première colonie, le nombre dynamique d'ouvrières et la tolérance n'influent pas vraiment sur les résultats. On peut cependant remarquer de légers changements dans le temps d'exécution avec la nature de la grille et la position de la première colonie.

4. Conclusion

Dans ce chapitre « Tests et résultats », nous avons tenté d'identifier l'influence de certains des principaux paramètres sur les résultats de la classification. Il s'est avéré que le nombre d'objets pour créer une classe (seuil de classe) joue un rôle primordial dans le nombre final de classes. C'est la raison pour laquelle l'utilisateur doit choisir soigneusement la valeur de ce seuil et cela par rapport aux nombres d'objets existants dans le benchmark utilisé.

Les tests ont montré que la méthode donne de bons résultats avec un taux de fiabilité élevé et un temps d'exécution réduit.

Conclusion générale

Résoudre un problème de classification non supervisée consiste à partitionner un ensemble de données en groupes homogènes, sans aucune connaissance préalable du nombre de groupes. Ce problème est reconnu NP-Difficile et il ne peut donc être résolu par des approches exactes pour des problèmes de grande taille. Les heuristiques sont alors les seules approches réalisables. Les méthodes inspirées de la biologie ont de plus en plus de succès car elles ont donné des résultats intéressants dans de nombreux cas. C'est dans ce cadre que s'inscrit notre travail dans lequel nous proposons un algorithme de résolution du problème de classification inspiré d'un certain type de fourmis naturelles.

Dans ce document, nous avons commencé par citer les différentes méthodes pour la résolution du problème de classification non supervisée. Dans un premier temps, nous avons défini le Data Mining, ses définitions, son utilité, et ses principales tâches. Dans le deuxième chapitre, nous avons abordé la classification, sa définition, ses types et ses méthodes de résolution. Dans le troisième chapitre, nous nous sommes intéressés aux fourmis artificielles. Ainsi nous avons exposé certains des algorithmes à base des fourmis artificielles les plus connus comme AntClass, AntClust, AntTree et AntPart.

Enfin nous avons proposé une nouvelle méthode qui s'inspire de l'organisation sociale chez les fourmis pour la recherche de nourriture et qui utilise de la phéromone comme mécanisme de communication entre les fourmis. L'approche développée est intégrée dans un environnement qui permet à l'utilisateur de tester la méthode sur des benchmarks supervisés à travers un certain nombre d'outils graphiques et statistiques que nous avons mis en œuvre dans notre environnement.

La méthode que nous avons proposée a donné des résultats satisfaisants à travers une série de tests que nous avons effectués. L'environnement développé permettra de tester la méthode avec plus de précision et de déterminer les paramètres les plus intéressants pour les différents types de problèmes à classifier.

Afin d'améliorer ou de compléter notre travail, plusieurs perspectives peuvent être proposées. Ces améliorations peuvent être:

- La parallélisation de la méthode dans le but d'accélérer le processus de classification, essentiellement quand la taille des échantillons devient considérable;
- Tenter de trouver des valeurs par défaut pour certains des paramètres, qui soient efficaces pour tous les benchmarks: nature de la grille (toroïdale ou non), la création de la première colonie (aléatoire ou sur la position du premier objet dans le benchamrk) et le nombre de fourmis ouvrières (fixe ou dynamique) ont montré qu'il n'ont pas une grande influence sur les résultats, alors ils peuvent être fixés au préalable et éviter à l'utilisateur de les fixer lui-même, afin d'alléger sa tâche de paramétrage.
- Lancer les fourmis exploratrices à chaque itération (enlever la période de lancement des exploratrices) tout comme les ouvrières. Les exploratrices ont montré une grande efficacité dans la réduction du temps d'exécution.
- Munir les ouvrières d'un rayon de liberté (dans les cas où elles se déplacent sans trouver d'objets). L'ouvrière n'est pas limitée par une tolérance dans ce rayon de liberté, au-delà de ce rayon la tolérance conditionne le déplacement de l'ouvrière.
- Enfin, adapter la méthode à des projets concrets tels que le traitement de l'information médicale ou dans un domaine bien précis comme le domaine bancaire, télécommunication ou bien les assurances.

Bibliographie

[ABD 03]	Abdelali Mouad – Oudri Hichem, « Création des règles d'association », Filière Informatique & Réseaux. 2^{cmc} année, Rapport de projet, Suivi France Telecom : DURAND Nicolas, ENSI - France, 2003. www.greyc.ensicaen.fr/ensicaen/Projets_AN2_Info_2002_2003/ND1.pdf
[AMO 06]	Amorkrane Samah, « Algorithme génétique pour les problèmes d'ordonnancement dans la synthèse de haut niveau pour contrôleur dédiés », Mémoire de Magister, Université de Batna, Algérie, 2006.
[AZZ 03]	Hanène Azzag, Nicolas Monmarché, M.Slimane, C.Guinot et G.Venturini, Algorithme AntTree : « Classification non supervisée par les fourmis artificielles », 35ème journée de statistiques, 16 Janvier 2003.
[AZZ 04]	AZZAG H., PICAROUGNE F., GUINOT C., VENTURINI G., « Un survol des algorithmes biomimétiques pour la classification, Classification Et Fouille de Donnée », pages 13-24, RNTI-C-1, Cépaduès. 2004. www.antsearch.univ-tours.fr/publi/azzag04survol.pdf
[BIA 01]	Jacques Biais, « L'extraction de données », Conférence Knowledge Management, Paris, 2001.
[CEL 04]	Pierre CÉLIER, « Le forage de données (*DATA MINING*) », Support de cours, Enset - Maroc, 26/04/2004. http://enset-media.ac.ma/cpa/datamining.htm
[CHI 03]	Salim Chitroub, « Classification contextuelle d'images de télédétection utilisant la programmation génétique », Conférence Internationale sur les Systèmes de Télécommunications, d'Electronique Médicale et d'Automatique CISTEMA'2003, Tlemcen, 27-29 Septembre 2003.
[CLE XX]	Manuel Clergue, « Réseaux de neurones Artificiels », Support de cours, Université de Nice Sophia Antipolis. deptinfo.unice.fr/twiki/pub/Minfo04/SystemesArtificielsComplexes/cRdNV8.ppt
[DAT 06]	Le site : www.web-datamining.net, Dernière mise à jour : 2006.
[DOR 99]	Marco Dorigo & Gianni Di Caro, « Ant Algorithms for Discrete Optimization », Artificial Life Volume 5, Number 2, pages 137-172, 1999.
[DRE 02]	Gérard Dreyfus, Manuel Samuelides, Jean-Marc Martinez, Mirta B. Gordon, Fouad Badran, Sylvie Thiria et Laurent Hérault, « Réseaux de neurones : Méthodologie et Applications », Edition Eyrolles, 2002.
[FAD 07]	Le site: www.fadelexpert.ici.ma, Dernière mise à jour: 7 Août 2007.
[FES 02]	Marie Fesneau et Fabien Ducher, « informatique decisionnelle », Rapport interne, l'EISTI - France, avril 2002. ms.eisti.fr/IMG/doc/theorie_du_decisionnel_intro.doc
[FIL 02]	Estelle FILMON et Yohann HUBERT, « Etude bibliographique Data Mining », Mémoire de Maîtrise en Informatique, LERIA- Faculté des Sciences d'Angers, mars 2002.
[GOG 01]	J.François Goglin, « la construction du data warehouse », Edition hermes, 2001.
[GRA XX]	Le site: www.grappa.univ_lille.fr.
[GRI XX]	Le site : www.m-grimaud.com, Dernière mise à jour : Juillet 2007.
[HEL 04]	Georges El Helou et Charbel Abou khalil, « Data Mining -Techniques d'extraction des connaissance », Mémoire pour l'obtention du Diplôme Universitaire " Modèles de l'économie numérique ", Université Panthéon-Assas –

	Paris, 2004.
[KOU 03]	Aby Kouacho & El Kourri adil, « post traitement des règles d'association », Filière Informatique & Réseaux. 2eme année, Rapport de projet, ENSI France, 2003. www.greyc.ensicaen.fr/ensicaen/Projets_AN2_Info_2002_2003/ND2.pdf
[LAB 06]	N.Labroche, Nicolas Monmarché et G.Venturini, « Modélisation de la fermeture coloniale chez les fourmis pour la classification non supervisée », Rapport interne, Laboratoire de l'Université de Tours, 2006. www.hant.li.univ-tours.fr/webhant/pub/LabMonVen02a.cap.pdf
[LAM 01]	Luc La Montagne, « Raisonnement a base de cas textuel pour la réponse automatique au courrier électronique », Thèse de Doctorat, Université de Montréal, Canada, 2001.
[LEN XX]	Philippe Lenca, « Extraction de connaissances à partir de données », Support de cours, ENST Bretagne. perso.enst-bretagne.fr/~lenca/enseignement/ecd/cours/intro_ecd.pdf
[LIA 06]	Bertrand Liaudet, « introduction du Data Mining », Support de cours pour 4ème année Option Ingénierie d'Affaires et de Projets, EPF-Paris, Juin 2006. http://bliaudet.free.fr
[LUM 94]	LUMER E., FAIETA B., « Diversity and Adaptation in Populations of Clustering Ants », CLIFF D., HUSBANDS P., MEYER J., W. S., Eds., *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, MIT Press, Cambridge, Massachusetts, 1994, p. 501–508.
[MAZ 04]	Maziz et Siad, « Classification à l'aide des colonies de fourmis », Mémoire d'ingéniorat, Institut National d'Informatique, 2004
[MON 00]	Nicolas Monmarché, « Algorithmes de fourmis artificielles : application à la classification et à l'optimisation », Thèse de Doctorat, Université François Rabelais Tours, 20 décembre 2000.
[MON 01]	Nicolas Monmarché, Mohamed Slimane et Gilles Venturini, « L'algorithme Antclass : Classification non supervisée par une colonie de fourmis artificielles », ECA – 1/2001. Apprentissage et évolution, pages 131 à 166.
[NAK 98]	Didier Nakache, « Data Mining sur Internet », Mémoire de DEST d'Informatique d'entreprise (diplôme d'études supérieures techniques, équivalent maîtrise), CNAM-Lille, 15-12-1998.
[NAP 06]	Amedeo Napoli, «Extraction de connaissances et fouille de données », Support de cours pour Master Informatique 2 - Spécialité PRIM, Nancy, 2006. www.loria.fr/~napoli/Master-2/ecd-121006-4.pdf
[NOB 04]	Kicolas Nobilis, « Un modèle case – Based Reasoning pour la détection d'intrusion », Rapport de stage, Laboratoire I3S, Université Nice Sophia Antipolis, septembre 2004. babylone.essi.fr/DEA-RSD/2003-04/rapports/nobelis.pdf
[OUA 06]	Ouadfel Salima, « Contributions à les segmentation d'images basées sur la résolution collective par colonies de fourmis artificielles, Thèse de Doctorat, Université Hadj Lakhdar de Batna, 2006.
[PER 96]	Aurora Pérez -Ángela Ribeiro Seijas, « An Approximation to Generic Knowledge Discovery in Database Systems», Rapport Interne, Université de Madrid. www.iai.csic.es/gpa/postscript/malfo96.ps
[PHI XX]	Le site: www.e-philo.univer-paris1.fr.
[PIL 07]	Le site : www.piloter.org, Dernière mise à jour : 2007.
[POL 00]	Xavier Polanco, «Convergence de produits logiciels et d'information en Fouille

	de Données et Extraction de Connaissance à partir de Bases de Données », Séminaire ADEST ,15 février 2000.
[PON XX]	Pascal Poncelet, «Knowledge Discovery in Database (KDD) and Data Mining (DM) », Support de cours, Ecole des Mines d'Alés. www.lirmm.fr/~laurent/CNAM/Cours_PPoncelet.ppt
[RAK 05]	Ricco Rakotomalala, «Arbres de décision », Revue MODULAD, n°33, pp 163-187, 2005.
[REC 07]	Le site : www.recherche.enac.fr, Dernière mise à jour : 28 Mars 2007.
[SAU 03]	Bertrand le Saux, Classification Non exclusive et personnalisation par apprentissage : « application à la navigation dans les bases d'images », Thèse de Doctorat, Université de Versailles Saint-Quentin, 10 juillet 2003.
[SEK 06]	Sekhi Massinissa et Zidouni Azeddine., « Résolution du problème de classification non supervisées par une méthode évolutive », Mémoire d'ingéniorat, INI –Oued Smar, 2006.
[SER 02]	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 pour obtenir le grade de Docteur de l'université Paris, Paris, 10 décembre 2002.
[TUF 02]	Stéphane Tufféry, « Data Mining et Scoring », Edition Dunod, 2002.
[USH XX]	Le site: www.chimique.usherbrooke.ca.
[VAN 05]	C. Vangenot, « Datawarehouse », Support de cours, Ecole Polytechnique Fédérale de lausanne,2005. lbdwww.epfl.ch/f/teaching/courses/SlidesIBD/DW.pdf
[WIK 07]	Le site: fr.wikipedia.org, Dernière mise à jour : 14 Août 2007.
[ZIG 03]	Zighed D.A., Rakotomalala R., "Extraction de connaissances à partir de données (ECD)", dans les Techniques de l'Ingénieur, H 3 744, 2003.

Annexe A

1. Le Data Warehouse

1.1. Définition

Un Data Warehouse est un entrepôt de données. Il s'agit d'un stockage intermédiaire des données [GOG 01]. Les données intégrées au sein d'un Data Warehouse sont issues de diverses sources: système d'information, web ou données externes [GRI XX].

Une définition a été faite par Bill Inmon4:

"Un Data Warehouse est une collection de données thématiques, intégrées, non volatiles et historisées organisées pour la prise de décision " [GOG 01]

Une définition des termes: thématiques, intégrées, non volatiles et historisées s'impose:

- Thématiques (orientées objet): on rassemblera les informations par thèmes et non par fonctions comme le font les modélisations traditionnelles. Par exemple, les informations concernant les activités "assurance vie", "assurance auto" et "assurance santé" sont distinctes dans les bases de données opérationnelles. Ainsi le Data Warehouse a la capacité de fournir des informations relatives à un ou plusieurs clients pour tous types d'assurances. Aussi, la vue offerte par le Data Warehouse sur les données est alors une vue orientée "clients" [GOG 01].

- Intégrées: les données sont mises en forme selon un standard afin de faciliter l'extraction par les outils d'analyse [GRI XX].
- Non volatiles: les données ne peuvent être supprimées ou modifiées, afin de permettre des analyses fiables, précises et pour conserver la traçabilité des informations et des données prises. [GRI XX, GOG 01].

---

[4] Connu comme étant le père du Data Warehousing.

- Historisées: les données sont positionnées dans le temps afin de permettre leur suivi dans le temps [GRI XX].

L'entrepôt de données peut structurer les informations selon deux types: [GRI XX].

1-En étoile ou flocon dans un SGBD relationnel (base de données classique)

2-En schéma multidimensionnel – les données sont stockées dans des cubes ou hypercubes de type OLAP – C'est le modèle que l'on retiendra pour la suite [GRI XX].

1.2. Définition OLAP

Les outils OLAP (On Line Analytical Process) reposent sur une base de données multidimensionnelle, destinée à exploiter rapidement les dimensions d'une population de données. La plupart des solutions OLAP reposent sur un même principe: restructurer et stocker dans un format multidimensionnel les données issues de fichiers plats ou de bases relationnelles [NAK 98].

1.3. Définition du Datamart

On ne peut pas parler de Data Warehouse sans évoquer le Datamart. Par définition, le Datamart est un magasin de données, c'est un sous ensemble du Data Warehouse [GOG 01]. C'est un ensemble d'extraits des données du Data Warehouse, classé selon une logique métier. Un Data Warehouse contient plusieurs Datamarts, dont chacun traite d'un métier ou d'une activité spécifique de l'entreprise. Ainsi, un Datamart est une base de données décisionnelle structurée et formatée en fonction d'un métier précis [GRI XX].

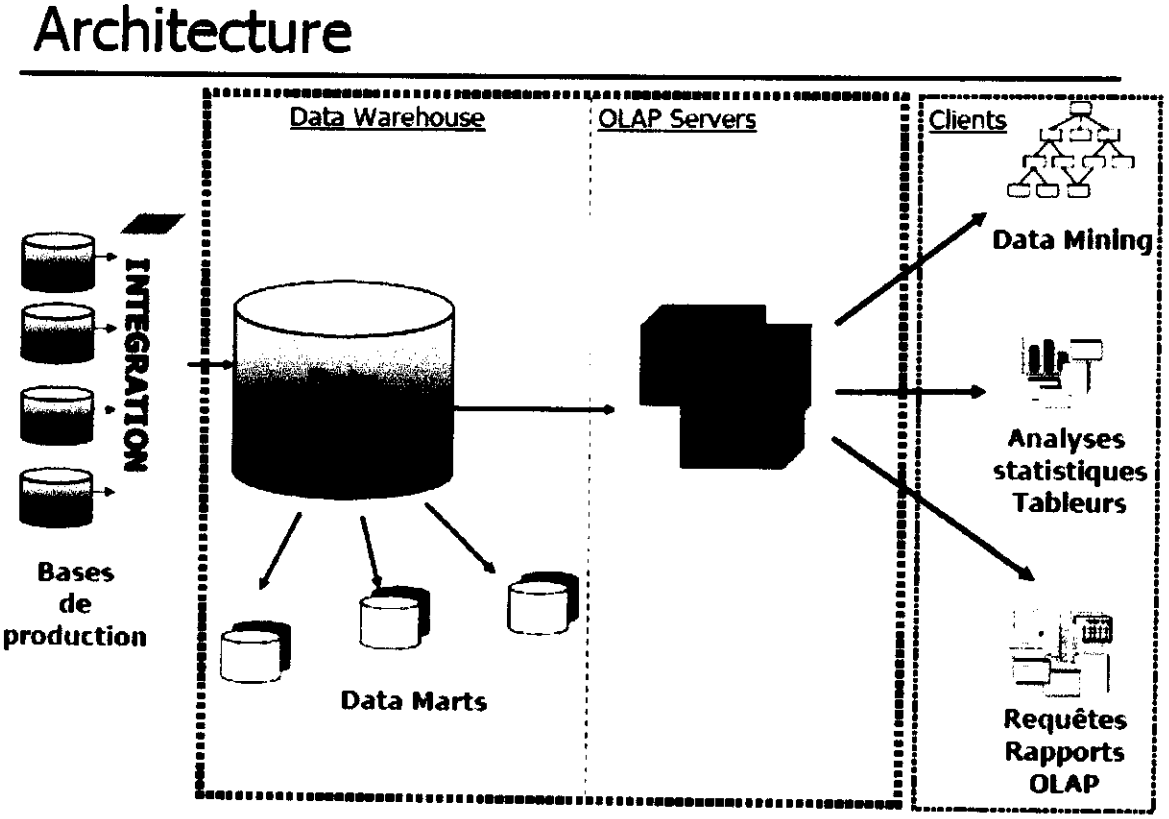


Figure 45. Fonction du Data Warehouse [VAN 05].

2. Le rapport entre le Data Mining et les Data Warehouses

Le Data Mining utilise une partie d'un Data Warehouse (quand il existe !) appelée: Datamart. Cet ensemble de données (Datamart) est la partie extraite du Data Warehouse qui contient l'ensemble des informations nécessaires à l'analyse. Lorsque ni le Datamart, ni le Data Warehouse n'est présent, le « Data Miner » devra créer sa propre base de données avant de les analyser. Le Data Warehouse précède donc le Data Mining dans l'extraction des connaissances des données, mais sa présence n'est pas indispensable [DAT 06].