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
du diplôme d'ingénieur d'état en informatique.
Option : Système d'information

Sujet :

Résolution du Problème du Data Mining par une approche évolutive

Présenté par : M^lle^ Boudjema Anissa
M^lle^ Boudjema Hadjer

Promoteur: D^r^ Koudil Mouloud.

Soutenue : Octobre 2005.

- Numéro/Juin 2005 -

Résumé :

Nous présentons dans notre travail une implémentation d'une méthode biomimétique permettant de créer des groupes au sein de données et de les visualiser dynamiquement. elle s'inspire des insectes volants se déplaçant en nuage en créant des mouvements complexes à partir de règles locales simples. Chaque insecte représente une donnée. Le déplacement des insectes vise à créer des groupes de données homogènes se déplaçant ensemble dans un espace à deux dimensions. Les groupes créés et visualisés en temps réel informent l'expert du domaine qui a fourni les données sur leur structuration en classe, par exemple, le nombre de classes possible, le regroupement de données similaires, et les données isolées représentant des cas « à part ».

Mots-Clés : classification non supervisée, fouille visuelle de données, visualisation de données, nuage d'insectes, intelligence en essaim.

Dédicace

A nos chers Parents qui nous ont toujours soutenu

A notre petite Sœur Isma

A toute la famille

A tous les amies

Nous dédions ce mémoire

Anissa et Hadjer

Remerciements

Nos remerciements les plus vifs vont à notre promoteur Mr Koudil pour son aide, ses orientations judicieuses et sa disponibilité, qu'il en soit humblement remercié.

Nous remercions également les membres du jury d'avoir bien voulu juger notre travail.

Table des matières

Introduction générale 1

Chapitre I **Data Mining**

1. Introduction 3
2. Définition 3
3. Processus d'extraction des connaissances 4
   3.1. Formulation du problème 4
   3.2. Acquisition des données 5
   3.3. Pré-traitement des données 5
      3.3.1. La sélection de lignes 5
      3.3.2. La sélection de colonnes 5
      3.3.3. Le nettoyage des données 6
         3.3.3.3.1. *Le traitement des données manquantes* 6
         3.3.3.3.2. *Le traitement des données aberrantes* 6
         3.3.3.3.3. *Le traitement des doublons* 6
      3.3.4. La transformation des attributs 7
   3.4. Data Mining (Fouille de données) 7
   3.5. Validation du résultat 8
      3.5.1. Validation croisée 9
      3.5.2. Bootstrep 9
   3.6. Intégration du résultat 9
4. Les techniques de Data Mining 9
   4.1. Les méthodes de classification et de prédiction 9
      4.1.1. Les réseaux de neurones 10
         4.1.1.1. *Neurone formel* 10
         4.1.1.2. *Perceptron multicouche(PMC)* 10
         4.1.1.3. *Principe de construction d'un réseau de neurones* 11
         4.1.1.4. *Avantage et Limites* 13
      4.1.2. Les plus proche voisins 13
         4.1.2.1. *Notion de distance* 14
         4.1.2.2. *Sélection de la classe* 14
         4.1.2.3. *Avantages et Limites* 15
      4.1.3. Les arbres de décision 15
         4.1.3.1. *Principe de construction d'un arbre de décision* 16
         4.1.3.2. *Avantages et Limites* 19
      4.1.4. Les algorithmes génétiques 19
         4.1.4.1. *Principe des algorithmes génétiques* 20
            4.1.4.1.1. *Fonction d'évaluation* 20
            4.1.4.1.2. *Opérateurs génétiques* 20
         4.1.4.2. *Application pour la classification* 22
         4.1.4.3. *Avantages et Limites* 23
   4.2. Les méthodes de classification automatique (clustering) 24
      4.2.1. Les méthodes momothétiques 24

Liste des figures

Figure I.1	Les étapes du processus d'extraction de connaissances à partir des données	4
Figure I.2	Le neurone formel	10
Figure I.3	Un perceptron multicouches	11
Figure I.4	Algorithme de rétropropagation	12
Figure I.5	Algorithme de classification par k-PPV	13
Figure I.6	Exemple d'arbre de décision	16
Figure I.7	Algorithme d'apprentissage par arbres de décision	17
Figure I.8	Fonctionnement général d'un AG de base	20
Figure I.9	Algorithme d'agglomération	25
Figure I.10	Algorithme des K-means	26
Figure II.1	Méthode de descente générique	33
Figure II.2	Méthode générique de recuit simulé	34
Figure II.3	Méthode générique de Tabou	35
Figure III.1	Schéma général de l'environnement	43
Figure III.2	Chargement de Benchmarks	44
Figure III.3	Editeur des données	44
Figure III.4	Algorithme décrivant le principe général	46
Figure III.5	Illustration du mouvement du nuage d'insectes: *(a) les insectes sont placés aléatoirement et orientés dans des directions désordonnées.* *(b) les insectes sont en groupes se déplaçant de manière cohérente*	47
Figure III.6	Calcul du déplacement d'un insecte *i*	48
Figure III.7	Comportement théorique de l'algorithme dans le cas de deux insectes en interaction	51
Figure III.8	Classes représentatives de la méthode	52
Figure III.9	Visualisation du développement des emplacements des insectes	55
Figure III.10	Algorithme de Classification	55
Figure III.11	Visualisation de la classification des insectes	57
Figure III.12	Sélection des données	58

4.2.2. Les méthodes polythétiques 24
4.2.2.1. *Les techniques hiérarchiques* 25
4.2.2.2. *Les techniques de partitionnement* 26
4.2.2.2.1. *La méthodes des K-means* 26
4.2.2.2.2. *La méthodes des K-medoïds* 27
4.3. Les règles d'association 27
4.3.1. Principe de construction des règles d'associations 27
4.3.2. Avantages et Limites 29
5. Principaux domaines d'application 30
6. Conclusion 31

Chapitre II **Métaheuristiques pour le Partitionnement**

1. Introduction 32
2. Méthodes à solution unique 32
2.1. Les méthodes de descentes (Hill Climbing) 33
2.2. Le recuit simulé (Simulated Annealing) 33
2.3. La recherche Tabou (Tabu Search) 25
3. Méthodes évolutionnaires 36
3.1 Algorithmes génétiques 36
3.2 Programmation génétiques (Genetic Programming) 37
4. Colonies de fourmis 38
5. Systèmes immunitaires artificiels 39
6. Conclusion 40

Chapitre III **Conception et Mise en œuvre**

1. Introduction 41
2. Description du problème 41
3. Description de la solution 41
4. Description de l'environnement mis en œuvre 43
4.1. Module de chargement des Benchmarks 43
4.2. Module de calcul du déplacement des insectes 45
4.2.1. Mesure de similarité 45
4.2.2. Coordonnées et vitesse d'un insecte 45
4.2.3. Algorithmes principal 46
4.2.4. Améliorations apportées à la méthode d'initialisation 47
4.2.5. Règle de comportement local 48
4.2.5.1. *Détermination du voisinage* 49
4.2.5.2. *Calcul de la nouvelle direction* 49
4.2.5.3. *Changement D'amplitude et déplacement final* 52
4.3. Module de visualisation des insectes 53
4.4. Module de classification des données 55
4.5. Module de sélection des données 57
4.6. Module d'affichage des résultats 58
5. Conclusion 49

Chapitre IV **Tests et Résultats**

1. Introduction 60

Figure III.13 Affichage des résultats..... 59

Figure IV.1 *Exemples de résultats obtenus au bout de 500 itérations avec différentes valeurs du paramètre d_{seuil} (lié notamment au calcul du voisinage) pour la base de données Segmentation*..... 64

Figure IV.2 *Un exemple des effets de l'initialisation orientée des insectes avec en (a) la base des Iris répartie aléatoirement, en (b) la même base avec initialisation orientée, et en (c) et (d) les résultats obtenus respectivement à partir de (a) et (b) au bout de 50 itérations avec un d_{seuil} de 0.05*..... 65

Figure IV.3 *Un exemple de résultat obtenu en utilisant le voisinage par grille sur la base Iris, (a) le résultat obtenu par l'algorithme principal, (b) le résultat obtenu par l'algorithme de classification en fonction des groupes d'insectes qui se sont formés en (a)*..... 66

Figure IV.4 *Un exemple d'un résultat en utilisant le voisinage par grille sur la base Segmentation, (a) le résultat obtenu par l'algorithme principal, (b) le résultat obtenu par l'algorithme de classification en fonction des groupes d'insectes qui se sont formés en (a)*..... 66

Liste des tableaux

Tableau I.1 Matrice représentant un ticket de supermarché..... 28
Tableau I.2 Matrice représentant le nombre de fois où un article apparaît..... 28
Tableau I.3 Matrice représentant le nombre d'associations..... 29
Tableau IV.1 Caractéristiques des différentes bases de données..... 60
Tableau IV.2 Paramètres et résultats sur le benchmark Soybean..... 62
Tableau IV.3 Paramètres et résultats sur le benchmark Iris..... 62
Tableau IV.4 Paramètres et résultats sur le benchmark Segmentation..... 62
Tableau IV.5 Paramètres et résultats sur le benchmark Heart..... 62

2. Jeux de données..... 60
3. Méthodologie de test..... 60
4. Exécution de la méthode..... 61
5. Interprétation des résultats..... 63
5.1. L'influence du paramètre d_{seuil} 63
5.2. L'influence de l'initialisation orientée..... 64
5.3. L'influence du voisinage..... 65
6. Conclusion..... 66

Conclusion générale et perspectives..... 67
Bibliographie..... 69

Introduction générale

Dans les années 70, l'invention du modèle relationnel, qui devint vite une norme pour les bases de données, l'émergence des SGBD, l'introduction du langage de recherche SQL et l'apparition des micro-ordinateurs dans les années 80 ont facilité le stockage et la restitution des données. Ceci a permis aux entreprises de développer des applications consacrées à la gestion: paie, facturation, comptabilité, etc. Ces applications sont regroupées sous le terme d'*informatique de production ou Informatique Opérationnelle*.

Dans un environnement de concurrence plus pressant, n'importe quel décideur dans une entreprise se pose différentes questions sur le profil de ses clients, sur l'évolution des ventes... Cependant, on ne peut pénaliser le système opérationnel pour répondre à des requêtes nécessitant un temps de calcul trop long, car ces systèmes sont réservés à de courtes requêtes indécomposables d'ajouts, de suppressions et de modifications d'enregistrements. Il s'avère donc nécessaire de développer des systèmes décisionnels permettant d'extraire des informations pertinentes des données, afin d'aider l'entreprise dans la prise des « bonnes » décisions.

De nouveaux systèmes de stockage (*data warehouse ou entrepôt de données)* ont été développé dans le but de garder un historique et restructurer les données de production. Le *data warehouse* est une architectures de sauvegarde de données réunit de sources multiples et hétérogènes. Ces données sont non volatiles (non modifiables) et spécialement organisées pour l'analyse.

Ces entrepôts de données disposent, bien sûr de capacité de reporting[1]. Ces logiciels de visualisation permettent, cependant rarement, de découvrir des associations ou des tendances dans une base de données.

Pour répondre aux besoins de découverte d'informations, un ensemble de techniques de compréhension intelligible des données, certaines nouvelles, d'autres existant depuis longtemps sont synthétisées dans des logiciels de fouille de données ***« data minig »***. Les algorithmes permettant d'extraire des connaissances à partir des données ont des origines diverses, certains sont issus des statistiques (régression...), d'autres de la recherche en Intelligence Artificielle (réseaux de neurones, arbre de décision..), certains

[1] : Présentation de données ou d'agrégats sous forme de tableaux ou de graphiques

encore s'inspirent de la théorie de l'évolution (algorithme génétique) ou de l'éthologie[2] (colonies de fourmis...).

Pour la résolution du problème de classification, qui est l'une des taches du *data mining*, Monmarché, Venturini et Guinot ont présenté un nouvel algorithme biomimétique[3] permettant de créer des groupes au sein de données et de les visualiser dynamiquement. Cet algorithme s'inspire des insectes volants se déplaçant en nuage en créant des mouvements complexes à partir de règles locales simples.

Notre travail s'inscrit dans cette problématique. Le travail qui nous a été proposé consiste à réaliser une implémentation de l'algorithme cité précédemment, en l'intégrant au sein d'un environnement permettant l'étude de cette approche et le test de différents jeux de paramètres.

Ce document est composé de quatre chapitres. Le premier chapitre comporte un certain nombre de définitions sur les tâches et les techniques du *Data Mining*. Le deuxième chapitre est dédié à l'étude de métaheuristique pour la classification. La conception et la mise en œuvre font l'objet du troisième chapitre. Enfin, nous présentons dans le chapitre quatre quelques tests et résultats.

[2] : Etude scientifique du comportement des animaux dans leur milieux naturel.
[3] : une nouvelle discipline basée non pas sur ce qui peut être extrait de la nature mais sur ce qui peut être appris d'elle.

Chapitre I

Data Mining

1. Introduction

Le développement des moyens informatiques de stockage et de calcul, ainsi que la popularisation de nombreux algorithmes (réseaux de neurones, arbres de décision,...) ont permis maintenant le traitement et l'analyse d'ensembles de données très volumineux.

Ce qui a conduit au développement et à la commercialisation de logiciels intégrant un sous-ensemble de méthodes statistiques et algorithmiques sous la terminologie de *Data Mining*.

2. Définition

Le *data mining* est apparu au début des années 90. Ce qui a le plus contribué à l'émergence du *data mining*, est la nécessité de valoriser les données collectées dans les bases (estimé à plusieurs tera-octets, 1tera=10^{12} octet). Dans ce qui suit, nous allons exposer quelque définitions générales proposées par les experts du *data mining* :

- L'extraction de connaissances à partir des données est un processus non trivial d'identification de structures inconnues, valides et potentiellement exploitables dans les bases de données. [FAY 96]
- Le *data mining* est la candidature d'algorithmes spécifiques pour extraire des modèles des données. [FAY 96]
- Le *data mining* est l'art d'extraction des connaissances à partir des données. [MOR XX]
- Un processus qui permet de découvrir dans de grosses bases de données des informations jusque-là inconnues (connaissances), qui peuvent être utiles, et d'utiliser ces informations pour soutenir des décisions commerciales tactiques et stratégiques. [MAT XX]
- 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. [LIN 97]

Une confusion subsiste encore entre le *data mining*, que nous appelons en français « fouille de données », et *knowledge discovery in data bases* (KDD), que nous appelons en français « extraction des connaissances à partir des données » (ECD). Le *data mining* est l'un des maillons de la chaîne de traitement pour la découverte des connaissances à partir des données. Sous forme imagée, nous pourrions dire que l'ECD est un véhicule dont le *data mining* est le moteur. [MOR XX]

3. Processus d'extraction de connaissances

Le processus comporte six phases: formulation du problème, acquisition des données, pré-traitement des données, *data mining* (fouille de données), validation et intégration du résultat.

Le processus est dit non linéaire car si le résultat d'une phase n'est pas satisfaisant, on peut toujours revenir aux phases précédentes. On estime que moins de 20% du temps d'un processus d'extraction est destiné à l'étape de *data mining*, plus de 80% du temps restant est consacré aux autres étapes.

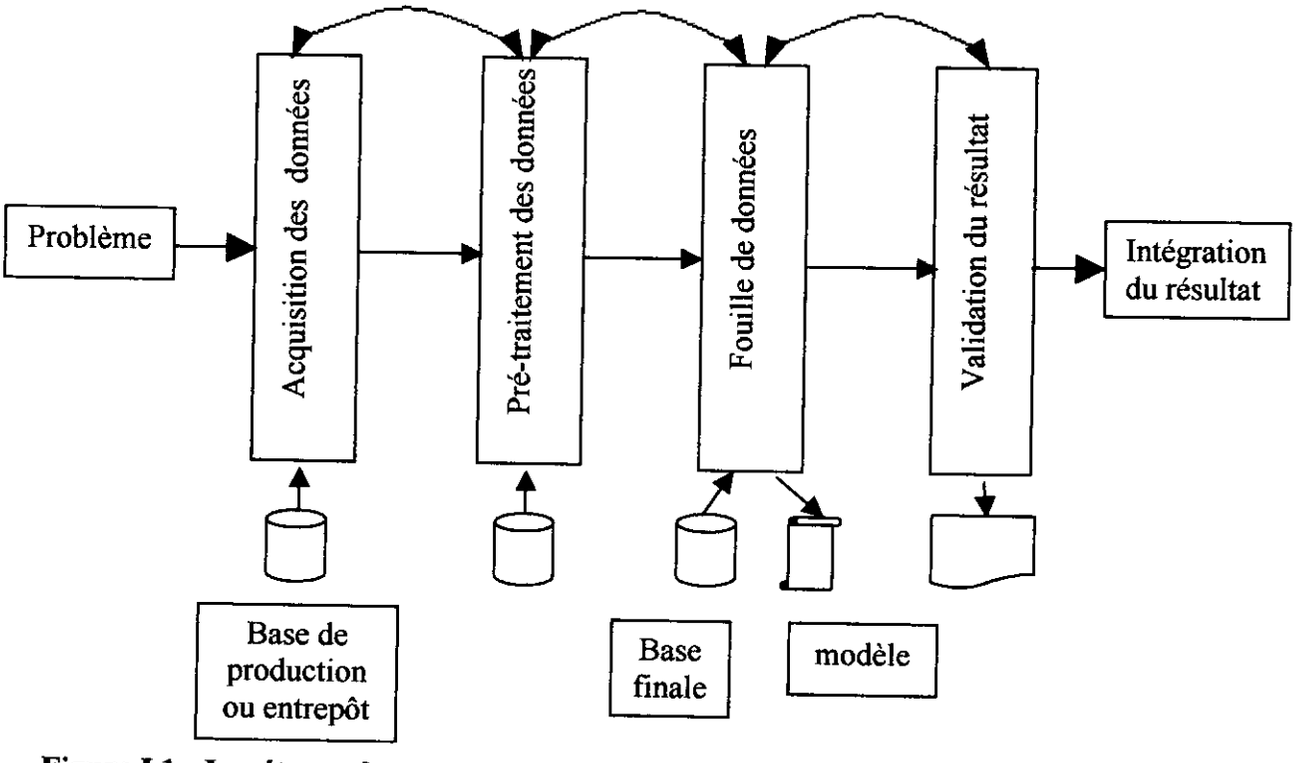


Figure I.1 : Les étapes du processus d'extraction de connaissances à partir des données

3.1. Formulation du problème

Dans cette première phase, il s'agit de formuler le problème à analyser, de définir ce que l'on attend et ce que l'on souhaite faire des connaissances (les objectifs) et de connaître les clients ou les individus qui utiliseront ces connaissances. L'obtention de ces information permettra de choisir la technique la mieux adaptée pour résoudre le problème.

3.2. Acquisition des données

Les données peuvent être stockées selon des architectures variées: dans des bases de données relationnelles, dans des *entrepôts de données*, sur le *web* ou dans des banques de données spécialisées (images, bibliothèques ou librairies numériques, base de données génomiques...). Elles peuvent être structurées ou non, selon différents types: données tabulaires ou textuelles, images, sons ou séquences vidéo. [MOR XX]

Cette phase a pour fonction de sélectionner les données ayant un rapport avec le problème posé. Elle nécessite l'utilisation de moteurs de recherche de données. Pour les données non structurées (les images, les textes, ...) on fait appel à des moteurs de recherche spécifiques ainsi qu'à des programmes pour recueillir les données de passage (qui ne sont pas mémorisées dans la base de l'entreprise par exemple).

3.3. Pré-traitement des données

Généralement, les données acquises n'ont pas une structure exploitable par les techniques de *data mining*. La préparation consiste à homogénéiser les données et à les disposer en tableau lignes/colonnes [MOR XX], car la plupart des techniques utilisent cette forme de données afin de résoudre les problèmes. Il existe d'autres méthodes plus exigeantes qui n'exploitent que les données binaires.

3.3.1. La sélection de lignes

L'analyste doit choisir entre étudier l'exhaustivité de la base et travailler sur un échantillon [LEF 01], l'étude de l'exhaustivité de données apporte une meilleur qualité et fiabilité des résultats, mais nécessite de puissantes machines. Un échantillon est utilisé dans la majorité des cas afin de détecter des tendances générale. Une base échantillonnée présente l'avantage de nécessiter un temps de réponse réduit, ceci est d'autant plus appréciable du fait que le *data mining* est souvent très itératif.

3.3.2. La sélection de colonnes

Il s'agit dans cette étape de sélectionner les attributs pertinents, car intégrer toutes les variables dans la phase de fouille de données entraînera un surdimensionnement du problème, on aura alors un temps de calcul trop long. Cette sélection nécessite souvent l'aide d'experts pour déterminer les attributs les plus aptes à décrire la problématique. [LEF 01] On peut également avoir recours à des techniques d'analyse (réseaux de neurones, associations, régression, ACP, ...) afin d'avoir les éléments les plus importants.

3.3.3. Le nettoyage des données

Certains outils du *data mining* n'acceptent pas l'absence, l'incohérence et la répétition des données, on dit alors que ces méthodes sont sensibles au bruit. Pour que les données soient plus adaptées aux méthodes, on effectue plusieurs traitements:

3.3.3.1. *Le traitement des données manquantes*

Il existe plusieurs solutions pour procéder à la génération et au remplacement de ces champs vides dont les principales sont: [LEF 01]

- *Exclure les enregistrements incomplets*: le but de cette méthode est d'éliminer tous les enregistrements dont une valeur manque.
- *Remplacer les données manquantes*: on peut remplacer la valeur de l'attribut concerner par différents moyens :
  - la valeur la plus fréquente de l'attribut concerné.
  - la moyenne des valeurs prises par cet attribut.
  - estimer par des méthodes d'induction (régression, réseau de neurones, graphes d'induction).

3.3.3.2. *Le traitement des données aberrantes*

Ce sont les données qui ont des valeurs anormales par rapport à leurs natures (leurs types). Par exemple, un client qui a une date de naissance égale à l'année en cours. Un contrôle sur les domaines des valeurs permet de retrouver ces valeurs.

3.3.3.3. *Le traitement des doublons*

Les doublons peuvent se révéler gênants parce que les outils utilisés pour l'extraction de connaissances vont donner plus d'importance aux valeurs répétées. Les répétitions des informations doivent donc être repérées.

3.3.4. Transformation des attributs

Pour la plupart des méthodes de fouille, des transformations sur les attributs sont primordiales pour obtenir des résultats satisfaisants. Les transformations les plus utilisées sont:

- Le regroupement est appliqué dans le but de diminuer le grand nombre de valeurs discrètes d'un attribut et d'obtenir un nombre raisonnable, par exemple transformer les adresses en villes ou en régions.
- Le changement de type d'attributs, par exemple, transformer les dates en durées.
- La discrétisation qui consiste à transformer des attributs continus en découpant le domaine de valeurs de ces attributs en intervalles afin d'obtenir des attributs qualitatifs. [MOR XX]
- Certaines variables continues ont des intervalles très étendus par rapport à d'autre variables, ce qui constitue un problème surtout pour les méthodes fondées sur la notion de distance. On procède donc à une normalisation d'échelle de tous les attributs continus.

3.4. Data mining (fouille de données)

La fouille de données concerne le *data mining* dans son sens restreint et est au cœur du processus d'ECD. [MOR XX]

Cette phase consiste à extraire la connaissance utile à partir d'un large volume de données et à la présenter sous une forme synthétique. Elle repose sur une recherche exploratoire, c'est à dire dépourvue de préjugés concernant les relations entre les données. [LEF 01]

Elle fait appel à plusieurs méthodes issues des statistiques, de l'analyse de données, de l'intelligence artificielle ou de la théorie de l'évolution, le choix de la méthode se fera en fonction :

- De la tâche à résoudre,
- De la nature et de la disponibilité des données,
- De la finalité du modèle construit. Pour cela, les critères suivants sont importants: complexité de la construction du modèle, complexité de son utilisation, ses performances, sa pérennité. [GRF XX]

Les différentes tâches seront expliquées comme suit:

- Classification non supervisée (Clustering): qui consiste à composer des

groupes ou classes (clusters) homogènes à partir d'un ensemble de données. Aucune information sur la classe d'une donnée n'est fournie au préalable.

- Association: Il s'agit de trouver des similarités ou des associations. Par exemple, *si j'achète des pâtes et de la purée de tomates, j'ai deux fois plus de chance d'acheter aussi du parmesan* (association). [GRF XX]
- Estimation: elle consiste à estimer la valeur d'un attribut continu (à prédire) en fonction de la valeur des autres attributs (prédictifs).
- Classification supervisée: il consiste à placer chaque individu de la population dans une classe, parmi plusieurs classes prédéfinies, en fonction des caractéristiques de l'individu indiqué comme variables explicatives. [TUF 03]

3.5. Validation du résultat

Après l'extraction du modèle et avant de l'utiliser, il faut l'évaluer, c'est-à-dire mesurer sa performance, sa justesse et son pouvoir de généralisation. Les techniques employées pour des opérations de validation diffèrent selon la nature du problème. Il existe deux types de validation: validation statistique, 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.

Pour les problèmes de classification et d'estimation, on utilise la validation statistique dont le but est d'évaluer la capacité de généralisation du modèle. On peut estimer la qualité d'ajustement du modèle sur l'échantillon d'apprentissage. C'est une estimation biaisé, car elle est trop optimiste de l'erreur de prédiction. Elle est liée aux données qui ont servi à l'ajustement du modèle et elle est d'autant plus faible que le modèle est complexe. [LSP XX] Pour estimer sans biais l'erreur de prédiction, il est recommandé de diviser l'échantillon en trois parties:

- Un ensemble d'apprentissage pour générer le modèle,
- Un ensemble de validation: qui sert à la comparaison de modèles au sein d'une même famille afin de sélectionner celui qui minimise l'erreur, [LSP XX]
- Un ensemble de test: qui est utilisé pour comparé entre eux les meilleurs modèles de chacune des méthodes considérées. [LSP XX]

Au moins deux ensembles sont nécessaires: l'ensemble d'apprentissage permet de générer le modèle, l'ensemble test permet d'évaluer l'erreur réelle du modèle sur un

ensemble indépendant. [GRF XX]

Dans le cas où l'échantillon n'est pas de taille importante, on construit le modèle sur cet échantillon et on utilise la *validation croisée* ou le *bootstrap* pour estimer l'erreur réelle de se modèle.

3.5.1. Validation croisée

La validation croisée est conceptuellement simple, efficace et largement utilisée pour estimer l'erreur. Elle implique cependant un surplus de calculs car le principe consiste à calculer la moyenne des erreurs estimer sur des parties de l'échantillon d'apprentissage.

3.5.2. Bootstrap

Il a été conçu par Bradly Efron (1982). Pour cette méthode, l'échantillon de validation est tiré aléatoirement avec remise. De façon analogue à la validation croisée, l'erreur du modèle est estimée par l'erreur moyenne réalisée sur les différents échantillons de validation. [MOR XX]

3.6. Intégration du résultat

Cette phase consiste à implanter les résultats dans les systèmes informatiques ou alors dans le processus de l'entreprise (transition du domaine des études au domaine opérationnel).

4. Les méthodes du Data mining

Les méthodes de *data mining* permettent de découvrir ce que contiennent les données comme informations ou modèles utiles. Nous avons classifier ces méthodes selon les tâches à résoudre:

- Les méthodes de classification supervisée et de prédiction.
- Les méthodes de classification non supervisée (clustering).
- Les règles d'association.

Chacune de ces famille de méthodes comporte plusieurs techniques, nous allons donner à présent un aperçu général sur les principales méthodes.

4.1. Les méthodes de classification supervisée et de prédiction

Ces méthodes ont pour objectif de rechercher à partir des données

disponibles un modèle explicatif ou prédictif entre, d'une part, un attribut particulier à prédire et, d'autre part, des attributs prédictifs, dans ce contexte, on parle d'apprentissage supervisé car l'attribut à prédire est déjà préétabli. Dans le cas où le modèle produit s'avérait valide, il pourrait alors être utilisé pour prédire une classe (Classification) ou estimer une valeur (Estimation) de manière automatique à partir des caractéristiques d'un individu.

Nous allons maintenant expliquer les différentes méthodes les plus utilisées:

4.1.1. Les réseaux de neurones

Les réseaux de neurones sont parmi les outils de modélisation les plus utilisés. [MOR XX] Ils sont issus de modèles biologiques. Ils sont constitués d'unités élémentaires (les neurones) organisées selon une architecture. [GRF XX] Les réseaux de neurones utilisés pour la classification et l'estimation sont les perceptrons multicouches (PMC)

4.1.1.1. *Neurone formel*

C'est une unité de calcul élémentaire dont le modèle est issu de certains principes de fonctionnement du neurone biologique. L'unité de calcul combine des entrées réelles x_1, ..., x_n en une sortie réelle o. [GRF XX] Chaque entrée du neurone est calculée par la somme pondérée des entrées $h(x,w) = \sum_i w_i x_i$ où w est un vecteur poids du neurone qui le relie avec les neurones en entrées. La sortie du neurone est une fonction d'activation $f(h)$, souvent sigmoïde.

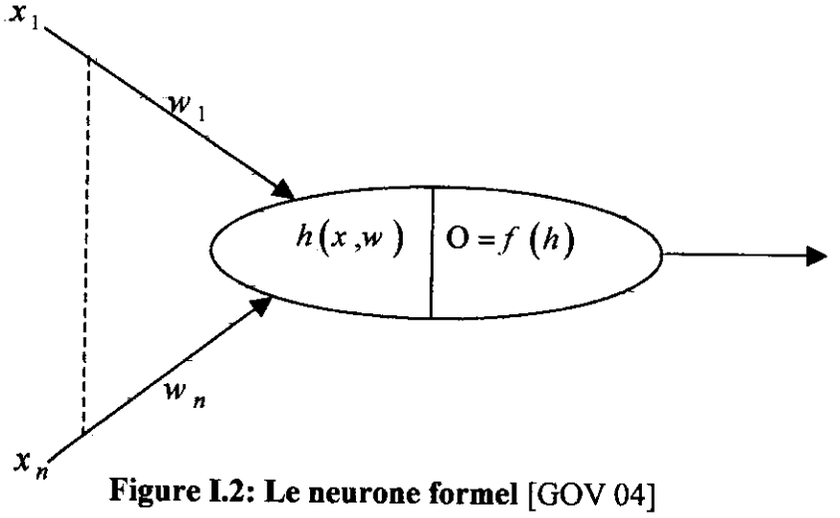


Figure I.2: Le neurone formel [GOV 04]

4.1.1.2. *Perceptron multicouches couches (PMC)*

Le perceptron multicouches est un modèle de réseaux de neurones. Les neurones sont répartis en couches successives, la première couche est la **couche d'entrée**, aucun calcul n'est effectué, une cellule d'entrée ne fait que copier son entrée vers sa sortie. Les entrées correspondent aux attributs (ou leurs codages) du problème considéré. Ensuite, viennent une ou plusieurs **couches cachées** constituées de neurones élémentaires. Tous les neurones d'une couche sont les entrées de chaque neurone de la couche suivante et de elle seulement, autrement dit, il n'y a pas de retour arrière et on passe d'une couche à la suivante. Enfin, la dernière couche est la **couche de sortie** qui contient un ou plusieurs neurones. Les sorties de ces neurones correspondent aux valeurs à estimer (ou leurs codages) pour le problème considéré. [GRF XX]

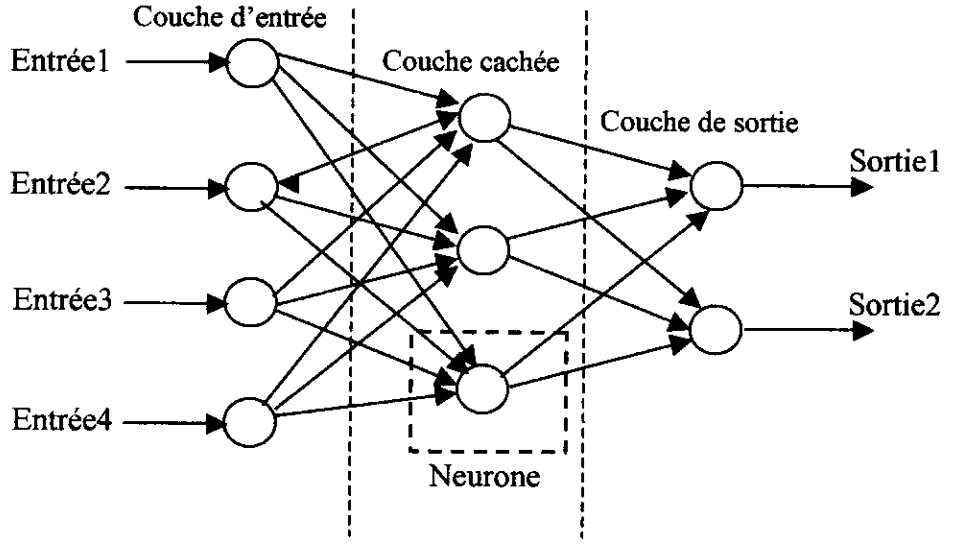


Figure I.3 : Un perceptron multicouches

4.1.1.3. *Principe de construction d'un réseau de neurones*

Au départ, le réseau de neurones est ignorant, et réalise à partir de ses exemples, un apprentissage et devient un modèle rendant compte du comportement observé en fonction des variables descriptives. La mise en œuvre d'un réseau de neurones passe par trois étapes: [GRF XX]

► **Codage du problème :**

Pour un bon fonctionnement des algorithmes, il est souvent nécessaire de normaliser les entrées dans l'intervalle [0,1]. Pour les données énumératives, plusieurs codages sont possibles, prenons l'exemple d'un attribut A prenant ses valeurs dans l'ensemble {1,2,3,4,5}, plusieurs codages sont envisageables:

- 5 entrées à valeurs binaires, dans ce cas, la donnée A=3 est codée par

00100.

- On peut coder 5 valeurs sur 3 bits, dans ce cas, la donnée A=3 est codée par 011.
- Une entrée réelle, dans ce cas, les valeurs 1, 2, 3, 4, 5 peuvent être codées par: 0, 0.25, 0.5, 0.75, 1.

Les PMC peuvent avoir une ou plusieurs sorties réelles. Pour un problème d'estimation d'une ou plusieurs variables, un neurone de sortie sert à estimer une des variables (normalisée). Pour un problème de classification, on peut créer une sortie par classe ou coder en binaire (pour 4 classes, 2 bits suffisent).

- **Choix de l'architecture et réglage des paramètres :**

Dans cette phase, nous définissons le nombre de couches et le nombre de neurones pour chaque couche. Si on souhaite avoir un réseau de neurones ayant un pouvoir de généralisation, il faut choisir une architecture assez riche mais pas trop, le choix se fait soit par expérience, soit par essais successifs.

- **Apprentissage :** [LEF 01]

Il consiste en la mise à jour itérative des poids des connexions jusqu'à l'obtention du comportement désiré. Dans ce qui suit, nous allons exposer le principe général de l'algorithme de rétropropagation **(Figure I.4)** qui est le plus utilisé.

1. **Initialisation** des poids au hasard (de préférence de petites valeurs entre (-0.5 et +0.5).
2. **Choix** d'un exemple en entrée.
3. **Propagation** du calcul de cette entrée à travers le réseau.
4. **Calcul** de la sortie de cette entrée.
5. **Mesure** de l'erreur de prédiction par différence entre sortie réelle et sortie prévue.
6. **Calcul** de la sensibilité d'un neurone (c à d sa contribution à l'erreur) à partir de la sortie.
7. **Correction** des poids des neurones pour atténuer l'erreur.
8. **Aller** à 2 jusqu'à atteindre un seuil minimal d'erreur ou nombre maximal d'itérations.

Figure I.4 : Algorithme de rétropropagation

4.1.1.4. *Avantages et Limites:*

- ✓ Les réseaux traitent des données réelles normalisées et même des données plus complexes (son, images, ...).
- ✓ Le réseau étant construit, le calcul d'une sortie à partir d'un vecteur d'entrée est un calcul très rapide.
- × Il est difficile d'interpréter le modèle obtenu, qui est présenté souvent comme une "boite noire".
- × La difficulté est dans le choix d'une bonne architecture (nombre de couches intermédiaire) et dans le nombre de paramètres qui est élevé (les coefficients synaptiques).
- × L'algorithme n'est pas incrémental, c'est-à-dire, que si les données évoluent avec le temps, il est nécessaire d'entamer une autre phase d'apprentissage.
- × Il faut passer un grand nombre de fois tous les exemples de l'échantillon d'apprentissage avant de converger et donc le temps d'apprentissage peut être long. [GRF XX]

4.1.2. Les plus proches voisins PPV (nearest neighbor)

Cette méthode est utilisée pour résoudre une tâche de classification ou d'estimation. La méthode PPV est une méthode de raisonnement à partir de cas. Elle part de l'idée de prendre des décisions en recherchant un ou des cas similaires déjà résolus en mémoire. Cette méthode ne nécessite pas une phase d'apprentissage, car le modèle est composé de l'échantillon d'apprentissage, de la fonction de distance et de la fonction qui choisit la classe en fonction des classes voisines. L'algorithme générique de classification d'un nouvel exemple par la méthode PPV est :

paramètre: le nombre k de voisins

Donnée: un échantillon de m enregistrements classés $(x^{\rightarrow}, c(x^{\rightarrow}))$

Entrée: un enregistrement $y^{\rightarrow}$

1. déterminer les k plus proches enregistrements de $y^{\rightarrow}$
2. combiner les classes de ces k exemples en une classe c

Sortie: la classe de $y^{\rightarrow}$ est $c(y^{\rightarrow}) = c$

Figure I.5:Algorithme de classification par k-PPV [GRF XX]

4.1.2.1. *Notion de distance*

Le choix de la distance est primordial au bon fonctionnement de la méthode. Pour définir la fonction de distance, on définit d'abord une distance sur chacun des champs, puis on combine ces distances pour définir la distance globale entre enregistrements. Commençons par étudier les choix possibles selon le type du champ. [GRF XX]

► La distance sur les champs:

Attributs numériques :

Soit x, y deux valeurs numériques :

- $d(x,y) = |x - y|$
- $d(x,y) = \frac{|x-y|}{d_{\max}}$ *où* $d_{\max}$ est la distance maximale entre deux réels du domaine considéré.

Attributs discrets:

Données binaires:

- $d(0,0) = d(1,1) = 0.$
- $d(0,1) = d(1,0) = 1.$

Données nominales:

Soit x, y deux données nominales :

- $d(x,y) = 1 \; si \; x = y$.
- $d(x,y) = 0 \; si \; x \neq y$.

► La distance ente deux enregistrements:

Soit o_1, o_2 deux objets de la base d'apprentissage et A le nombre d'attributs :

- La distance Euclidienne: $d(o_1, o_2) = \left(\sum_{k=1}^{A} \left| x_1^k - x_2^k \right|^2 \right)^{\frac{1}{2}}$.
- La distance de Manhattan: $d(o_1, o_2) = \sum_{k=1}^{A} \left| x_1^k - x_2^k \right|$.

4.1.2.2. *Sélection de la classe*

Si k =1, c'est la méthode 1-PPV qui consiste simplement à trouver l'enregistrement le plus proche et à prendre la même décision. Cette méthode est employée pour des problèmes simples pour lesquels les points (les enregistrements) sont bien répartis en

groupes denses d'enregistrements de même classe. [GRF XX]

Pour que le résultat soit plus performant, il est préférable de choisir k >1. Le problème consiste à choisir entre les k classes des voisins les plus proches. Une des solutions les plus utilisée propose d'attribuer la classe majoritaire.

La méthode des plus proches voisins peut être utilisée pour une estimation, il ne s'agit plus de choisir une classe mais de prédire une valeur. Une des solutions consiste à rechercher les plus proches voisins pour appliquer une régression linéaire afin d'estimer cette valeur.

4.1.2.3. *Avantages et Limites*

- ✓ La lisibilité des résultats, car on peut expliquer le choix en montrant les plus proches voisins.
- ✓ Applicable à tout type de données. Il suffit juste de définir une fonction de distance.
- ✓ Peut traiter un grand nombre d'attributs à condition que le nombre d'exemples soit plus important par rapport au nombre d'attributs.
- × La méthode ne nécessite pas de recherche du modèle, par contre, le temps de classification est assez important par rapport aux autres méthodes qui nécessitent une phase d'apprentissage mais effectuent une classification rapide.
- × Le modèle est l'échantillon, il faut donc un espace mémoire important pour le stocker.

4.1.3. Les arbres de décision

Un arbre de décision est une représentation graphique d'une procédure de classification, elle comporte des nœuds internes et des feuilles. Les nœuds internes sont appelés nœuds de décision car ils effectuent un test sur les attributs prédictifs et les feuilles sont étiquetées par les classes à prédire. Les réponses possibles au test correspondent aux labels des arcs issus de ce nœud. Dans le cas de nœuds de décision binaires, les labels des arcs sont omis et, par convention, l'arc gauche correspond à une réponse positive au test. [GRA XX]

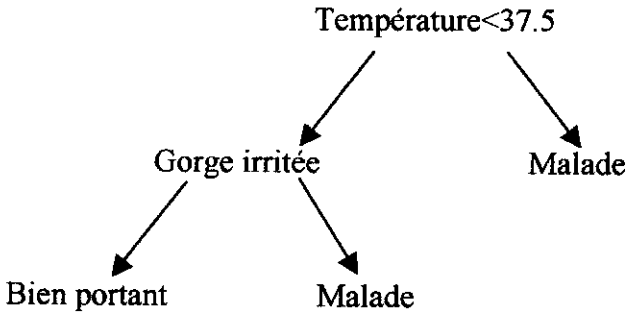


Figure I.6: Exemple d'arbre de décision [GRA XX]

Pour tout nouvel enregistrement à classifier, il suffit d'avoir une description de l'enregistrement pour parcourir l'arbre de la racine jusqu'à la feuille, en suivant les réponses des tests et la classe associée à la feuille sera associée à l'enregistrement.

Les particularités les plus importantes des arbres de décision sont les suivantes :

- Procédure de classification simple et compréhensible par l'utilisateur;
- Pouvoir justifier la classification d'un nouvel enregistrement;
- Lors de la construction de l'arbre, l'algorithme choisit les tests sur les attributs pertinents.

4.1.3.1. *Principe de construction d'un arbre de décision*

L'algorithme d'apprentissage par arbre de décision utilise comme donnée d'entrée, un échantillon S de m exemples (enregistrements) classés à priori. Il fournit en sortie un arbre de décision, cet arbre est le modèle qui servira à classifier de nouveaux enregistrements.

Entrée : Un échantillon S de m enregistrements classés.

Initialisation :

Arbre vide ;

Nœud courant ◄— racine ;

Echantillon courant ◄— S ;

Répéter

Si le nœud courant est terminal **Alors**

Etiqueter le nœud courant par une feuille ;

Sinon

Sélectionner un test et créer le sous arbre ;

Fsi

Nœud courant ◄— Nœud non encore étudié ;

Echantillon courant ◄— Echantillon atteignant nœud courant ;

Jusqu'à production d'un arbre de décision

Elaguer l'arbre de décision obtenu ;

Fin.

Sortie : arbre de décision élagué.

Figure I.7 : Algorithme d'apprentissage par arbres de décision [GRF XX]

Les différentes étapes de l'algorithme seront expliquées de la façon suivante:

▶ Décider si le nœud courant est terminal:

C'est à dire décider si un nœud doit être étiqueté comme une feuille. [GRA XX] La décision est basée sur deux conditions:

- Il n'y a plus d'attributs disponibles; c'est-à-dire que, sur le chemin menant de la racine au nœud courant tous les tests disponibles ont été utilisés.
- Les enregistrements courants sont dans la même classe.

▶ Etiqueter le nœud courant par une feuille:

Affecter au nœud courant la classe majoritaire. Si par exemple, il y a 10 enregistrements de la classe 1 et 50 enregistrements de la classe 0, alors on attribue à la feuille la classe 0. Pour certains problèmes, il est plus approprié de définir le coût ou le risque d'une mauvaise classification.

▶ Sélectionner un test:

Le choix du test ne peut être informel. Par conséquent, il faut introduire des quantités qui permettent de comparer les différents choix possibles. [GRA XX]

Considérons l'exemple suivant: [GRF XX]

S contient 100 exemples, 60 de classe 0 et 40 de classe 1. Le nœud courant sera étiqueté par le couple (60,40). Supposons que deux tests soient disponibles, et que ces deux tests déterminent les répartitions suivantes:

$$(60,40) \rightarrow A\,(30,10)(30,5)(0,25)$$
$$(60,40) \rightarrow B\,(40,20)(20,20)$$

Pour choisir le test, on utilise des fonctions qui mesurent le << degré de mélange >> des différentes classes. Pour les problèmes à deux classes, on peut utiliser une des fonctions suivantes :

- La fonction de Gini : $Gini(x) = 4x(1\text{-}x)$.
- La fonction Entropie : $Entropie(x) = -x \log x - (1\text{-}x) \log (1\text{-}x)$.

où x désigne la proportion d'éléments dans l'une des deux classes. Ces deux fonctions sont à valeurs dans l'intervalle réel [0,1], prennent leur minimum pour x=0 ou x=1 (tous les exemples sont dans une même classe) et leur maximum lorsque x=1/2 (les exemples sont également répartis entre les deux classes).

Choisissons, par exemple, la fonction de Gini. Pour le nœud courant, x=60/100 et $Gini(x)$=4 × 60/100 × 40/100 = 0.96. Si on choisit le test A, pour le premier fils (le plus à gauche), x=3/4 et $Gini\ (x)$ =0.75. Pour le second fils, x=6/7 et $Gini\ (x)$ =0.49. Pour le troisième fils, $Gini\ (x)$ =0. Pour comparer les trois tests, on estime le "degré de mélange espéré" en pondérant les degrés de mélange des fils par la proportion des exemples allant sur ce fils, on obtient:

- Pour A: 40/100 × 0.75 + 35/100 × 0.49 + 25/100 × 0 = 0.47
- Pour B: 60/100 × 0.89 + 40/100 × 1 = 0.93

Il reste à définir une fonction permettant de choisir le test qui doit étiqueter le nœud courant [GRA XX]. Pour cela, on introduit une fonction gain qui est calculée par la soustraction entre le degré de mélange du nœud courant et le degré de mélange espéré par l'introduction du test, on choisit alors le test qui apporte le gain maximal.

Quand le nombre de classe est supérieur à deux, les fonctions Entropie et Gini seront calculées de la façon suivante :

$$Entropie\,(p) = -\sum_{k=1}^{c} P\left(k/p\right) \times \log\left(P\left(k/p\right)\right)$$

$$Gini(p) = 1 - \sum_1^c P\left(k/p\right)^2$$

$$= 2\sum_{k<k'} P\left(k/p\right) P\left(k'/p\right)$$

► **Elaguer l'arbre de décision obtenu:** [GRF XX]

L'objectif d'une procédure de classification est de bien classer des exemples non encore rencontrés, on parle de pouvoir de généralisation. Si l'algorithme fournit en sortie un arbre très grand qui classe bien l'échantillon d'apprentissage, on se trouve confronté au problème de sur-spécialisation : on a appris << par coeur >> l'ensemble d'apprentissage, mais on n'est pas capable de généraliser. L'objectif de la phase d'élagage est d'obtenir un arbre plus petit (on élague des branches, c'est-à-dire que l'on détruit des sous-arbres) dans le but d'obtenir un arbre ayant un meilleur pouvoir de généralisation (même si on fait augmenter l'erreur sur l'ensemble d'apprentissage).

4.1.3.2. *Avantages et Limites*

- ✓ L'avantage principal de ces produits est sans conteste la lisibilité du modèle construit. [LEF 01]
- ✓ Pouvoir justifier la classe d'un exemple en présentant son chemin de la racine jusqu'à la feuille.
- ✓ Applicable sur des bases d'exemples volumineuses où les attributs sont de types différents (continus, discrets, ...).
- ✓ Adapté pour la sélection des attributs pertinents dans la phase de pré-traitement des données.
- ✗ L'algorithme n'est pas incrémental, c'est-à-dire que si les données évoluent avec le temps, il est nécessaire de relancer une phase d'apprentissage.
- ✗ Lorsque le nombre de classes augmente, la performance du modèle diminue.

4.1.4. Les algorithmes génétiques

Les algorithmes génétiques sont basés sur le principe de survie de structures les mieux adaptés et les échanges d'informations. A chaque génération, un nouvel ensemble de créature artificiels (chromosomes ou individus), codées sous forme de chaînes de bits, est construit à partir des meilleurs éléments de la génération précédente en utilisant des techniques dérivées de la génétique et de l'évolution naturelle: reproduction, croisement, mutation. Ce sont des algorithmes itératifs dont le but

est d'optimiser une fonction appelée " Fonction d'évaluation" qui associe un coût à chaque solution. [GOL 94]

4.1.4.1. *Principe des algorithmes génétiques*

Le principe général des algorithmes génétiques est représenté dans l'organigramme suivant :

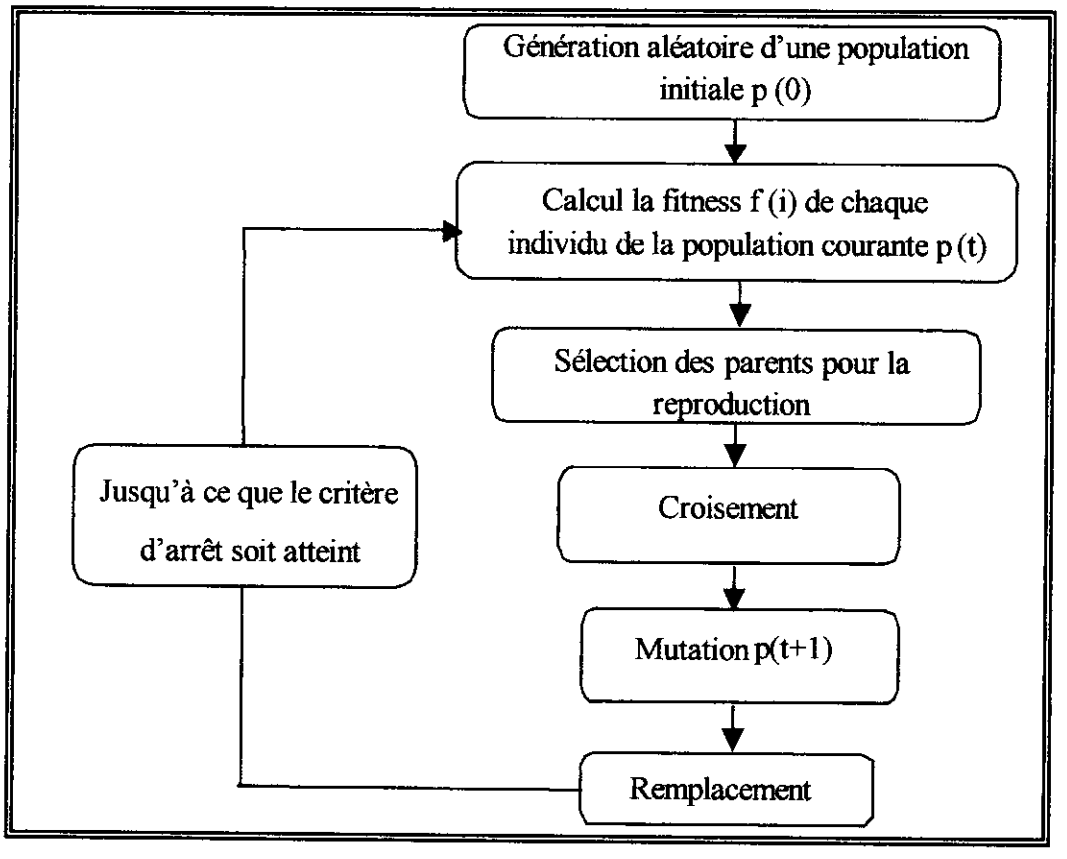


Figure I.8: Fonctionnement général d'un AG de base [JOU 03]

4.1.4.1.1. *Fonction d'évaluation*

La fonction d'évaluation quantifie la qualité de chaque chromosome par rapport au problème. Elle est utilisée pour sélectionner les chromosomes pour la reproduction. Les chromosomes ayant une bonne qualité (fitness) ont plus de chance d'être sélectionnés pour la reproduction et donc plus de chance que la population suivante hérite de leur matériel génétique. [JOU 03]

4.1.4.1.2. *Opérateurs génétiques*

Les opérateurs génétiques travaillent directement sur les individus composant la population :

► Opérateur d'initialisation:

Cet opérateur est utilisé pour générer la population initiale de l'algorithme génétique [JOU 03]. La population initiale doit être bien répartie pour disposer d'une bonne variation des solutions, souvent ces solutions sont générées de façon aléatoire.

► Opérateur de sélection:

Il consiste à définir les solutions qui serviront à la reproduction de la population. Lors de la sélection, les individus les plus adaptés sont privilégiés par rapport au reste de la population, c'est une heuristique utilisée par l'algorithme génétique: les bonnes solutions sont supposées être les plus prometteuses pour la génération de descendants. [VEN 96]

Les méthodes de sélection les plus utilisées sont:

- **Sélection par fitness:**

La sélection par fitness (ou sélection par proportionnalité) est en général implémentée comme une simulation d'une roulette biaisée.[GOL 89]

Chaque individu est représenté par une partie de la roue de taille proportionnelle à sa fitness. Ainsi un individu c_i a la probabilité suivante d'être sélectionné:

$$P_{sélection} = \frac{F_{évaluation}(c_i)}{\sum_{j=1}^{n} F_{évaluation}(c_j)}$$

où n représente la taille de la population de l'algorithme génétique.

Lors de la sélection, on fait tourner la roue, lorsqu'elle s'arrête, le curseur pointe sur une partie de la roue, on choisit alors l'individu correspondant à cette partie.

- **Méthode élitiste:**

Elle consiste à trier la population par ordre décroissant de leur fitness, et sélectionner les individus ayant les meilleurs fitness. Elle présente beaucoup d'inconvénients: une variance nulle et une diversité inexistante. [SOU 04]

- **Sélection par tournois:** [SOU 04]

Cette méthode est celle avec laquelle on obtient les résultats les plus satisfaisants, le principe de cette méthode est le suivant : on effectue un tirage avec remise de deux individus de P, et on les fait "combattre". Celui qui a la fitness la plus élevée l'emporte avec une probabilité p comprise entre 0.5 et 1. On répète ce processus n fois de manière a obtenir les n individus de P' qui serviront de parents.

► Opérateur de croisement (crossover): [SOU 04]

Le croisement permet de reproduire une nouvelle population plus ou moins distincte de la population précédente. Cet opérateur travaille sur les individus sélectionnés par l'opérateur de sélection. On sélectionne de manière aléatoire des couples (parents), puis on recombine leurs chromosomes pour obtenir des fils. Les méthodes utilisées pour le croisement sont:

- Le croisement un point détermine aléatoirement un point de coupure et échange la deuxième partie des deux parents.
- Le croisement deux points (qui peut être étendu à x points) possède 2 points (ou x) de coupures qui sont déterminés aléatoirement.
- Le crossover uniforme échange chaque bit avec une probabilité fixé à 1/2.

➢ Opérateur de mutation:

Cet opérateur consiste simplement à inverser la valeur d'un bit du chromosome avec une probabilité de mutation P_m très faible. La mutation transforme donc aléatoirement les solutions d'une population. Cet opérateur introduit une diversité nécessaire à l'exploration de l'espace de recherche en permettant d'explorer des solutions dans des régions à priori sans intérêt.[VEN 96]

➢ Opérateur de remplacement: [JOU 03]

Cette dernière étape du processus itératif consiste en l'incorporation des nouvelles solutions dans la population courante. Les nouvelles solutions sont ajoutées à la population courante en remplacement (total ou partiel) des anciennes solutions. Généralement, les meilleures solutions remplacent les plus mauvaises ; il en résulte une amélioration de la population. Lorsque la nouvelle population n'est constituée que de nouvelles solutions, on parle d'algorithme génétique générationnel.

4.1.4.2. *Application pour la classification*

Il existe plusieurs travaux de recherches des règles de prédiction par algorithme génétique. Les travaux dans [ROM 04] présentent un exemple d'utilisation des algorithmes génétiques. Le principe de ces travaux est décrit comme suit :

- **Le codage:** chaque individu représente une règle de prédiction. Chaque individu représente plus précisément l'antécédent (IF part) d'une règle de prédiction. Par conséquent, pour prédire les différentes valeurs d'un attribut, on a besoin d'exécuter l'algorithme génétique plusieurs fois, une fois pour chaque valeur de chaque attribut. Les règles de prédiction sont représentées en *logique floue*, c'est à dire

qu'une discrétisation est appliquée sur les attributs explicatifs continus, de telle sorte que des termes linguistiques flous remplacent les valeurs de ces attributs.

Un individu représente une conjonction de conditions spécifiant un antécédent de règle de déduction. Chaque condition est représentée par un gène qui consiste en un doublet attribut valeur de la forme $Ai = V\,i\,,\,j$ où Ai est le i-ème attribut et $V\,i\,,\,j$ est la j-ème valeur appartenant au domaine de Ai.

De manière à simplifier l'encodage des conditions dans le génome, on utilise un encodage positionnel où la i-ème condition est encodée dans le i-ème gène. Par conséquent, il est seulement nécessaire de représenter la valeur $V\,i\,,\,j$ de la i-ème condition dans le génome puisque l'attribut de la i-ème condition est implicitement déterminée par la position du gène dans le génome.

De plus, chaque antécédent contient un flag booléen qui indique si la i-ème condition est présente ou pas dans l'antécédent de la règle. Par conséquent, bien que tous les individus aient la même longueur pour leur génome, des individus différents représentent des règles de longueurs différentes selon la valeur des flags booléens.

- **La fonction de fitness:** de nombreux algorithmes de découverte de règles de prédiction se concentrent sur l'évaluation de la précision des règles de prédiction, sans chercher à savoir si les règles découvertes sont réellement intéressantes pour l'utilisateur. Par conséquent, la fonction de fitness prend en compte deux critères:
  - Le critère de précision de la règle, noté Acc.
  - Le critère d'intérêt de surprise, noté *Surp.*

  La fonction de fitness est calculée par la formule suivante:

  Fitness (i) = Acc (i) * Surp (i).

- **Sélection, croisement et mutation:** la méthode des tournois a été utilisée pour la sélection. Le croisement uniforme a également été mis en œuvre. Une transformation aléatoire de la valeur d'un attribut (gène) a été appliquée pour la mutation.

4.1.4.3. *Avantage et Limite*

- ✓ Les AG produisent des résultats qui peuvent être expliqués par les gènes contenus dans l'individu.
- ✓ Les AG peuvent s'appliquer à, presque, tous les types de données, il suffit de représenter les données par des chaînes de bits de longueurs fixes.
- ✗ La tâche la plus difficile dans l'utilisation des AG est que le problème doit être

codé afin qu'il puisse être représenté par des individus de longueurs fixes à optimiser en se servant d'une fonction d'adaptation. Le type de codage affecte la qualité du résultat.

- Les AG peuvent exiger une grande puissance de calcul, cet inconvénient peut être lié à une population trop importante ou à une fonction d'évaluation trop coûteuse.
- Il peut arriver que les AG convergent vers des solutions sous optimales. Cela arrive en particulier lorsque la population est trop petite, dans ce cas une solution peut apparaître rapidement et dominer alors que l'espace n'a pas été suffisamment exploré.

4.2. Les méthodes de classification automatique (clustering)

La classification automatique a pour but d'obtenir une représentation simplifiée des données initiales [GOV 04]. Il s'agit de sectionner un ensemble de données en classes homogènes (groupes ou clusters). La classification automatique appelé aussi clustering peut prendre plusieurs autre termes selon le domaine d'application, en science naturelle on parle de *systématique*, de *taxinomie* ou de *taxonomie*; en marketing, on parle de *typologie*; en médecine, pour classifier les maladies, on utilise le mot *nosologie*.

Les techniques employées pour des opérations de classification automatique relèvent de ce que nous appelons *l'apprentissage non supervisé*. Nous parlons d'apprentissage non supervisé car l'utilisateur ne sait pas *a priori* quelles classes, groupes ou catégories il va obtenir. Ce mode d'apprentissage est également appelé « apprentissage sans professeur ».[MOR XX] Les méthodes de classification automatique (cluster analysis) peuvent être réparties en deux groupes:

4.2.1. Les méthodes monothétiques

Pour ces méthodes, le regroupement est basé sur le partage de certaines caractéristiques entre les individus d'une même classe, de telle sorte que chacune de ces caractéristiques représente un attribut d'une valeur constante ou de très faible variance. Considérons, par exemple, le nombre de doigts d'un être vivant et comparons le singe et l'homme: sur ce critère de comparaison (et sur bien d'autres) les deux espèces seront jugées semblables. [GOV 04]

4.2.2. Les méthodes polythétiques

Ces méthodes sont basées sur la notion de proximité (distance, dissimilarité). Les individus regroupés dans la même classe ont des caractéristiques proches par rapport aux autres individus. Contrairement à l'approche monothétique, l'approche polythétique prend en compte tous les attributs. Il existe deux techniques principales :les techniques hiérarchiques et les techniques de partitionnement. Ces techniques sont décrites dans ce qui suit :

4.2.2.1. *Les techniques hiérarchiques*

Les méthodes hiérarchiques construisent une suite de partitions imbriquées. Il existe deux types de méthodes hiérarchiques :

- Descendantes :elles consistent à diviser l'ensemble des individus en classes et de continuer la division jusqu'à obtenir des classes contenant un seul individu.
- Ascendantes :chaque individu représente une classe. Ces classes sont regroupées successivement jusqu'à obtenir un cluster qui comprend la totalité des individus. La méthode par agglomérations est la méthode hiérarchique ascendante la plus utilisée.

1- Initialisation:

- représenter chaque individu dans une classe (singleton).
- calculer la distance entre les classe.

2- Répéter

- regrouper les deux classes les plus proches.
- calculer les distances entre les nouveaux clusters et les anciens clusters non encore regroupés.

Jusqu'à ce que le nombre de classes soit égal à 1.

Figure I.9 :Algorithme d'agglomération

On a ainsi construit une suite de partitions de l'échantillon d'entrée où le nombre k de groupes varie de m à 1. Pour choisir la valeur de k, à chaque étape, on calcule la distance intra groupe et la distance inter groupes. La distance intra groupe peut être une moyenne des distances entre éléments du groupe. Pour calculer la distance inter groupes, il existe trois possibilités :

- Le rattachement simple où la distance entre deux groupes est distance entre

- Le rattachement complet où la distance entre deux groupes est donnée par la distance entre les membres les plus éloignés.
- La distance entre les centres.

On choisit alors la valeur de *k* qui minimise la distance intra groupe et maximise la distance entre groupes. [GRF XX]

4.2.2.2. *Les techniques de partitionnement*

Le but est de diviser les objets en *k* clusters, on ne construit alors qu'une seul partition des objets. Pour le partitionnement, deux approches heuristiques sont souvent utilisées:

4.2.2.2.1. *La Méthode des K-means*

Connue aussi sous le nom des k-moyennes ou des centres mobiles. L'algorithme est fondé sur le concept de similarité : deux points sont similaires, s' ils sont proches pour la distance considérée(la mesure de distance est définie dans la section 4.1.2.1).

Supposant que le nombre k de groupes soit connu a priori, on choisit alors *k* enregistrements, soit *k* points de l'espace appelés les centres. L'algorithme parcourt alors toute la base en affectant chaque élément au cluster dont le centre est le plus proche. La moyenne des points de chaque clusters est ensuite calculée pour obtenir de nouveaux centres. Le procédé est répété jusqu'à atteindre une certaine stabilité dans les clusters formés.

Paramètres: *k* le nombre de clusters

Entrée: un une base d'apprentissage de m enregistrements (points) $x_1, \ldots x_m$

1. sélectionner *k* points comme les représentants initiaux
2. Pour $i = 1$ à m, affecter l'élément x_i au centre le plus similaire (ceci donne *k* clusters)
3. Pour $j = 1$ à *k*, recalculer le centre du cluster C_j
4. répéter les étapes **2** et **3** jusqu'à ce qu'il n'y ait plus (ou peu) de changements dans les clusters.

Figure I.10 :Algorithme des K-means

Avantages et Limites:

- ✓ Appliqué à tout type de données; il suffit juste de définir la distance pour chaque type.
- ✓ Vu sa facilité, il est implanté dans tous les logiciels de *data mining*;

- ✓ Vu sa facilité, il est implanté dans tous les logiciels de *data mining*;
- ✓ Les données nécessitent seulement une normalisation avant l'application de la méthode.
- × Les performances de la méthode sont dépendantes du choix d'une bonne mesure de similarité. C'est une tâche délicate, surtout lorsque les données sont de types différents. [GRF XX]
- × Besoin de préciser k à l'avance.

4.2.2.2.2. *La Méthode des K-medoïds*

Elle est appelé également ***méthode des nuées dynamiques***. Le cluster n'est plus représenté par le centre de gravité comme le k-moyennes mais plutôt par l'un des éléments du cluster.

4.3. Les règles d'association

La recherche d'association vise à construire un modèle fondé sur des règles conditionnelles, à partir d'un fichier de données [LEF 01]. La règle est représentée sous la forme: Si **condition** alors **résultat**.

Les règles d'association sont traditionnellement liées au secteur de la distribution car leur principale application est « *l'analyse du panier de la ménagère* » qui consiste en la recherche d'associations entre produits sur les tickets de caisse. Le but de la méthode est l'étude de ce que les clients achètent pour obtenir des informations sur qui sont les clients et pourquoi ils font certains achats. La méthode recherche *quels produits tendent à être achetés ensemble.*[GRF XX]

Les règles engendrées par cette méthode peuvent être non triviales, mais elles peuvent par contre être évidentes et sans aucun intérêt pour l'utilisateur. La recherche de règles d'association est une méthode non supervisée, car on ne dispose en entrée que de la description des données [GRF XX]

4.3.1. Principe de construction des règles d'association

Nous allons nous baser sur un exemple des tickets de caisse dans un supermarché (**Tableau I.1**) pour expliquer le processus de construction des règles d'association. [LEF 01]

Ticket1	Ticket2	Ticket3	Ticket4
Farine	Œuf	Farine	Œuf
Sucre	Sucre	Œuf	Chocolat
Lait	Chocolat	Sucre	Thé
		Chocolat	

Tableau I.1 : Matrice représentant des tickets de supermarché

Une association est une implication de la forme X ⟶ Y.

Le ticket1 contient :

Farine ⟶ Sucre Sucre ⟶ Farine Sucre ⟶ Lait

Farine ⟶ Lait Lait ⟶ Farine Lait ⟶ Sucre

Une association s'apprécie au travers de deux indicateurs :

- **Le niveau de confiance :**il correspond au nombre de fois où l'association X ⟶ Y est présente, rapporté au nombre de présence de l'article X.
- **Le niveau de support :**il correspond au nombre de fois où l'association X ⟶ Y est présente, rapporté au nombre de tickets comportant l'article X ou Y.

➢ **L'extraction des associations pertinentes :**

Le processus d'extraction des associations se déroule en deux phases distinctes : il isole les articles présentant un niveau de support supérieur à un certain seuil, puis combine les articles les plus représentés pour générer les associations.

Tout d'abord, on dénombre le nombre de fois où un article est présent dans l'ensemble des tickets **(Tableau I.2)**.

Article	Fréquence
Farine	2
Sucre	3
Lait	1
Œuf	3
Chocolat	3
Thé	1

Tableau I.2 : Matrice représentant le nombre de fois où un article apparaît

Si l'on décide par exemple, de retenir un taux de support supérieur à 30%, alors les articles Lait et Thé, qui ont un taux de support de 25% sont éliminés.

La seconde étape combine les articles restants pour former l'ensemble de toutes les associations et leur dénombrement **(Table I.3)**:

Association de niveau 2	Fréquence
Farine – Sucre	2
Farine – Œuf	1
Farine – Chocolat	1
Sucre – Œuf	2
Sucre – Chocolat	2
Œuf – Chocolat	3

Tableau I.3 : Matrice représentant le nombre d'associations

De la même manière, on élimine les associations qui présentent un taux de support inférieur à 30% c'est à dire Farine – Œuf et Farine – Chocolat.

La troisième étape consiste à créer les triplets possibles. Comme Farine n'est présent que dans un seul couple il ne contribue pas à la création d'un triplet. Il ne reste qu'un seul triplet Sucre – Œuf – Chocolat qui représente un taux de support de 50%. La constitution de quadruplets est impossible dans cet exemple, l'algorithme se termine donc à ce niveau.

► **Identification des associations les plus fortes :**

- Farine ⟶ Sucre : taux de confiance 100% et support 66%
- Sucre ⟶ Farine : taux de confiance 66% et support 66%
- Sucre ⟶ Œuf : taux de confiance 66% et support 50%
- Oeuf ⟶ Sucre : taux de confiance 66% et support 50%
- Sucre ⟶ Chocolat : taux de confiance 66% et support 50%
- Chocolat ⟶ Sucre : taux de confiance 66% et support 50%
- Oeuf ⟶ Chocolat : taux de confiance 100% et support 100%
- Chocolat ⟶ Oeuf : taux de confiance 100% et support 100%
- Œuf – Chocolat ⟶ Sucre : taux de confiance 66% et support 50%

4.3.2. Avantages et Limites

✓ La lisibilité du modèle conçu (facile à interpréter).

- ✓ La méthode est l'une des rares méthodes qui prend en entrée des achats qui sont des listes d'articles de taille variable. [GRF XX]
- ✓ La méthode est facile à implémenter et elle existe dans la plupart des environnements du *data mining.*
- ✗ Les règles d'association sont très coûteuses en temps de calcul.

5. Principaux domaines d'application

Les principales applications du *data mining* sont relatives aux clients (domaines du marketing, fidélisation client, détection de fraude), en effet, dans un environnement plus concurrentiel et avec des clients plus exigeants, l'entreprise doit connaître de manière approfondie ses clients pour proposer les meilleurs services qui répondent aux besoins de ses clients et augmentent son rendement. Pour mieux répondre à la demande du client, la connaissance de son comportement est décisive.

Selon les domaines, on présentera les principales applications du *Data Mining* :[LEF 01]

➢ **Grande distribution et vente par correspondance (VPC)**

- Analyse des comportements des consommateurs.
- Recherche des similarités des consommateurs en fonction de critères géographiques ou sociodémographiques.
- Prédiction des taux des réponses en marketing direct.
- Vente croisée et activation sélective dans le domaine des cartes de fidélité.
- Optimisation des réapprovisionnements.

➢ **Laboratoires pharmaceutiques**

- Modélisation comportementale et prédiction de médication ou de visites.
- Optimisation des plans d'actions des visiteurs médicaux pour le lancement de nouvelles molécules.
- Identification des meilleures thérapies pour différentes maladies.

➢ **Banques**

- Recherche de formes d'utilisation de cartes caractéristiques d'une fraude.
- Modélisation prédictive des clients partants.
- Détermination de pré-autorisations de crédits revolving.
- Modèles d'arbitrage automatique basé sur l'analyse de formes historiques des cours.

➢ **Assurances**

- Modèles de sélection et de tarification.
- Analyse des sinistres.
- Recherche des critères explicatifs du risque ou de fraude.
- Prévision d'appel sur les plates formes d'assurances directes.

6. Conclusion

Nous avons essayer dans ce chapitre de donner un aperçu général sur la technologie de *Data Mining*, sur ces principales méthodes et ces importantes applications dans les entreprises.

Le *Data Mining* exploite des techniques dont le but est d'aboutir à des connaissances opérationnelles qui accélère la prise de décision et les actions de l'entreprise. Cependant, il faut pas voir le *Data Mining* comme solution à tout problème d'entreprise, car il correspond seulement à une nouvelle technologie de valorisation de grande masse d'information.

Chapitre II

Métaheuristiques pour le Partitionnement

1. Introduction

Si on souhaite résoudre le problème de partitionnement des données par énumération des partitions(solutions) possibles par ordinateur, le nombre de partitions noté $S(N,K)$ tel que N est le nombre d'objets et K le nombre de groupes serait alors calculé de la façon suivante [JAI88]:

$$S(N,K)=\frac{1}{K!}\sum_{i=1}^{K}(-1)^{K-i}\binom{K}{i}(i)^{N}$$

par exemple, S (10,4) =34 105 et S (20,4) =11 259 666 000 (!).

Si le nombre d'objets à classer est grand, le problème ne pourra pas être résolu par une approche exacte en un temps raisonnable. Le problème de partitionnement pourrait donc être exprimé comme un problème d'optimisation combinatoire[1].

Depuis les années 70, afin de trouver un compromis entre la qualité des solutions obtenues et le temps de calcul utilisé, des méthodes sont apparues, souvent inspirées de systèmes naturels (biologie de l'évolution, physique, éthologie...). Ces méthodes assurent un compromis entre diversification – quand il est possible de déterminer que la recherche se concentre sur de mauvaises zones de l'espace de recherche – et intensification – on recherche les meilleures solutions dans la région de l'espace de recherche en cours d'analyse. Ces méthodes ont été appelées métaheuristiques.

Dans ce chapitre, nous citons les métaheuristiques les plus connues et nous verrons pour chaque méthode les différentes applications en classification.

2. Méthodes à solution unique

Les méthodes itératives à solution unique sont toutes basées sur un algorithme de recherche de voisinage qui commence avec une solution initiale, puis l'améliore pas à pas en choisissant une nouvelle solution dans son voisinage. [BAC 99]

[1] :L'optimisation combinatoire consiste à trouver le meilleur entre un nombre fini de choix.

Nous présenterons ici les méthodes les plus utilisées et leur application en classification: les méthodes de descente, le recuit simulé et la recherche tabou.

2.1 Les méthodes de descentes (Hill Climbing)

L'illustration la plus simple des méthodes itératives est ce que l'on appelle les méthodes de descentes (*Hill-Climbing*: HC). Elles sont issues des travaux de [PAP 76, PAP 82] et doivent leur succès à leur simplicité et leur rapidité.

A chaque itération, l'algorithme s'améliore vers une solution voisine de meilleure qualité par rapport à la solution courante et de tous les autres voisins. Il s'arrête lorsque aucun voisin n'est meilleur que la solution courante (un optimum local est atteint).

Procédure: φ fonction de coût;
Variable locale: S solution courante;
Choix d'une solution initiale S_0;
Solution courante $S \leftarrow S_0$;
(a) Génération des candidats par voisinage;
Choix du meilleur candidat C dans ce voisinage;
Si $\varphi(C) < \varphi(S)$ **Alors**
$S \leftarrow C$;
Aller en **(a)**;
Fin Si
Return S

Figure II.1 :Méthode de descente générique

Dans le cadre de la résolution du problème de partionnement, on peut évoquer l'algorithme des K_means qui est très utilisé pour le clutering [FUK 90].

2.2 Le recuit simulé (Simulated Annealing)

Méthode mise au point par des chercheurs de la société IBM [KIR 83]. Le recuit simulé s'appuie sur l'algorithme de Metropolis, qui permet de décrire l'évolution d'un système thermodynamique . [HAS XX]

L'analogie historique s'inspire du recuit des métaux en métallurgie: un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un optimum local pour un problème d'optimisation combinatoire. Si on le refroidit

lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalente à un optimum global. [JOU 03]

Au début de l'algorithme, une solution initiale peut être prise aléatoirement dans l'espace de recherche, une température initiale T0 est affecté au paramètre T (température du système) qui décroît tout au long de l'algorithme pour tendre vers 0.

A chaque itération, la solution générée est affectée à la solution courante si elle de meilleur qualité, sinon elle sera accepté avec une probabilité p, évitant ainsi de tomber dans un optimum local.

Les principaux inconvénients du recuit simulé résident dans le choix des nombreux paramètres, tels que la température initiale, la loi de décroissance de la température, le critère d'arrêt. Ces paramètres sont souvent choisis de manière empirique. [HASXX]

i = 0; T = Température initiale
X_c = Solution initiale
Répéter
  Répéter
    Générer une autre solution X_n à partir de X_c (dans son voisinage)
    Si $f(X_c) < f(X_n)$ Alors $X_c = X_n$
    Sinon r=Nombre aléatoire compris entre [0,1]
      $p = \exp\left[\frac{f(X_n) - f(X_c)}{T}\right]$
      Si $r < p$ Alors $X_c = X_n$
    Finsi
  Jusqu'à un REP /* REP nombre d'itérations prédéfini
  Diminuer la température de T selon une stratégie
  i = i+1
Jusqu'à critère d'arrêt
Fin

Figure II.2: Méthode générique de recuit simulé

Dans le cadre de la classification, Brown et al. [BRO 90] proposent SINICC (Simulation of Near-Optima for Internal Clustering Criteria) pour réaliser la classification hiérarchique.L'opérateur de perturbation utilisé est très similaire au K_means: il prend un point au hasard et l'affecte aléatoirement à un autre cluster. Les auteurs ont travaillé sur différentes fonctions objectif et ont appliqué SINICC à un problème réel: la surveillance radar. [JOU03]

2.3 La recherche Tabou (Tabu Search)

Méthode d'optimisation présentée par F Glover [GLO 86]. Elle n'a aucun caractère stochastique et utilise la notion de mémoire pour éviter de tomber dans un optimum local [JOU03].

A chaque itération, l'algorithme explore tout le voisinage de la solution courante et sélectionne la solution voisine qui minimise la fonction objectif, même si cette solution est moins bonne que la solution courante. Ceci permet d'échapper à l'optimum local.

Le risque cependant est de rentrer dans des cycles. C'est pourquoi il faut que l'heuristique dispose d'une mémoire. Le mécanisme consiste à interdire (d'où le nom de "tabou") de revenir sur les dernières positions explorées .[HAT XX]

```
Initialiser S de manière aléatoire
Sopt ← S
Ajouter S à la liste tabou
Tantque la condition d'arrêt n'est pas vérifiée Faire
   S' =Voisin (S) //Fonction qui retourne le meilleur voisin de S
   Tant que S' est dans la liste tabou Faire
      S' =Voisin (S)
   Fin Tantque
   Si la liste tabou est pleine alors
      Remplacer le dernier élément de la liste tabou par la solution S'
   Sinon
      Ajouter S' à la liste tabou
   Finsi
   Si S' est meilleur que Sopt alors
      Sopt ← S'
   Finsi
   S ← S'
 Fin Tantque
```

Figure II.3: Méthode générique de Tabou

Al-Sultan [ALS95] applique la recherche Tabou pour classifier des modèles. Un ensemble de solutions tests est généré à partir de la solution courante. Pour chaque exemple, un nombre aléatoire $0<=R<=1$ est généré. Si ce nombre est supérieur ou égal à une probabilité P_i, alors le modèle est changé aléatoirement de classe (P_i est un seuil de probabilité fixé); sinon on partitionne comme dans la meilleure solution. De la meilleure à la moins bonne solution, si le critère d'aspiration est satisfait ou une condition taboue évitée, alors la solution test est choisie comme la meilleure solution courante et chaque exemple affecté au cluster i est stocké dans la liste taboue. Si toutes

les solutions tests sont taboues, alors les solutions sont régénérées à partir de la meilleure solution trouvée. [JOU 03]

3. Méthodes évolutionnaires

Les méthode évolutionnaire se basent sur les grands principes rencontrés dans la nature et notamment celui de l'évolution des espèces et de la sélection naturelle énoncés par Charles Darwin [DAR 59].

Deux types d'algorithmes évolutionnaires ont été développés isolément par différents scientifiques: les Algorithmes Génétiques [HOL 75] et la programmation génétique [KOZ 92].

3.1 Algorithmes génétiques

Les algorithmes génétiques simulent le processus d'évolution d'une population. A partir d'une population de N solutions du problème représentant des individus, on applique des opérateurs simulant les interventions sur le chromosome telles que le croisement (crossover) ou la mutation pour arriver à une population de solutions de mieux en mieux adaptée au problème. Cette adaptation est évaluée grâce à une fonction coût [JOU 03]. Nous avons détaillé plus amplement les algorithmes génétiques dans le Chapitre I section 4.1.4.

Il existe plusieurs applications des AG pour résoudre la tâche de patitionnement, nous présentons juste quelques travaux :

Les travaux présentés dans [VON 91], chaque individu de la population représente une partition d'objets et la population regroupe un ensemble de partitions. Soit N le nombre d'objets à regrouper et K le nombre de groupes, une partition est alors représentée par une chaîne de N entiers $(o_1,.....,o_N)$ où $o_i \in \{1,.....,K\}\ \forall i$. Si $o_i = o_j$ alors les objets o_i et o_j se trouvent dans le même groupe. Si par exemple un individu est codé de la façon suivante : $(2,1,2,3,3,2)$ alors la partition représentée par cet individu est : $groupe_1 = \{o_2\}$, $groupe_2 = \{o_1,o_3,o_6\}$, $groupe_3 = \{o_4,o_5\}$.

D'autres représentations ont été proposées dans [JON 91], cette approche introduisant des séparateurs dans la chaîne d'entiers représentant une partition :par

exemple si N =6, l'individu $(4,3,7,1,5,2,8,6)$ donne la partition $groupe_1 = \{o_4, o_3\}$, $groupe_2 = \{o_1, o_5, o_2\}$, $groupe_3 = \{o_6\}$ avec 7 et 8 comme séparateurs.

Faulkenauer a fait remarquer que ces représentations prenaient en compte uniquement l'affectation des objets aux classes et a proposé un codage des solutions incluant les groupes [FAL 94], par exemple, les deux partitions suivantes :

- Partition1 : $groupe_1 = \{o_2\}, groupe_2 = \{o_1, o_3, o_6\}$, $groupe_3 = \{o_4, o_5\}$ sera codée par $(2,1,2,3,3,2 : 2,1,3)$.
- Partition2 : $groupe_1 = \{o_1, o_5, o_6\}$ et $groupe_2 = \{o_2, o_3, o_4\}$ sera codée par $(1,2,2,2,1,1 : 1,2)$.

où la première partie (avant les deux points) représente l'affectation des objets aux groupes et où la deuxième représente les groupes présents. Le principal avantage de cette représentation apparaît lorsque l'opérateur de croisement est appliqué :seules les deuxièmes parties sont croisées, les premières parties en subissent les conséquences. [MON 00]

3.2 Programmation génétique (Genetic Programming)

Le principal promoteur de ce paradigme est J. Koza [KOZ 92], l'algorithme consiste à faire évoluer une population constituée d'un grand nombre de programmes. La plupart des algorithmes de programmation génétique travaille avec une population modélisée sous forme d'arbres. Lors de la recherche, la profondeur de ces arbres peut augmenter fortement en taille. Au départ, la population est constituée de programmes créés aléatoirement. Chaque programme est évalué selon une méthode propre au problème posé. A chaque itération (génération), on classe les programmes en fonction des notes qu'ils ont obtenues, et on créée une nouvelle population, où les meilleurs programmes auront une plus grande chance de survivre ou d'avoir des enfants que les autres. Ce principe est le même qu'en algorithmique génétique classique, mais les opérateurs de croisement et de mutation sont différents. En effet, ils travaillent directement sur la structure d'arbre du programme.[JOU 03]

Selon nos recherches, nous n'avons pas trouvé d'article qui traite la classification non supervisée avec la PG.

4. Colonies de fourmis

Les fourmis artificielles font partie de ce que les chercheurs nomment l'intelligence en essaim ou intelligence collective artificielle, dont le comportement proposé par Deneubourg et al. [DEN 83, DEN 89] est le suivant: les fourmis se déplacent du nid à la recherche de nourriture. Elles déposent sur le sol une matière odorante appelée phéromone. Cette dernière aidera les fourmis à retrouver le chemin du retour vers leur nid et permettra à d'autres fourmis de choisir le même chemin afin de trouver les sources de nourriture détectées par leurs congénères.

Fondée sur ces observations, l'ACO (Ant Colony Optimization) est une méthode d'optimisation [DOR 91]. Elle a été développée au début pour le problème du voyageur de commerce. Elle a été, par la suite, largement appliquée à d'autres problèmes d'optimisation combinatoire. Cependant, ce modèle n'a pas été utilisé pour résoudre le problème de partitionnement où il a fallu exploiter un autre comportement [DEN 90]: les fourmis artificielles se déplacent sur un plan. Les objets à rassembler sont répartis sur ce plan. Une fourmi ne dispose que d'une perception locale de ces objets et ne communique pas avec les autres. Les objets sont ramassés et déposés avec des probabilités fixes. Ces principes relativement simples font qu'il apparaît des regroupements d'objets (cet algorithme permet alors de trier des objets).

Le pas qui sépare le tri d'objets de la classification a ensuite été franchi dans [LUM 94]. Ils ont proposé un algorithme en utilisant une mesure de dissimilarité (distance euclidienne) entre les objets (enregistrements). Ces objets sont répartis aléatoirement sur une grille 2D, les fourmis déposées sur la grille ne perçoivent que les objets se trouvant dans leur voisinage. Un objet est ramassé avec une probabilité d'autant plus grande qu'il est peu similaire aux objets voisins. L'objet est déposé dans une région comportant des objets qui lui sont similaire.

AntClass est une amélioration de l'approche précédente, présenté dans [MON 99], elle permet de construire des ensembles d'objets, chaque ensemble (Tas) est présent dans une seule case qui représentera à la fin de l'algorithme une classe et l'ensemble des tas représenteront la partition. Lors du déplacement, une fourmi ramasse, selon sa capacité (nombre d'objets pouvant être transportés par une fourmi) les objets les plus dissimilaires, elle les dépose dans des tas comportant les objets qui leurs sont similaires.

Cet algorithme inclut une hybridation avec des centres mobiles (K-Means) pour accélérer la convergence et améliorer la partition obtenue par AntClass.

Enfin, dans [AZZ 03], un nouveau modèle permettant d'effectuer rapidement une classification hiérarchique a été introduit. Il s'agit de copier la manière dont les fourmis construisent des structures vivantes en s'accrochant les unes aux autres en fonction de critères locaux (la forme de la structure influençant le comportement d'accrochage ou de décrochage). Dans ce modèle, chaque fourmi artificielle représente une donnée. Les fourmis sont placées initialement à la racine de l'arbre et vont pouvoir se déplacer dans cet arbre et s'accrocher afin de construire une structure hiérarchique dont chaque nœud représente une donnée. L'objectif est d'obtenir la propriété suivante: chaque nœud o de l'arbre est une catégorie composée de toutes les données portées par les sous-arbres de o. Les sous catégories (représentées par les nœuds connectés à o) doivent être très similaires à leur mère dans l'arbre, mais également les plus dissimilaires entre elles. Les résultats obtenus sont très compétitifs par rapport à la classification ascendante hiérarchique notamment. [AZZ 04]

5. Systèmes immunitaires artificiels

Dans les années 90, les systèmes immunitaires artificiels (AIS) sont apparus. Ils sont inspirés du fonctionnement du système immunitaire humain et animal en utilisant les principes suivants: les agents (lymphocytes) qui produisent des anticorps sont les seuls qui différencient le soi du non soi (antigène). Pour cela, les agents subissent des tests de sélection: si une liaison est possible entre les anticorps et les molécules du soi, alors les agents générant ces anticorps sont éliminés, ne gardant ainsi que les lymphocytes rejetant le non soi. Cette phase est appelée sélection négative. Il y a nécessité de produire un grand nombre d'anticorps du fait que la quantité d'antigènes est importante. Pour ce faire, les lymphocytes activés se multiplient par un processus de sélection par clonage.

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.

En ce qui concerne la classification, les principes des systèmes immunitaires sont, à un niveau général, les suivants (voir par exemple le système aiNet [CAS 00]): les

données $d_1,...,d_n$ 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 active 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. Ces anticorps vont également 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. [AZZ 04]

6. Conclusion

Nous venons juste d'évoquer les principales applications de méthaheuristique pour le partitionnnement. Pour les méthodes issues des méthaheuristiques à solution unique, le K-means est la méthode la plus utilisée par les experts.

Dû sûrement à leurs succès autant que méthode d'optimisation, les AG pour le partitionnement ont fait objet de plusieurs travaux par rapport à tout les autres méthodes. Dans les AG, un individu représente une partition, la population un ensemble de partitions et la solution le meilleur individu de la population, ce qui fait que les AG ont un fonctionnement centralisé par rapport à d'autres méthodes comme les colonies de fourmis qui sont des systèmes auto-organisés et non centralisés(la solution est l'ensemble des individus et utilise des comportements locaux agissant en parallèle sur la partition). Les recherches ne se base pas seulement sur des méthodes permettant des regroupements de données mais également une visualisation de ces regroupements, cela permet à l'expert du domaine d'interpréter directement et interactivement les résultats et de formuler graphiquement des requêtes sur ces données.

Chapitre III

Conception & Mise en œuvre

1. Introduction

Dans ce chapitre, la méthode que nous avons mise en œuvre pour la résolution du problème de classification est introduite. Cette méthode est basée sur un modèle biologique: les nuages d'insectes volants. Elle a été intégrée dans un environnement complet qui est décrit dans ce chapitre.

2. Description du problème

Le problème de la classification non supervisée peut être formulé de la manière suivante: étant donné un ensemble fini de N objets, on cherche à partitionner cet ensemble en K classes (la valeur de K n'est pas connue à priori) de telle sorte que chaque élément (objet) appartienne à une et une seule classe et que les objets d'un même groupe soient plus similaire entre eux qu'avec les objets des autres groupes.

3. Description de la solution

De nombreux algorithmes de classification utilisent des principes inspirés de la biologie; par exemple: les algorithmes génétiques et les algorithmes à base de population de fourmis artificielles qui furent utilisés dans plusieurs travaux pour la résolution du problème du partitionnement des données (Chapitre II).

Il existe encore d'autres algorithmes inspirés de systèmes biologiques, mais qui n'ont pas encore été appliqués et testés sur des problèmes de classification. On peut citer par exemple les automates cellulaires ou encore les nuages d'animaux volants ou nageants.

La méthode réalisée par [MON 02] se concentre sur ce dernier modèle qui s'inspire du comportement observé chez certains animaux ayant un comportement social pour le déplacement. En effet, dans ces systèmes biologiques, des nuages d'oiseaux peuvent se former et évoluer de manière très spectaculaire et utile pour les individus ayant ce comportement: l'effet de masse peut repousser des prédateurs éventuels ou alors certaines configurations précises permettent d'économiser de l'énergie (exemple du vol des canards). Des comportements similaires se retrouvent dans la nage collective de certaines espèces de poissons pour la chasse en groupe par exemple (banc de thons) où

le banc prend la forme d'un entonnoir, ou a des mouvements d'explosion pour échapper à un ennemi; ou encore dans le vol en groupe chez des insectes (criquets). En informatique, des modélisations directement issues de ces comportements sont caractérisées par une "***intelligence en essaim***" ("***swarm intelligence***") [BON99], qui fait qu'un groupe d'entités plutôt simples et obéissant à des règles locales de coordination pour leurs déplacements, sont capables d'engendrer des comportements et des mouvements globaux beaucoup plus complexes [BRO 87, REY 87].

Chaque individu réagit à des règles, souvent identiques pour toute la population, qui ne prennent en compte que le voisinage immédiat de l'individu qui ne peut pas percevoir l'ensemble du nuage. A partir du déplacement local de l'individu, des formes complexes pour le nuage d'animaux vont pouvoir apparaître, formes qui dépassent le cadre de la règle locale. Cette propriété d'émergence est recherchée dans de nombreux problèmes d'informatique puisqu'elle va permettre une parallélisation de l'algorithme avec des comportements élémentaires simples tout en obtenant globalement un résultat complexe.

Dans le domaine de la classification, on peut assimiler un individu du nuage d'animaux volants à une donnée. Les individus sont placés sur un plan et sont caractérisés par trois éléments: leurs coordonnées dans l'environnement, leur vecteur vitesse et des règles comportementales pour gérer les déplacements.

Ces règles sont communes à tous les individus et utilisent le voisinage local d'un insecte pour décider de changer le vecteur vitesse. Elles prennent en compte la similarité des données portées par les insectes afin de former des nuages de données homogènes. Autrement dit, la proximité des insectes, à la fois en termes de position et de vitesse dans l'espace des déplacements 2D, va correspondre à la similarité des données dans leur espace de description. Ce type d'algorithme permet à l'expert du domaine d'obtenir une visualisation dynamique dans laquelle les distances entre données ont un sens du point de vue de la représentation des données.

La méthode que nous avons mise en œuvre peut donc aussi bien être considérée du point de vue de la classification que de la visualisation de données ou fouille visuelle de données (***visual data-mining***).

4. Description de l'environnement mis en oeuvre

Un environnement a été réalisé pour mettre en œuvre cette technique. Il est composé de six modules qui interagissent. La figure III.1 illustre le schéma général de l'environnement, et montre les différents modules qui ont été conçus pour pouvoir implémenter la méthode de fouille visuelle et de classification des données par insectes volants.

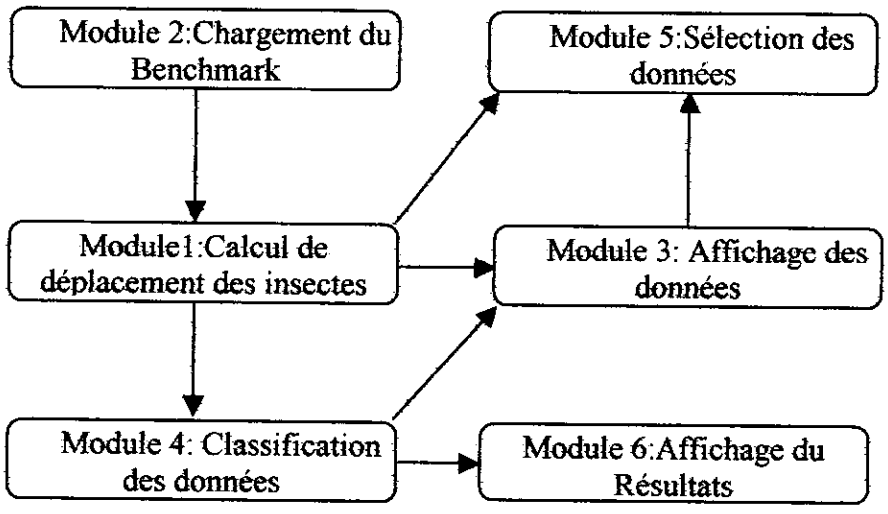


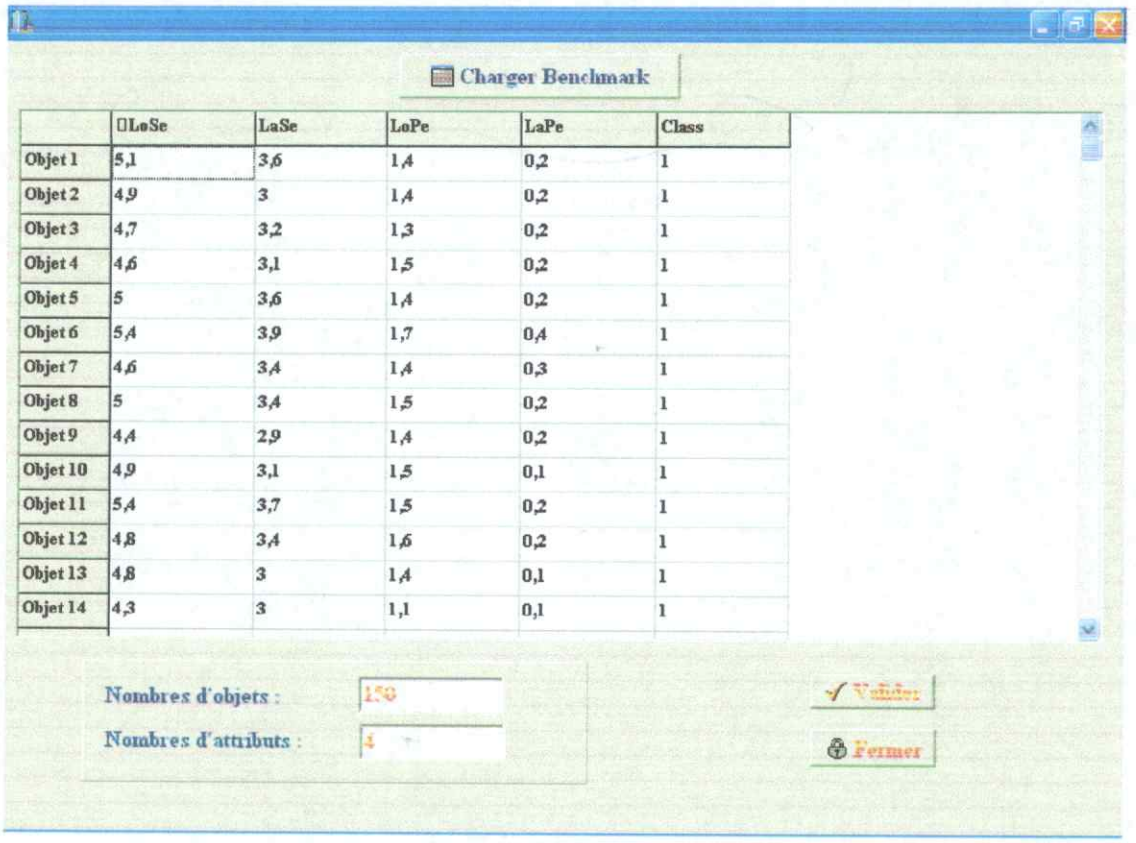
Figure III.1: Schéma général de l'environnement

Dans ce qui suit, nous revenons sur une description plus détaillée des modules composant l'outil.

4.1. Module de chargement des Benchmarks

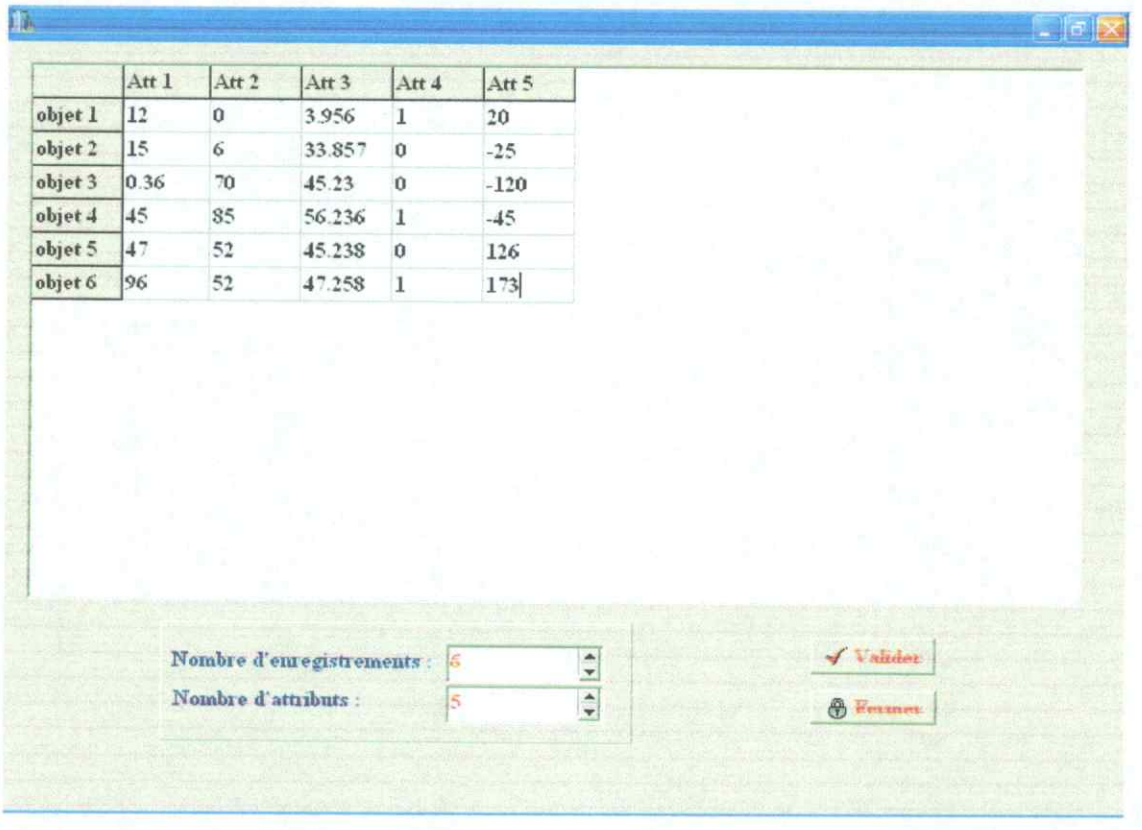
Afin de pouvoir tester la qualité des résultats obtenus par cette méthode, nous avons utilisé des exemples constitués à partir de bases de données supervisées (chaque objet est déjà affecté à une classe). Cela a permis de comparer les résultats obtenus à la classification réelle des exemples.

Ce module a donc pour rôle de permettre à l'utilisateur de choisir un benchmark particulier et de le charger (**Figure III.2**). L'utilisateur a également la possibilité de créer son propre jeu d'essai, à l'aide d'un éditeur pour la saisie de données qui est mis à sa disposition(**Figure III.3**).



	LoSe	LaSe	LoPe	LaPe	Class
Objet 1	5,1	3,6	1,4	0,2	1
Objet 2	4,9	3	1,4	0,2	1
Objet 3	4,7	3,2	1,3	0,2	1
Objet 4	4,6	3,1	1,5	0,2	1
Objet 5	5	3,6	1,4	0,2	1
Objet 6	5,4	3,9	1,7	0,4	1
Objet 7	4,6	3,4	1,4	0,3	1
Objet 8	5	3,4	1,5	0,2	1
Objet 9	4,4	2,9	1,4	0,2	1
Objet 10	4,9	3,1	1,5	0,1	1
Objet 11	5,4	3,7	1,5	0,2	1
Objet 12	4,8	3,4	1,6	0,2	1
Objet 13	4,8	3	1,4	0,1	1
Objet 14	4,3	3	1,1	0,1	1

Figure III.2:Chargement de Benchmarks



	Att 1	Att 2	Att 3	Att 4	Att 5
objet 1	12	0	3.956	1	20
objet 2	15	6	33.857	0	-25
objet 3	0.36	70	45.23	0	-120
objet 4	45	85	56.236	1	-45
objet 5	47	52	45.238	0	126
objet 6	96	52	47.258	1	173

Figure III.3 :Editeur des données

4.2. Module de calcul du déplacement des insectes

Ce module représente le noyau du logiciel. Les différents déplacements des insectes sur le plan sont calculés, afin d'obtenir des regroupement de ces insectes.

La première notion présentée est la mesure de similarité qui permet de décider dans quelle mesure deux objets sont semblables entre eux.

4.2.1. Mesure de similarité

Soit un ensemble de n données $(e_1, \ldots, e_n)$ que l'on souhaite regrouper en classes. La fonction qui calcule la similarité *Sim* (i, j) entre deux données e_i et e_j est donnée par la formule suivante:

$$Sim(i,j) = 1 - \frac{1}{A}\sum_{k=1}^{A}\left|e_i^k - e_j^k\right|$$

où A est le nombre d'attributs décrivant les données. e_i et e_j sont deux objets;

e_i^k représente la valeur du *k-ième* paramètre de l'objet e_i.

Une procédure nommée ***calcul_similarité*** a été réalisée. Elle calcule la similarité entre chaque couple d'insectes. Les résultats sont sauvegardés dans une table dont les valeurs sont comprises entre [0,1]. Une valeur de '1' correspond à deux données totalement identiques, alors que '0' correspond à deux données totalement dissemblables. Cependant, avant de calculer cette similarité, il faut être attentif à la normalisation des échelles pour les différents attributs de la table des données. Si un attribut prend ses valeurs dans l'intervalle [0,1] et un autre dans l'intervalle [0,100000], le second est privilégié et cela perturbe la mesure de similarité. Pour cela, il faut que tous les attributs de type numérique soient normalisés (dans l'intervalle [0,1]). La formule utilisée dans la procédure de normalisation est:

$$x_i = \frac{x_i - \min x}{\max x - \min x}$$

où max x et min x représentent respectivement la valeur maximale et minimale d'un attribut x. Le paramètre x_i indique la valeur d'un champ de cet attribut.

4.2.2. Coordonnées et vitesse d'un insecte

Un insecte est visuellement caractérisé par ses coordonnées et sa vitesse. Soit une population de n insectes où l'insecte i représente la donnée e_i. Les insectes se déplacent dans un espace de dimension 2. Un insecte i est caractérisé par ses coordonnées réelles

(x_i, y_i) appartenant à ([0,1] × [0,1]) ainsi que par un vecteur vitesse $\mathbf{v}_i = A_i \hat{\mathbf{v}}_i$ d'amplitude A_i $(A_i \geq 0)$ et de direction normalisée $\hat{\mathbf{v}}_i$ $\left(\|\hat{\mathbf{v}}_i\| = 1\right)$. De ce fait, une classe insecte a été définie, contenant les champs suivants:

- Emplacement de l'insecte sur le plan.
- Vecteur direction.
- Amplitude de la vitesse.

4.2.3. Algorithme principal

L'algorithme suivant explique le principe général de la méthode:

(1) Lire les ***n*** données d'entrée $\mathbf{e_1}$,....., $\mathbf{e_n}$.
(2) Placer initialement les ***n*** insectes de façon aléatoire dans l'environnement 2D.
(3) Calculer la distance idéale entre chaque couple d'insectes.
(4) **Tantque** itération < nbre_itération **faire**
(5) Calculer un déplacement pour chacun des n insectes.
(6) Déplacer tous les insectes.
(7) **FinTantque.**

Figure III.4: Algorithme décrivant le principe général [MON 02]

L'algorithme principal (illustré en **Figure III.4**) fonctionne de la façon suivante: les insectes sont placés initialement à une position choisie aléatoirement et avec une direction initiale $\hat{\mathbf{v}}_i$ aléatoire.

Les distances idéales sont calculées pour chaque couple d'insectes (voir section 4.2.5). Ensuite, au gré de leur déplacement, les insectes vont se croiser et vont décider suivant une règle de comportement local de se rapprocher ou de s'éloigner les uns des autres, d'aller ou non dans les mêmes directions (les calculs concernant cette règle sont détaillés dans la section 4.2.5).

La règle locale tend intuitivement vers deux buts:

- Etablir une distance *2D* entre insectes qui soit représentative de la similarité des données qu'ils représentent.
- Déplacer dans la même direction des insectes représentant des données similaires.

À partir de cette règle locale vont émerger des groupes d'insectes se déplaçant ensemble et permettant de définir des regroupements dans les données.

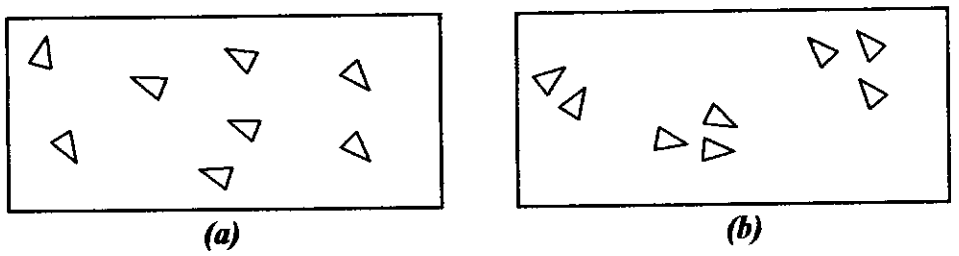


Figure III.5: Illustration du mouvement du nuage d'insectes: [MON 02]
(a) les insectes sont placés aléatoirement et orientés dans des directions désordonnées.
(b) les insectes sont en groupes se déplaçant de manière cohérente.

4.2.4. Améliorations apportées à la méthode d'initialisation

Lors de cette initialisation, il peut exister une manière plus judicieuse de définir les positions de départ des insectes. L'utilisateur peut, par exemple, utiliser deux attributs numérique de la base de donnée: les positions sont alors déterminées par les valeurs des deux attributs (ou d'un seul s'il n'y en a qu'un).

Les directions initiales des vecteurs vitesse sont fixées dans le sens centrifuge par rapport au milieu de l'espace *2D*. Ainsi, en supposant que les deux attributs sont bien choisis, les insectes vont bénéficier d'une configuration initiale très avantageuse par rapport à l'aléatoire. Non seulement les positions relatives ont statistiquement plus de chance de se rapprocher de la réalité, mais en plus les vecteurs vitesse auront des orientations pratiquement parallèles pour deux insectes voisins (la notion d'insectes voisins est expliquer dans la section 4.2.5).

Nous avons donc offert à l'utilisateur la possibilité de sélectionner parmi les options suivantes:

- des directions initiales aléatoires ou orientées vers le centre,
- ainsi que des positions initiales aléatoires ou déterminées par deux attributs.

Une autre possibilité également offerte à l'utilisateur, utiliser des positions et des directions déjà enregistrées lors d'exécutions précédentes. En effet, cela permet de partir d'un résultat intermédiaire jugé intéressant, et cela peut être avantageux surtout si les positions ont donné de bons résultats. Cela permet également de tester l'effet des différents paramètres de manière plus objective que lors du choix de positions initiales générées aléatoirement. Nous avons donc la possibilité d'arrêter l'exécution et de la reprendre plus tard avec seulement la sauvegarde des dernières positions et directions des insectes.

4.2.5. Règle de comportement local

Le comportement d'un insecte est déterminé par une règle individuelle, identique pour tous les insectes et décrite par l'algorithme de la Figure III.6.

(1) Déterminer le voisinage de i:

(2) $\mathcal{V}(i) = \{ j / j \in [1,n], j \neq i, d(i,j) \leq d_{seuil} \}$

(3) Calculer une nouvelle direction $\hat{v}_i(t+1)$:

(4) **Si** $\mathcal{V}(i) = \phi$ **alors**

(5) $\hat{v}_i(t+1) \leftarrow \hat{v}_i(t)$

(6) **Sinon**

(7) **Pour** chaque insecte $j \in \mathcal{V}(i)$ **faire**

(8) **Si** $d(i,j) > d^*(i,j)$ **alors**

(9) $\beta(i,j) \leftarrow 4 \times \left(\frac{d(i,j) - d^*(i,j)}{d_{seuil} - d^*(i,j)} \right)^2$ /* *attraction* */

(10) **Finsi**

(11) **Si** $d(i,j) = d^*(i,j)$ **alors**

(12) $\beta(i,j) \leftarrow 0$

(13) **Finsi**

(14) **Si** $d(i,j) < d^*(i,j)$ **alors**

(15) $\beta(i,j) \leftarrow -4 \times \left(1 - \frac{d(i,j)}{d^*(i,j)} \right)^2$ /* *répulsion* */

(16) **Finsi**

(17) l'influence de j sur i est alors:

(18) $v_{résultant}(i,j) \leftarrow \underbrace{\hat{v}_j}_{alignement} + \underbrace{\beta(i,j) \times \hat{v}_{ij}}_{attractionourépultion}$

(19) **Finpour**

(20) Calculer la somme des influence:

(21) $v_{résultant}(i,.) \leftarrow \sum_{j \in \mathcal{V}(i)} v_{résultant}(i,j)$

(22) Normaliser: $\hat{v}_{résultant}(i,.) \leftarrow \frac{v_{résultant}(i,.)}{\| v_{résultant}(i,.) \|}$

(23) Nouvelle direction: $\hat{v}_i(t+1) \leftarrow \hat{v}_{résultant}(i,.)$ en limitant l'angle entre $\hat{v}_i(t+1)$ et $\hat{v}_i(t)$ à 90°

(24) **Finsi**

(25) Calculer une nouvelle amplitude:

$$A_i(t+1) = \frac{1}{5} \times d_{seuil} + \frac{1}{20} \frac{d_{seuil}}{|\mathcal{V}(i)| + 1}$$

(26) Calculer le déplacement de l'insecte i (les insectes sont placés tous en même temps):

$$(x_i(t+1), y_i(t+1)) = (x_i(t), y_i(t)) + A_i(t+1) \hat{v}_i(t+1)$$

Figure III.6: Calcul du déplacement d'un insecte *i* [MON 02]

4.2.5.1. Détermination du voisinage

Pour chaque insecte *i*, il faut déterminer un voisinage $v(i)$ correspondant à l'ensemble des autres insectes j se trouvant à une distance inférieure ou égale à d_{seuil}. Cette notion est prépondérante puisqu'elle définit le seuil en dessous duquel un insecte peut en percevoir un autre et être influencé par celui-ci et réciproquement.

Pour calculer ce voisinage, une fonction nommée ***Déterminer_voisinage*** a été développée. Elle admet en entrée un insecte *i* et retourne l'ensemble des voisins de *i*. La distance euclidienne est utilisée pour le calcul du voisinage, elle est notée $d(i,j)$.

Le coût le plus élevé en termes de temps de calcul correspond au calcul du voisinage de chaque insecte. Pour simuler les *n* déplacements, la construction de ces voisinages est proportionnelle à n^2 du fait qu'il faille, pour chaque insecte, tester si tous les autres sont dans son voisinage ou non. Pour ramener la complexité à o(*n*), il a été envisagé d'utiliser la technique suivante [MON02] :Les insectes sont affectés aux cases de dimension $\frac{1}{d_{seuil}}$ d'une matrice carrée .

Le calcul du voisinage d'un insecte *i* consiste alors simplement à tester les 9 cases se trouvant autour de la case contenant *i*, Pour cela nous avons implémenter une fonction appelée ***Déterminer_voisinage_grille*** .

Les auteurs de la méthode limitent le nombre d'insectes contenus dans chaque case à 20. Le calcul est donc indépendant du nombre d'insectes. Si peu d'insectes sont présents dans le voisinage de *i*, cela ne change rien à l'exécution de l'algorithme. Par contre, si plus de 20 insectes se trouvent dans la grille, seulement 20 d'entre eux choisis aléatoirement seront considérés. D'après les auteurs, cette approximation ne change pas globalement le comportement car lorsque beaucoup d'insectes sont présents dans une zone réduite, un échantillonnage de cette population suffit statistiquement pour effectuer le calcul des interactions.

Nous offrons ainsi à l'utilisateur la possibilité d'utiliser les deux manières afin de calculer le voisinage.

4.2.5.2. Calcul de la nouvelle direction

Selon la taille du voisinage $v(i)$, deux cas peuvent se produire:

- Si $\nu(i)$ est un ensemble vide $(\nu(i)=\phi)$, alors l'insecte i ne changera pas de direction dans l'étape $t+1$ (ligne 5).
- Si des insectes sont présents dans le voisinage de l'insecte i, alors chacun de ces insectes influencera le prochain déplacement de i (de la ligne 7 à la ligne 19). L'influence de j sur i dépend de la distance $d(i,j)$ par rapport à une distance idéale $d^*(i,j)$.

L'idée de distance idéale développée ici est la suivante: La distance $d(i,j)$ doit converger vers $d^*(i,j)$ qui tient compte de la similarité entre e_i et e_j. Si cette similarité est importante et que les insectes sont trop éloignés par rapport à la distance idéale, alors il faut les rapprocher. Inversement, si cette similarité est faible et que les insectes sont trop proches, alors il faut les éloigner. Cette distance idéale dépend elle-même de la similarité $Sim(i,j)$ entre les données e_i et e_j de la manière suivante:

$$d^*(i,j)=\frac{1-Sim(i,j)}{1-Sim_{seuil}}\times d_{seuil}$$

Autrement dit, si $Sim(i,j)$ est exactement égale au seuil Sim_{seuil}, on va tenter de placer idéalement les deux insectes exactement à la limite de leur voisinage $(d^*(i,j)=d_{seuil})$. Si $Sim(i,j)<Sim_{seuil}$ alors on va tenter de séparer ces deux voisins $(d^*(i,j)>d_{seuil})$. Au contraire, si $Sim(i,j)>Sim_{seuil}$, on va alors laisser les deux insectes dans leur voisinage et à une distance donnée $(d^*(i,j)<d_{seuil})$.

Sim_{seuil} a été calculée par la formule $Sim_{seuil}=\frac{1}{2}(Sim_{moy}+Sim_{\max})$ où Sim_{moy} et $Sim_{\max}$ correspondent respectivement à la similarité moyenne et maximale entre les données.

Pour calculer l'influence de j sur la direction de i, il faut donc comparer la distance actuelle $d(i,j)$ entre les deux insectes à la distance idéale $d^*(i,j)$ et prendre en compte le vecteur direction de l'insecte j. Cette influence prend la forme d'un vecteur $v_{résultant}(i,j)$ qui, une fois normalisé, modifie la direction $\hat{v}_i$ de l'insecte i. Ce vecteur a deux composantes:

- l'une représentant un alignement entre les deux vecteurs vitesses et qui pousse i à aligner son déplacement sur j,
- l'autre représentant une attraction ou une répulsion entre les deux insectes.

On distingue donc les trois cas suivants:

- Si la distance entre les deux insectes est plus grande que la distance idéale $\left(d(i,j) > d^*(i,j)\right)$, alors une attraction a lieu entre i et j (voir ligne 9).
- Si $\left(d(i,j) = d^*(i,j)\right)$, alors le vecteur vitesse de i se rapproche simplement de celui de j en restant à la distance idéale (voir ligne 12).
- Enfin, Si $\left(d(i,j) < d^*(i,j)\right)$, alors une répulsion a lieu entre i et j puisqu'ils sont trop proches l'un de l'autre (voir ligne 15).

L'influence de i sur j est alors calculée avec la formule générale (voir ligne 18):

$$v_{résultant}(i,j) \leftarrow \underbrace{\hat{v}_j}_{alignement} + \underbrace{\beta(i,j) \times \hat{v}_{ij}}_{attractionourépultion}$$

où $\hat{v}_{ij}$ est un vecteur unitaire pointant de i vers j, et où β prend une valeur positive, nulle ou négative suivant les trois cas énumérés précédemment.

Dans le cas de plusieurs insectes voisins de i, on effectuera simplement la somme des vecteurs résultants $v_{résultant}(i,j)$ et on obtiendra $v_{résultant}(i,.)$ (ligne 21), qui sera normalisé pour obtenir $\hat{v}_{résultant}(i,.)$ (ligne22) qui représente le nouveau vecteur direction de i, mais il faut cependant limiter les changements de direction qu'ils peuvent réaliser (ligne23).

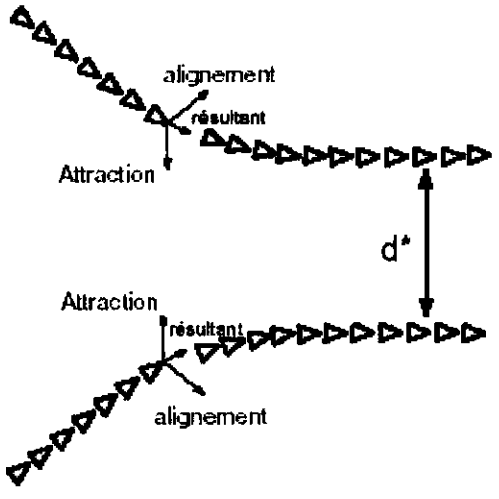


Figure III.7: Comportement théorique de l'algorithme dans le cas de deux insectes en interaction [MON 02]

L'effet de cette règle locale de comportement peut être constaté sur la figure III.5 qui illustre un résultat prévisible: lorsque deux insectes similaires vont se rencontrer (et en l'absence d'autres interactions), Les insectes ont tendance à maintenir entre eux une distance égale à la distance idéale et à se déplacer dans la même direction.

4.2.5.3. Changement d'amplitude et déplacement final

Pour déterminer l'amplitude du vecteur vitesse d'un insecte *i*, on applique la formule de la ligne 25 de l'algorithme. La première composante de l'amplitude fixe la vitesse minimale des insectes à $1/5^{ème}$ de la distance définissant le voisinage. Cela garantit d'abord qu'aucun insecte ne reste immobile dans l'environnement sans faire de rencontre avec d'autres insectes. De plus, un insecte ne se déplace pas suffisamment vite pour venir par exemple pour se placer directement au milieu d'un groupe. Cela pourrait en effet faire éclater un groupe homogène si l'intrus placé directement au centre est vraiment dissimilaire. Au lieu de cela, un insecte sera progressivement rejeté ou attiré aux abords d'un groupe. La deuxième composante fait que des insectes se déplaçant en un groupe homogène auront une vitesse légèrement plus faible que des insectes seuls qui pourront ainsi naviguer plus vite et augmenter leurs chances de trouver un groupe qui les accepte.

Pour déplacer tous les insectes en même temps, on considère qu'ils effectuent des mouvements toutes les unités temps (voir ligne 26).

Pour la modélisation du comportement des insectes dans leurs déplacements, nous avons définit une classe nommée ***insectes_volants***.

Figure III.8: Classes représentatives de la méthode

La classe insectes_volants regroupe les variables et les méthodes suivantes:

- *n*:le nombre d'insectes.
- insectes: un vecteur de type insecte et de taille *n*, il contient tout les insectes à représenter sur le plan, les indices de ce vecteur seront utilisées comme numéros représentant les insectes.

- d_{seuil} : un paramètre qui définit le seuil en dessous duquel un insecte peut en apercevoir un autre et être influencer par celui-ci et réciproquement
- nbre_itération: c'est le nombre de fois où les insectes effectueront des déplacements.
- Table des données: table contenant les données à regrouper en classes
- Table des similarités: table regroupant les mesures de similarité entre les données.
- Table des distances idéales: table calculée en utilisant le d_{seuil} et la table des similarités.

Elle contient principalement des méthodes de:

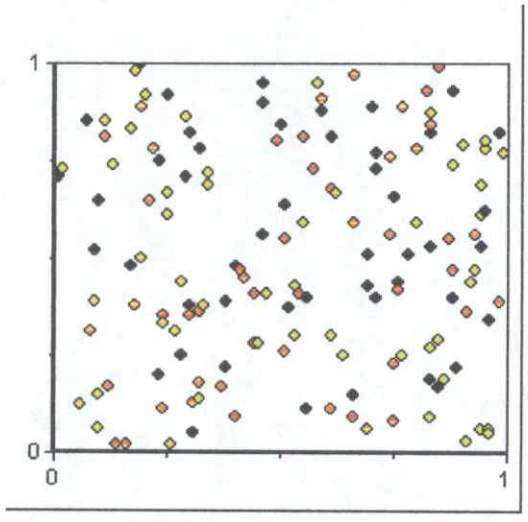
- Normalisation de la table des données.
- Calcul des similarités.
- Calcul des distances idéales.
- Calcul des positions et des directions initiales.
- Calcul du voisinage.
- Calcul de la nouvelle direction: c'est la méthode qui génère les emplacements et les directions des insectes à chaque itération.

4.3. Module de visualisation des insectes

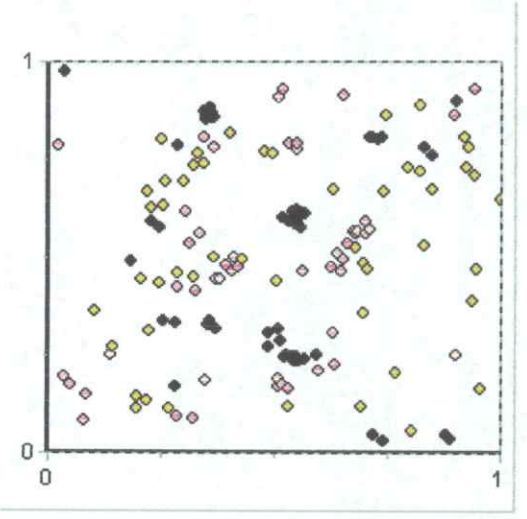
La visualisation dynamique des données en déplacement et en interaction les unes avec les autres offre à l'utilisateur la possibilité de percevoir instantanément un grand nombre d'informations. En effet, la répartition bidimensionnelle des données offre des informations sur les groupes de données et leur taille respective, sur le centre des groupes qui vont contenir des données très similaires entre elles (distance idéale faible) et sur la périphérie des groupes où les données seront plus distantes et donc un peu moins similaires. Nous avons donc mis en œuvre une procédure nommée ***Affichage_nuage_insectes*** qui se concentre sur l'affichage simultané des insectes dans des emplacements générés dans le module 1 à la fin de chaque itération.

La Figure III.9 montre un exemple de l'application de la méthode sur la base d'Iris qui contient 150 données répartis en 3 classes, sur le plan, nous avons placé ces données dans des positions aléatoire (**Figure III.9.(a)**), les insectes représentés par la même couleur sont dans la même classe. Les insectes se déplace selon l'algorithme principale.

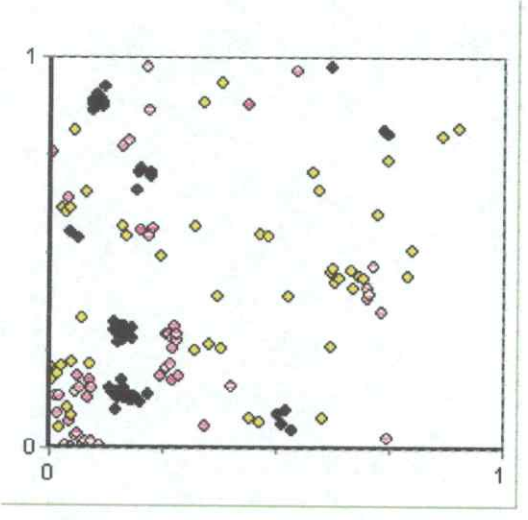
Nous avons pris des images du plan au bout de chaque 50 itérations pour montrer l'évolution des emplacements des insectes afin de formés des groupes.



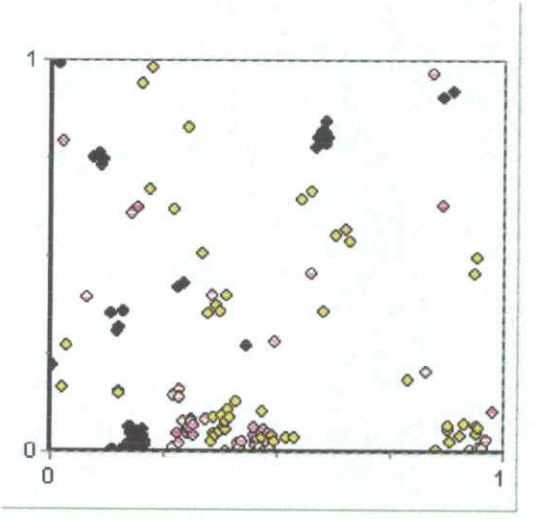
(a)



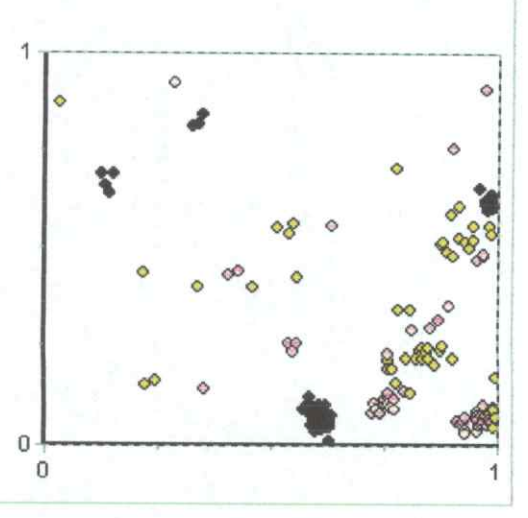
(b) 50 itérations



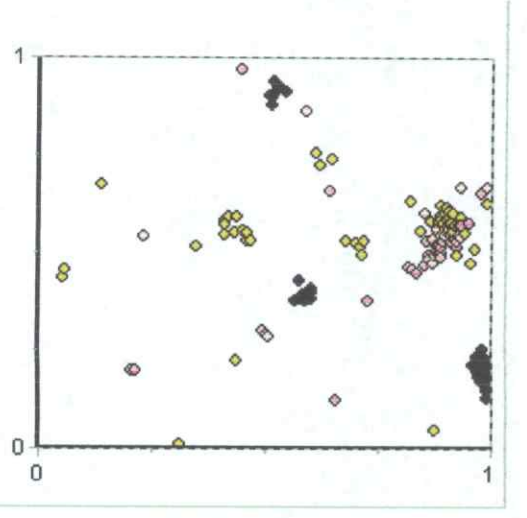
(c) 100 itérations



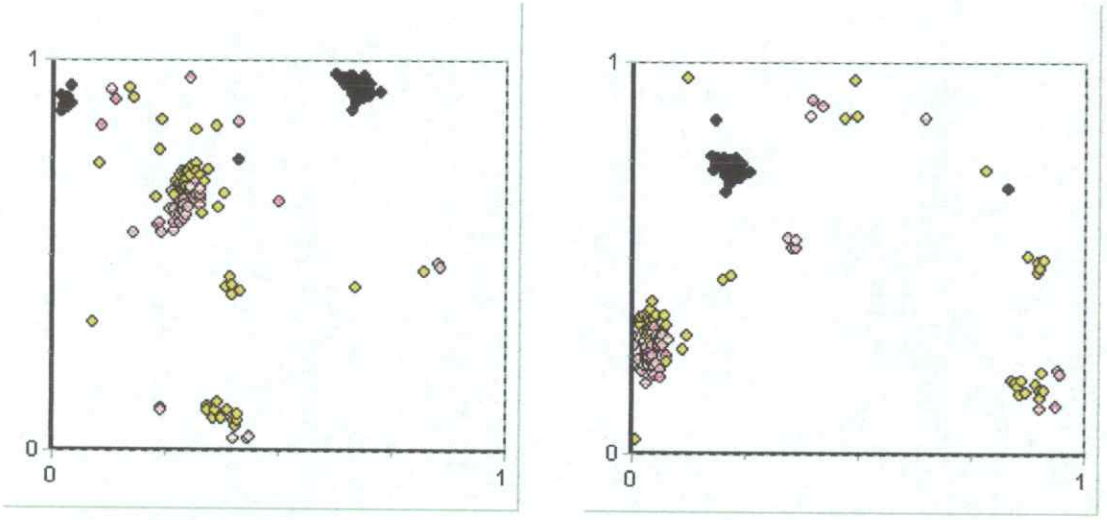
(d) 150 itérations



(e) 200 itérations



(f) 250 itérations



(g) 300 itérations *(h) 350 itérations*

Figure III.9 : Visualisation du développement des emplacements des insectes

4.4. Module de classification des données

Les déplacements des insectes (module1) donnent des regroupements entre les données. Cependant, la position de ces données ne spécifie pas l'affectation de ces données à une classe particulière. Il est possible qu'à la fin du traitement, certains individus soient encore proches d'un groupe d'individus tout en étant dissimilaires d'eux. Pour cela, il est nécessaire de recourir à une opération de classification dont le rôle est d'affecter les objets qui se ressemblent aux mêmes classes. Pour réaliser cette opération, nous avons mis en œuvre un algorithme proposé par [MON02] pour calculer une classification effective des données. Cet algorithme opère comme suit:

(1) Soit i un insecte n'ayant pas encore de classe. On crée une nouvelle classe contenant seulement i.
(2) Tous les insectes du voisinage $v(i)$, qui n'ont pas encore de classe définie et qui sont tels que $Sim(i,j) > Sim_{seuil}$, sont affectés à cette classe.
(3) Cette opération est lancée de manière récursive sur tous les insectes ainsi ajoutés à la classe.
(4) Lorsque plus aucun insecte ne peut être affecté à cette classe, on recommence à l'étape (1) avec un nouvel insecte sans classe. Si tous les insectes ont été classés, l'algorithme passe à l'étape (5).
(5) Le cas des insectes isolés est traité ici comme suit: toute classe contenant moins de $\frac{n}{20}$ données est détruite, et les données ainsi libérées sont placées selon la donnée classée qui leur est la plus similaire.

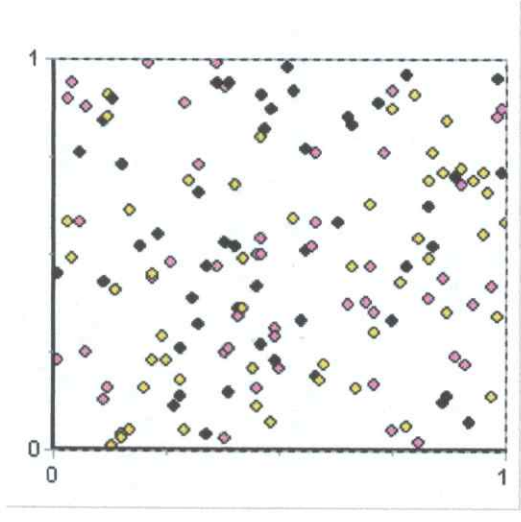
Figure III.10: Algorithme de Classification [MON 02]

Pour l'implémentation de l'algorithme de classification, nous avons construit une méthode ***Classification_insectes*** qui fait appelle à d'autres fonctions, ***Affecter_insectes*** , son rôle est d'affecter les insectes aux classes appropriées, ***Detruire_classes*** , son rôle est de détruire les classes ayant un nombre d'objets inférieur à $\frac{n}{20}$ et ***Reaffecter_insectes*** qui réaffecte à d'autres classes les insectes dont les classes étaient détruites.

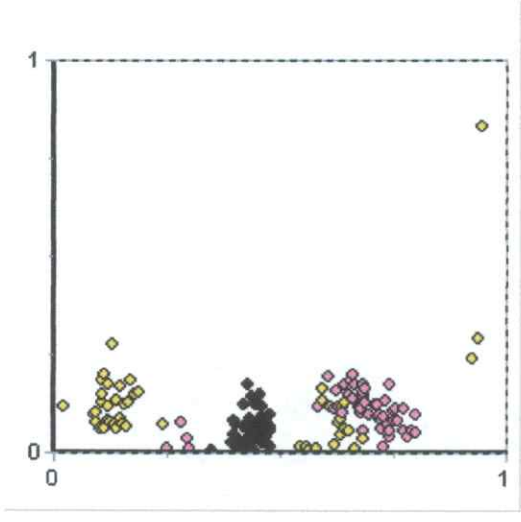
Nous allons montrer un exemple de classification des données, en utilisant la base d'Iris. La FigureIII.11.(a) représente les positions aléatoires des insectes, FigureIII.11.(b) représente les nuages de groupes formés à partir de l'algorithme principal, en utilisant un d_{seuil} de 0.05 avec 500 itérations.

FigureIII.11.(c) représente le résultat de l'application de l'algorithme de classification à partir des emplacements des insectes de la FigureIII.11.(b).

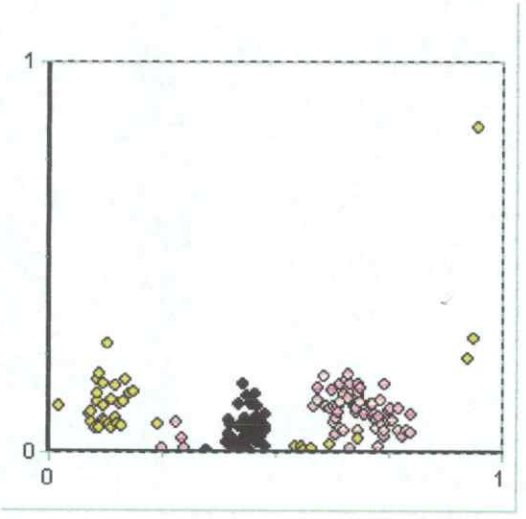
Dans la FigureIII.11.(a)et(b) les différentes couleurs représentent les classes d'origines de la base Iris, alors que dans la FigureIII.11.(c) les couleurs représentent les classes retrouvés par la méthode des insectes volants.



(a)



(b)



(c)

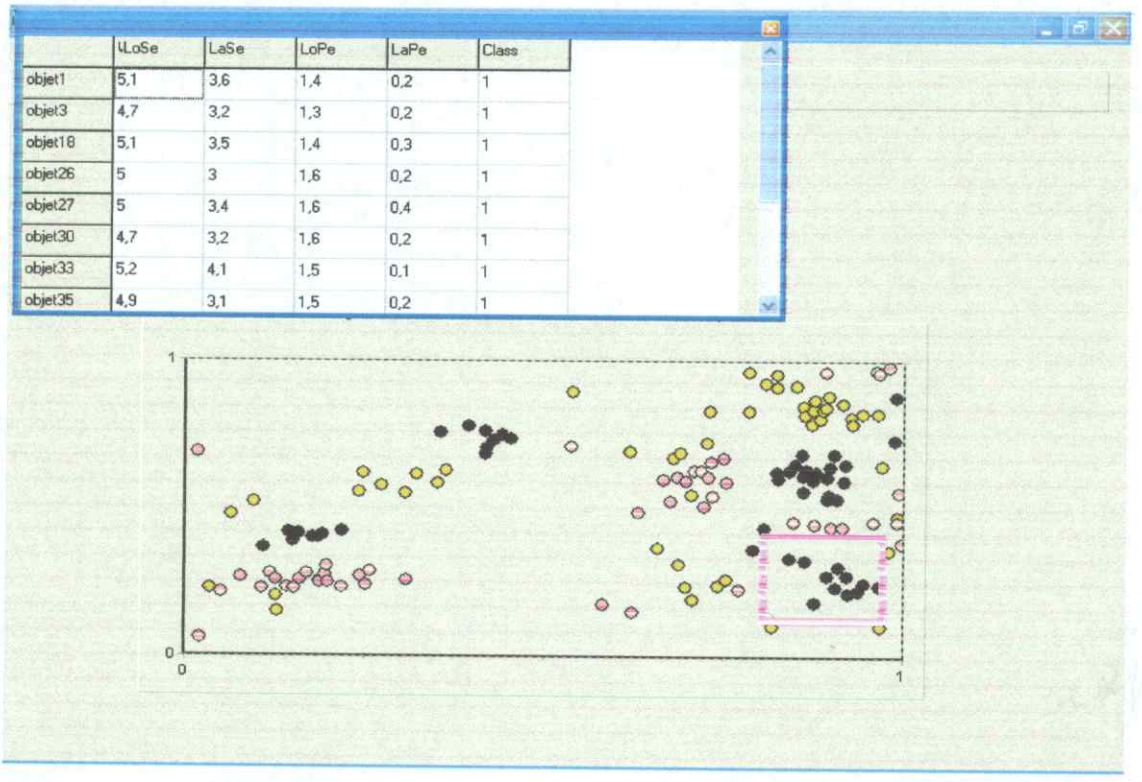
Figure III.11 : Visualisation de la classification des insectes

4.5. Module de sélection des données

L'approche à base de nuages d'insectes se caractérise par l'obtention d'une visualisation dynamique des données. Cette visualisation permet à l'utilisateur d'observer le mouvement des insectes et d'agir sur le déroulement du traitement. Il peut par exemple, geler l'affichage et sélectionner un ou plusieurs insectes afin d'obtenir l'affichage des données correspondantes. La possibilité est également offerte à l'utilisateur de sélectionner un groupe afin d'obtenir toutes les données qui le caractérisent. Cette fonctionnalité peut être utilisée pour obtenir des informations sur les insectes situés, par exemple, au centre d'un groupe, et qui sont donc très similaires aux membres du groupe. Elle permet de détecter des objets qui auraient été classés dans un groupe différent des objets qui l'entourent.

La sélection peut être effectuée à la fin de tous les calculs de déplacements dans le module 1. Il est également possible de geler momentanément l'affichage en suspendant les calculs, afin de visualiser des informations intermédiaires.

Pour cette sélection l'utilisateur a la possibilité de tracer un rectangle sur le plan, d'y cliquer dessus et tous les données correspondant aux insectes qui se trouvent dans le rectangle seront affichées dans une autre fenêtre avec la description de chacune des données.



	LoSe	LaSe	LoPe	LaPe	Class
objet1	5,1	3,6	1,4	0,2	1
objet3	4,7	3,2	1,3	0,2	1
objet18	5,1	3,5	1,4	0,3	1
objet26	5	3	1,6	0,2	1
objet27	5	3,4	1,6	0,4	1
objet30	4,7	3,2	1,6	0,2	1
objet33	5,2	4,1	1,5	0,1	1
objet35	4,9	3,1	1,5	0,2	1

Figure III.12:Sélection des données

4.6. Module d'affichage des résultats

Ce module a pour rôle d'afficher les résultats de l'opération de classification. Les informations que l'utilisateur peut ainsi consulter sont les suivantes:

- le nombre de classes trouvées.
- le temps d'exécution.
- le taux de fiabilité.
- une visualisation sur le plan: les insectes représentant les données d'une même classe sont visualisés sur le plan avec la même couleur, le nombre de couleurs correspond au nombre de classes obtenues.

la figure suivante montre la fenêtre qui regroupe tout les informations sur le résultat de la classification.

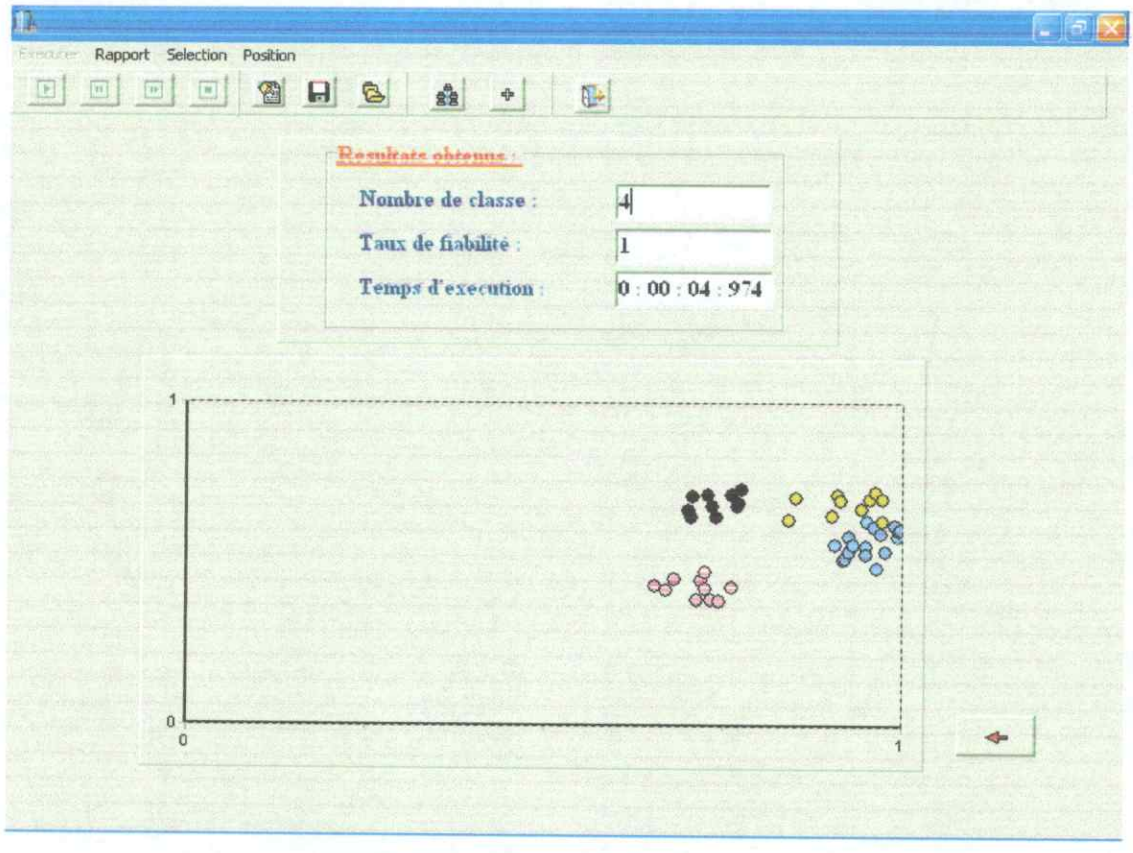


Figure III.13 :Affichage des résultats

L'utilisateur peut consulter un rapport détaillé qui contient des informations sur les paramètres d'entrée utilisés par la méthode ainsi que tous les résultats:

- nombre de classes,
- temps d'exécution,
- taux de fiabilité,
- la partition obtenue
- le centre de gravité de chaque classe…

5. Conclusion

Dans le travail réalisé, nous avons choisi d'implémenter la méthode des insectes volants du fait qu'elle ne résout pas seulement le problème de classification, mais elle permet aussi un affichage dynamique et interactif des individus. Elle permet donc à l'utilisateur d'obtenir des informations plus précises (nombre de groupes, données isolées ou bruitées …) textuelles et visuelles.

Chapitre IV

Tests et Résultats

1. Introduction

Dans ce chapitre, nous allons présenter quelques uns des résultats obtenus lors des tests effectués avec l'approche de classification par insectes volants sur différent Benchmarks. Tous les résultats présentés sont une moyenne de 10 essais.

2. Jeux de données

Dans le but d'évaluer cette méthode, nous avons choisi d'utiliser des bases de données réelles issues du *Machine Learning Repository* [BLA 98]. Ces bases de données sont supervisées (pour chaque objet on connaît la classe) afin de pouvoir comparer la classification obtenue par rapport à la classification réelle des données. Les caractéristiques principales des bases de données sont résumées dans le Tableau IV.1.

Nom du benchmark	Nombre d'exemples	Nombre d'attributs	Valeurs manquantes	Nombre de classes
Soybean	47	35	Non	4
Iris	150	4	Non	3
Segmentation	210	19	Non	7
Heart	270	13	Non	2

Tableau IV.1 :Caractéristiques des différentes bases de données

3. Méthodologie de test

Comme nous avons choisi des données dont la classe est connue à l'avance, il est possible de comparer la classification trouvée à la classification réelle des données. Cette comparaison a lieu en fonction du nombre de classes trouvées (qualité des solutions trouvées) mais aussi en fonction d'une erreur calculée pour évaluer la qualité de la classification. Cette erreur consiste à énumérer tous les couples d'exemples de la base et à compter combien de couples sont mal classés. Un couple est considéré comme mal classé si les deux exemples sont classés ensemble alors qu'ils ne l'étaient pas dans la classification réelle, ou inversement. Cette erreur est divisée par le nombre total de couples. L'erreur de classification E_c est calculée de la façon suivante :

Pour chaque objet o_i, on note sa classe d'origine $c(o_i)$ et la classe obtenue par $c'(o_i)$.

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

Où

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

4. Exécution de la méthode

Les quatre tableaux suivants résument les différents jeux de paramètres et les résultas obtenus. L'explication des différentes abréviations utilisées dans les tableaux est la suivante:

d_{seuil} : paramètre qui définit le seuil en dessous duquel un insecte peut en apercevoir un autre et être influencer par celui-ci et réciproquement.

Nb_It : Nombre d'itération.

NBC : Nombre de classes obtenues par la méthode.

Init_Pos : Initialisation des positions des insectes sur le plan.

Pos_Al : Pour une initialisation aléatoire.

Pos_Att : Pour une initialisation déterminée par les valeurs de deux attributs.

Init_Vit : Initialisation des directions des insectes.

Vit_Al : Pour une initialisation aléatoire.

Vit_Ort : Pour des directions orientées vers le centre.

Vois : voisinage d'un insecte est calculé soit par grille (**Vois_G**) soit par rapport à toute la population (**Vois_P**).

TF : Taux de fiabilité de la partition ($TF = 1 - E_c$; E_c erreur de classification citée précédemment).

TExe: Temps d'Exécution obtenu par la méthode.

Benchmark	Paramètres		Initialisation		Vois	Résultats		
	d_{seuil}	Nb_It	Init_Pos	Init_Vit		NBC	TF	TExe
Soybean	0.01	500	Pos_Al	Vit_Al	Vois_P	1	0.25	00.00.988
	0.05	500	Pos_Al	Vit_Al	Vois_P	7	0.93	00.01.097
	0.1	500	Pos_Al	Vit_Al	Vois_P	5	0.95	00.01.986
	0.1	50	Pos_Att	Vit_Ort	Vois_P	6	0.96	00.00.052
	0.1	50	Pos_Al	Vit_Al	Vois_P	6	0.91	00.00.042
	0.1	50	Pos_Al	Vit_Al	Vois_G	4	0.95	00.00.036

Tableau IV.2 : Paramètres et résultats sur le benchmark Soybean

Benchmark	Paramètres		Initialisation		Vois	Résultats		
	d_{seuil}	Nb_It	Init_Pos	Init_Vit		NBC	TF	TExe
Iris	0.01	1000	Pos_Al	Vit_Al	Vois_P	0	-	00.11.065
	0.05	1000	Pos_Al	Vit_Al	Vois_P	3	0.80	00.13.016
	0.1	1000	Pos_Al	Vit_Al	Vois_P	2	0.77	00.16.984
	0.05	50	Pos_Att	Vit_Ort	Vois_P	3	0.72	00.00.508
	0.05	50	Pos_Al	Vit_Al	Vois_P	1	0.32	00.00.478
	0.05	100	Pos_Al	Vit_Al	Vois_G	2	0.77	00.04.936
	0.05	30	Pos_Att	Vit_Ort	Vois_G	2	0.77	00.01.100

Tableau IV.3 : Paramètres et résultats sur le benchmark Iris

Benchmark	Paramètres		Initialisation		Vois	Résultats		
	d_{seuil}	Nb_It	Init_Pos	Init_Vit		NBC	TF	TExe
Segmentation	0.01	1000	Pos_Al	Vit_Al	Vois_P	0	-	00.13.562
	0.05	1000	Pos_Al	Vit_Al	Vois_P	5	0.80	00.17.992
	0.1	1000	Pos_Al	Vit_Al	Vois_P	3	0.69	00.21.951
	0.05	150	Pos_Att	Vit_Ort	Vois_P	3	0.57	00.01.870
	0.05	150	Pos_Al	Vit_Al	Vois_P	2	0.44	00.01.980
	0.05	50	Pos_Al	Vit_Al	Vois_G	2	0.38	00.06.962
	0.05	60	Pos_Att	Vit_Ort	Vit_G	2	0.38	00.07.230

Tableau IV.4 : Paramètres et résultats sur le benchmark Segmentation

Benchmark	Paramètres		Initialisation		Vois	Résultats		
	d_{seuil}	Nb_It	Init_Pos	Init_Vit		NBC	TF	TExe
Heart	0.01	1000	Pos_Al	Vit_Al	Vois_P	0	-	00.15.068
	0.05	1000	Pos_Al	Vit_Al	Vois_P	2	0.53	00.20.053
	0.1	1000	Pos_Al	Vit_Al	Vois_P	2	0.53	00.25.957
	0.05	500	Pos_Att	Vit_Ort	Vois_P	2	0.54	00.11.011
	0.05	100	Pos_Al	Vit_Al	Vois_G	1	0.50	00.15.025

Tableau IV.5 : Paramètres et résultats sur le benchmark Heart

5. Interprétation des résultats :

Selon les résultats mentionnés dans les tableaux ci-dessus, on peut constater les points suivants :

5.1 L'influence du paramètre d_{seuil}

Les trois premier tests de chaque tableau ont été effectués sur le paramètre d_{seuil}. Il est possible de remarquer que ce paramètre est prépondérant pour cette approche. Il a donc une grande influence sur la qualité du résultat.

Pour un d_{seuil} faible $\left(d_{seuil} = 0.01\right)$, on remarque que la convergence vers un partitionnement des données est difficile à atteindre parce que ce d_{seuil} entraîne des tailles de voisinages réduites et des déplacements d'insectes faibles, ce qui rend la probabilité d'une rencontre entre insectes plus faible (**Figure IV.1.(a)**). Au contraire, avec un d_{seuil} plus élevé $\left(d_{seuil} = 0.05\right)$, on obtient de meilleurs résultats car un tel d_{seuil} augmente les tailles des voisinages et l'ampleur des déplacements, ce qui donne plus de possibilité aux insectes de se rencontrer et de s'influencer les uns les autres.

On peut remarquer de la Figure IV.1.(b) que les groupes formés sont très compacts. Ceci est dû au fait que ce d_{seuil} implique des petites distances idéales entre les insectes d'un même groupe. A partir d'un d_{seuil} plus important $\left(d_{seuil} = 0.1\right)$, la taille de l'environnement ne permet plus aux insectes non similaires entre eux de s'éloigner suffisamment pour sortir de leurs voisinages respectifs. La convergence a donc tendance à se détériorer (**Tableau IV.3**). Avec un tel d_{seuil}, les distances idéales sont plus importantes et les insectes d'un même groupe sont plus dispersés (**Figure IV.1.(c)**).

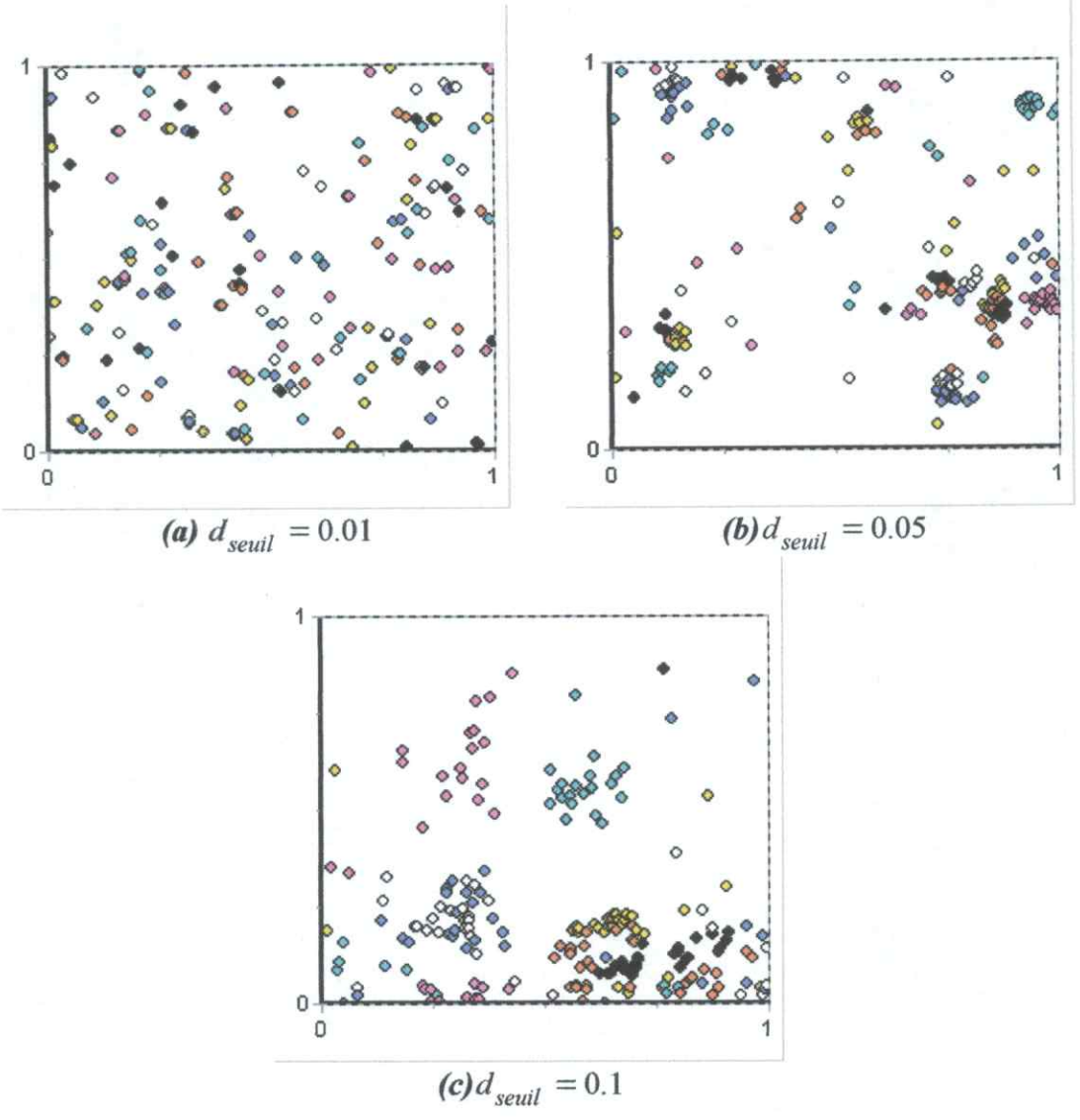


Figure IV.1: *Exemples de résultats obtenus au bout de 500 itérations avec différentes valeurs du paramètre d_{seuil} (lié notamment au calcul du voisinage) pour la base de données Segmentation.*

5.2 L'influence de l'initialisation orientée

On remarque que l'utilisation d'une initialisation orientés (**Pos_Att,Vit_Ort**) pour la base Iris(**Table IV.3** et **Figure IV.2**) fait converger l'algorithme au bout seulement de 50 itérations. On peut expliquer cela par le fait qu'Iris ne contient que quatre attributs continus. Choisir deux de ces attributs s'approche statistiquement de la représentation réelle de ces données sur le plan.

Lorsque le nombre d'attributs est important, les données ne peuvent pas être correctement représentées par seulement deux attributs. C'est pour cela que cette

initialisation n'apporte pas une grande amélioration par rapport à une initialisation aléatoire (**Tableau IV.2**, **Tableau IV.4** et **Tableau IV.5**).

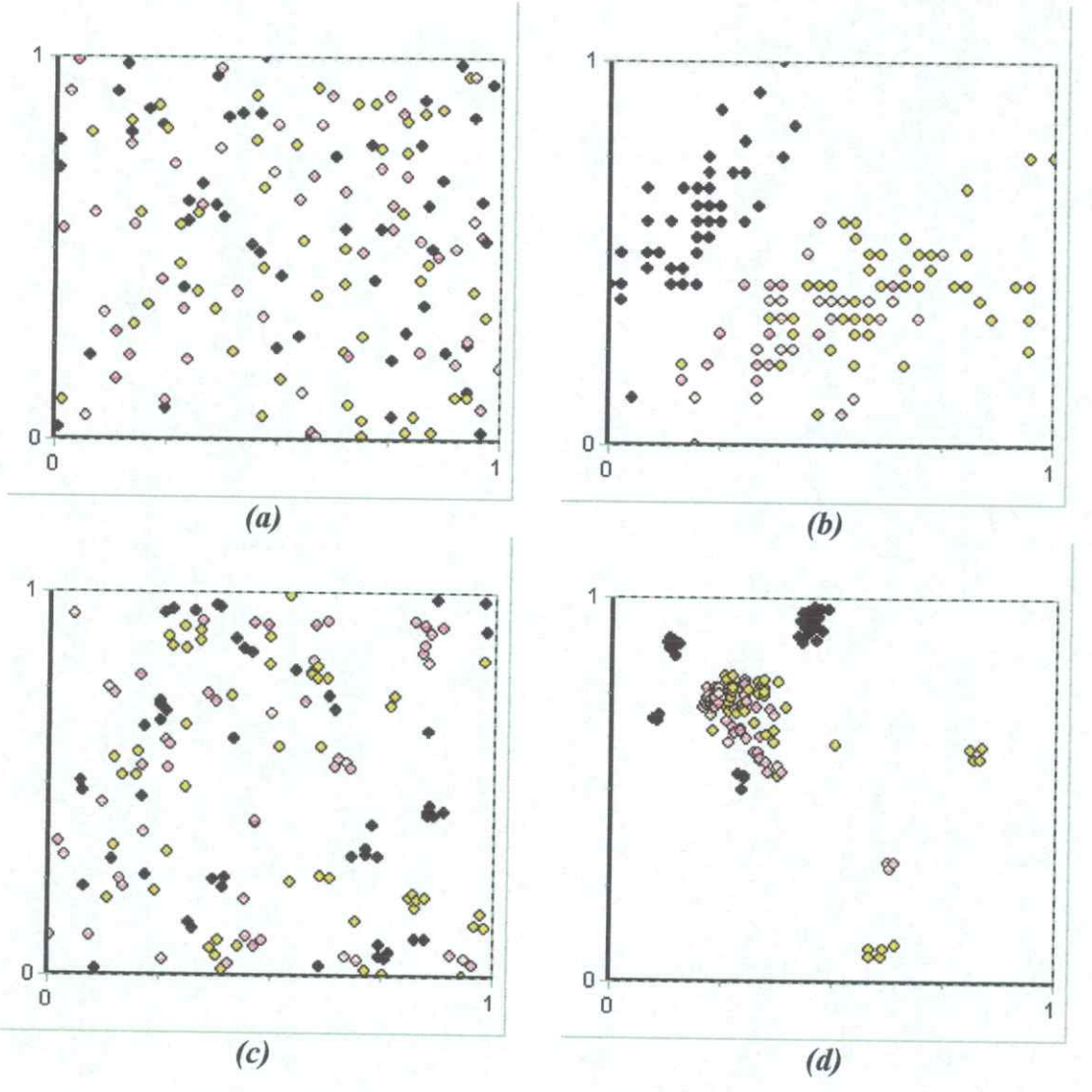


Figure IV.2 : *Un exemple des effets de l'initialisation orientée des insectes avec en (a) la base des Iris répartie aléatoirement, en (b) la même base avec initialisation orientée, et en (c) et (d) les résultats obtenus respectivement à partir de (a) et (b) au bout de 50 itérations avec un* d_{seuil} *de 0.05.*

5.3 L'influence du voisinage

On constate sur les benchmarks (Soybean, Iris) que l'utilisation du calcul de voisinage par grille améliore largement le temps d'exécution, et ceci sans dégradation des performances de l'algorithme en termes de classification (**Tableau IV.2**, **Tableau IV.3** et **Figure IV.3**).

Par contre, les résultats des benchmarks (Heart, Segmentation) ne sont pas très satisfaisants (**Tableau IV.4**, **Tableau IV.5** et **Figure IV.4**).

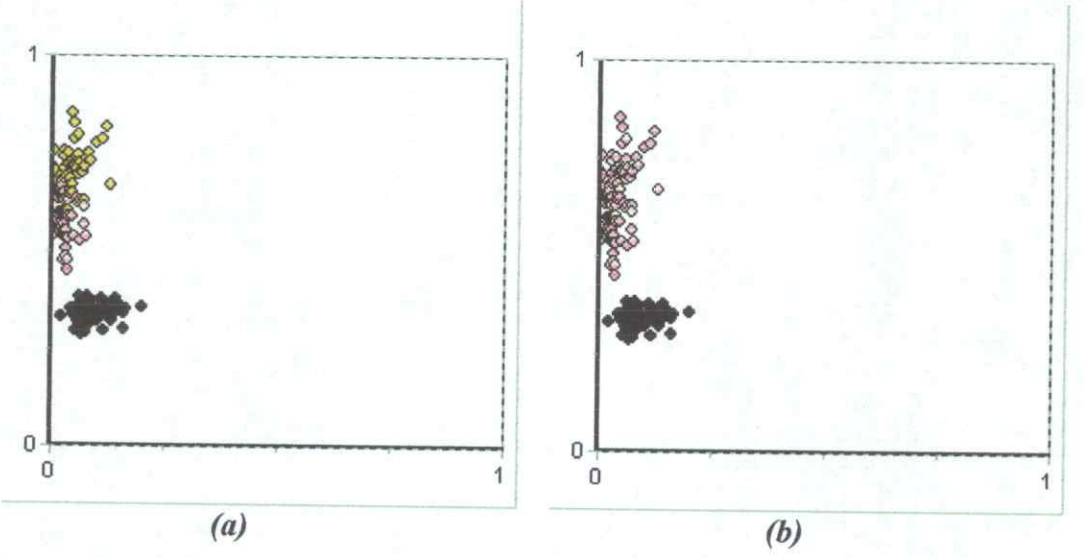


Figure IV.3 :*Un exemple de résultat obtenu en utilisant le voisinage par grille sur la base Iris, (a) le résultat obtenu par l'algorithme principal, (b) le résultat obtenu par l'algorithme de classification en fonction des groupes d'insectes qui se sont formés en (a).*

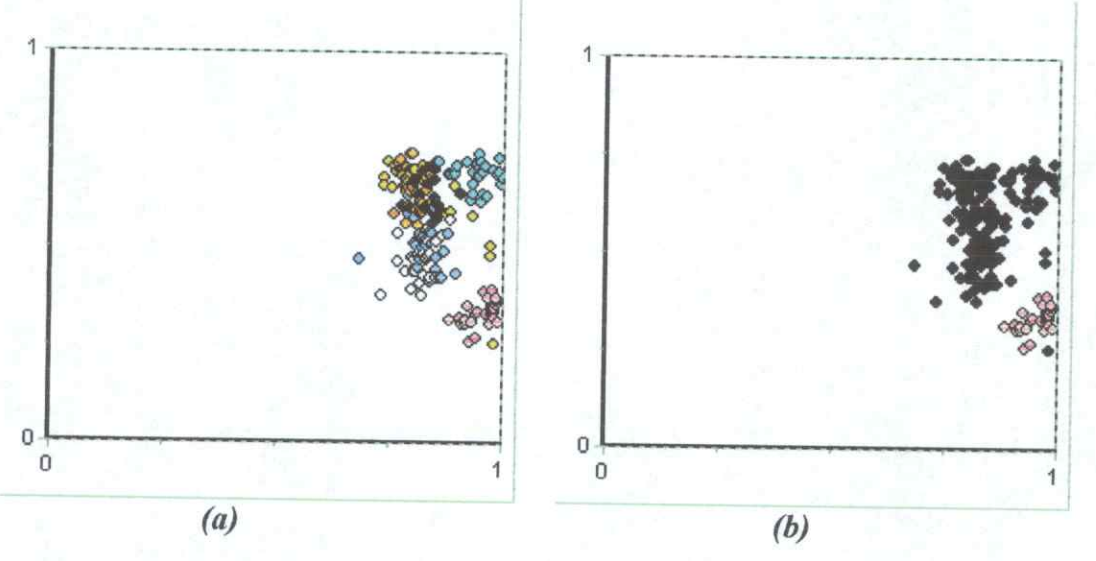


Figure IV.4 :*Un exemple d'un résultat en utilisant le voisinage par grille sur la base Segmentation, (a) le résultat obtenu par l'algorithme principal, (b) le résultat obtenu par l'algorithme de classification en fonction des groupes d'insectes qui se sont formés en (a).*

6. Conclusion

A travers ses tests, nous avons essayé de montrer l'influence du paramètre d_{seuil} sur le résultat et nous avons constaté qu'un d_{seuil} de 0.05 apporte une meilleure classification. Nous avons également montré l'influence de l'initialisation et du calcul de voisinage sur le temps de convergence de l'algorithme.

Conclusion générale et perspectives

A travers ce document, on s'est surtout intéresser au problème de classification non supervisée qui est l'une des tâches de *Data Mining*. 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.

Dans ce travail, nous avons commencer par présenter des généralité sur le *Data Mining* : définition du *Data Mining*, définition des tâches (classification, estimation,...) et explication des méthodes les plus utilisés dans ce domaine. Dans le chapitre II, nous avons exposé des métaheuristiques (colonies de fourmis, algorithmes génétiques...) et les travaux qui ont été développé pour résoudre le problème de classification.
Nous avons ensuite présenté l'approche de classification par nuage d'insectes volants et les différentes classes et méthodes que nous avons conçu pour son implémentation. Cette approche permet non seulement une classification des données mais en plus un affichage interactif et dynamique des groupes trouvés ce qui offrira rapidement à l'utilisateur des informations sur le nombre de groupes, leur forme et la similarité des données qui les composent et les cas de données isolées. L'environnement mis en œuvre offre la possibilité d'arrêter l'exécution de la méthode et de la reprendre plus tard, il est évolutif et permet d'ajouter de nouvelles méta-heuristiques en réutilisant l'ensemble des outils offerts à l'utilisateur.

Les tests effectués sur des Benchmarks supervisés (Chapitre IV) ont prouvé qu'on peut arriver à de bon résultats car le nombre de classes trouvés est toujours très proches du nombre réel et avec un taux de fiabilité très satisfaisant.

Plusieurs points importants restes à étudier :

- Tester de nouvelles méthodes de placement initial des données (ACP ou ACM).
- L'étude de la méthode sur un espace à 3 dimension selon le même protocole que celui présenté pour savoir si l'augmentation de la dimension de l'espace de déplacement influe ou non sur le résultat.
- Des groupes d'insectes, une fois formés, ont par moment tendance à se heurter. Il faudrait certainement introduire ici une notion comme un deuxième voisinage pour faire en sorte que les groupes dévient de leur route avant d'entrer en contact par le voisinage défini par d_{seuil} surtout si les groupes sont dissimilaires .

- Le paramètre d_{seuil} devrait être local à chaque insecte et s'adapter ainsi à la position de celui-ci. Un insecte au cœur d'un groupe homogène n'a pas besoin d'un grand voisinage pour savoir dans quelle direction aller. Par contre, un insecte seul a tout intérêt à agrandir son voisinage pour essayer de rejoindre un groupe. Des règles locales doivent alors être ajoutées pour déterminer comment augmenter ou diminuer le rayon du voisinage.

Bibliographie

[ALS 95] K. Al-Sultan. A tabu search approach to the clustering problem.*Pattern Recognition*, 28 (9): 1443–1451, 1995.

[AZZ 03] N. Azzag, H. Monmarché, M. Slimane, G. Venturini, et C. Guinot. Anttree: a new model for clustering with artificial ants. In IEEECongress on Evolutionary Computation, Canberra, Australia, 08-12 December 2003.

[AZZ 04] Hanene Azzag, Fabien Picarougne, Christiane Guinot, Gilles Venturini. Un survol des algorithmes biomimétiques pour la classification, 2004. www.antsearch.univ-tours.fr/publi/azzag04survol.pdf

[BAC 99] V. Bachelet. *Métaheuristiques parallèles hybrides: Application au QAP*. PhD Thesis, USTL LIFL France, 1999.

[BLA 98] Blake C.and Merz C.(1998)."UCI Repository of machine learning databases, University of California,Irvine,Dept.of Information and ComputerSciences".www.ics.uci.edu/~mlearn/MLRepository.html

[BON 99] BONABEAU E., DORIGO M., THERAULAZ G., *Swarm Intelligence : From Natural to Artificial Systems*, Oxford University Press, New York, 1999.

[BRO 87] BROADWELL P., « Plasm : A Fish Sample », Demonstrated at the Artificial Life Workshop, Sept. 21-25, Los Alamos, 1987.

[BRO 90] D. E. Brown and C. L. Huntley. A practical application of simulated annealing to clustering. Technical Report IPC-91-03, Institute for Parallel Computation, University of Virginia, USA,1990.

[CAS 00] L.N. de Castro et F.J. Von Zuben. An evolutionary immune network for data clustering. In In Proceedings of the IEEE SBRN'00 (Brazilian Simposium on Artificial Neural Networks), pages 84–89, 2000.

[DEN 83] J.L. Deneubourg, J.M. Pasteels, and J.C. Verhaeghe, "Probabilistic behaviour in ants: a strategy of errors ", *Journal of Theoretical Biology*, 105: 259–271, 1983.

[DEN 89] J.L Deneubourg and S. Goss. Collective patterns and decision-making. *Ethology and Evolution*, pages 295–311, 1989.

[DEN 90] J.-L. Deneubourg, S. Goss, N.R. Franks, A. Sendova-Franks, C. Detrain, et L. Chretien. The dynamics of collective sorting: robot-like ant and ant-like robots. In Proceedings of the First International Conference on Simulationof Adaptive Behavior, pages 356–365, 1990.

[DOR 91] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical report, Technical Report 91016, Dipartimento di Elettronica e Informatica, Politecnico di Milano, italie, 1991.

[FAL 94] Falkenauer, E. "A new representation and operators forgenetic algorithms applied to grouping problems", *Evolutionary Computation*, 2 (2): 123–144.

[FAY 96] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth et R. Uthurusamy (Eds.) ; Advances in Knowledge Discovery and Data Mining ; MIT Press ; 1996.

[FUK 90] K. Fukunaga, "*Introduction to Statistical Pattern Recognition*", Academic Press, San Diego, CA, 1990.

[GRA XX] w.ww.grappa.univ-lille3.fr/poly/apprentissage.

[GRF XX] www.grappa.univ-lille3.fr/poly/fouille.

[GLO 86] F. Glover. Future paths for integer programming and links to artificial intelligence.*Computers and Operations Research*, 13: 533–549, 1986.

[GOL 89] D. E. Goldberg. Genetic Algorithms - in Search, Optimization and Machine learning. Addison-Wesley Publishing Company, 1989.

[GOL 94] Golberg, Algorihmes génétiques, exploration, optimisation et apprentissage automatique

[GOV 04] Gérard Govaert, Christophe Ambroise. Analyse des données et data mining, 2004.

[HAS XX] www.hadida.de\recuit simulé.

[HAT XX] www.hadida.de\recherche avec tabous.

[HOL 75] Holland J., "*Adaptation in natural and artificial systems*", University of Michigan Press, Ann Arbor, 1975.

[JAI 88] Jain, A. and Dubes, R. *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series, 1988.

[JON 91] Jones D. and Beltrano M., "Solving partitioning problems with genetic algorithms", In (Belew and Booker, 1991), pages 442–449.

[JOU 03] Laetitia Jourdan 2003. " Métaheuristiques Pour L'Extraction De Connaissances: Application à la Génomique "Thèse pour obtenir le grade de Docteur de L'U.S.T.L.

[KOZ 92] Koza, J."Genetic Evolution and Co-Evolution of Computer Programs". In (Langton et al., 1992), pages 603–629.

[KNI 02] T. Knight et J. Timmis. On data clustering with artificial ants. In J. Garibaldi A. Lotfi et R. John, editors, Proceedings of the 4th International Conference on Recent Advances in Soft Computing, pages 266–271, Nottingham,UK., December 2002.

[KIR 83] S. Kirkpatrick, D.C. Gelatt, and M.P.Vechhi. Optimization by simulated annealing. *Science*, 220: 671–680, May 1983.

[LEF 01] René Lefébure, Gilles Venturi. Data-mining: gestion de la relation client personnalisation de sites web, 2001.

[LIN 97] Data Mining : techniques appliquées au marketing, à la vente et au service client ; Michael J.A. Berry & Gordon Linoff 1997.

[LSP XX] www.lsp.ups-tlse.fr/Besse/pub/Appren_stat.pdf.

[LUM 94] E.D. Lumer et B. Faieta. Diversity and adaptation in populations of clustering ants. In Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, pages 501–508, 1994.

[MAT XX] www.math.mcmaster.ca.

[MON 99] N. Monmarché, M. Slimane, et G. Venturini. On improving clustering in numerical databases with artificial ants. In D. Floreano, J.D. Nicoud, et F. Mondala, editors, 5th European Conference on Artificial Life (ECAL'99), Lecture Notes in Artificial Intelligence, volume 1674, pages 626–635, Swiss Federal Institute of Technology, Lausanne, Switzerland, 13-17 September 1999. Springer-Verlag.

[MON 00] Nicolas Monmarché 2000; "Algorithmes de fourmis artificielles : application à l'optimisation et la classification" thèse pour obtenir le grade de docteur de l'université de TOURS.

[MON02] N. Monmarché, C. Guinot, et G. Venturini. Fouille visuelle et classification de données par nuage d'insectes volants. RSTI-RIA-ECA: Méthodes d'optimisation pour l'extraction de connaissances et l'apprentissage, (6):729–752, 2002.

[MOR XX] http: //morgon.univ-lyon2.fr/Introduction_au_datamining_cours.htm

[NAS 02] O. Nasaroui, D. Dasgupta, et F. Gonzalez. The fuzzy artificial immune system: Motivations, basic concepts, and application to clustering and web profiling. In Proceedings of the IEEE International Conference on Fuzzy Systems at WCCI, pages 711–716, May 12-17 2002.

[PAP 76] C. H. Papadimitriou. *The complexity of combinatorial optimization problems*. PhD thesis, Princeton, 1976.

[PAP 82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[REY 87] REYNOLDS C. W., « Flocks, Herds, and Schools : A Distributed Behavioral Model », *Computer Graphics (SIGGRAPH '87 Conference Proceedings)*, vol. 21, n4, 1987, p. 25–34.

[ROM 04] Wesley Romão, Alex A. Freitas, Itana M. de S. Gimenes. Discovering Interesting Knowledge from a Science & Technology Database with a Genetic Algorithm. 2004. Computer Science Dept., Universidade Estadual de Maringa, Brazil Computing Laboratory, University of Kent, United Kingdom

[SOU 04] Souquet Amédée, Radet François-Gérard. Algorithmes génétiques, 2004.

[TUF 03] Stéphane Tufféry . Data-mining & Scoring, Editions Dunod, 2003.

[VEN 96] Venturini G., "Algorithmes génétiques et apprentissage", Revue d'intelligence artificielle, 10 (2-3): 345–387, 1996.

[VON 91] Von Laszewski, G. «Intelligent Structural Operators for the k-Way Graph Partioning Problem». In Belew and Booker, 1991, pages 45–52.