République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE D'ORAN Mohamed Boudiaf

Faculté des Sciences

Département d'informatique

Spécialité: Informatique ***Option:*** Systèmes, Réseaux et Bases de Données ***(SRBDD)***

MEMOIRE

Présenté par

Alaoui Abdiya

Pour l'obtention du diplôme de Magister en informatique

Thème

Application des techniques des métaheuristiques pour l'optimisation de la tâche de la classification de la fouille de données

Soutenu le ..... devant la commission d'examen composée de:

Qualité	**Nom et Prénom**	**Grade**	**Etb d'origine**
Présidente	Mme Belbachir Hafida	Professeur	USTO
Rapporteur	Mr. Belkadi Khaled	M.C.A	USTO
Examinateur	Mr. Rahal Sid Ahmed	M.C.A	USTO
Examinatrice	Mme Mekki Rachida	M.C.A	USTO

Année Universitaire: 2011/2012

Remerciements

Je souhaite en premier lieu remercier la personne qui m'a encadré et aidé durant ce mémoire Mr le Docteur KHALED BELKADI de m'avoir donné l'opportunité de faire cette thèse de magister ainsi que pour ses conseils judicieux qui m'ont permis d'avancer dans mon travail.

Je remercie Mme le professeur Belbachir Hafida notre responsable de la post-graduation, et qui m'a fait l'honneur de présider ce jury.
Je remercie Mme Mekki Rachida et Mr Rahal Sid Ahmed, qui m'ont fait l'honneur d'accepter de participer à ce jury

Je remercie tous mes enseignants de la post-graduation Systèmes, Réseaux et Bases de Données

Bien entendu, je remercie mes parents et toute la famille pour le soutien moral qu'ils m'ont apporté tout au long de ce travail.

Dédicaces

Je dédie ce modeste travail en premier lieu à :
Mes parents pour tout le soutien et la patience dont ils ont fait preuve.
Mes frères :

Cheikh
Abdelkader
Mustapha

Mes sœurs:

Fadhila
Nadjet

Mes amis (es)

Résumé:

La sélection d'attributs est une étape de prétraitement qui joue un rôle important dans la fouille de données. Elle permet de représenter un sous ensemble de données à partir d'un ensemble volumineux de données et d'éliminer les données redondantes, non pertinentes ou bruitées. Il y a plusieurs avantages de la sélection de sous ensemble d'attributs : Elle facilite la visualisation des données et fournit une meilleure compréhension. Elle réduit la complexité de données d'apprentissage qui va conduire à la réduction du temps de l'algorithme d'apprentissage. Un autre facteur important est la réduction de la dimension du problème, l'amélioration de la performance de la prédiction et la compréhension du modèle d'apprentissage. Ceci est réalisé en supprimant les attributs non pertinents à partir de l'ensemble total des attributs en préservant les avantages mentionnés ci-dessus.

Appliquée à la tâche de la classification supervisée, la sélection d'attributs améliore la précision et la compréhension du classifieur. La recherche d'un sous ensemble d'attributs est un problème d'optimisation NP-difficile qui peut être résolu par les méta-heuristiques. Dans ce travail, nous proposons un algorithme de sélection de sous ensemble d'attributs pertinents à l'aide d'une métaheuristique « Optimisation par colonies de Fourmis » et des arbres de décisions plus précisément C4.5 pour construire un modèle d'apprentissage robuste. Les expérimentations sont réalisées sur des bases de données de l'UCI (University of California, Irvine). Les résultats expérimentaux de notre approche sont comparés à ceux obtenus par : l'Algorithme Génétique, la Recherche par Dispersion et C4.5. Les résultats obtenus sont compétitifs.

Mots clés:

Fouille de données (DataMining), Métaheuristiques, Sélection d'attributs, Classification supervisée, Optimisation par colonies de fourmis, Arbres de décisions (C4.5).

Table des matières

Introduction Générale 1

Chapitre I : La Fouille de données

1.Introduction 3
2. Extraction de connaissance à partir de données (ECD) 3
2.1 Le processus ECD 4
3. Fouille de données (Datamining) 6
3.1 Fouille de données vs statistiques 7
3.2 Fouille de données vs informatique décisionnelle (Business intelligence) 7
3.3 Classification des méthodes de fouille de données 8
3.4 Qu'est ce qu'une donnée 9
3.4.1 Les différentes natures d'attributs 9
3.4.2 Les différentes natures de valeurs d'attributs 9
3.5 Les tâches de fouille de données 10
3.5.1 Classification supervisée 10
3.5.2 Segmentation (clustering) 10
3.5.3 Règles d'Association 10
4. Classification supervisée 11
4.1 Validation croisée 13
4.2 Les méthodes de la classification supervisée 14
4.2.1 Classifieur bayésien naïf 14
4.2.2 Réseaux de Neurones 15
4.2.3 Séparateurs à Vaste Marge (SVM) 15
4.2.4 Plus Proche Voisin (PPV) 16
4.2.5 Bagging 16
4.2.6 Boosting 17
4.2.7 Arbres de Décisions 17
5 Conclusion 20

Chapitre II : La Sélection d'Attributs

1. Introduction 21
2. Objectifs de la sélection d'attributs 21
3. Pertinence d'un attribut 23
4. Redondance d'attributs 24
5. Sélection d'attributs 26
6. Schéma général de la sélection d'attributs 28
6.1 Génération de sous ensemble 28

6.1.1 Point de départ 29
6.1.2 Stratégie de recherche 29
6.1.2.1 La recherche complète 29
6.1.2.2 La recherche séquentielle 29
6.1.2.3 La recherche aléatoire 30
6.2 Evaluation de sous ensemble 31
6.2.1 Les critères indépendants 31
6.2.2 Les critères de dépendances 33
6.2.3 Les méthodes hybrides 34
6.3 Critères d'arrêt 34
6.4 Procédures de validation 34
7. Cadre de catégorisation 35
7.1 Les méthodes Filtrantes 36
7.2 Les méthodes Wrapper 36
7.3 Les méthodes hybrides 38
8. Exemples des algorithmes 38
8.1 Filter 38
8.2 Wrapper 39
9. Conclusion 39

Chapitre III : Les Métaheuristiques

1. Introduction 40
2. Optimisation difficile 40
3. Algorithmes d'optimisation approchés 42
3.1 Heuristiques 42
3.2 Métaheuristiques 42
4. Classification des métaheuristiques 43
4.1 Inspirés de la nature vs non inspiré de la nature 43
4.2 Basé sur la population des solutions vs une solution unique 43
4.3 Dynamique vs une fonction objectif statique 44
4.4 Une structure de voisinage vs des structures divers de voisinages 44
4.5 Utilisation de mémoire à long terme vs mémoire à court terme 44
5. Extension 44
6. Présentation des principales Métaheuristiques 45
6.1 Méthodes de trajectoires 45
6.1.1 Méthode de Descente (Hill Climbing) 45
6.1.2 Recuit Simulé (Simulated Annealing) 46
6.1.3 Recherche Tabou (Tabu Search) 46
6.1.4 Méthode GRASP (Greedy Randomized search Procedure) 48

6.1.5 Recherche à Voisinages Variables (VNS) 49
6.2 Méthodes basées sur les population 49
6.2.1 Algorithmes de Colonies de Fourmis (Ant Colony Optimization) 49
6.2.2 Algorithmes Génétiques (Genetic Algorithm) 49
6.2.3 Algorithme d'Estimation de Distribution (EDA) 50
6.2.4 Optimisation par Essaims particulaires (Particle Swarm Optimization) 51
6.2.5 Recherche par Dispersion (Scatter Search) 52
7. Algorithmes de Colonies de Fourmis (ACO) 53
7.1 Optimisation naturelle par les fourmis 53
7.2 Exemple du problème du voyageur de commerce (PVC) 54
7.3 Variantes 56
7.4 Formalisation d'un algorithme de Colonie de Fourmis 57
8. Conclusion 59

Chapitre IV: Implémentation de l'approche proposée (ACO/C4.5)

1.Introduction 60
2. Exemples des métaheuristiques utilisées pour la sélection d'attributs 60
3. Arbres de décisions C4.5 62
3.1 Fonction d'évaluation 63
4. L'approche proposée 64
4.1 Description de la stratégie 65
4.1.1 Espace de recherche 65
4.1.2 Initialisation des valeurs de phéromones 66
4.1.3 Sélection d'attributs 67
4.1.4 Valeur heuristique 67
4.1.5 Fonction d'évaluation fitness (taux d'erreur du chemin) 67
4.1.6 Mise à jour de phéromone 68
4.2 Approche hybride ACO/C4.5 (Algorithme IV.2) 68
5. Expérimentation 70
5.1 Présentation de weka 70
5.1.1 Utilisation de notre propre code sous weka 71
5.2 Jeux de données de l'UCI 72
5.3 Résultats expérimentaux 74
5.3.1 Algorithme Génétique 76
5.3.2 Recherche par dispersion 77
5. Conclusion 79

Conclusion Générale 80
Bibliographie 81
Annexes 89

Liste des figures

Figure I.1 : Le processus d'extraction des connaissances à partir de données.....4
Figure I.2 : Schéma de validation croisée d'ordre c.....14
Figure I.3 : Un exemple d'arbre de décision (jouer au tennis).....19

Figure II.1 : Espace d'attributs de 4 dimensions.....22
Figure II.2 : Catégorisation possible des attributs.....26
Figure II.3 : Processus de sélection d'attributs avec validation.....28
Figure II.4 : Résumé des méthodes de sélection d'attributs.....31
Figure II.5 : Schéma de validation croisée pour l'évaluation d'un processus de sélection – classification.....35
Figure II.6 : Deux approches de la sélection d'attributs.....37

Figure III.1 : Une solution locale minimale et une solution globale minimale.....42
Figure III.2 : Etape expérimentale pour observer le comportement des fourmis.....54
Figure III.3 : Situation de début de recherche-comportement de fourragement.....54
Figure III.4 : Comportement du fourragement après un certain temps.....54

Figure IV.1 : Espace de recherche N*N pour le chemin de fourmis.....66
Figure IV.2 : Interface de Weka.....72
Figure IV.3 : Comparaison du taux d'erreur entre ACO/C4.5 et C4.5.....75

Figure A.1 : Diagramme de Hasse représentant le treillis des itemsets.....90

Liste des Tableaux

Tableau I.1 : Quelques méthodes de fouille de données 8
Tableau I.2 : Jeu de données (jouer au tennis) 19

Tableau II.1 : Comparaison des fonctions d'évaluation 33

Tableau IV.1 : Les Jeux de données 73
Tableau IV.2: Comparaison de notre approche hybride et C4.5 75
Tableau IV.3 : Les trois algorithmes utilisés dans l'implémentation 78
Tableau IV.4 : Nombre d'attributs et le taux d'erreur de la classification des trois approches hybrides 79

Tableau A.1 : Contexte d'extraction des Règles d'association D 89
Tableau B.3: Cadre de catégorisation 94

Liste des Algorithmes

Algorithme I.1 : Algorithme d'apprentissage générique 17

Algorithme II.2 : Algorithme Wrapper 37

Algorithme III.1 : Méthode de Descente 45
Algorithme III.2 : Recuit Simulé 46
Algorithme III.3 : Recherche Tabou 47
Algorithme III.4 : Méthode GRASP 48
Algorithme III.5 : Recherche à Voisinages Variables 49
Algorithme III.6 : Algorithme Génétique 50
Algorithme III.7 : Algorithme d'Estimation de Distribution 51
Algorithme III.8 : Optimisation par les Essaims Particulaires 52
Algorithme III.9 : Recherche par Dispersion 53
Algorithme III.10 : Algorithme de Colonies de Fourmis de base : le « Ant System » 56
Algorithme III.11 : Optimisation par Colonies de Fourmis 57

Algorithme IV.1 : Principe de l'algorithme : C4.5 62
Algorithme IV.2 : ACO/C4.5 69

Algorithme B.1 : Algorithme Filter 92
Algorithme B.2: Algorithme Hybride 93

Introduction Générale

Au fil des années, la capacité de stockage et l'acquisition de l'information sont devenues moins chères. Cela nous permet de stocker toutes les données possibles liées au problème en question. Toutefois, il y a des données qui ne contiennent pas nécessairement des informations utiles à extraire. Avec l'augmentation exceptionnelle de la quantité des données pouvant être stockées, l'exploration de nouvelles méthodes qui sont en mesure de traiter des données automatiquement est nécessaire suivant un processus d'extraction de connaissances à partir de données. Ce processus permet l'intégration et la collecte des données ; la sélection, le nettoyage et le traitement des données ; l'analyse des données pour l'extraction des motifs et des modèles appropriés ; évaluation et interprétation des modèles construits ; consolidation des connaissances disponibles à l'emploi. A ce stade ces modèles correspondent à la fouille de données. Il existe différentes techniques de fouille de données, y compris la classification supervisée, la recherche des règles d'association et la segmentation.

Une technique importante de la fouille de données est la classification supervisée. L'objectif de la classification est de construire un ou plusieurs modèles sur des données d'apprentissage, qui peut prédire correctement la classe d'objets d'essai.

Il existe plusieurs problèmes à partir d'une grande échelle des domaines qui peuvent être exprimés sous la forme de problèmes de classification. La classification a plusieurs applications importantes de notre vie comme les applications de diagnostique médicales, de la détection des fraudes et de détection biométrique…

Un des algorithmes le plus populaire de la fouille de données est les arbres de décisions. Ils permettent de produire des procédures de classification compréhensibles. Ils représentent sous forme de graphes un ensemble de règles facilement interprétables.

La sélection d'un sous ensemble d'attributs est un processus qui permet la représentation d'un sous ensemble des données à partir d'un ensemble volumineux des données dont laquelle l'ensemble d'origine peut contenir de milliers d'attributs. Chaque attribut peut porter un peu d'information, il devient difficile de traiter tous les attributs. Il est important d'extraire ou de choisir les attributs importants à partir d'une base de données. Il y a plusieurs avantages de la sélection de sous ensemble d'attributs : Elle facilite la visualisation des données et fournit une meilleure compréhension. Elle réduit la complexité de données d'apprentissage qui va conduire à la réduction du temps de l'algorithme d'apprentissage. Un autre facteur important est la réduction de la dimension du problème, l'amélioration de la performance de la prédiction et la compréhension du modèle d'apprentissage. Ceci est réalisé en supprimant les attributs non pertinents à partir de l'ensemble total des attributs en préservant les avantages mentionnés ci-dessus. La sélection d'un sous ensemble d'attributs est un problème NP-difficile qui peut être résolu par les métaheuristiques.

Les métaheuristiques constituent une classe de méthodes qui fournissent des solutions de bonnes qualités en un temps raisonnable à des problèmes difficiles pour lesquels il n'existe pas des méthodes classiques plus efficaces. Elles sont généralement des méthodes stochastiques itératives qui progressent vers un optimum global en évaluant une fonction « objectif ». L'Optimisation Par Colonies de Fourmis est une branche de l'intelligence en essaim (une métaheuristique basée sur la population), inspirée par le comportement des fourmis réelles.

L'algorithme de colonie de fourmis est bien adapté pour des problèmes d'optimisation discrets tels que les problèmes d'affectation quadratiques, l'ordonnancement des jobs, le routage de réseaux, la coloration des graphes, la bioinformatique et la fouille de données.

La problématique de ce mémoire de magister est de voir quel est l'apport des Métaheuristiques dans l'amélioration des tâches de la fouille de données (DataMining). Plus précisément on résout un problème d'optimisation NP-difficile qui est la sélection d'un sous ensemble d'attributs pertinents pour améliorer la classification supervisée (un taux de précision élevé) de la fouille de données, en utilisant les métaheuristiques.

Dans ce travail, nous proposons un algorithme de sélection de sous ensemble d'attributs pertinents à l'aide d'une métaheuristique « Optimisation par colonies de Fourmis » et des arbres de décisions plus précisément C4.5 pour construire un modèle d'apprentissage robuste.

Le présent mémoire est organisé en quatre chapitres:

Dans le premier chapitre on va présenter le processus de l'extraction de connaissance à partir des données (ECD), le DataMining, ses tâches en se concentrant sur la classification supervisée.

Dans le deuxième chapitre, nous présenterons la sélection d'attributs, ses objectifs, ainsi que les différentes méthodes proposées dans la littérature.

Dans le troisième chapitre, on va définir les problèmes d'optimisation difficile, l'heuristique et la métaheuristique avec une classification de cette dernière et une présentation des principales métheuristiques connues dans la littérature en se basant sur les algorithmes de colonies de fourmis.

Le quatrième chapitre est consacré à la résolution d'un problème NP-difficile qui est « la sélection d'attributs » par une hybridation entre l'algorithme de colonie de fourmis et les arbres de décisions C4.5 avec une implémentation de l'approche proposée en présentant les résultats expérimentaux obtenus.

Enfin, on termine par une conclusion générale et quelques perspectives.

Chapitre I:

La Fouille de données

1. Introduction

Ce chapitre a pour objectif de présenter le processus de l'extraction de connaissances à partir des données, la fouille de données, ses tâches et plus précisément la classification supervisée. La fouille de données désigne en réalité un ensemble de traitements différents menant à la découverte de connaissances variées. Ces traitements sont la classification, la segmentation et les règles d'association.

La classification est une tâche centrale de l'étape de fouille de données dans le processus d'extraction de connaissances à partir de données. Elle consiste à construire un classifieur à partir d'un ensemble d'exemples étiquetés par leur classe (phase d'apprentissage) et ensuite à prédire la classe de nouveaux exemples avec le classifieur (phase de classement).

2. Extraction de connaissances à partir de données (ECD)

L'extraction de connaissance à partir de données est un domaine de connaissance qui est très vaste. Son objectif est de trouver des modèles intelligibles à partir de données.
Le défi est de travailler avec une grande masse de données de sorte que le processus devrait être entièrement ou partiellement automatique et d'utiliser des techniques permettant d'analyser les données et d'extraire des connaissances utiles.

L'automatisation du processus est confrontée à la complexité des données, ainsi que d'autres problèmes tels que le bruit, les valeurs manquantes et l'imprécision.

Plusieurs définitions ont été avancées pour essayer de cerner le processus ECD. On peut retenir que l'ECD est un processus non trivial d'extraction de connaissances cachées et potentiellement utiles au sein d'entrepôts de données volumineux et à grande dimensionnalité **[han et al, 1992]**

La non trivialité réfère au fait que, contrairement à la statistique qui est confirmatoire, la fouille de données est plutôt exploratoire **[wijsen, 2001]**. Avec l'ECD, on ne sait pas a priori ce qu'on pourrait apprendre des données. Cette connaissance a priori caractérise les résultats mis à nus qui sont plutôt cachés. La non-trivialité se justifie également par le fait que la découverte de connaissances passe par plusieurs étapes.

Les résultats d'une fouille de données devraient être non seulement utiles mais compréhensibles par les utilisateurs du domaine (les résultats devraient servir de support au processus de décision).

L'entrepôt contient un volume important de données et ces données sont décrites par plusieurs attributs.

Il est important que l'ECD est un processus, c.à.d. un ensemble d'étapes et d'actions dont la finalité est l'extraction des tendances et des corrélations au sein des données.

Contrairement aux idées reçues, l'ECD ne se limite pas exclusivement à la fouille de données **[miller et al, 2001]** qui en constitue toutefois la partie visible.

Ce dernier se compose en effet d'un ensemble d'étapes allant de la compréhension des domaines d'étude, à l'exploitation des résultats de la fouille en passant par la fouille de données elle même **[fayyad et al, 1996]**. Il est important de noter que le processus d'extraction de connaissances en ce sens qu'il est interactif et itératif.

- par itératif, il faut entendre que l'ECD n'est pas un processus linéaire où chaque étape est appliquée une seule fois pour aboutir à la fin avec le modèle de connaissance recherché cela pourrait être le cas dans le meilleur des scenarios possibles mais cela arrive assez rarement dans la pratique.

-l'ECD est un processus « human-centric » c.à.d. que l'homme est au cœur du processus d'où le qualificatif interactif. Il est important de relativiser le qualificatif automatique à tord attribué à la fouille de données. Les outils de fouille ne sont pas des rebots qui seuls doivent parcourir de larges ensembles de données afin d'y extraire quelques informations utiles à l'organisation **[fayyad, 1998]**, biens au contraire, il s'agit d'un ensemble d'interactions entre l'utilisateur et les outils de fouille afin que les résultats obtenus au bout du processus soient non seulement compréhensibles mais utiles.

2.1 Processus ECD

Le processus d'extraction des connaissances à partir de données présenté en **Figure I.1** est organisé en 05 phases:

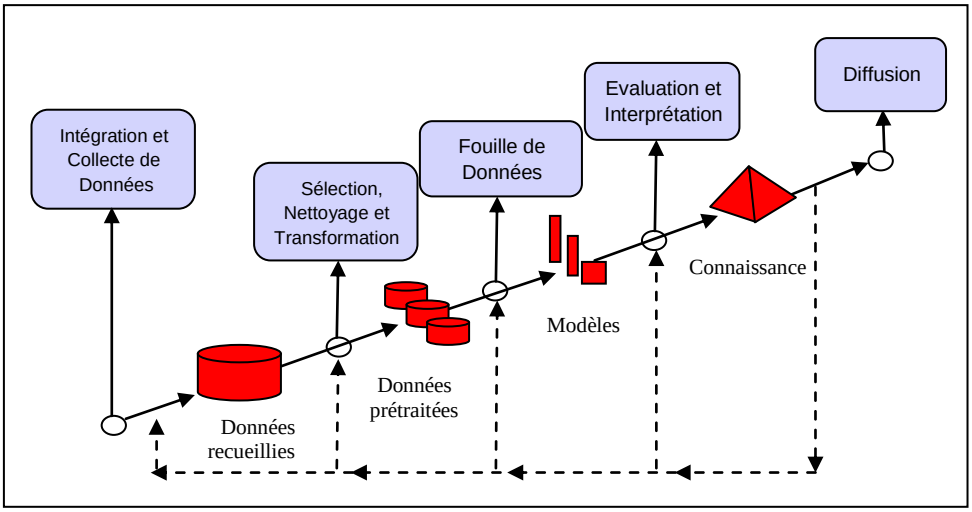

Figure I.1: Le processus d'extraction des connaissances à partir de données

Phase 01 (intégration et collecte de données)

Cette étape consiste à regrouper un ensemble de données relatives au système cible de l'étude, en unique source : l'entrepôt de données pour faciliter les différents traitements. Ces données sont le plus souvent hétérogènes. Cette première phase se concentre sur la collecte des données nécessaires pour mener à bien un processus ECD.

Phase 02 (sélection, nettoyage et transformation)

Cette étape consiste à sélectionner, dans l'entrepôt, les données qui seront retenues pour construire le modèle.

Le nettoyage de données consiste à gérer la qualité des données et plus particulièrement les imprécisions et les incertitudes qu'elles peuvent contenir. Le développeur pourra choisir de corriger ou d'ignorer les données manquantes et erronées.

Quelques unes des tâches de cette phase sont :

- la détection des valeurs erronées.

Les valeurs incohérentes parce qu'il y a des erreurs (délibérées ou non), des omissions, des duplications qui ont été introduites, … (erreur de mesure par exemple). Elles peuvent être dues à une erreur dans les données, comme des cas particuliers. Selon les tâches qu'on va effectuer après la détection de ces anomalies, soit à prendre en compte ou à éliminer.

- le traitement des valeurs manquantes.

Les valeurs manquantes sont inconnues, non nécessaires, non enregistrées, …

Les valeurs manquantes doivent être traitées spécialement, parfois une valeur manquante doit être détectée et ajoutée. Il est intéressant de les conserver car elles signifient qu'il y a un problème dans les données (examen médical par exemple) et dans certains cas, on peut ignorer les données avec les valeurs manquantes.

- l'échantillonnage.

Dans le cas où la quantité des données est importante, on peut travailler avec un échantillon des données. Cela permet d'augmenter l'efficacité du processus de la fouille de données

- la sélection d'attributs.

La sélection d'attributs est une étape de prétraitement qui permet d'améliorer l'efficacité du processus et de donner une plus grande lisibilité du problème à étudier.

La transformation consiste à préparer les données brutes et à les convertir en données appropriées. Elle structure les données dans un formalisme attendu par les algorithmes qui seront appliqués par la suite. Les opérations les plus connues sont la discrétisation des variables continues, la binarisation des variables nominales, l'agrégation de données. Cette étape facilite les tâches de la fouille de données. Plusieurs algorithmes de fouille de données sont contraignants sur la forme des données qu'ils acceptent

La discrétisation de variables continues est un exemple de transformation d'attributs. Il s'agit de transformer un attribut continu en divisant son domaine en intervalles finis. Ainsi, le domaine de l'attribut transformé devient un ensemble discret.

L'agrégation de données est un autre type de transformation. L'agrégat d'un attribut est la transformation de ce dernier par une règle ou une équation. Par exemple, on veut analyser

les salaires annuels des employés, et que l'on dispose seulement des salaires mensuels, un nouvel attribut agrégat serait le salaire multiplié par douze.

Phase 03 (la fouille de données)

La fouille de données est le cœur du processus car elle permet d'extraire l'information (utile et inconnue) de gros volumes de données stockées dans des bases ou des entrepôts de données. La section 3 est consacrée à la fouille de données et ces tâches (la classification, …).

Phase 04 (évaluation et interprétation)

Les résultats obtenus par la phase précédente doivent être évalués pour mesurer la qualité des modèles construits. Il n'y a pas de critère unique d'évaluation, mais il en existe plusieurs. Le choix dépend du contexte dans lequel nous sommes: par exemple dans la classification, on utilise souvent la précision et la régression prédictive, le critère utilisé est l'erreur quadratique moyenne de la valeur prédite à partir de la valeur réelle.

L'interprétation concerne l'analyse des résultats de l'étape de la fouille de données par un expert du domaine assisté par le développeur. Celui-ci pourra décider de modifier les réglages effectués lors des étapes précédentes (données sélectionnées, seuils de discrétisation, …) et de relancer le processus.

Phase 05 (diffusion)

Une fois le modèle est construit, il y a plusieurs tâches à effectuer: Soit de prendre des décisions futures, soit d'appliquer sur un ensemble de nouvelles données.

Dans le cours du temps, on doit mesurer l'évolution du modèle. Il peut émerger de nouveaux cas qui provoquent une dégradation progressive du modèle. Par exemple dans les applications liées au commerce, où les goûts des consommateurs sont influencés par des facteurs externes non prévisibles.

3. Fouille de données (Datamining)

Il existe plusieurs définitions de la fouille de données, nous citons quelques unes d'entre elles :

« La découverte d'informations originales, auparavant inconnues et potentiellement utiles, à partir des données » **[frawly et al, 1992]**

« Un processus d'aide à la décision où les utilisateurs cherchent des modèles d'interprétations dans les données » **[kamran parsaye, 1996]**

« L'exploration et l'analyse, par des moyens automatiques ou semi automatiques, d'un large volume de données afin de découvrir des tendances ou des règles » **[Michael et al, 1997]**

La fouille de données désigne l'ensemble des algorithmes et des méthodes destinés à l'exploration et à l'analyse de grandes bases de données informatiques en vue de détecter dans ces données: des règles, des associations et des tendances inconnues (non fixées a

priori) et des structures particulières, restituant de façon concise, l'essentiel de l'information utile pour l'aide à la décision. Elle utilise des méthodes statistiques avancées comme le partitionnement des données (rassemblement des données en paquets homogènes), et emploie régulièrement des mécanismes de l'intelligence artificielle ou des réseaux de neurones.
Le but de la fouille de données est de découvrir des relations inconnues dans les données, spécialement quand les données proviennent de bases de données différentes.

La fouille de données a aujourd'hui une grande importance économique du fait quelle permet d'optimiser la gestion des ressources (humaines et matérielles). Elle est utilisée par exemple :
- diagnostic médical : « les patients ayant tels et tels symptômes et demeurant dans des agglomérations de plus de 10^4 habitants développent couramment telle pathologie ».
- analyse de génome et bioinformatique plus généralement.
- commerce électronique, recommandation de produits.
- analyser les pratiques et les stratégies commerciales et leurs impacts sur les ventes.
- extraction d'informations depuis des textes : fouille de textes…

Après avoir défini la fouille de données, il convient de préciser ce qui la diffère des domaines d'analyse connexes avec lesquels on pourrait quelque fois la confondre

3.1 Fouille de données vs statistiques

Contrairement à la méthode statistique, la fouille de données ne nécessite jamais que l'on établisse une hypothèse de départ qu'il s'agira de vérifier. C'est des données elles même qui se dégageront les corrélations intéressantes et le logiciel n'étant là que pour les découvrir. La fouille de données se situe à la croisée des statistiques (Analyse Exploratoire, Modélisation Paramétrique/Non-paramétrique,…), de l'intelligence artificielle (Apprentissage Symbolique, Reconnaissance des formes, Réseaux de Neurones,…) et des bases de données (BD relationnelles, Entrepôts de données,…). Les programmes d'analyse sont lancés sur la base de données, sans objectifs du genre « trouver la corrélation entre telle et telle données »

3.2 Fouille de données vs informatique décisionnelle (Business intelligence)

Business intelligence également « informatique décisionnelle » englobe les solutions informatique apportant une aide à la décision avec, en bout de chaine, rapports et tableaux de bord de suivi à la loi analytique et prospectif.
Le but est de collecter, consolider, modéliser et de restituer les informations disponibles au sein des bases de données de l'entreprise et de permettre aux responsables de la stratégie d'une entreprise d'avoir une vue d'ensemble d'activités traitées :
- sélection des données.

- tri, regroupage ou répartition de ces données selon certains critères.
- élaboration des calculs récapitulatifs simples.
- présentation synthétique des résultats.

La fouille de données introduit une dimension supplémentaire qui est la modélisation exploratoire (détecter les liens de cause à effet et valider leur reproductibilité).

3.3 Classification des méthodes de fouille de données

Les méthodes de la fouille de données peuvent être classées selon le type d'apprentissage utilisé dans les méthodes de la fouille

- Fouille supervisée: comprenant à la fois des données d'entrée et de sortie dont le but est de classer correctement un nouvel exemple.
- Fouille non supervisée: c'est un processus dans lequel l'apprenant reçoit des exemples d'apprentissage (pas notion de classe) ne comprenant que des données d'entrées dont le but est de regrouper les exemples en segment (cluster) d'exemples similaires.

On peut classer les méthodes de fouille de données selon les objectifs :

- Classification : elle permet de prédire si une instance de donnée est membre d'un groupe ou d'une classe prédéfinie.
- Segmentation : elle consiste à partitionner logiquement la base de données en clusters (Former des groupes homogènes à l'intérieur d'une population).
- Règles d'associations : elle consiste à déterminer les corrélations (ou relations) entre les attributs. Des exemples des méthodes de fouille de données classés selon cette organisation sont présentés dans **Tableau I.1**:

	Supervisées	**Non supervisées**
Classification	▪ Arbres de décision ▪ Réseaux de Neurones avec perceptron. ▪ Modèles/ Réseaux Bayésiens. ▪ Machines à vecteur supports (SVM).	▪ K plus proches voisins (ppv-raisonnement à partir de cas). ▪ Règles temporelles ▪ Reconnaissance des formes.
Segmentation		▪ K-means. ▪ K plus proches voisins (ppv-raisonnement à partir de cas). ▪ Réseaux de Neurones avec cartes de kohonen.
Association		▪ Règles d'Association

Tableau I.1: Quelques méthodes de fouille de données

On peut classer les méthodes de fouille de données en deux grandes familles

Méthodes prédictives et méthodes descriptives. **[Usama M. Fayyad, 1996]**

- Les méthodes descriptives: elles permettent de décrire la situation actuelle, elles caractérisent les propriétés générales des données dans la base de données et mettent l'accent sur la compréhension et l'interprétation de ces dernières.

- Les méthodes prédictives: qui, en apprenant sur le passé, simulent le futur. Elles utilisent les données avec des résultats connus pour développer des modèles permettant de prédire les valeurs des autres données.

3.4 Qu'est ce qu'une donnée

Une donnée est un enregistrement au sens des bases de données, que l'on nomme aussi « individu » (au sens des statistiques) ou « instance » (terminologie orientée objet au sens informatique). Une donnée est caractérisée par un ensemble d'attributs.

3.4.1 Les différentes natures d'attributs

Un attribut peut être de nature qualitative ou quantative en fonction de l'ensemble des valeurs qu'il peut prendre. Un attribut est qualitatif si on ne peut pas en faire une moyenne, sa valeur est d'un type défini en extension (une couleur, une marque de voiture, …) sinon, l'attribut est de nature quantative : un entier, un réel, … ; il peut représenter un salaire, une surface, un nombre d'habitants, … ; on peut donc appliquer les opérateurs arithmétiques habituels sur les attributs quantatifs, ce qui n'est pas le cas des attributs qualitatifs.

Un attribut peut également être un enregistrement (une date par exemple), donc composé lui-même de sous attributs (jour, mois et année dans le cas d'une date), sur lequel on peut définir les opérateurs arithmétiques habituels:

Donc quantatif ne signifie pas forcément « numérique » et, réciproquement, numérique ne signifie pas forcement quantatif : un code postal est numérique, mais pas quantatif.

3.4.2 Les différentes natures de valeurs d'attributs

- **attributs à valeurs nominales:** les valeurs sont des symboles (des noms) dont aucune relation (ordre ou distance) existe entre les nominaux.

Exemple01: Les valeurs de temps : ensoleillé, neigeux, pluvieux, gris.

Une opération (règle) qu'on peut exécuter: Si temps = neigeux alors voyage = non.

- **attributs à valeurs ordinales**: une notion d'ordre s'impose sur les ordinaux mais il n'est pas possible de calculer directement des distances entre les valeurs ordinales, les opérations d'addition et de soustraction ne sont pas possibles.

Exemple02:

Les températures sont décrites par les adjectifs (chaud, froid, moyen)

et chaud > moyen > froid,

Une règle qu'on peut exécuter: si température > froid alors voyage = oui

- attributs de type intervalles: les intervalles impliquent une notion d'ordre, et les valeurs sont mesurées dans des unités spécifiques et fixées. La somme, la différence et le produit de deux intervalles ne sont pas possibles (car le point zéro n'existe pas).
Exemple03: La température est exprimée en degrés Celsius ou Fahrenheit.
- attributs de type rapport (ratio): toutes les opérations mathématiques sont autorisées sur les attributs de ce type.
Exemple04: L'attribut distance, par exemple on peut comparer deux distances

3.5 Les tâches de fouille de données

3.5.1 Classification supervisée

Elle permet de prédire si un élément est un membre d'un groupe ou d'une catégorie donnée (voir section 4).

3.5.2 Segmentation (clustering)

Elle consiste à former des groupes (clusters) homogènes à l'intérieur d'une population. Pour cette tâche, il n'y a pas de classe à expliquer ou de valeur à prédire définie à priori, il s'agit de créer des groupes homogènes dans la population (l'ensemble des enregistrements). Après l'application de l'algorithme et lorsque les groupes ont été construits, d'autres techniques ou une expertise doivent dégager leur signification et leur éventuel intérêt. Parmi les algorithmes de clustering, il y a K-means **[MacQueen, 1967]**.

La segmentation est une tâche d'apprentissage non supervisée car on ne dispose d'aucune autre information préalable que la description des exemples. Elle joue un rôle exceptionnel dans les applications de fouilles de données telles que l'exploration des données scientifiques, la fouille de texte, les applications de bases de données spatiales, l'analyse du web, le marketing, la biologie et d'autres [**berkhin, 02**].

Les propriétés des algorithmes de clustering dans le Datamining

Les caractéristiques désirables des algorithmes de clustering dépendent du problème particulier à étudier **[kumar 2000, berkhin 2002]**, voir l'annexe A

3.5.3 Règles d'Association

La découverte des règles d'association (la recherche des corrélations fréquentes entre un ensemble d'objets) à partir d'une base de données transactionnelle est proposée par les auteurs **[agrawal et al, 1993]**. Cette approche est considérée aujourd'hui comme faisant parti des approches d'apprentissage symbolique non supervisé utilisé dans le domaine de fouille de données et d'extraction de connaissances.

Elle peut être appliquée à tout secteur d'activité pour lequel il est intéressant de rechercher des groupements potentiels des produits ou de services : services bancaires, services de télécommunication, l'organisation de l'accès aux sites internet. Par exemple, elle peut être également utilisée dans le secteur médical pour la recherche de complications

dues à des associations de médicaments ou à la recherche de fraudes en recherchant des associations inhabituelles. Une caractéristique principale de la méthode est la clarté des résultats produits.

L'extraction de règles d'association est un processus itératif et interactif constitué de plusieurs phases allant de la sélection et la préparation de données jusqu'à l'interprétation des résultats. La pluparts des approches proposées dans la littérature pour l'extraction des itemsets fréquents reposent sur ces phases (voir l'annexe A). Parmi les algorithmes d'extraction des itemsets fréquents, on trouve l'algorithme Apriori **[Agrawal, 94]**.

4. Classification supervisée

Le fonctionnement de la classification supervisée se décompose en deux points :
Le premier est la phase d'apprentissage, tout ce qui est appris par l'algorithme est représenté sous la forme des règles de classification que l'on appelle le modèle d'apprentissage.
Le second point est la phase de la classification proprement dite, dans laquelle les données tests vont être utilisées pour estimer la précision des règles de classification générées pendant la première phase. Si la précision du modèle est considérée comme acceptable, la règle pourra être appliquée à des nouvelles données.

La classification supervisée est une tâche de fouille de données qui utilise la connaissance à priori sur l'appartenance d'un exemple à une classe. Elle permet d'apprendre à l'aide d'un ensemble d'entrainement, une procédure de classification qui permet de prédire l'appartenance d'un nouvel exemple à une classe.

On commence par la définition du problème de classification supervisée. Ensuite, on donne quelques exemples des domaines d'application.

Définition01: on dispose d'un ensemble x de N exemples, i.e. des couples (donnée, étiquette). Chaque donnée $x_i \in D$ est caractérisée par p attributs et par sa classe $y_i \in Y$. Dans un problème de classification supervisée, la classe prend sa valeur parmi un ensemble y fini. Le problème consiste alors, en s'appuyant sur l'ensemble d'exemples $X=\{(x_i,y_i)_{i \in \{1,\ldots,N\}}\}$, à prédire la classe de toute nouvelle donnée $x \in D$.
On parle d'une classification binaire quand le nombre de classes $|y|$ est 2 et il peut être quelconque. Dans tous les cas, il s'agit d'un attribut qualitatif pouvant prendre un nombre fini de valeurs.
Si une donnée appartient à plusieurs classes, c'est le cas du problème multi classe.
L'apprentissage supervisé est l'utilisation d'un ensemble d'exemples pour prédire la classe de nouvelles données.

-hypothèse de représentativité des exemples.

Les exemples dont on dispose sont représentatifs de l'ensemble de données. En général, il est difficile de déterminer si cette hypothèse est vérifiée pour un jeu d'exemples donné.

Typiquement, dans un problème de classification, le nombre d'exemples (les données pour lesquelles on dispose de leur classe) est petit : l'étiquetage des données est en général effectué à la main, ce qui entraine un cout élevé de cet étiquetage, donc l'obtention d'un nombre important d'exemples. C'est le problème de statistique où l'échantillon est petit.

Définition02: un classifieur est un algorithme qui, à partir d'un ensemble d'exemples, produit une prédiction de la classe de toute donnée.

D'une manière générale, un classifieur procède par « induction » : à partir d'exemples (cas particuliers), on construit une connaissance plus générale. La notion d'induction de connaissance implique la notion de « généralisation » de la connaissance : à partir de connaissances éparses, les exemples, on induit une connaissance plus générale. Naturellement, même si l'on suppose que la classe des étiquettes n'est pas erronée, il y a un risque d'erreur lors de généralisation, ce risque est quantifié par la notion de « taux d'échec », ou d' «erreur de généralisation ».

« Généraliser » : il faut déterminer, au moins implicitement, la pertinence d'attributs pour prédire la classe d'une donnée quelconque. Il implique de construire un modèle de données, la taille de ce modèle est un paramètre important.

- sur apprentissage : la taille du modèle est plus gros que l'ensemble des exemples, on a rien appris rien généralisé et on est incapable d'effectuer une prédiction fiable pour une donnée qui ne se trouve pas dans l'ensemble des exemples.
- apprentissage difficile (on n'est plus capable d'effectuer une prédiction fiable pour une donnée particulière), on peut n'avoir appris que les propositions des différentes classes dans l'espace de données : par exemple, 1/3 des données sont bleu et les autres sont rouge, cela sans lien avec la description des données, prédire la classe revient alors à tirer la classe au hasard avec ces proportions un tiers/ deux tiers.
-entre les deux extrêmes, il y a un milieu où le modèle a pris du recul par rapport aux exemples, a su extraire les informations pertinentes du jeu d'exemples pour déterminer la classe de n'importe quelle donnée avec une probabilité élevée de succès.

Le modèle est alors de taille modérée et la probabilité d'erreur de ce modèle est la plus faible que l'on puisse obtenir : on a un modèle optimisant par rapport qualité/prix, i.e. la probabilité d'effectuer une prédiction correcte/cout du modèle. La recherche d'un modèle optimisant ce rapport est l'objectif de l'apprentissage automatique lequel est l'un des outils indispensables pour la réaliser de la fouille de données.

Domaines d'application

Il existe de nombreux domaines d'application de ce problème :

L'attribution de crédit bancaire, la reconnaissance de gènes, la prédiction des sites archéologiques, le diagnostic médical, détection de fraudes fiscales, etc.

4.1 Validation croisée

Pour éviter l'apprentissage par cœur en classification supervisée et pour avoir une estimation non optimiste de l'erreur de classification, il faut utiliser en test des échantillons qui ont servi à l'apprentissage (utiliser une base d'apprentissage et une base de test). Les performances des méthodes de la classification dépendent généralement du nombre d'échantillons d'apprentissage : plus ce nombre est élevé, plus fiable seront les règles de classification. En même temps, il est nécessaire de conserver un nombre significatif d'échantillons de test pour que l'évaluation de ces performances soit significative. La technique de la validation croisée est fréquemment utilisée pour répondre à ces deux besoins: elle consiste à diviser l'ensemble de départ en un certain nombre de sous ensembles de taille égale, chaque sous ensemble étant alors utilisé comme base de test, alors que l'union de tous les autres sous ensembles est utilisée comme base d'apprentissage **Figure I.2**. Soit r ce nombre de sous ensembles, on parle alors de la validation croisée d'ordre r, naturellement r est supérieur ou égal à 2.

Soit D un ensemble de données de n échantillons (la classe est connue), chacun comportant d attributs. L'ensemble D est alors découpé en r sous ensembles disjoints de taille identique [E(n/r) ±1], la représentation statistique de chaque classe étant préservée par rapport à celle de l'ensemble D. À chaque itération, le nombre d'erreurs de classification réalisées est comptabilisé et le taux d'erreur de la classification totale est obtenu en divisant ce nombre d'erreurs par le nombre d'échantillons n.

Etant donné que la classe de chaque échantillon est connue, nous pouvons également produire la matrice de confusion qui recense pour chaque classe réelle, le nombre d'échantillons que le classifieur a classé dans chaque classe ce qui permet de détecter les éventuelles confusions que pourrait faire le système. Si c est le nombre de classe de D, la matrice de confusion est une matrice de taille cxc où la cellule à l'intersection de ligne i et de la colonne j est le nombre d'éléments de classe réelle i classés par le système dans la classe j.

On appelle « Leave One Out » le cas particulier pour lequel le nombre de sous ensemble de validation croisée est égale au nombre de mesures de l'ensemble (r=n).

L'apprentissage se fait alors sur la base complète sauf 1 échantillon et le test est effectué sur l'élément restant. Cela permet, dans le cas d'un faible ensemble de données, de maximiser l'ensemble d'apprentissage mais présente l'inconvénient de se rapprocher de l'apprentissage « par cœur ». La difficulté de cette méthode est de trouver la bonne valeur de d en fonction de la taille de l'ensemble de départ et de la complexité du problème : plus le problème est complexe, plus le système a besoin d'exemples pour apprendre.

-si r est petit, l'ensemble d'apprentissage sera trop faible et le système de classification ne généralisera pas assez, entrainant une augmentation du taux d'erreur. Par contre, il y a moins de phase d'apprentissage à réaliser et l'obtention des résultats est plus rapide. Cette solution est intéressante quand on dispose de beaucoup de mesures.
-si r est grand, l'ensemble de test sera plus faible par rapport à l'ensemble d'apprentissage et on risque le phénomène de sur-apprentissage (cas extrême: « Leave One Out »).

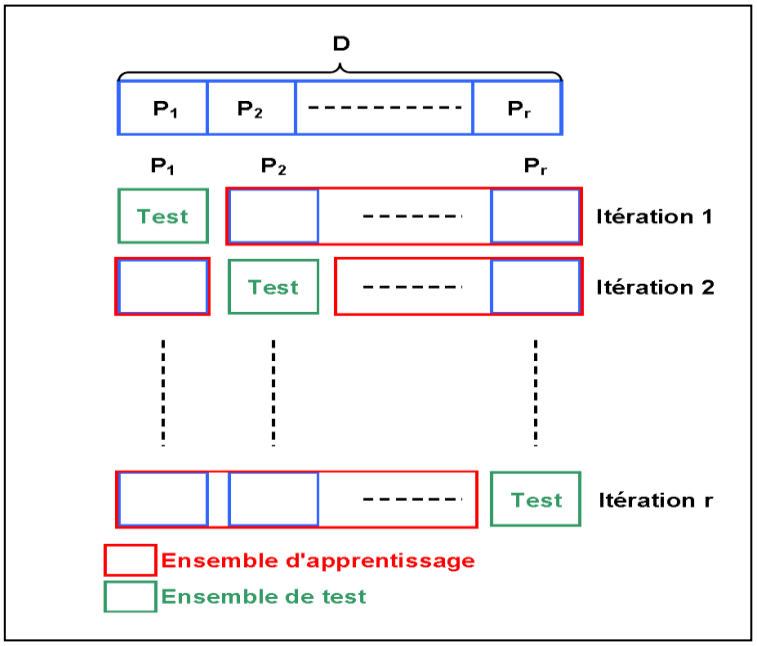

Figure I.2 : Schéma de validation croisée d'ordre c

4.2 Les méthodes de la classification supervisée

4.2.1 Classifieur bayésien naïf

Le classifieur bayésien naïf fournit une approche simple, avec une sémantique claire de représenter, utiliser, et d'apprendre des connaissances probabilistes. La méthode est conçue pour l'utiliser dans des tâches d'apprentissage supervisé dont le but est de prédire la précision de la classe pour les données de test et dans lequel les données d'apprentissage incluent l'information de classes. C'est une forme de réseaux bayésien, le terme naïf est utilisé car le classifieur repose sur des hypothèses d'indépendance des données.

Il suppose que les attributs sont indépendants pour une classe donnée et que aucun des attributs cachés ou latents a une influence sur le processus de prédiction **[John et Langely, 1995],** Malgré ses hypothèses, le classifieur performe généralement bien avec peu de données d'apprentissage. Puisque les données sont considérées indépendantes, seule la variance de chaque caractéristique doit être calculée (non pas la matrice de covariance en entier). Il trouve son fondement théorique dans le théorème de bayes **[bayes, 1763]**.

Etant donné les hypothèses d'indépendance des variables, le classifieur est défini comme suit :

$$classe(f_1,\ldots,f_n) = plusgrand_c\left(p(C=c)\prod_{i=1}^{n} p(F_i = f_i \mid C=c)\right)$$

La fonction plusgrand ne fait que sélectionner la plus grande valeur parmi les probabilités que l'objet appartienne à chacune des classes. On obtient cette probabilité par la multiplication de la probabilité d'obtenir cette classe (parmi les échantillons d'apprentissage) avec la probabilité d'obtenir chaque caractéristique pour cette classe, selon la valeur moyenne et la variance préalablement calculée. La classe qui obtient la plus grande probabilité en fonction des caractéristiques de l'objet présenté est choisie en tant que classe prédite pour l'objet.

4.2.2 Réseaux de Neurones

Les réseaux de neurones sont une très grande famille de techniques d'intelligence artificielle dont une branche très importante permet de faire de la classification. Les premières notions de réseaux de neurones sont apparues dans les années 1950 lorsque les chercheurs ont voulu simuler le fonctionnement du cerveau. Un réseau de neurones est constitué d'un graphe pondéré orienté dont les nœuds symbolisent les neurones. Ces neurones possèdent une fonction d'activation (fonction signe, fonction sigmoïde, …) qui permet d'influencer les autres neurones du réseau. Les liens synaptiques sont les connexions entre les neurones, ces liens propagent l'activité de neurones avec une pondération caractéristique de la connexion (poids synaptique). C'est une technique très employée en classification, notamment dans le Domaine médical **[ye et al, 2002].**

Il existe plusieurs types de réseaux de neurones, la plus courante est le perceptron multicouche. L'interconnexion entre les neurones permet un calcul global complexe qui en classification se traduit par des frontières de décision aux formes complexes. La phase d'entrainement de réseau consiste à régler les poids synaptiques grâce à l'ensemble d'apprentissage.

4.2.3 Séparateurs à Vaste Marge (SVM)

Les SVM sont considérés comme l'une des méthodes robustes et précises **[vapnik, 1995]**, ils disposent d'un fondement théorique simple, les SVM ont été largement utilisés dans la reconnaissance des formes, la classification, la régression et l'estimation de densité.

Le principe de base est de trouver l'hyperplan optimal qui sépare deux classes dans l'espace de description. Cet espace est celui qui maximise la distance entre les deux classes (la marge), le SVM utilise des fonctions de noyau qui, dans un espace augmenté, permettent une séparation optimale des points en différentes catégories. Les données d'apprentissage sont utilisées pour découvrir l'hyperplan qui séparera au mieux les points.
L'idée du SVM est d'utiliser sa fonction de noyau pour reconsidérer le même problème dans un espace de dimension plus élevée. Cet espace est caractérisé par la possibilité d'y

trouver un séparateur linéaire qui permet de classer les points dans les deux groupes appropriés. Le séparateur linéaire peut ensuite être projeté dans l'espace d'origine où il devient habituellement non linéaire. Le critère d'optimisation est la largeur de marge entre les classes (l'espace vide de chaque coté des frontières de décision). La largeur de marge est caractérisée par la distance jusqu'aux échantillons d'entrainement le plus près. Ces échantillons s'appellent vecteurs de supports, ils définissent la fonction discriminante qui permet la classification. Le nombre de vecteurs de support est minimisé en maximisant la marge.

4.2.4 Plus Proche Voisin (PPV)

Le plus proche voisin est l'un des algorithmes les plus populaires utilisés pour le classement dans les différents domaines de la reconnaissance des formes et de la fouille de données, c'est une méthode basée sur la notion de proximité (voisinage) entre exemples et sur le raisonnement à partir de cas similaire pour prendre une décision.

Un objet est classifié selon un vote majoritaire par ses voisins, l'objet obtient la classe qui est la plus commune chez ses K plus proche voisins dans l'espace des caractéristiques. k doit donc un entier positif, généralement petit. On choisit souvent un k impair pour éviter l'égalité dans le vote. La distance utilisée pour le calcul de la proximité des voisins est le plus souvent la distance euclidienne. Les exemples d'apprentissage sont des vecteurs dans un espace multidimensionnel. La phase d'apprentissage consiste simplement à conserver ces données dans un format qui permettra le calcul efficace des distances et la recherche des voisins. Plusieurs algorithmes de KNN et ses variantes ont été présentés dans la littérature afin de diminuer la charge de calcul dédié à la recherche des voisins les plus proches d'un nouvel échantillon **[shakhnarovich et al, 2006]**.

4.2.5 Bagging

La méthode du Bagging a été introduite par **[Breiman, 1996]**. Le mot Bagging est la contraction des mots Bootstrap et Aggregating. Le principe du bagging est d'entraîner un algorithme d'apprentissage sur plusieurs bases d'apprentissage obtenues par tirage avec remise (ou bootstrap) de m exemples d'apprentissage dans l'échantillon d'apprentissage S. Pour chaque tirage b, une hypothèse h_b est obtenue. L'hypothèse finale est basée sur les moyennes des hypothèses obtenues. Son avantage est qu'on améliore la performance des classifieurs instables en calculant la moyenne de leurs réponses.

Ainsi, si les hypothèses h_b calculées pour chaque tirage b ont une variance importante, alors l'hypothèse finale aura une variance réduite. Un classifieur est dit instable si un petit changement dans les données d'apprentissage provoque un gros changement dans le comportement du classifieur. Le but du bagging est d'atténuer l'instabilité inhérente à certains classifieurs.

4.2.6 Boosting

Le boosting est une méthode d'agrégation de classifieurs faibles pour obtenir un classifieur performant **[Schapire, 1990]**. Son principe consiste à assigner un poids égal à chaque exemple d'apprentissage (1/n) (appelé pondération) où n est le nombre d'échantillons. Ce poids indique la difficulté de prédire la classe de cet exemple. L'algorithme s'exécute T fois construisant T classifieurs sur les exemples d'apprentissage pondérés.

Chaque fois qu'on produit un classifieur on change les poids des nouveaux exemples utilisés pour le classifieur suivant, on augmente le poids de ceux dont la classe est mal prédite et on diminue le poids des autres. Ensuite, on calcule l'erreur (e) du modèle sur l'ensemble pondéré. Si e est égale à zéro ou e est supérieure à 0.5 on termine la construction du classifieur. L'idée est de forcer le nouveau classifieur à diminuer l'erreur attendue. Le classifieur final se forme en utilisant un schéma de vote.

4.2.7 Arbres de Décisions

Il est essentiel de produire des procédures de classification compréhensibles par l'utilisateur. Dans l'exemple du diagnostic médical, le médecin doit pouvoir comprendre et interpréter les raisons du diagnostic. Les arbres de décisions répondent à cette contrainte car ils représentent graphiquement un ensemble de règles facilement interprétables. Bien sûr, souvent la taille de l'arbre (plusieurs centaines de nœuds) empêche de saisir la procédure de classification dans son ensemble mais en partant de la description d'un élément, sa classification est toujours compréhensible. L'induction avec des arbres de décision est l'une des formes d'algorithme d'apprentissage les plus simples et pourtant les plus efficaces **[russell et norvig, 2003]**. Les arbres de décisions sont une des techniques les plus populaires de l'apprentissage automatique et de la fouille de données.

Apprentissage des arbres de décisions:

Le schéma général des algorithmes est le suivant :

Algorithme I.1: Algorithme d'apprentissage générique

Entrées : langage de description ; échantillon S
Début
    Initialiser l'arbre à vide ; // la racine est le nœud courant
    Répéter
        Décider si le nœud courant est terminal ;
        Si le nœud est terminal
        Alors affecter une classe ;
        Sinon sélectionner un test et créer le sous-arbre ;
        Passer au nœud suivant non exploré s'il existe ;
    Jusqu'à obtenir un arbre de décision ;
Fin

Le principe consiste à diviser récursivement et le plus efficacement possible les exemples de l'ensemble d'apprentissage par des tests définis à l'aide des attributs jusqu'à ce que l'on obtienne des sous-ensembles d'exemples ne contenant (presque) que des exemples appartenant tous à une même classe.

Dans toutes les méthodes, on trouve les trois opérateurs suivants:

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

Les méthodes vont différer par les choix effectués pour ces différents opérateurs, c'est-à-dire sur le choix d'un test (par exemple, utilisation du gain et de la fonction entropie) et le critère d'arrêt (quand arrêter la croissance de l'arbre, soit quand décider si un nœud est terminal).

Avec un tel algorithme, on peut calculer un arbre de décision dont l'erreur apparente est faible, voire nulle. Un arbre de décision parfait est un arbre de décision tel que tous les exemples de l'ensemble d'apprentissage soient correctement classifiés. Un tel arbre n'existe pas toujours (s'il existe deux exemples tels que à deux descriptions identiques correspondent deux classes différentes). L'objectif est de construire un arbre d'erreur de classification la plus petite possible. Mais malheureusement :

• l'erreur apparente est une vision très optimiste de l'erreur réelle,

• trouver un arbre de décision d'erreur apparente minimale est, en général, un problème NP-complet.

Exemple de construction d'un arbre de décision

Le jeu de données utilisé: un ensemble de jours (un jour = un exemple), chacun caractérisé par un numéro et ses conditions météorologiques (température, humidité de l'air, force du vent, ciel), l'attribut cible étant « jouer au tennis ? », dont les valeurs possibles sont oui et non. Une fois l'arbre de décision construit, on pourra classer une nouvelle donnée pour savoir si on joue ou non ce jour-là (**Tableau I.2**).

La **Figure I.3** présente un exemple d'arbre de décision.

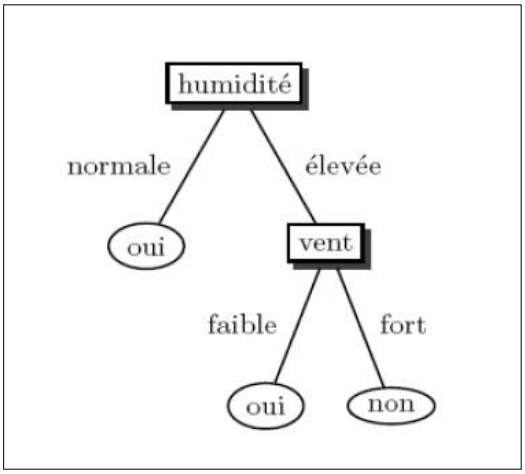

Figure I.3: Un exemple d'arbre de décision (jouer au tennis)

La donnée 3 du **Tableau I.2** est prédite comme « oui » car son attribut « humidité » vaut « Elevée », donc on suit la branche droite partant de la racine et on atteint le nœud « Vent », et l'attribut « Vent » vaut « Faible », ce qui nous fait suivre la branche gauche de ce nœud et atteindre une feuille étiquetée « oui » (celle du milieu sur la figure). De même, la donnée 2 sera prédite comme de classe « non » et la donnée 5 sera prédite de classe « oui » en s'aiguillant directement depuis la racine vers la feuille « oui » à la gauche de la figure.

Jour	Ciel	Température	Humidité	Vent	Jouer au tennis ?
1	Ensoleillé	Chaude	Elevée	Faible	Non
2	Ensoleillé	Chaude	Elevée	Fort	Non
3	Couvert	Chaude	Elevée	Faible	Oui
4	Pluie	Tiède	Elevée	Faible	Oui
5	Pluie	Fraîche	Normale	Faible	Oui
6	Pluie	Fraîche	Normale	Fort	Non
7	Couvert	Fraîche	Normale	Fort	Oui
8	Ensoleillé	Tiède	Elevée	Faible	Non
9	Ensoleillé	Fraîche	Normale	Faible	Oui
10	Pluie	Tiède	Normale	Faible	Oui
11	Ensoleillé	Tiède	Normale	Fort	Oui
12	Couvert	Tiède	Elevée	Fort	Oui
13	Couvert	Chaude	Normale	Faible	Oui
14	Pluie	Tiède	Elevée	Fort	Non

Tableau I.2: Jeu de données (jouer au tennis)

Elagage

Un des problèmes connus lors de la phase de construction et de classement est que la taille de l'arbre grandit de manière linéaire avec la taille de la base d'apprentissage. De plus, les arbres de décisions complexes peuvent avoir des taux d'erreur très élevés à cause de sur-apprentissage qui peut se produire lorsque l'ensemble d'apprentissage contient des données bruitées ou qu'il ne contient pas certains exemples importants, ou encore lorsque les exemples sont trop spécifiques. L'élagage est l'une des solutions pour réduire ces taux d'erreurs en simplifiant l'arbre par suppression de quelques branches.
Il existe plusieurs techniques d'élagage **[Mingers, 1989]** pour éviter le sur-apprentissage.

Les algorithmes d'apprentissage

Les principaux algorithmes de construction d'un arbre de décision sont : ID3 **[Quinlan, 1986]**, C4.5 **[Quinlan, 1993]** et CART **[Breimann et al, 1984]**.
Nous traitons dans la suite (chapitre IV) plus particulièrement l'algorithme C4.5, qui est un algorithme performant pour générer un arbre de décision, il s'agit d'une amélioration de l'algorithme ID3.

5. Conclusion

Nous avons présenté une introduction à un domaine de recherche en plein essor « l'extraction de connaissances à partir de données ». En effet, les quantités de données collectées dans divers domaines deviennent de plus en plus importantes et leur analyse de plus en plus demandée. Nous avons cité le processus d'extraction de connaissances et nous avons insisté sur une étape importante qui est la fouille de données. Ensuite, nous sommes focalisés sur une tâche de cette étape de fouille de données, qui est la tâche de la classification supervisée.

Pour assurer une réduction de dimension du problème en améliorant la précision de la prédiction et de la compréhension du classifieur, le chapitre suivant sera consacré à une étape de prétraitement « la sélection d'attributs » qui joue un rôle important dans la fouille de données.

Chapitre II:
La Sélection d'Attributs

1.Introduction

Dans le DataMining (la fouille de données), la taille des données manipulées est généralement importante, de ce fait l'exploitation de ces données se révèle difficile en pratique. Les données qui sont souvent traitées pour effectuer des tâches de classification ont connu une croissance exponentielle. Dernièrement pour le nombre d'attributs, la classification est passée du traitement de quelques douzaines d'attributs vers des centaines voire des milliers d'attributs.

La sélection d'attributs est l'une des techniques importantes et fréquentes utilisée dans le prétraitement de données pour la fouille de données.

Dans ce chapitre, nous allons présenter la sélection d'attributs qui peut être décrite par un schéma général, ses objectifs, ainsi que les différentes méthodes proposées dans la littérature.

2.Objectifs de la sélection d'attributs

Si l'amélioration des performances est l'un des principaux atouts de la sélection d'attributs, d'autres sont également importants suivant les caractéristiques du problème que l'on a traité

- cout d'acquisition des attributs réduit: Diminution de la consommation de ressources par la mesure du sous ensemble de variables plus petit.
- mise en exergue des variables pertinentes pour la classification pour augmenter la précision : Utiliser un sous ensemble plus petit permet d'améliorer la classification si on élimine les attributs non pertinents cela permet aussi une meilleure compréhension des phénomènes étudiés.
- facilité d'interprétation des modèles moins complexes, plus simple à interpréter: il sera facile d'analyser un modèle contenant moins d'attributs. Des études ont montré que la sélection des attributs en amont de l'induction d'arbres de décision permettait non seulement d'améliorer les performances en classification **[Perner, 2001]**, mais également de réduire la taille des arbres induits **[Seban et Nock , 2001].**
- durée d'apprentissage réduite : L'algorithme d'apprentissage est plus rapide si l'espace de recherche est restreint par la réduction du nombre d'attributs

Les auteurs **[Langley et Ibu, 1993]** ont par exemple montré que pour l'algorithme du plus proche voisin, le nombre d'exemples nécessaires pour atteindre une erreur en généralisation donnée croit de façon exponentielle par rapport au nombre d'attributs non pertinents. Une fois que les meilleurs attributs sont identifiés, le temps d'apprentissage et d'exécution sont réduits et en conséquence l'apprentissage est moins couteux.

- des petits sous ensemble d'attributs permettent une meilleure généralisation des données en évitant le sur-apprentissage.

Le processus de sélection d'attributs pour la classification peut être considéré comme un problème de recherche dans un espace d'états, où chaque état peut être représenté comme un vecteur d'une taille égale au nombre des attributs dans le problème et chaque élément vecteur peut prendre la valeur 1 si l'attribut correspond est sélectionné et 0 sinon. La taille de l'espace de recherche du sous ensembles d'attributs d est $O(2^d)$, la **Figure II.1** représente les 16 états dans le cas de d=4 reliés par le fait d'inclure ou d'exclure l'un des attributs.

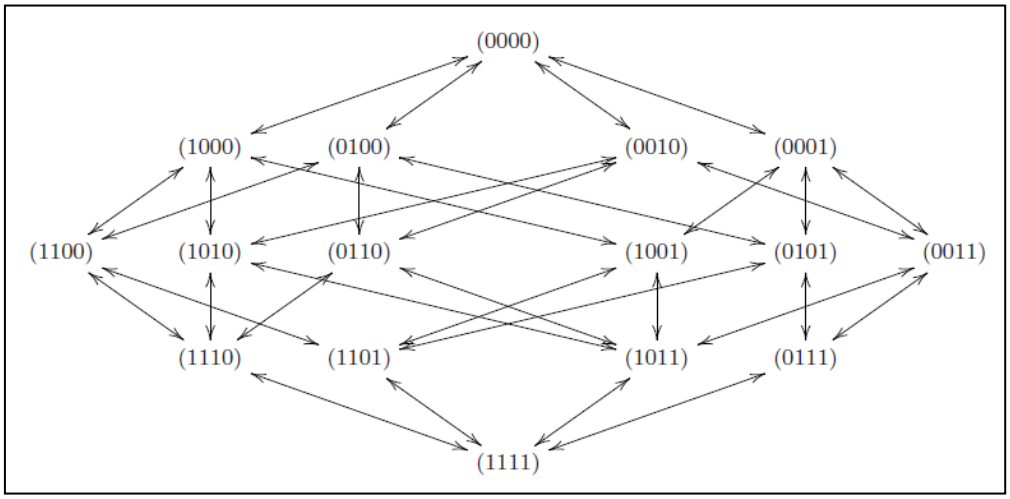

Figure II.1: Espace d'attributs de 4 dimensions **[Jourdan, 2003]**.

Ne retenir que certains attributs impose de faire des choix, de décider quels attributs conservés en fonction de certains critères, quels que soient les objectifs exacts que l'on cherche à atteindre, il semble naturel de privilégier les attributs les plus pertinents au regard de la tâche finale à effectuer. L'étude du comportement de divers classifieurs sur des bases de données artificielles a par ailleurs permis de mettre en évidence la dégradation des performances de ces classifieurs en présence d'attributs non pertinents **[Kohavi et John, 1997].** La notion de la pertinence joue un rôle central dans la sélection d'attributs.

En présence de centaines voire de milliers d'attributs il y a beaucoup de chances pour que des attributs soient corrélés et expriment des informations similaires, on dira alors qu'ils sont redondants, d'un autre coté, les attributs qui fournissent le plus d'information pour la classification seront dits pertinents.

L'objectif de la sélection est donc de trouver un sous ensemble optimal d'attributs qui ait les propriétés suivantes : il doit être composé d'attributs pertinents et il doit chercher à éviter les attributs redondants. De plus cet ensemble doit permettre de satisfaire au mieux l'objectif fixé c'est à dire la précision de l'apprentissage, la rapidité de l'apprentissage ou bien encore l'explicabilité du classifieur proposé **([dash et Liu, 1997]**, **[Kohavi et john, 1997]**, **[Liu et Motoda,2008]**).

3.Pertinence d'un attribut

Avant de définir le processus de sélection d'attributs, il est nécessaire de connaitre la différence entre les types des attributs puisqu'il nous permet une meilleure compréhension de la performance d'une stratégie de la sélection d'attributs.

La notion de pertinence, malgré les efforts déployés par les différents auteurs, n'a pas de définition formelle qui est acceptée par tous **[Bell et Wang, 2000]**. Parce que le concept de la pertinence est associé à une tâche particulière dans l'apprentissage automatique **[Blum et Langely, 1997].**

Une proposition importante est donnée par Michalski **[Michalski, 1983]**, qui distingue trois situations différentes concernant la pertinence des attributs utilisée en apprentissage automatique :

- **pertinence complète :** si les attributs sont directement pertinents pour le problème, dans ce cas tous les attributs interviennent dans la tâche de la classification.
- **pertinence partielle :** lorsque des objets ont des attributs non pertinents ou redondants, par conséquent, pour mener à bien la tâche d'apprentissage, il devra sélectionner les attributs les plus pertinents.
- **pertinence indirecte :** s'il ya des attributs qui ne sont pas pertinents, mais les nouveaux attributs générés par eux sont pertinents.

Il existe dans la littérature plusieurs définitions de la pertinence d'un attribut. Celles-ci dépendent de la nature des données.

- Gennari et Fisher **[Gennari et Fisher, 1989]:** Ils définissent les attributs pertinents comme ceux dont les valeurs des attributs changent systématiquement en fonction de l'appartenance de données à telle ou telle catégorie, qui peut être exprimé comme suit : un attribut F_i est pertinent si et seulement s'il existe une valeur f_i de cet attribut et une valeur de classe C, c, vérifiant $p(F_i = f_i)>0$ et tel que :

$p(C=c \backslash F_i = f_i) \neq p(C=c)$ avec $p(F_i = f_i)$ qui exprime le fait que les données contiennent au moins une instance ayant la valeur f_i pour l'attribut F_i , est différente de la probabilité de prédire C sans connaitre la valeur de F_i .

- une définition plus large de **[Kohavi et John, 1997]**, ils ont défini les attributs pertinents comme ceux dont les valeurs varient systématiquement avec les valeurs de classe :

Un attribut F_i est pertinent si la connaissance de sa valeur change les probabilités sur les valeurs de la classe C, mais cette définition peut être précisée pour distinguer les attributs fortement pertinents et les attributs faiblement pertinents grâce aux définitions suivantes:

Soit F un ensemble complet d'attributs, F_i un attribut et $S_i=F-\{F_i\}$, on suppose que l'on travaille avec un espace probabilisé où la probabilité est notée p.

$p(C \backslash S)$ est la probabilité de la classe C connaissant les attributs de l'ensemble S. Donc :

Un attribut F_i est fortement pertinent si et seulement si :

$p(C\backslash F_i, S_i) \neq p(C\backslash S_i)$ ……..……(1)

Un attribut F_i est faiblement pertinent si et seulement si :

$p(C\backslash F_i, S_i) = p(C\backslash S_i)$ et s'il existe $S_i' \subset S_i$ tel que $p(C\backslash F_i, S'_i) \neq p(C\backslash S'_i)$ ..……..…….. (2)

Un attribut F_i est non pertinent si et seulement si :

$p(C\backslash F_i, S_i) = p(C\backslash S_i)$ et quelque soit $S_i' \subset S_i$

tel que $p(C\backslash F_i, S'_i) = p(C\backslash S'_i)$ ……..……(3)

Les attributs fortement pertinents sont indispensables et ils devraient figurer dans tout sous ensemble optimal sélectionné, car leur absence devrait conduire à un défaut de reconnaissance de la fonction cible.

La faible pertinence suggère que l'attribut n'est pas toujours important, mais il peut devenir nécessaire pour un sous ensemble optimal dans certaines conditions.

La non pertinence se définit simplement par rapport à (1), (2) et indique qu'un attribut n'est pas du tout nécessaire dans un sous ensemble optimal d'attributs.

- **[Almullin et Diettrich, 1991]** (tous les attributs sont booléens)

  * un attribut F_i est dit pertinent pour un concept C si F_i apparait dans chaque formule booléenne qui représente C, il est dit non pertinent sinon

  * un attribut F_i est pertinent, s'il existe deux instances (x_1, y_1) et (x_2, y_2) appartenant à deux classes différentes ($y_1 \neq y_2$). Tel que la valeur F_i est différente pour les deux instances et les valeurs des autres attributs sont identiques.

- Les auteurs **[Dash et liu, 1998]** définissent un attribut non pertinent comme celui n'affectant pas la structure fondamentale des données et un attribut redondant comme celui n'apportent de rien de nouveau pour décrire la structure fondamentale des données.

  * le processus de la sélection d'attributs comporte plusieurs étapes et son espace de recherche peut être modélisé sous forme d'un treillis de gallois.

- Les auteurs **[Liu et Yu, 2005]** définissent la pertinence dans le cas d'attributs et de fonctions booléennes et en supposant que les données sont non bruitées.

4.Redondance d'attributs

- Elle est exprimée en termes de corrélation entre les attributs. Deux attributs sont redondants (entre eux) si leurs valeurs sont complètement corrélées.

Cette définition ne se généralise pas directement pour un sous ensemble d'attributs.

On trouve dans **[Koller et Sahami, 1996]** une définition formelle de la redondance qui permet de concevoir une approche pour identifier et éliminer les attributs redondants. Cette formalisation repose sur la notion de couverture de Markov (markov Blanket) d'un attribut qui permet d'identifier les attributs non pertinents et redondants.

- les études empiriques sur des bases de données artificielles, ainsi que des exemples données par **[Guyon et Elisseeff, 2003]** ont mis en avant les effets néfastes sur les performances d'un classifieur qui pouvaient causer des attributs non pertinents, mais aussi des attributs qu'ils qualifient de redondants. Diverses définitions de la redondance ont été données dans la littérature et souvent basées sur le concept de corrélation entre attributs.

Nous ne donnons ici que la définition due aux auteurs Yu et Liu **[YU et Liu ,2004]**. Celle-ci s'appuie sur le concept de couverture de Markov introduit par les auteurs Koller et Sahami **[Koller et Sahami, 1996]**.

Définition: soit F l'ensemble total d'attributs et C la classe.

Soit F_i un attribut, et M_i un sous ensemble d'attributs qui ne contient pas F_i , c'est-à-dire : $M_i \subset F$ et $F_i \notin M_i$.

M_i est une couverture de Markov pour F_i Si et seulement si :

$P(F-M_i-\{F_i\}, C\backslash F_i, M_i) = p(F-M_i-\{F_i\}, C\backslash M_i)$

La définition de couverture de Markov impose que M_i subsume non seulement l'information que F_i apporte sur C mais aussi l'information qu'il apporte sur tous les autres attributs.

Les auteurs Koller et Sahami **[Koller et Sahami, 1996]** ont montré que un sous ensemble d'attributs optimal peut être obtenu par une procédure d'élimination descendante, appelée Filtrage par Couverture de Markov (Markov Blanket Filtering) :

Soit G l'ensemble d'attributs courant (G=F au départ).

A chaque étape de la procédure, s'il existe une couverture de Markov pour l'attribut de F_i dans l'ensemble G courant, F_i est enlevé de G.

Ce processus garantit qu'un attribut enlevé dans une étape précédente peut trouver une couverture de Markov dans une étape postérieure.

Les attributs fortement pertinents ne peuvent trouver aucune couverture de Markov. Par contre les attributs non pertinents doivent être enlevés de toute façon, et il n'est pas donc nécessaire de s'y intéresser dans la définition des attributs redondants cela conduit à la définition suivante de la redondance **[YU et Liu ,2004]:**

Soit G l'ensemble d'attributs courant, un attribut F_i est redondant et par conséquent peut être enlevé de G si et seulement s'il est faiblement pertinent et qu'il possède une couverture de Markov dans G.

A fin de synthétiser les différentes notions de pertinence et redondance que l'on vient de présenter, une catégorisation des attributs est présentée dans la **Figure II.2**. Dans ce schéma un ensemble initial d'attributs peut être partitionné en quatre catégories :

A = {attributs non pertinents}.

B_1 = {attributs faiblement pertinents et redondants}.

B_2 = {attributs faiblement pertinents et non redondants}.

C = {attributs fortement pertinents}.

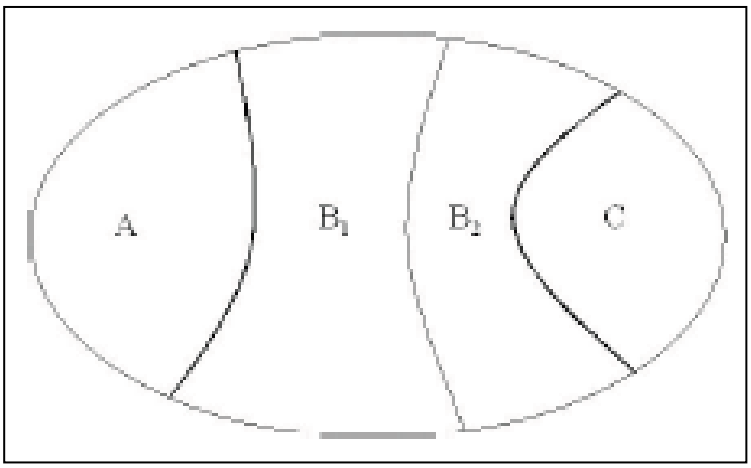

Figure II.2 : Catégorisation possible des attributs.

Un sous ensemble d'attributs optimal contient essentiellement tous les attributs des parties B_2 et C. Il est important de préciser que pour un sous ensemble initial donné, le processus de Filtrage par Couverture de Markov peut conduire à différents découpages donnant les parties B_1 et B_2 (qui sont disjoint). Ces formalisations sont intéressantes pour comprendre les notions de redondance et de pertinence. Néanmoins les définitions probabilistes qu'on a vu ne permettent pas de proposer un processus de sélection d'attribut applicable sur des données de grande dimension, par la suite on va présenter ce processus du point de vue plus opérationnel.

5.Sélection d'Attributs

- La sélection d'attribut est une technique permettant de choisir les caractéristiques, variables ou mesures les plus intéressantes pertinentes, à un système donné, pour la réalisation de la tâche pour laquelle il a été conçu. Cette phase est généralement un module important d'un système complexe.
- La sélection d'attributs est un processus qui sélectionne un sous ensemble à partir de l'ensemble original d'attributs que l'on peut mesurer par des critères d'évaluation si un sous ensemble est optimal. Trouver un sous ensemble optimal d'attribut est habituellement difficile **[Kohavi et John, 1997]** et de nombreux problèmes reliés à la sélection d'attributs ont été montrés pour être NP-difficile **[Blum et Rivest, 1992].**

Le problème de sélection d'un sous ensemble d'attributs peut être vu comme une recherche dans un espace d'hypothèses (ensemble de solutions possibles) **[Blum et Langely, 1997]**.

Etant donné un ensemble initial X de n attributs, la sélection d'un bon sous ensemble d'attributs nécessite d'examiner potentiellement 2^n-1 sous ensembles possibles.

La qualité d'un sous ensemble sélectionné est évaluée selon un critère de performance que l'on notera J.

Dans le cas d'un problème de classification supervisée, ce critère est très souvent la précision d'un classifieur construit à partir de l'ensemble d'attributs sélectionné.

La recherche d'un sous ensemble d'attributs, optimal pour le critère J que l'on s'est donné, est alors un problème NP-difficile **[Cotta et Moscato, 2003].**

Plusieurs approches peuvent être envisagées pour contourner cette difficulté **[molina et al, 2002] :**

Soit X un ensemble d'attributs

Soit J une mesure d'évaluation qui attribue à tout sous ensemble de X un score :

$J: \bar{x} \subset X \rightarrow R$

J doit être optimisé (maximiser ou minimiser suivant la nature de J),

On supposera dans la suite que J doit être maximisée.

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

- Nombre d'attributs fixé : pour un nombre m fixé, avec m<n, on cherche à trouver $\bar{x} \subset X$ tel que $J: |\bar{x}| = m$ et que $J(\bar{x})$ soit maximum.
- Le seuil de performance fixé : on se donne une valeur seuil J_{opt} , c'est-à-dire , le minimum acceptable pour J , et on cherche à trouver $J: \bar{x} \subset X$ tel que le cardinal de X soit le plus petit possible que $J(\bar{x}) \geq J_{opt}$.
- compromis de performance et nombre d'attributs. Trouver un compromis entre le fait de minimiser le nombre d'attributs $|\bar{x}|$ et le fait d'optimiser $J(\bar{x})$.

* La première stratégie consiste à passer d'un ensemble initial de n attributs à un sous ensemble de m attributs sélectionnés qui donne une performance au moins égale ou meilleure à celle obtenue avec l'ensemble complet.

Cela suppose qu'on connait le nombre optimal d'attributs à sélectionner.

La première difficulté est de définir a priori un nombre m. Ce nombre dépend de la taille, de la quantité et de la qualité de l'information disponible.

Si m est fixé, une deuxième difficulté consiste alors à examiner toutes les combinaisons possibles. La recherche d'un sous ensemble de m attributs parmi n donne un nombre de combinaisons $\binom{m}{n}$

A titre indicatif, le nombre de combinaisons sans répétition c_{49}^{6} vaut 13983816 (il s'agit là du nombre de combinaisons possibles pour un tirage du loto).

La croissance exponentielle de $\binom{m}{n}$ rend la recherche très couteuse et une exploration exhaustive n'est pas envisageable, même pour des valeurs modérées de m et n.

* dans le deuxième cas on fixe un seuil de performance J_{opt} à respecter.

On cherche donc un sous ensemble de cardinalité minimale dont la performance soit meilleure que J_{opt} qui peut être une valeur observée avec une certaine représentation du problème et on se fixe l'objectif de trouver une représentation utilisant un nombre minimum d'attributs mais garantissant une performance au moins égale à J_{opt}. Par exemple les méthodes évolutionnaires (Algorithme génétique) peuvent être utilisées pour cet objectif.

* Dans le troisième cas, on considère un problème d'optimisation bi-critère où l'on cherche à la fois de maximiser la fonction J_{opt} tout en minimisant le nombre d'attributs retenus.

Exemple : sélection d'attributs de puces de l'ADN, il faut considérer le bon compromis entre la performance et la taille de sous ensemble final en prenant les critères précédemment cités.

6.Schéma général de la sélection d'attributs

Les différentes méthodes proposées dans la littérature pour la sélection d'attributs peuvent être décrites par un schéma général **[Dash et Liu, 1997]** dans lequel on trouve les éléments clés suivants (**Figure II.3**):

1) une procédure de génération de sous ensembles candidats qui détermine l'exploration de l'espace de recherche.
2) Une fonction d'évaluation donnant la qualité de sous ensembles candidats.
3) Une condition d'arrêt.
4) Un processus de validation pour vérifier si l'objectif souhaité est atteint.

Par la suite on va détailler les étapes importantes de ce schéma.

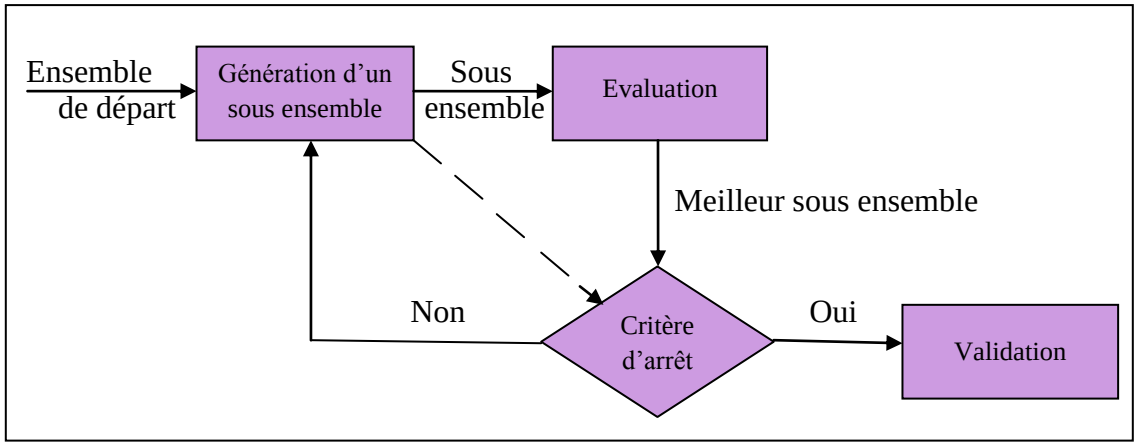

Figure II.3 : Processus de sélection d'attributs avec validation

6.1 Génération de sous ensemble

C'est un processus essentiel de recherche heuristique, avec chaque état dans l'espace de recherche spécifie un ensemble candidat pour l'évaluation.

La nature de ce processus est déterminée par les deux étapes suivantes **[Liu et Yu, 2005]**

6.1.1 Point de départ: est le point dans l'espace du sous ensemble des attributs, à partir duquel on commence la recherche, et ce même point va affecter la direction de recherche. Pour n attributs, l'espace de recherche contient 2^n-1 ensembles possibles.

- **Option Forward** : on commence la recherche par un ensemble vide, et on ajoute successivement des attributs.
- **Option Backward :** on commence par l'ensemble complet (tous les attributs, et on supprime successivement des attributs).
- **Option Stepwise (bi-directional):** on commence avec les deux extrémités, puis on ajoute et on supprime des attributs simultanément.

La recherche peut également commencer par un sous ensemble choisi au hasard afin d'éviter d'être piégé par des optima locaux.

6.1.2 Stratégie de recherche

C'est une procédure qui permet d'explorer l'espace des combinaisons des attributs. Pour n attributs, l'espace contient 2^n-1 sous ensemble d'attributs possibles. Cet espace de recherche est exponentiellement prohibitif pour la recherche exhaustive.

Par conséquent, différentes stratégies ont été explorées : recherche complète, séquentielle et aléatoire.

6.1.2.1 La recherche complète

Elle garantie le résultat optimal par rapport au critère d'évaluation utilisée.

La recherche ne doit pas être exhaustive pour qu'elle soit complète. Différentes heuristiques et fonction peuvent être utilisées pour réduire l'espace de recherche sans avoir de risques de perdre les résultats optimaux:

- Branch and Bound, **[Narendra et Fukunag, 1977]**
- Beam Search, **[Doak 1992], …**

6.1.2.2 La recherche séquentielle

Elle ne garantie pas le résultat optimal, elle consiste à rajouter ou éliminer itérativement des attributs. Il existe plusieurs variations de l'approche de Greedy hill Climbing (méthode de descente) comme **[Liu et Motoda, 1998]**

- Forward Selection.
- Sequential Backward Elimination
- Bidirectional Selection

Plusieurs stratégies de recherches heuristiques peuvent être envisagées.

Dans une stratégie de recherche séquentielle ascendante (Forward Selection), l'ensemble de départ est l'ensemble vide et chaque étape de génération de sous ensemble considère tous les attributs qui ne sont pas encore sélectionnés pour retenir le meilleur entre eux, et l'ajouter à l'ensemble courant. De la même façon, une recherche descendante part de l'ensemble complet d'attributs et retire à chaque étape un attribut.

Ces méthodes de nature gloutonne ne font pas de retour arrière (et ne sont pas alors complètes). Leur principal avantage est qu'elles sont de complexité quadratique. Sur des données de taille modérée, elles peuvent être appliquées efficacement.

Toutes les approches ajoutent ou suppriment des attributs un à un (one at a time).
Une autre alternative est d'ajouter (ou supprimer) p attributs en une seule étape et de supprimer (ou ajouter) q attributs à l'étape suivante (p>q) **[Doak, 1992]**
Les algorithmes avec la recherche séquentielle sont simples à mettre en œuvre et rapide à produire des résultats dont l'ordre de l'espace est généralement $O(N^2)$ ou moins.

6.1.2.3 La recherche aléatoire

Elle commence par un sous ensemble choisi au hasard.
Il y a deux chemins :
L'un à suivre la recherche séquentielle exemple : Random Start, Méthode de descente et Recuit Simulé **[Doak, 1992]**
L'autre, chaque ensemble d'attributs est généré de manière complètement aléatoire (c'est-à-dire le sous ensemble courant n'est pas issu d'une augmentation ou diminution d'attributs du sous ensemble précédent).
Exemple Algorithme LasVigas **[Brassard et Bratly, 1996].**
Pour toutes les approches, l'utilisation de l'aléatoire aide pour éviter des optima locaux dans l'espace de recherche et le sous ensemble optimal sélectionné dépend de ressources disponibles et le choix des paramètres.

La **Figure II.4** illustre la classification des méthodes de sélection d'attributs selon leur stratégie de recherche **[Dash et Liu, 1997].**
- La stratégie de recherche complète est divisée en « exhaustive » et « «non exhaustive ».
  - La catégorie exhaustive : une méthode peut évaluer tous les sous ensembles 2^n où elle peut faire la recherche 'Breadth first' pour arrêter la recherche dès qu'un sous ensemble optimal est trouvé.
  - La catégorie non exhaustive : on trouve différents techniques de recherches comme: Branch and Bound , Best First et Beam search.
- La stratégie de générations heuristiques : elle est divisée en sélection « Forward », sélection « Backward » et sélection hybride Forward/bacward « Instance Based ».
- De même les stratégies de génération aléatoire sont divisées en deux types:
  - **Type01 :** la probabilité de génération d'un sous ensemble reste constante et elle est la même pour tous les sous ensembles.
  - **Type02 :** la probabilité de génération d'un sous ensemble se change lors de l'exécution de programme.

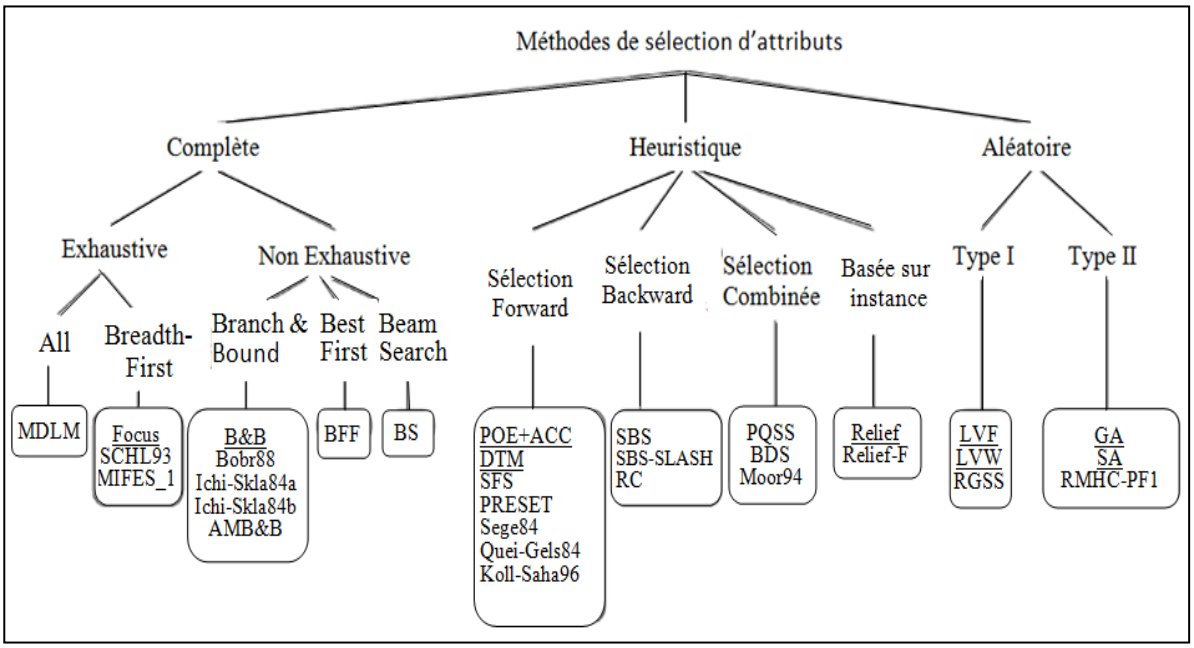

Figure II.4 : Résumé des méthodes de sélection d'attributs.

6.2 Evaluation de sous ensemble

Chaque nouveau sous ensemble généré a besoin d'être évalué par des critères d'évaluation. Un meilleur sous ensemble est déterminé par certains critères (peut être optimal pour un critère donné et pas pour d'autres).

Les critères d'évaluation sont regroupés en deux grandes classes d'algorithmes: celles fondées uniquement sur les données et leurs caractéristiques (filter) et celles qui utilisent l'algorithme d'apprentissage pour évaluer les sous-ensembles générés (wrapper).

6.2.1 Les critères indépendants

Ils sont utilisés dans les algorithmes de type Filter. Une méthode filtrante consiste à évaluer les meilleurs attributs ou les sous ensembles des attributs par l'exploitation des caractéristiques intrinsèques de données d'apprentissage sans impliquer des algorithmes **[Liu et Yu, 2005],** les plus populaires sont :

✓ **les mesures de distance** (mesure de séparabilité ou divergence ou discrimination)

Pour deux classes de problèmes, un attribut x est préféré à un autre attribut y si x induit la plus grande différence dans la distribution de probabilités conditionnelles des deux classes que y, en essayant de trouver l'attribut qui peut séparer les deux classes aussi loin que possible.

x et y sont indistincts si la différence est égale à zéro

Les exemples de cette mesure sont : la distance euclidienne, la distance de Mahalanobis, la distance de Bhattacharya et le critère de Fisher.

✓ **Les mesures d'information**

Ces mesures déterminent typiquement le gain d'information de l'attribut x.

Le gain d'information de x est défini comme la différence entre l'incertitude antérieure et postérieure prévue en utilisant cet attribut.

L'attribut x est préféré de l'attribut y si le gain de x est plus grand que celui de y, exemple : mesure entropie de Shannon.

✓ **Les mesures de dépendances (mesure de corrélation ou de similitude)**

Elles mesurent la capacité de prévoir la valeur d'une variable à partir des valeurs des autres. Le coefficient est la mesure de dépendance classique et peut être utilisé pour trouver la corrélation entre l'attribut et la classe C.

Si la corrélation de x avec C est élevé que celle de y avec C, l'attribut x est préféré de l'attribut y.

Une variation légère de ceci est de déterminer la dépendance de l'attribut sur d'autres attributs, cette valeur indique le degré de redondance de l'attribut.

Toutes les fonctions d'évaluation basées sur les mesures de dépendances peuvent être divisées entre mesure de distance et mesure de l'information. Ces mesures de dépendances sont encore conservées dans une catégorie distincte parce qu'elles représentent un point de vue différent.

Pour la sélection d'attribut dans la classification, on cherche comment un attribut est fortement lié à une classe.

Pour le clustering, l'association entre deux attributs aléatoires mesure la similarité entre les deux. Exemple : coefficient de corrélation, information mutuelle.

✓ **Les mesures de consistance**

Caractéristiquement différentes des autres méthodes à cause de leurs fortes relations avec la classe d'information dans la sélection du sous ensemble d'attributs. Ces mesures essayent de trouver le nombre minimal d'attributs qui séparent les classes comme la consistance de l'ensemble complet d'attributs. L'inconsistance est définie comme deux instances ayant les mêmes valeurs d'attributs mais avec différentes étiquettes de classes.

Tableau II.1 montre une comparaison des différentes fonctions d'évaluation, indépendamment du type de la procédure de génération utilisée.

Les différents paramètres utilisés pour les comparaisons sont :

- Généralité : comment le sous ensemble approprié est sélectionné pour différents classifieurs (pas pour un seul classifieur).
- Complexité en temps : le temps pris pour la classification de sous ensemble d'attributs.
- La précision : est la prédiction utilisant le sous ensemble sélectionné.

Le signe – signifie qu'il n'y a pas de conclusion correspond à la fonction d'évaluation.

à l'exception du « taux d'erreur de classification », la précision de toutes les autres fonctions d'évaluation dépend de l'ensemble de données et de classifieurs utilisés (pour la classification après la sélection d'attributs).

Les résultats de ce tableau montrent une tendance non surprenante (la précision est très élevée si la complexité en temps est élevée). Ce tableau nous indique également ce que la mesure devrait être utilisée dans des circonstances différentes.

Exemple : avec des contraintes de temps

Etant donné un ensemble de classifieurs pour choisir un d'entre eux, le taux d'erreur du classifieur ne doit pas être choisi comme une fonction d'évaluation.

Fonction d'évaluation	Généralité	Complexité en temps	La précision
Mesure de Distance	Oui	Faible	_
Mesure d'Information	Oui	Faible	_
Mesure de dépendance	Oui	Faible	_
Mesure de Consistance	Oui	Modérée	_
Taux d'erreur du classifieur	Non	élevée	Très élevée

Tableau II.1: Comparaison des fonctions d'évaluation.

6.2.2 Les critères de dépendances

Ils sont utilisés dans le modèle Wrapper qui requiert un algorithme de recherche prédéterminée pour la sélection d'attributs.

Les Wrappers ont été introduits plus récemment par **[John et al, 1994],** le principe est de générer des sous ensembles candidats et de les évaluer grâce à l'algorithme d'induction.

Exemple :

Dans la classification : la prédiction de la précision est largement utilisée comme une mesure primaire. Elle est peut être utilisée comme un critère de dépendance pour la sélection d'attributs. Comme un classifieur utilise les attributs choisis pour prédire les classes des instances non connues, la précision est généralement élevée. Mais ce calcul est assez couteux pour estimer la précision de chaque sous ensemble.

Dans le clustering :

Un modèle w de sélection d'attributs tente d'évaluer le meilleur sous ensemble d'attributs par la qualité de clusters résultante de l'application de l'algorithme de clustering sur le sous ensemble sélectionné.

Il existe plusieurs critères heuristiques pour estimer la qualité de résultat de clustering (Cluster Compactness, Scatter Separability , Maximum Likelihood).

- ✓ **Les mesures de taux d'erreurs des classifieurs**

Les méthodes qui utilisent ce type de la fonction d'évaluation sont appelées les méthodes Wrappers (le classifieur est une fonction d'évaluation).

Pour la prédiction des classes des instances non connues, le niveau de précision est très élevé bien que très couteux en calcul.

6.2.3 Les méthodes hybrides

Elles utilisent les mesures d'indépendances (pour décider le sous ensemble pour une cardinalité donnée) et les algorithmes d'induction (pour choisir le meilleur sous ensemble final à travers différentes cardinalités) pour évaluer les sous ensembles d'attributs.

6.3 Critères d'arrêt

L'initialisation et le critère d'arrêt définissent les bornes de la recherche.

Le critère d'arrêt peut être un temps de calcul fixé, un nombre d'itérations fixé, l'absence de gain de performance par rapport aux solutions déjà trouvées ou encore le fait que les sous ensembles candidats deviennent trop homogènes (dans le cas d'algorithmes à base de populations).

6.4 Procédures de validation

Pour évaluer la sélection d'attributs, on aura besoin d'appliquer un classifieur et d'examiner les performances de la classification.

Dans les applications du monde réel, on doit compter sur quelques méthodes indirectes en surveillant le changement des performances par rapport aux changements des attributs,

Par exemple, si nous employons le taux d'erreur de classification comme un indicateur de performance pour le traitement, pour un sous ensemble d'attributs choisi, on peut suivre l'expérience « avant et après » pour comparer le taux d'erreur du classifieur sur l'ensemble complet d'attributs et sur le sous ensemble choisi.

Le modèle de validation pour la sélection et la classification d'attributs est présenté dans la **Figure II.5**, qui montre la partition de données en un ensemble d'échantillons d'apprentissage et un ensemble d'échantillon de test.

Cette méthode peut être utilisée pour évaluer n'importe quel type de méthode de sélection, soit de filtrage (Filter), soit d'enveloppe (Wrapper) ou soit d'approche intégrée.

On note tout de même que cette validation exige de recommencer le processus de la sélection à chaque itération de validation croisée ce qui peut être couteux en temps de calcul pour une méthode enveloppe ou intégrée.

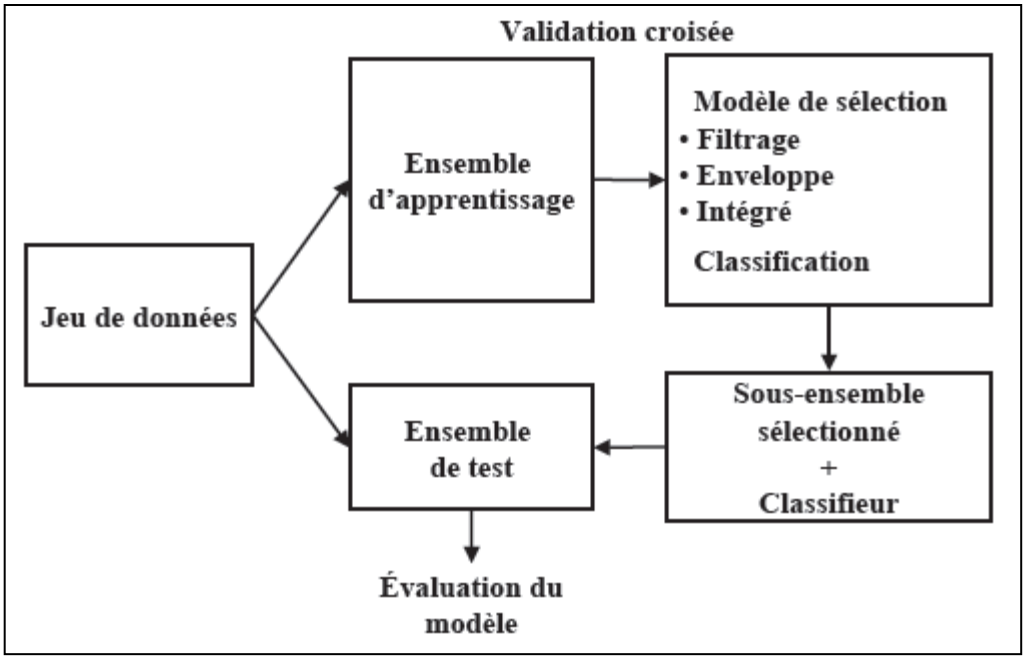

Figure II.5 : Schéma de validation croisée pour l'évaluation d'un processus de sélection –classification.

7.Cadre de catégorisation

[Liu et Yu, 2005] ont développé un cadre de catégorisation en trois dimensions. Les critères d'évaluation et les stratégies de recherche sont des facteurs dominants pour la conception des algorithmes de sélection d'attributs (certains exemples d'algorithme sont présentés en annexe B)

- Suivant les stratégies de recherches : les algorithmes sont classés par catégories en complet, séquentiel et aléatoire.
- Sous les critères d'évaluation : les algorithmes sont classés par catégories Filter, Wrapper et Hybride.
- La troisième dimension est les algorithmes d'induction car la disponibilité de l'information de classe dans la classification ou clustering affecte un critère d'évaluation utilisé dans les algorithmes de sélection des attributs.

Avec la catégorie Filter, on distingue des critères d'évaluation spécifique : distance, information, dépendance et cohérence.

Avec la catégorie Wrapper, il y a la prédiction de la précision (utilisée pour la classification) et cluster goodness (utilisé pour le clustering).

Trois rôles présentés pour ce cadre :

1. Trouver les relations entre les algorithmes.
2. Choisir un algorithme de sélection d'attributs pour une tâche donnée parmi ces algorithmes.

3. Trouver les combinaisons inexplorées des procédures de génération et d'évaluation.

7.1 Les méthodes Filtrantes

Ces méthodes sélectionnent les attributs en utilisant les différentes approches et les différents critères pour calculer la pertinence d'un attribut avant le processus d'apprentissage c'est-à-dire la construction d'un classifieur, voir l'Annexe B (**Algorithme Filter [Liu et Yu, 2005])**

7.2 Les méthodes Wrapper

Ces méthodes se servent de l'algorithme d'induction comme d'une boite noire : l'apprentissage est effectué avec les variables sélectionnées et les performances sont estimées à partir de l'erreur de généralisation. La méthode Wrapper conduit une recherche dans l'espace des paramètres possibles.

Une recherche requiert**:**

- un espace d'états où chaque état représente un sous ensemble d'attributs. Pour n attributs, il y a n bits dans chaque état et chaque bit indique si l'attribut est présent ou absent.
- un état initial : lorsque l'on fait une sélection Forward, la recherche commence avec un ensemble vide d'attributs, lorsqu'on fait une élimination « backward », la recherche commence avec l'ensemble complet d'attributs.
- une condition d'arrêt.
- une méthode de recherche.

L'Algorithme Wrapper [Liu et Yu, 2005] utilise l'algorithme d'induction au lieu de la mesure d'indépendance M pour l'évaluation de sous ensemble S. Il évalue chaque sous ensemble généré S par sa qualité utilisant l'algorithme d'induction sur les données et évalue la qualité des résultats (**Algorithme II.1**).

1) L'approche de type **Wrapper** utilise le classifieur pour évaluer le sous ensemble d'attributs choisis **Figure I.6**

Les avantages : la méthode Wrapper peut être utilisée lorsqu'on travaille avec un très grand nombre d'attributs car elle est de complexité raisonnable. Elle ne tient que des informations présentées dans les données et elle est indépendante du processus de la classification.

Les inconvénients : la méthode Wrapper repose sur le choix d'un seuil pour le critère de pertinence choisi ou d'un nombre d'attributs à choisir, le choix de ces paramètres n'est pas facile à réaliser.

Algorithme II.2: Algorithme Wrapper

Entrées: $D(F_0, F_1, \ldots, F_{n-1})$ //l'ensemble des données d'apprentissage avec N Attributs
S_0 // un sous ensemble initial pour commencer la recherche
δ // critère d'arrêt
Sorties: S_{best} // un sous ensemble optimal
Début
Initialiser : $S_{best}= S_0$;
γ_{best}=eval(S_0, D, A); //évaluer S_0 par un algorithme de recherche A
Répéter
S=générer(D) ; // générer un sous ensemble pour l'évaluation
γ=eval(S, D, A); //évaluer le sous ensemble actuel S par A
si (γ est meilleur que γ_{best}) alors
$\gamma_{best}= \gamma$;
S_{best}=S ;
Jusqu'à (δ est atteinte) ;
Retourner S_{best};
Fin

2) L'approche de type Filter utilise une fonction spécifique pour évaluer le sous ensemble d'attributs choisi **Figure II.6**

Les avantages: Le sous ensemble choisi est parfaitement adapté au classifieur.

Les inconvénients: Il y a un risque de Sur-Apprentissage. Elle est plus couteuse en temps de calcul (construction du classifieur à chaque évaluation de sous ensemble candidats). La complexité de calcul dépend de la complexité du modèle d'apprentissage utilisé.

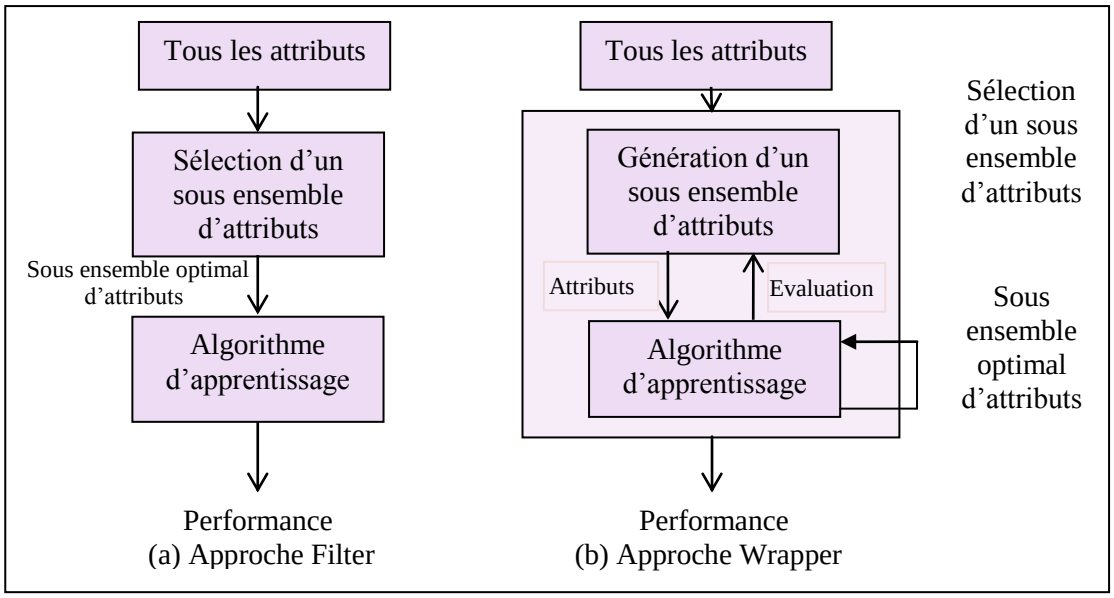

Figure II.6: Deux approches de la sélection d'attributs.

7.3 Les méthodes hybrides

Elles sont récemment proposées pour une large base de données **[Das, 2001].** L'algorithme hybride commence la recherche à partir d'un sous ensemble S_0 dans la sélection séquentielle « Forward ». Cet algorithme fait l'itération pour trouver le meilleur sous ensemble à chaque augmentation de cardinalités, Voir l'annexe B (**Algorithme Hybride [Liu et Yu, 2005]).**

8. Exemples des algorithmes

8.1 Filter

FOCUS:

Cet algorithme est présenté par **[Almuallim et Dietterich, 1991].** Il est de type Stepwise Forward Sélection. Il fait l'exploration exhaustive de tous les sous ensembles de variables afin de déterminer le plus petit sous ensemble qui est suffisant pour déterminer l'appartenance à une classe de toutes les instances.

Originalement défini pour des données booléennes non bruitées, l'algorithme de base est restreint à deux classes. De plus **[Dash et Liu, 1998]** indiquent que l'algorithme prend du temps si la taille de sous ensemble reste importante. Il a une complexité en temps $O(N^M)$.

BRANCH AND BOUND:

Il est développé par **[Narendra et Fukunaga, 1977],** la méthode de séparation et d'évaluation pour la sélection d'attributs a été reprise par **[Dash et Liu 1998].** Les fonctions d'évaluation utilisées sont la distance de Mahalanbis, la fonction discriminante, le critère de Fisher, la distance de Bhattacharya et la divergence (exemple : Kullback – Liebbi).

L'algorithme commence avec l'ensemble total d'attributs et enlève un attribut à la fois. L'algorithme garantit d'arriver à la solution optimale sans parcourir tous les sous ensembles de variables.

Il est inefficace pour réduire l'espace de recherche et la fonction d'évaluation doit être monotone.

LVF (Las Vigas Algorithm):

C'est un algorithme probabiliste, pour la sélection d'attributs. LVF fait des choix probabilistes de sous ensembles dans la recherche de l'ensemble optimal, il garde le plus petit sous ensemble d'attributs généré aléatoirement dont le taux d'inconsistance satisfait un seuil. Il est rapide pour faire décroitre le nombre d'attributs **[Chen et Liu, 1999].**

LVF est aveugle et génère des sous ensembles non intéressants, ceci est du au problème de dégradation et il est de plus en plus lent au fur et à mesure qu'il s'approche d'une solution optimale.

LVF trouve un sous ensemble d'attributs même s'il y a du bruit dans les données. De plus, l'utilisateur obtient rapidement un bon sous ensemble.

8.2 Wrapper

SFS (Sequential Forward Selection):

SFS est pour la recherche en avant. Il consiste à ajouter les attributs un à un à partir d'un ensemble vide d'attributs, en prenant à chaque étape celui qui permet de maximiser un critère J. Lorsque le nombre d'attributs attendu est plutôt faible, SFS s'avère nettement moins couteux.

SBS (Sequential Backward Selection):

SBS est pour la recherche en arrière. Cet algorithme part de l'ensemble complet d'attributs, et les supprime un à un, à chaque étape. Il supprime celui dont l'absence permet de maximiser un critère J. La génération des successeurs qui est effectuée à chaque itération ne requiert que l'application d'un seul opérateur de base, l'algorithme SBS est performant et conduit vite à des optima locaux. Mais les premières itérations de SBS sont très couteuses. SBS permet d'évaluer l'influence de chaque attribut sur la classe en présence des autres attributs.

BFS (Best First Search) [Russel et norvig, 1995]:

L'idée est de choisir le nœud le plus promoteur qui n'a pas été étendu.

9. Conclusion

Nous avons présenté une étape de prétraitement « la sélection d'attributs » qui joue un rôle important dans la fouille de données. Elle permet de construire un modèle décrivant les données en supprimant les attributs redondants, non pertinents ou bruités.

Appliquée à la tâche de la classification dans la fouille de données, elle assure une réduction de la dimension du problème, ce qui permet de réduire la durée de l'apprentissage et de simplifier le modèle appris. Cette simplification facilite généralement l'interprétation de ce modèle. Elle permet aussi d'éviter le phénomène de sur-apprentissage améliorant la précision de la prédiction et la compréhension du classifieur.

La recherche d'un sous ensemble d'attributs optimal à partir d'un ensemble d'attributs de dimension élevée est un problème d'optimisation NP-difficile. Le chapitre suivant est consacré aux méthodes dites les méta-heuristiques.

Chapitre III:
Les Métaheuristiques

1. Introduction

Il existe un grand nombre de métaheuristiques d'optimisation. Elles se distinguent classiquement en deux groupes : les méthodes de recherche locale et les méthodes de recherche globale.

Dans ce chapitre, nous allons définir les problèmes d'optimisation difficile, l'heuristique et la métaheuristique suivis par une classification de cette dernière et on va voir que les métaheuristiques se prêtent à toutes sortes d'extensions. On va présenter un état de l'art sur les principales métaheuristiques connues dans la littérature en se basant sur la métaheuristique ACO (Ant Colony Optimization). Les algorithmes de colonie de Fourmis ont connu un essor important dans la résolution des problèmes d'optimisation difficile.

2. Optimisation difficile

Plusieurs problèmes d'optimisation dépendent du choix d'une meilleure configuration de l'ensemble de variables pour atteindre ses objectifs, ils peuvent se découper en deux catégories : les problèmes où les solutions sont codées avec des valeurs réelles de variables et les problèmes où les solutions sont codées avec des variables discrètes, parmi les derniers, on trouve une classe des problèmes appelée les problèmes d'optimisation combinatoire (CO).

Nous sommes à la recherche d'un objet à partir d'un ensemble dénombrable fini ou infini, cet objet est généralement un nombre entier, un sous ensemble, une permutation ou une structure de graphe **[Papadimitriou et Steiglitz, 1982]**

Définition01: un problème d'optimisation combinatoire p(S, f) peut être défini par :

- Un ensemble de variables $X=\{x_1,\ldots,x_n\}$;
- Les domaines variables $D_1,\ldots, D_n$;
- Les contraintes entre les variables ;
- Une fonction « objectif » à minimiser (ou à maximiser) telle que $f: D_1 \times \ldots \times D_n \rightarrow R^+$;
- L'ensemble de toutes les affectations possibles à réaliser
  $S=\{s= \{(x_1,v_1),\ldots,(x_n,v_n)\}/v_i \in D_i$, satisfait toutes les contraintes$\}$.

S est généralement appelé un espace de recherche (solutions), chaque élément de l'ensemble peut être vu comme une solution candidate, pour résoudre un problème d'optimisation combinatoire on doit trouver une solution $s^* \in S$ avec une fonction objectif de valeur minimale qui est $f(s^*) < f(s)$.

$\forall\ s^* \in S$, s^* est nommé une solution optimale globale de (S, f) et $s \subseteq S$ est nommé l'ensemble de solutions optimales globales.

Parmi les ensembles des problèmes combinatoires, on cite :

- Le Problème du Voyageur de Commerce (PVC) qui consiste à trouver le plus court chemin et qui permet de passer par un certain nombre de villes et revenir à la ville de départ.
- Le Problème d'Affectation Quadratique (PAQ) : ce problème possède de nombreuses applications telles que la répartition de bâtiments ou de services, l'affectation des portes d'aéroport, la synthèse d'images.
- Les problèmes d'ordonnancement,…

De nombreux algorithmes ont été développés pour résoudre les problèmes d'optimisation combinatoire en raison de leur importance pratique. Ces algorithmes peuvent être classés en algorithmes exactes et algorithmes approchés.

Les méthodes exactes permettent de résoudre certains problèmes en un temps fini **[Papadimitriou et Steiglitz, 1982] et [Nemhauser et Wolsey, 1988]**. Ces méthodes nécessitent généralement un certain nombre de caractéristiques de la fonction objectif, comme la stricte convexité, la continuité ou encore la dérivabilité. On peut citer par exemple : la méthode de la programmation linéaire, quadratique ou dynamique, la méthode du gradient, la méthode de newton, … **[Dréo, 2006].** Pour les problèmes d'optimisation combinatoire qui sont NP-difficiles (optimisation difficile) **[Garey et Johnson, 1979]**, il n'existe pas un algorithme exact polynomial, supposant que $P \neq NP$.

Quand le nombre de combinaisons possibles devient exponentiel par rapport à la taille du problème, le temps de calcul devient rapidement critique (un temps de calcul trop élevé). L'utilisation des méthodes approchées pour résoudre les problèmes d'optimisation combinatoire a reçu de plus en plus d'attention au cours des dernières années.

Si les méthodes exactes permettent de trouver une ou plusieurs solutions dont l'optimalité est garantie, dans certaines situations, on peut obtenir de solutions de bonnes qualités sans garantie d'optimalité mais avec un temps de calcul réduit (les méthodes approchées). Parmi les méthodes approchées, on distingue les méthodes constructives et les méthodes de recherches locales.

Les méthodes constructives génèrent les solutions en ajoutant les composants jusqu'à ce que une solution soit complète, une solution partielle initialement vide, elles sont généralement plus rapide mais avec des solutions de qualité inférieure par rapport aux algorithmes de recherche locale.

Les algorithmes de recherche locale commencent par une solution initiale et essayent de remplacer la solution actuelle, d'une manière itérative, par une meilleure solution dans le voisinage qui est défini d'une manière formelle comme suit :

Défintion02

Une structure de voisinage est une fonction $N : S \rightarrow 2^S$ qui associe à chaque $s \in S$ un ensemble de voisinage $N(s) \subseteq S$. N(s) est appelé voisinage de S.

L'introduction d'une structure de voisinage nous permet de définir la notion du minimum local.

Défintion03

Une solution s∈ S est un minimum local relativement à la structure de voisinage N si f(s) ≤ f(s') ∀ s'∈ N (S).

Une solution s∈ S est un minimum global si f(s) ≤ f(s') ∀ s'∈ S, voir **Figure III.1**.

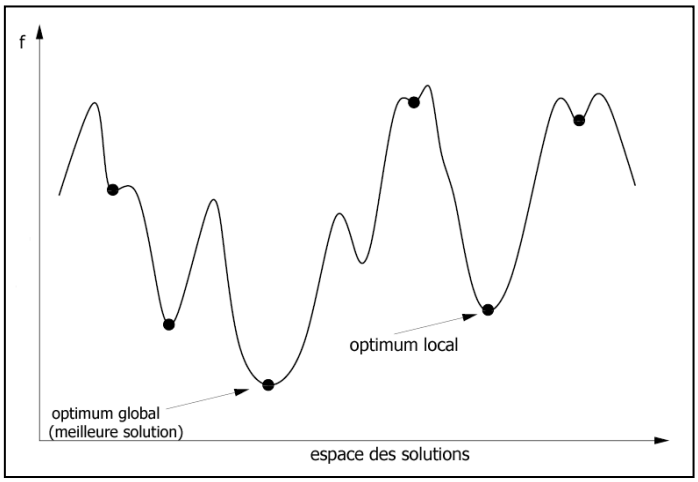

Figure III.1: Une solution locale minimale et une solution globale minimale

3. Algorithmes d'optimisation approchés

3.1 Heuristiques

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

3.2 Métaheuristiques

Plusieurs définitions ont été proposées pour expliquer clairement la notion de métaheuristique, nous citons parmi elles:

« Un processus itératif qui subordonne et qui guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque optimales » **[Osman et Laporte, 1996].**

« Les métaheuristiques sont généralement des algorithmes stochastique itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction objectif » **[Verel, 2008].**

Les métaheuristiques constituent une classe de méthodes qui fournissent des solutions de bonne qualité en un temps raisonnable à des problèmes combinatoires réputés difficiles pour lesquels on ne connait pas de méthode classique plus efficace. Elles sont

généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est à dire l'extremum global d'une fonction en évaluant une certaine fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème à fin d'en trouver une approximation de la meilleure solution d'une manière proche des algorithmes d'approximations.

L'intérêt croissant apporté aux métaheuristiques est tout à fait justifié par le développement des machines avec des capacités calculatoires énormes, ce qui a permis de concevoir des métaheuristiques de plus en plus qui ont fait preuve d'une certaine efficacité lors de la résolution de plusieurs problèmes à caractère NP-difficile.

4. Classification des métaheuristiques

Il existe plusieurs façons de classer et de décrire les algorithmes de métaheuristiques. Selon les caractéristiques choisies, plusieurs classifications sont possibles, on peut distinguer **[blum et roli, 2003]:**

4.1 Inspirées de la nature vs non inspirées de la nature

Une manière, plus intuitive, de classer les métaheuristiques consiste à séparer celles qui sont inspirées d'un phénomène naturel, de celles qui ne le sont pas.

Les algorithmes génétiques et les algorithmes de colonies de fourmis sont inspirés de la nature tandis que l'algorithme de Recherche Tabou et l'algorithme de Recherche Locale Itérative sont non inspirés de la nature, d'après les auteurs **[blum et roli, 2003]**, cette classification n'est pas significative :

- de nombreux algorithmes hybrides récents ne correspondent pas à ces deux classes (les deux classes en même temps).
- la difficulté de classer une méthode dans certains cas, par exemple, l'utilisation de la mémoire dans la recherche tabou n'est pas inspirée de la nature.

4.2 Basées sur la population des solutions vs une solution unique

Une autre manière de la classification est de partager les métaheuristiques en deux grandes classes : les métaheuristiques à solution unique (Recherche Locale, Recherche Tabou, Recuit Simulé,….) et celle à une population de solutions (Algorithmes Evolutionnaires, Recherche par Dispersion, Optimisation par Essaim Particulaires, …).

Les méthodes d'optimisation à population de solutions améliorent, au fur et à mesure des itérations, une population des solutions. L'intérêt de ces méthodes est d'utiliser la population comme un facteur de diversité.

Les méthodes d'optimisation à solution unique sont appelées les méthodes trajectoires (décrire une trajectoire en l'espace de recherche au cours du processus de recherche).

4.3 Dynamique vs une fonction objectif statique

Selon l'utilisation de la fonction 'objectif', on peut classer les métaheuristiques. Etant donné un problème d'optimisation consistant à minimiser une fonction f sur un espace s de solutions, les méthodes qui travaillent directement sur f sont statiques, les méthodes sont dynamiques s'ils utilisent une solution g obtenue à partir de f en ajoutant quelques composantes qui permettent de modifier la topologie de l'espace de solutions, ces composantes additionnelles pouvant varier durant le processus de recherche.

4.4 Une structure de voisinage vs des structures diverses de voisinages

Etant donné qu'un minimum local relativement à un type de voisinage ne l'est pas forcement pour un autre type de voisinage, il est peut être utile d'utiliser des métaheuristiques basées sur plusieurs types de voisinage c'est le cas de la Recherche à Voisinage Variable où la diversification est faite.

4.5 Utilisation de mémoire à long terme vs mémoire à court terme

Une caractéristique importante est l'utilisation de l'historique pour la classification, les algorithmes sans mémoire sont des processus markoviens puisque l'action à réaliser est totalement déterminée par la situation courante.

Une mémoire à long terme conserve une trace de mouvements récemment effectués, des solutions visitées ou en générale des décisions prises.

Une mémoire à court terme est une accumulation de paramètres de synthèse sur la recherche.

5. Extension

Les métaheuristiques se prêtent à toutes sortes d'extensions, on cite notamment celles proposées par les auteurs Clerc et Siarry **[Clerc et Siarry, 2004]:**

Le recours à des implémentations parallèles: de multiples modes de parallélisations ont été proposés pour les différentes métaheuristiques. Certaines techniques se veulent générales ; D'autres, en revanche, tirent parti de particularités du problème. Ainsi, dans les problèmes de déplacement de composants, les tâches peuvent être réparties naturellement entre plusieurs processeurs : chacun d'eux est chargé d'optimiser une zone géographique donnée, et des informations sont échangées périodiquement entre les processeurs voisins.

L'hybridation : elle s'efforce de tirer parti des avantages spécifiques de métaheuristiques différentes en les combinant **[Talbi, 2002].**

L'optimisation multiobjectif [Collette and Siarry, 2002]: il s'agit d'une optimisation simultanée de plusieurs objectifs contradictoires. La recherche vise à trouver un ensemble d'optima « optimales au sens de pareto » qui forme la « surface de compromis » du problème considéré.

L'optimisation dynamique: la fonction « objectif » varie temporellement au cours de l'optimisation, il faut approcher au mieux la solution optimale à chaque pas du temps.

L'optimisation multimodale: on cherche un ensemble de meilleurs globaux et/ou locaux, les algorithmes évolutionnaires sont particulièrement bien adaptés à cette tâche, de par leur nature distribuée. Les variantes de multi-population exploitent parallèlement plusieurs populations qui s'attachent à repérer des optimums différents.

6. Présentation des principales Métaheuristiques

Il existe un grand nombre de métaheuristiques d'optimisations. Nous allons nous appuyer sur la classification qui distingue les méthodes de trajectoires (qui exploitent séquentiellement un seul voisinage) des méthodes basées sur des populations de solutions (qui exploitent plusieurs à la fois).

6.1 Méthodes de trajectoires

6.1.1 Méthode de Descente (Hill Climbing)

Les Méthodes de Descente sont assez anciennes. Leur succès revient à leur simplicité et à leur rapidité **[Papadimitriou, 1976].** La principale caractéristique de la méthode de descente est sa grande simplicité de mise en œuvre. La plupart du temps, elle ne fait que calculer f(s+i)-f(s), où, i correspond à un déplacement élémentaire, et si cette expression peut se simplifier algébriquement, alors on pourra évaluer très rapidement cette différence.
Le principe consiste à choisir une solution s' dans le voisinage d'une solution s en améliorant la recherche tel que f(s')< f(s).

On peut décider soit d'examiner toutes les solutions de voisinage et prendre la meilleure de toutes (ou prendre la meilleur trouvée), soit d'examiner un sous ensemble de voisinage.
La méthode de Descente peut être décrite comme suit :

Algorithme III.1: Méthode de Descente
Procédure Descente Simple (solution initiale s)
Répéter :
Choisir s' dans N(s)
Si f(s')<f(s) alors s→ s'
Jusqu'à ce que f(s') >= f(s), ∀ s' ∈ S

L'ensemble S définit l'ensemble des points pouvant être visités durant la recherche. La structure de voisinage N donne les règles de déplacement dans l'espace de recherche.
La fonction f induit une topologie sur l'espace de recherche.
Une variante consiste à parcourir N(s) et à choisir la première solution s' rencontrée telle que f(s') <f(s) (pour autant qu'une telle solution existe).

6.1.2 Recuit Simulé (Simulated Annealing)

La méthode du Recuit Simulé est issue d'une analogie entre le phénomène physique de refroidissement lent d'un corps en fusion, qui le conduit à un état solide, de basse énergie. Il faut abaisser lentement la température, en marquant des paliers suffisamment longs pour que le corps atteigne l'équilibre thermodynamique à chaque palier de température pour les matériaux, cette énergie se manifeste par l'obtention d'une structure régulière, comme dans les cristaux et l'acier **[Syari et al, 1983].**

Par analogie avec le processus physique, la fonction à minimiser est l'énergie du système E, on utilise aussi un paramètre « la température du système T ».

A partir de l'espace de solution, s_0 est choisi aléatoirement, on associe avec cette solution une énergie initiale $E=E_0$ et une température initiale $T=T_0$ élevée.

La solution est modifiée à chaque itération de l'algorithme, ce qui entraine une variation ΔE de l'énergie du système. Si cette variation diminue l'énergie du système (ΔE est négative) on l'accepte, sinon on l'accepte avec une probabilité $e^{-\Delta E\backslash T}$, La température est restée constante.

Cet algorithme commence par s_0, et continue jusqu'à k_{max} (maximum d'étapes) ou jusqu'à un état ayant une énergie e_{max}, l'appel voisin (s) engendre un état aléatoire voisin d'un état s, l'appel aléatoire renvoie une valeur aléatoire dans l'intervalle [0,1]. L'appel temp(r) renvoie la température à utiliser selon la fraction r du temps total déjà dépensé.

Algorithme III.2 : Recuit Simulé

```
s←s0 ;
e←E(s) ;
k←0 ;
Tantque k<kmax et e>emax
  s←voisin(s);
  en←E(sn) ;
  si en<e ou aléatoire () < p(en-e, temp (k/kmax)) alors
    s←sn ;
    e←en ;
    k←k+1 ;
Retourner s
```

6.1.3 Recherche Tabou (Tabu Search)

La Recherche Tabou a été introduite par l'auteur Glover **[Gloer, 1986]. A**insi l'idée de base se trouve dans **[Hansen, 2000],** elle n'a aucun caractère stochastique et utilise la notion de mémoire pour éviter de tomber dans un optimum local.

Dans la recherche Tabou et à partir d'une solution donnée, on examine toutes ses voisines, on retient la meilleure solution trouvée même s'elle reste non améliorante.

On ne s'arrête pas à un optimum trouvé pour la première fois. Cette stratégie peut entrainer des cycles, par exemple un cycle de longueur 3 s←s'←s''←s←s'←s''…. pour éviter ce type de cycle, on utilise la notion de mémoire (liste Tabou), on sauvegarde les k dernières configurations visitées dans une mémoire à court terme et on interdit tout mouvement qui conduit à une de ces configurations. Cette mémoire nous permet d'éviter tous les cycles de longueur inférieure ou égale à k (une valeur qui dépend du problème à résoudre). La mémorisation de configurations serait trop couteuse en termes du temps de calcul et de place mémoire, ce qui diminue l'efficacité de la méthode. La liste Tabou mémorise des caractéristiques de configurations complètes. C'est-à-dire lorsqu'un mouvement vient d'être effectué, c'est généralement la caractéristique perdue par la configuration courante qui devient Tabou.

Lorsque les listes Tabou font intervenir des caractéristiques de modifications, les interdictions qu'elles engendrent peuvent s'avérer trop fortes et restreindre l'ensemble des solutions admises à chaque itération d'une manière jugée trop brutale. Un mécanisme particulier, l'aspiration, est utilisé pour résoudre ce problème, il permet de lever le statut Tabou d'une configuration, sans introduire un risque de cycles dans le processus de recherche. La fonction d'aspiration est définie de différentes manières. La plus simple consiste à révoquer le statut Tabou d'un mouvement si ce dernier permet d'atteindre une meilleure solution que celle trouvée.

Etant donnée s une solution courante, T : liste tabou

$N^T(s)$ est un ensemble de solutions de N(s) vers laquelle la Recherche Tabou accepte de se diriger. Cet ensemble contient toutes les solutions qui ne sont pas taboues ainsi que celles qui le sont mais dont le statut tabou est levé en raison de critère d'aspiration.

Le critère d'arrêt, par exemple, est de fixer un nombre maximum d'itérations sans améliorer s^* ou de fixer un temps limite après lequel la recherche doit s'arrêter.

Algorithme III.3: Recherche Tabou

```
s←s0 ;
T← ∅;
s*←s ;
Tant que (arrêt n'est pas satisfait)
  Déterminer une solution s' qui minimise f(s') dans N^T(s)
   Si f(s') < f(s*) alors
    s*←s' ;
    s ←s' ;
    mise-à-jour T ;
Fin tq
```

6.1.4 Méthode GRASP (Greedy Randomized search Procedure)

La méthode GRASP est une méthode itérative introduite par les auteurs Feo et Resende **[FEO, 1989] et [FEO, 1994].** GRASP est une hybridation puisqu'elle combine les avantages de la Méthode de Descente, la recherche aléatoire et la méthode de Voisinage.

C'est une méthode itérative (**Algorithme III.4**) qui se déroule en deux phases :

Une phase de construction d'une solution et une phase d'amélioration d'une solution.

Durant la phase de construction et d'une manière itérative une solution est construite : chaque itération ajoute un élément, l'élément rajouté est choisi à partir d'une liste des candidats, on utilise une liste des meilleurs candidats retenus avec un critère donné (fonction gloutonne ou une heuristique permettant de mesurer le bénéfice qu'on peut espérer).

Cette liste est appelée RCL (Restricted Candidate List), RCL est dynamiquement mise à jour.

- La phase de construction continue jusqu'à ce qu'une solution complète soit obtenue.
- La phase d'amélioration : à partir de la solution trouvée par la phase précédente, une Méthode de Descente, Recuit Simulé ou Recherche Tabou est appliquée pour améliorer la solution.

Algorithme III.4: Méthode GRASP

$f^* \leftarrow \infty$ (une valeur de meilleure solution rencontrée)

Tant que (critère d'arrêt n'est pas vérifié)

S← ∅;

Tant que (la solution courante n'est pas complète)

- évaluer l'ensemble des éléments pouvant être rajoutés à s
- déterminer la liste RCL
- choisir un meilleur élément dans RCL et l'ajouter à S.

Fin tq

- Appliquer une Méthode de Descente, Recuit Simulé ou de Recherche Tabou à s : s'← s ;

si f(s') <f(s^*) **alors**

$s^* \leftarrow s'$;

f(s^*)←f(s') ;

Fin tq

Une procédure de GRASP répète les deux phases et retourne à la fin une meilleure solution trouvée. GRASP est une méthode simple qui permet d'améliorer les solutions par l'intervention d'une heuristique spécialisée. Mais elle est moins performante par rapport à

d'autres Métaheuristiques (elle peut tomber sur le minimum local et ceci est du à l'absence de mémoire).

6.1.5 Recherche à Voisinages Variables (VNS)

L'idée de base de la méthode de VNS est le changement systématique de la structure de voisinage avec une méthode heuristique de recherche locale au lieu d'utiliser une structure de voisinage unique, elle a été proposée par **[Mladenovic et Hansen, 1997]**.

Etant donné un ensemble de structures de voisinages (qui sont souvent imbriquées), une solution est générée aléatoirement dans le voisinage de la solution courante dont laquelle une Méthode de Descente locale est effectuée. Si l'optimum local obtenu n'est pas le meilleur que la solution courante, on passe vers le voisin suivant (répétition de procédure de recherche). On recommence la recherche à partir du premier voisinage où il y a d'autres solutions meilleures qui n'ont pas été explorées ou on explore chaque structure de voisinage (**Algorithme III.5**). Tel que $N^{(t)}(S)$ est l'ensemble des solutions dans le $t^{ème}$ voisinage de s.

Algorithme III.5: Recherche à Voisinages Variables

Choisir une solution s ∈ S ;
  t←1;
Tant que (critère d'arrêt n'est pas vérifié)
  - Choisir aléatoirement une solution s' dans $N^{(t)}(S)$
  - Appliquer une procédure de recherche locale à s'
  - s'' ← s' ;
  si f(s'') < f(s) **alors** s← s'' ; t←0 ;
  si t<T **alors** t←t+1 ; **sinon** t←1 ;
Fin tq

6.2 Méthodes basées sur les populations

6.2.1 Algorithmes de Colonies de Fourmis (Ant Colony Optimization)

Une colonie de fourmis communiquent indirectement via des modifications dynamiques des pistes de phéromones et construisent une solution à un problème donné. La description de l'algorithme de colonie de Fourmis sera présentée dans la **section07.**

6.2.2 Algorithmes Génétiques (Genetic Algorithm)

Les Algorithmes Génétiques sont des algorithmes d'optimisation qui s'appuient sur des techniques dérivées de la génétique et de l'évolution naturelle : sélection, croisement, mutation. Ils ont été inventés dans les années de 60 **[Holland, 1962].** L'auteur Goldberg a étudié les Algorithmes Génétiques et il a enrichit sa théorie par **[Goldberg, 1989]**:

- Un individu est lié à un environnement par son code d'ADN.
- Une solution est liée à un problème par son indice de qualité.

- Une bonne solution à un problème donné peut être vue comme un individu susceptible de survivre dans un environnement donné.

Les éléments suivants sont nécessaires pour utiliser un Algorithme Génétique:

- Le principe du codage d'un élément d'une population : généralement après une phase de modélisation du problème à étudier, on associe à chaque point de l'espace d'états une structure de données. Le succès de l'Algorithme Génétique est lié à la qualité du codage. Il y'a le codage réel (utilisé pour l'optimisation du problème à variables réelles) et le codage binaire qui est le plus utilisé.
- La génération d'une population initiale : le choix d'une population initiale est nécessaire pour les générations futures et principalement pour converger rapidement vers un optimum global.
- Les opérateurs permettent la diversification de la population au cours des générations et l'exploration d'espace d'états. L'opérateur de croisement recompose les gènes d'individus existant dans la population. L'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états.
- Les paramètres de dimensionnement : la taille de la population, le critère d'arrêt, les probabilités d'application des opérateurs de croisement et de mutation.

Un Algorithme Génétique recherche le ou les extrema d'une fonction définie sur un espace de données (**Algorithme III.6**).

Algorithme III.6: Algorithme Génétique

- Générer aléatoirement une population d'individus.
- Pour passer d'une génération k à une génération k+1, on applique les trois opérations pour tous les éléments de la population k :

  - sélectionner les couples de parents p_1, p_2 en fonction de leur adaptation.

  - appliquer l'opérateur de croisement avec une probabilité p_c et générer les couples d'enfants c_1 et c_2, certain éléments sont choisis en fonction de leur adaptation, on applique l'opérateur de mutation avec une probabilité p_m et on génère les individus p'

  -évaluer c_1 et c_2 , p' pour l'insérer dans la nouvelle population.
- Le critère d'arrêt : Soit le nombre de générations est fixé à l'avance ou un temps limité pour trouver une solution où il n'y a pas d'évolution.

6.2.3 Algorithme d'Estimation de Distribution (EDA)

Cet algorithme a été introduit par l'auteur Baluja en 1994 **[Baluja, 19994]** dans ses travaux sur PBI (Apprentissage Incrémental Basé sur Population) et puis par les auteurs muhlenbein et Paab en 1996 **[Mühlenbein et paass, 1996]**

Cet algorithme utilise les informations collectées tout au long du processus d'optimisation pour construire des modèles probabilistes des bonnes régions de l'espace de recherche et pour engendrer des nouvelles solutions **[Larranaga et Lozano, 2001]**. Elle n'utilise pas les opérateurs de croisement et de mutation utilisés par les algorithmes évolutionnaires.
On construit une nouvelle solution à partir des solutions des générations précédentes (**Algorithme III.7**).

Algorithme III.7: Algorithme d'Estimation de Distribution

Initialiser aléatoirement une population X^0
Calculer le modèle P^0 ; $X^t \leftarrow X^0$;
Tant que (le critère d'arrêt n'est pas vérifié)
- Sélection de la sous famille X^t_{parent} de X^t
- Calcul de p^{t+1} à partir de X^t_{parent}
- Echantillonnage de p^{t+1} pour créer $X^t_{offspring}$
- Remplacement de certains éléments de X^t par les éléments de $X^t_{offspring}$ pour créer X^{t+1}

$t \leftarrow t+1$;
Fin tq

- On génère une famille de solutions X^0.
- On construit un modèle probabiliste p^0.
- La boucle tant que regroupe quatre étapes :
  - choix d'une sous famille de meilleures solutions.
  - construction du modèle probabiliste p^1 à partir de cette sous famille.
  - génération d'une nouvelle famille de solution $X^1_{offspring}$ par l'échantillonnage du modèle p^1.
  - remplacement de certains éléments de X^0 par les éléments de $X^1_{offspring}$ pour créer X^1.
- on itère le processus jusqu'à ce que le critère d'arrêt soit satisfait.

La distribution probabiliste est calculée par différentes manières suivant le modèle considéré (gaussienne, …) **[Larranaga et Lozano, 2001].**

6.2.4 Optimisation par Essaims particulaires (Particle Swarm Optimization)

L'optimisation par Essaims particulaires a été introduite par les auteurs Russel et James en 1995 **[Kennedy et Eberhart, 1995].** Cet algorithme s'inspire à l'origine du monde du vivant. Il s'appuie notamment sur un modèle développé par le biologiste Craig Reynolds à la fin des années 1980 pour la simulation du déplacement d'un groupe d'oiseaux.

L'algorithme PSO (**Algorithme III.8)** déplace un essaim de particules dans l'espace de recherche. Le déplacement de chaque particule est influencé par sa vitesse, la meilleure position qui a été retenue et la meilleure position connue par toutes les particules de l'essaim.

Soit x_i un vecteur de position de la $i^{ème}$ particule de l'essaim

v_i un vecteur de vitesse de cette particule. D la dimension de ce problème.

x_i et v_i sont des vecteurs à D éléments dont la $j^{ème}$ est notée respectivement x_{ij} et v_{ij}

Soit p_i un vecteur de dimension D qui correspond à la meilleure position atteinte par la particule i et p_{ij} sa coordonnée sur la dimension j. on note g le vecteur de dimension D qui correspond à la meilleure position connue de l'essaim.

Algorithme III.8: Optimisation par Essaims Particulaires

Initialiser aléatoirement un essaim.
Evaluer la fonction « objectif » pour chaque essaim
$x_i \leftarrow p_i$; i=1,…N (taille de l'essaim)
Calcul g
Tant que (le critère d'arrêt n'est pas vérifié)
  Mise-à-jour x_i et v_i selon les équations (01) et (02)
  Evaluer la fonction « objectif »
  mise-à-jour p_i
  mise-à-jour g
Fin tq

A l'itération t+1 le déplacement des particules est calculé à l'aide des équations :

$$v_{ij}(t+1)=w.v_{ii}(t) +c_1.r_1.(p_{ij}-x_{ij}(t))+ c_2.r_2.(g_j-x_{ij}(t)) \quad (01)$$

$$x_{ij}(t+1)=x_{ij}(t)+v_{ij}(t+1) \quad (02)$$

w : coefficient de l'inertie

c_1, c_2 : coefficients d'accélération.

r_1,$r_2 \in [0,1]$ (générés aléatoirement).

La méthode d'optimisation par essaim particulaires partage beaucoup de similarité avec l'Algorithme Génétique dans le sens où les propriétés d'un individu sont influencées par les caractéristiques des autres.

6.2.5 Recherche par Dispersion (Scatter Search)

C'est une stratégie évolutionnaire, elle a été proposée par l'auteur Glover **[Glover, 1977]**. Elle est appliquée à un grand nombre de problèmes d'optimisation. Cet algorithme (**Algorithme III.9**) démarre avec une population de solutions dont lequel un sous ensemble de solutions est sélectionné pour un ensemble de références *ensRef* qui évolue par les mécanismes d'intensification et de diversification. Les solutions de cet ensemble sont combinées pour générer des nouvelles solutions mettant à jour l'ensemble de références (la combinaison est guidée, elle n'est pas aléatoire contrairement à l'Algorithme

Génétique). L'ensemble de références dans l'algorithme de Recherche par Dispersion est de taille petite que la taille des populations dans les algorithmes évolutionnaires.

Algorithme III.9: Recherche par Dispersion

Procédure
Début
- Générer une population (InitPOP)
- Générer un ensemble de référence (RefSet)

Répéter
Répéter
Sélection d'un sous ensemble (subset) ;
Combinaison (subset , s) ;
Amélioration (s, s+) ;
Jusqu'à (le critère 01 est satisfait)
Mise à jour ensemble de référence (refSet)
Jusqu'à (le critère 02 est satisfait)
Fin

7. Algorithmes de Colonies de Fourmis (ACO)

7.1 Optimisation naturelle par les fourmis

Les fourmis, les abeilles et les termites sont des insectes sociaux qui présentent un comportement de groupe intelligent et complexe. Le comportement collectif des insectes apparait à partir d'une interaction simple et indirecte des membres d'un groupe appelé les colonies. Les fourmis étant semi ou complètement aveugles, elles communiquent entre elles utilisant un produit chimique volatil appelé phéromone (produite par les glandes situées dans les abdomens des fourmis). Elles sont attirées par ces substances, qu'elles perçoivent grâce à des récepteurs situés dans leurs antennes. La phéromone est utilisée pour des alertes de sécurité et pour guider les autres fourmis vers la source de nourriture. Le comportement coopératif de recherche de nourriture était un intérêt particulier pour les chercheurs qui ont constaté que la colonie est en mesure de trouver le plus court chemin entre le nid et une source de nourriture même si les fourmis individuelles n'ont pas une vision globale du chemin.

La **Figure III.2** illustre un exemple d'une colonie de fourmis qui est connectée à une source de nourriture par deux branches différentes. Les fourmis effectuent une recherche autour du nid. Elles empruntent les différents chemins dont initialement il n'y a pas de traces de phéromones **Figure III.3**.

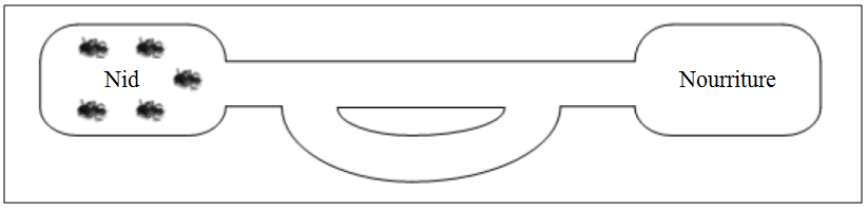

Figure III.2: Etape expérimentale pour observer le comportement des fourmis

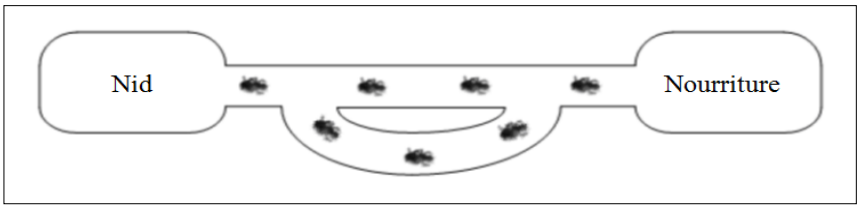

Figure III.3: Situation de début de recherche-comportement de fourragement

Quand une fourmi atteint la source de nourriture, elle retrace son chemin de retour vers le nid par une quantité de phéromone pour avertir les autres fourmis qu'il y a de la nourriture à la fin du chemin. Les fourmis les plus rapidement arrivées au nid après avoir visité la source de nourriture sont celles qui empruntent le chemin le plus court, aussi la quantité de phéromone présente sur le plus court chemin est plus importante que celle présente sur le chemin le plus long. Une piste qui présente une plus grande concentration en phéromone est plus attirante par les fourmis, elle a une probabilité plus grande d'être empruntée. La piste courte va alors être plus renforcée que la plus longue, et elle sera choisie par la quasi-totalité des fourmis (**Figure III.4**).

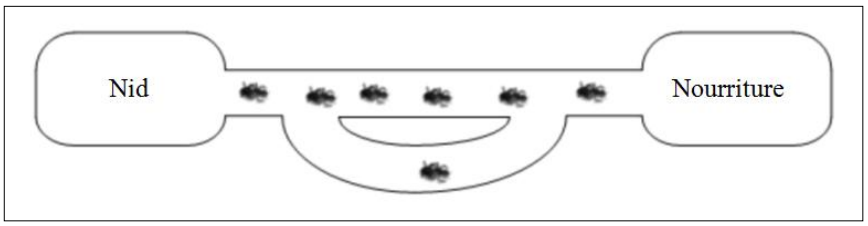

Figure III.4: Comportement du fourragement après un certain temps

Cette approche est initialement adaptée pour l'optimisation par les auteurs Colorni, dorigo et Maniezzo **[colorni et al., 1992]**.

7.2 Exemple du problème du voyageur de commerce (PVC)

Le problème du PVC consiste à trouver le plus court chemin permettant de relier un ensemble de villes en ne passant qu'une et une seule fois par une ville. Le problème revient à trouver le cycle hamiltonien minimal dans un graphe complet pondéré où les sommets sont les villes et les arcs sont les chemins entre les villes.

La résolution par Ant System (AS) [colorni et al, 1992] :

Le TSP a fait l'objet de la première implémentation d'un algorithme de colonie de fourmis, par suite une description de l'algorithme :

Soit G = (V,E) un graphe complet.

A chaque itération t ($1 \leqslant t \leqslant t_{max}$) chaque fourmi k=1,..., m parcourt le graphe et construit un trajet complet de n=|V| étapes (|V| étant le cardinal de l'ensemble de sommets V).

Pour chaque fourmi, le trajet entre la ville i et la ville j dépend de :

- La liste des villes que la fourmi k peut visiter quand elle est sur la ville i (c'est-à-dire les villes qui sont reliées à i par une arrête et que la fourmi k n'a pas encore visité) : J_i^k
- L'inverse de la distance entre les villes: $\boldsymbol{\eta}_{ij}=1/d_{ij}$ appelée la visibilité ou la valeur heuristique.
- La quantité de phéromones déposée sur l'arrête reliant les deux villes : $\boldsymbol{\tau}_{ij}$

On utilise des lois de transition aléatoires où $p_{ij}^k(t)$ est la probabilité que la fourmi k, située sur la ville i à l'itération t se déplace vers la ville j

$$p_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^{\alpha}.(\eta_{ij}(t))^{\beta}}{\sum_{l\epsilon J_i^k}((\tau_{il}(t))^{\alpha}.(\eta_{il}(t))^{\beta})} & si\ j \in J_i^k \\ 0 & sinon \end{cases} \qquad (03)$$

Où α et β sont deux paramètres à ajuster contrôlant l'importance relative de l'intensité de $\boldsymbol{\tau}_{ij}(t)$ et de la visibilité $\boldsymbol{\eta}_{ij}$

α=0, seule la visibilité de la ville qui a l'influence.

B=0, seule les pistes de phéromones influent.

Pour éviter une sélection trop rapide d'un trajet, on doit procéder au réglage de ces paramètres en jouant sur les comportements d'intensification et de diversification.

Après un tour complet, chaque fourmi laisse une quantité de phéromone $\Delta\boldsymbol{\tau}_{ij}^k(t)$ sur l'ensemble de son parcours.

Cette quantité dépend de la qualité de la solution trouvée.

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & si\ (i,j) \in T^k(t) \\ 0 & sinon \end{cases} \qquad (04)$$

Où $T^k(t)$ est le trajet effectué par la fourmi k à l'itération t

$L^k(t)$ est la longueur du tour et Q un paramètre fixé.

Enfin, le processus d'évaporation entre en jeu pour éviter de piéger dans une solution sous optimale en faisant oublier au système les mauvaises solutions. A chaque itération la règle de mise à jour est comme suit :

$$\tau_{ij}(t+1) = (1-p).\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \quad (05)$$

Où p : est le taux d'évaporation et m : nombre de fourmis.

La quantité de phéromones sur chaque arrête est une distribution uniforme d'une petite quantité $\boldsymbol{\tau}_0 \geqslant 0$

Le pseudo code de Ant system est:

Algorithme III.10: Algorithme de Colonies de Fourmis de base: le « Ant System ».

Pour t = 1,……, t_{max}
  Pour chaque fourmi k = 1,……,m
    Choisir une ville au hasard
    Pour chaque ville non visitée i
      Choisir une ville j, dans la liste J_i^k des villes restantes, selon la formule (03)
    Fin Pour
    Déposer une piste $\Delta\tau_{ij}^{k}(t)$ sur le trajet $T^k(t)$ conformément à l'équation (04)
  Fin Pour
  Évaporer les pistes selon la formule (05)
Fin Pour

7.3 Variantes

Ant system & élitisme: Cet algorithme a été introduit par les auteurs **[Dorigo et al, 1996].** Il utilise des fourmis élitistes. La fourmi qui effectue un chemin plus court dépose une quantité plus grande de phéromone pour l'exploration de la meilleure solution par les autres fourmis.

Ant-Q: cette approche utilise une règle de mise à jour locale inspirée de Qlearring **[Gambardella et Dorigo, 1995],** c'est une préversion de « Ant Colony System »

Ant Colony System: Cet algorithme a été introduit par les auteurs **[Dorigo et Gambardella, 1997].** C'est une variante la plus connue de Ant System pour améliorer les performances de cet algorithme sur des problèmes de grandes tailles où une nouvelle règle de déplacement (règle pseudo-aléatoire proportionnelle) s'ajoute un processus de mise à jour locale des éléments des pistes de phéromones, l'objectif de ce mécanisme étant d'augmenter la diversification de la recherche.

Max Min Ant System: c'est une variante plus efficace de « Ant System », où seules les meilleures fourmis tracent des pistes et le dépôt de phéromone est limité par une borne supérieure (empêchant une piste d'être trop renforcée) et une borne inférieure (laissant la possibilité d'être explorée à n'importe quelle solution).

Cet algorithme atteint de meilleurs résultats que l'original, et évite notamment une convergence prématurée **[Stutzle and Hoos, 1997].**

7.4 Formalisation d'un algorithme de Colonie de Fourmis [Dorigo, 2007]

Définition formelle d'un problème d'optimisation

Etant donné un problème d'optimisation (s, Ω, f) :

- S : est un espace de recherche défini sur un ensemble fini de variables de décision discrètes
- Ω : est un ensemble de contraintes sur les variables.
- f : $s \rightarrow R^+$ une fonction objectif à minimiser.

L'ensemble S_Ω des solutions réalisables est l'ensemble des éléments de S qui vérifient toutes les contraintes de Ω. Un élément $s^* \in S_\Omega$ est un optimum global ssi :

$s \in S_\Omega$, $f(s^*) \leqslant f(s)$.

L'espace de recherche S est défini sur un ensemble de variables discrètes $X_i(i=1,...,n)$ ayant chacune $v_i^j \in \{ v_i^1,...., v_i^{|D_i|}\}$. L'ensemble S^*_Ω

Optimisation par colonie de fourmi

Résoudre un problème d'optimisation combinatoire par colonie de fourmis revient à parcourir un graphe de construction complet. On note c_{ij} chaque variable instanciée $x_i=v_i^j$ et C l'ensemble des c_{ij} .

On obtient le graphe de construction $G_c(V,E)$ en associant C à l'ensemble des sommets V ou l'ensemble des arrêtes E. une trace de phéromone $\boldsymbol{\tau}_{ij}$ est associée à chaque c_{ij} .

Les fourmis se déplacent de sommet en sommet le long des arrêtes du graphe de construction en exploitant les informations que fournissent les traces de phéromones, et construisent ainsi peu à peu à une solution. De plus, chaque fourmi dépose une certaine quantité de phéromones pour chaque c_{ij} rencontré lors de son parcours. Cette quantité $\boldsymbol{\Delta\tau}$ peut dépendre de la qualité de la solution trouvée. Les fourmis suivantes utilisent l'information fournie par les traces de phéromones pour se guider et atteindre les régions les plus prometteuses de l'espace de recherche.

L'algorithme suivant présente une description générale :

Algorithme III.11: Optimisation par Colonies de Fourmis

Définir les paramètres, initialiser les traces de phéromones

Tant que (critère d'arrêt n'est pas vérifié)

- Construction des solutions par les fourmis
- Actions Spécifiques {facultatif}
- Mise à jour de phéromones

Fin tq

L'algorithme de colonies de fourmis se décompose en trois étapes qui se répètent à chaque itération :

- Construction des solutions par les fourmis
- Actions Spécifiques
- Mise à jour de phéromones

Généralement, le critère d'arrêt est défini par un nombre maximal d'itérations ou un temps CPU maximal.

Construction des solutions par les fourmis: Dans cette étape, m fourmis artificielles construisent des solutions à partir de l'ensemble des composantes de solution possibles $C = \{c_{ij}\}$ ($i = 1,\ldots, n$; $j = 1,\ldots, |D_i|$, n étant soit le nombre de sommets, soit le nombre d'arêtes, selon que l'on a associé C à V ou à E).

La construction d'une solution commence par une solution partielle vide $s^p = \emptyset$. Puis, à chaque étape de la construction, cette solution partielle s^p est étendue en y ajoutant une composante de solution parmi l'ensemble des voisins réalisables $N(s^p) \subseteq C$.

Le choix d'une composante dans $N(s^p)$ se fait de manière probabiliste à chaque étape de la construction. Chaque composante $c_{ij} \in N(s^p)$ a une probabilité $p(c_{ij}/s^p)$ d'être choisie. Les lois de probabilités dépendent de la variante utilisée.

Actions Spécifiques: Une fois l'étape de construction effectuée par les fourmis, et avant l'étape de mise à jour des traces de phéromones, des actions spécifiques au problème ou des actions centralisées (qui ne peuvent pas être effectuées séparément par chaque fourmi) peuvent être effectuées. Le plus souvent, ces actions consistent en une recherche locale parmi les solutions construites, et où seules les solutions localement optimisées sont utilisées dans la mise à jour des traces de phéromones.

Mise à jour de phéromones: Le but de la mise à jour des traces de phéromones, dernière étape d'une itération, est d'augmenter les valeurs de phéromones associées à de bonnes solutions, tout en réduisant celles associées à des mauvaises.

En général, la mise à jour se fait :

- en réduisant toutes les valeurs de phéromones par un procédé appelé évaporation
- en augmentant les valeurs de phéromones associées à un ensemble de bonnes solutions choisies S_{upd}.

A la fin d'une itération t, on effectue donc la mise à jour suivante :

$$\forall (i,j) \in E,\ \tau_{ij}(t+1) = \underbrace{(1-\rho)\tau_{ij}(t)}_{\text{évaporation}} + \rho \sum_{s \in S_{upd} | c_{ij} \in s} F(s)$$

Où $\rho \in [0, 1]$ est le taux d'évaporation et $F : S \rightarrow R^+$ est une fonction, appelée la fonction d'évaluation ou fonction fitness, telle que :

$$\forall s \neq s' \in S,\ f(s) < f(s') \Rightarrow F(s) \geq F(s')$$

Stigmergie (une forme de communication indirecte)

Le choix de la méthode d'implémentation des pistes de phéromone est important pour obtenir les meilleurs résultats. Ce choix est lié principalement aux possibilités de représentation de l'espace de recherche, chaque représentation pouvant apporter une façon différente d'implémenter les pistes. Les pistes de phéromone décrivent à chaque pas l'état de la recherche de la solution par le système. Les agents modifient la façon dont le problème va être représenté et perçu par les autres agents. Cette information est partagée par le biais des modifications de l'environnement des fourmis, grâce à la stigmergie.

Intensification/Diversification

L'intensification est l'exploitation de l'information rassemblée par le système à un moment donnée (dans le cas des Algorithmes de Colonies de Fourmis, cela revient à favoriser les pistes de phéromones).

La diversification est au contraire l'exploration de régions de l'espace de recherche imparfaitement prises en compte. Le piège d'un mauvais réglage entre intensification et diversification serait de tomber rapidement dans une solution sous-optimale (intensification trop forte), ou de ne jamais trouver une solution (diversification trop forte).

Dans le cas des Algorithmes à Colonies de Fourmis, plus la valeur de α sera élevée, plus l'intensification sera forte (les fourmis favoriseront les arcs avec beaucoup de phéromones). Inversement, plus α sera faible, plus la diversification sera forte.
Le paramètre β influe de la même manière sur l'intensification/diversification.

Parallélisme

Du fait que les solutions émergent des interactions indirectes ayant cours dans le système et que chaque fourmi ne prend en compte que des informations locales de son environnement (les pistes de phéromones), la parallélisation des algorithmes de fourmis est facile. On peut imaginer une solution multi-agent dans laquelle les fourmis seraient des agents purement réactifs.

8. Conclusion

Dans ce chapitre, nous avons décrit un état de l'art des méthodes d'optimisation basées sur les métaheuristiques les plus connues. Dans un premier temps nous avons défini les problèmes d'optimisation difficile, l'heuristique et la métaheuristique.

Un intérêt particulier a été porté à la méthode d'optimisation par colonie de fourmi. Cet algorithme forme une classe de métaheuristique proposée pour les problèmes d'optimisation difficile. L'algorithme de Colonie de Fourmis s'inspire des comportements collectifs de dépôt et de suivi de pistes observées.

Dans le chapitre suivant, nous allons voir la capacité de résoudre un problème NP-difficile (la recherche d'un sous ensemble d'attributs pertinents) par la métaheuristique ACO (Ant Colony Optimization).

Chapitre IV: Implémentation de l'approche proposée (ACO/C4.5)

1. Introduction

La sélection d'attributs est une technique permettant de choisir un sous ensemble d'attributs pertinents pour la construction des modèles d'apprentissage robustes. Elle fournit une meilleure compréhension des données par la sélection d'attributs importants au sein des données. La sélection d'attributs peut être implémentée comme une évaluation exhaustive de tous sous ensembles possibles.

Dans ce chapitre, nous proposons une résolution d'un problème NP-difficile « la sélection d'attributs » par une combinaison d'une métaheuristique basée sur une population « Optimisation par Colonie de fourmis » et les algorithmes de classification « Arbres de Décisions C4.5 » dont nous utilisons l'approche Wrapper (voir chapitre II).

Nous allons implémenter l'approche proposée sous Java et un logiciel libre Weka (il a été développé en java à l'université de Waikato, en Nouvelle Zélande).

2. Exemples des métaheuristiques utilisées pour la sélection d'attributs

Il existe des méthodes différentes qui sont implémentées pour la sélection d'attributs parmi elles :

- une méthode utilisant l'Algorithme de Colonie de Fourmis est proposée par les auteurs **[Deriche, 2009]**. C'est une approche hybride qui donne l'importance à la performance d'attributs et à la recherche locale. Cette méthode combine l'approche Wrapper pour améliorer la performance d'attributs et l'approche Filter pour la recherche locale.

Dans une première itération, les fourmis sélectionnent aléatoirement les sous ensembles d'attributs avec certains nombres d'attributs, seul le meilleur sous ensemble qui a une influence est sélectionné à l'itération suivante. À l'itération suivante les attributs sont sélectionnés par l'approche Filter, les attributs qui ont une meilleure maximisation du critère de performance sont sélectionnés. L'approche Filter est utilisée pour fournir l'information heuristique d'attribut donc les meilleurs attributs avec les meilleures qualités sont choisis à l'itération suivante. Les auteurs ont implémenté cette technique à la segmentation de la parole.

- La sélection d'attributs utilisant l'algorithme génétique est présentée par **[Yahang et Honavar, 1997]. C**'est un problème d'optimisation multicritère. Cette approche est une aide à la conception automatique des réseaux de neurones. Ces derniers sont utilisés pour la classification et la découverte de connaissances. Cette approche est utilisée avec un algorithme d'apprentissage des réseaux de neurones. L'Algorithme Génétique est utilisé avec une stratégie de sélection basée sur le classement. La population regroupe des individus représentant une solution candidate du problème de sélection de sous ensembles d'attributs.

- Les auteurs **[Garcia et al, 2004]** ont utilisé la Recherche par Dispersion qui se diffère de l'Algorithme Génétique par la taille de l'ensemble de solutions évaluées et la façon de

combiner les solutions existantes pour fournir de nouvelles solutions. Les auteurs ont étudié le problème de la sélection d'attributs pour obtenir un meilleur sous ensemble améliorant la tâche de la classification et réalisant ces objectifs : prendre des informations pertinentes, éviter les informations redondantes, préparer les règles et les algorithmes de classification les plus efficaces.

- Les auteurs **[Cuntu-paz, 2004]** ont résolu un problème de la sélection d'attributs par un Algorithme Génétique hybride avec une méthode basée sur une séparabilité de classes, ses objectifs sont de déterminer si cet algorithme a des performances en termes de la précision de la classification et l'accélération dont la classification est réalisée avec les réseaux bayesiens. L'approche Filter est plus rapide que l'approche Wrapper mais elle donne des résultats décevants car elle ignore complètement l'algorithme d'induction.
- D'après les auteurs **[wang et al, 2007]**, l'optimisation par essaim particulaires (PSO) comparée avec l'Algorithme Génétique n'a pas besoin de paramètres complexes (croisement, mutation,…). La PSO est efficace pour la sélection d'attributs en se basant sur la théorie des ensembles.
- Les auteurs **[Huang et Dun, 2008]** ont implémenté l'optimisation par essaim particulaires avec le classifieur Séparateur à Vaste Marge (SVM) via une architecture distribuée utilisant le web service pour réduire le temps de calcul dont le résultat est la sélection correcte d'attributs avec une précision élevée de la classification.
- L'auteur Wassem Shahzad **[Shahzad, 2010]** a proposé un algorithme hybride entre l'optimisation par colonie de fourmis et l'algorithme ID3 dont il a utilisé l'information de gain comme une valeur heuristique.

Pour notre étude, nous allons proposer une approche hybride entre l'Algorithme de Colonies de Fourmis et les Arbres de Décisions C4.5. C'est une sélection d'attributs basée sur une approche Wrapper où chaque fourmi construit une solution candidate (la sélection de certains attributs à partir de l'ensemble total d'attributs présents dans la base de données). Nous allons utiliser le rapport de gain « Gain Ratio » comme une valeur heuristique. Cet ensemble d'attributs est sélectionné à l'aide de phéromone et une valeur heuristique associée à chaque attribut. Le classifieur utilisé est l'algorithme C4.5 dont la procédure 10-validation croisée est réalisée pour vérifier le mérite d'un attribut. La valeur de phéromone est mise à jour en fonction de la fitness retournée par l'évaluation du classifieur d'apprentissage. A la fin, nous allons extraire le meilleur sous ensemble d'attributs à partir de l'ensemble total de données.

Par la suite nous allons présenter l'algorithme de la classification supervisée C4.5 et le rapport de gain utilisés dans notre approche.

3. Arbres de décisions C4.5

La construction d'un arbre de décision commence par le choix du test qui figure à la racine de l'arbre et la création des branches correspondant au test et continue, d'une façon récursive, par la création des sous-arbres pour chacune des branches (**Algorithme IV.1**). Ce type de stratégie de recherche, proposé initialement dans **[Hun et al, 1996]**, est connu sous le nom "diviser pour régner" (divide and conquer).

Algorithme IV.1 : Principe de l'algorithme : C4.5

```
C4.5 (ensemble_apprentissage)
crée le nœud racine
construit_arbre (ensemble_apprentissage, racine)
erreur_ prédite = élague_arbre (racine)

construit_arbre(T, nœud)
si tous les exemples de T ∈ à une même classe C
   alors étiquette nœud avec C
 sinon si T est vide
          alors étiquette nœud avec la classe du père (nœud)
        sinon
          choix du meilleur test X pour nœud
          pour chaque branche i de X faire
             soit Ti l'ensemble d'exemples correspondant à cette branche
             crée nœudi
             construit_arbre (Ti, nœudi)
          fin pour
       fin si
fin si
```

Les arbres de décision construits en suivant cette procédure sont souvent sur-spécialisés et présentent, donc, un assez faible pouvoir prédictif. Pour cette raison, C4.5 comporte une deuxième étape, d'élagage, au cours de laquelle des sous-arbres sont remplacés par des feuilles ou par une de leurs branches. La recherche d'une représentation élaguée commence par les feuilles terminales de l'arbre de décision et remonte vers sa racine, en effectuant tous les remplacements qui améliorent l'estimation du taux d'erreur qui sera réalisé sur de nouveaux exemples (fonction d'évaluation).

3.1 Fonction d'évaluation

Pour évaluer la qualité d'un test, C4.5 utilise le "rapport de gain" (gain ratio), qui représente le rapport entre le gain d'information utile pour la classification, apporté par l'ajout d'un test, et l'information potentielle générée par la partition des exemples.

Soient C_j, j=1..k, les classes d'un problème d'apprentissage et T l'ensemble des exemples qui restent à classer dans une branche de l'arbre de décision. L'information nécessaire pour décider qu'un exemple appartient à la classe C_j est :

$$-log_2\left(\frac{fréq\ (C_j, T)}{|T|}\right)\text{bits}, \tag{01}$$

où fréq (C_j ,T) et |T| dénotent le nombre d'exemples de la classe C_j et le nombre total d'exemples dans T.

L'information moyenne nécessaire pour connaître la classe d'un exemple arbitraire de cette branche est donc :

$$info(T) = -\sum_{j=1}^{k}\frac{fréq\ (C_j, T)}{|T|} \times log_2\left(\frac{fréq\ (C_j, T)}{|T|}\right)\text{bits}. \tag{02}$$

Soient T_i, i=1..n les ensembles d'exemples correspondant à chaque branche i obtenue par l'ajout d'un nouveau nœud sur la branche initiale. L'information nécessaire pour classer un exemple, après avoir rajouté ce nœud, correspond à la moyenne pondérée de l'information nécessaire pour chaque nouvelle branche i :

$$info_X(T) = \sum_{i=1}^{n}\frac{|T_i|}{|T|} \times info(T_i)\,\text{bits}. \tag{03}$$

Le gain d'information pertinente pour la classification obtenue par l'ajout du nœud *X* est donc :

$$gain(X) = info(T) - info_X(T) \tag{04}$$

Cette mesure du gain informationnel a été utilisée pendant longtemps en tant que fonction d'évaluation par ID3 **[Quinlan, 1986]**, l'ancêtre de C4.5. Ce dernier apporte un raffinement supplémentaire, fondé sur l'observation que le gain informationnel favorise d'une façon injustifiée les tests avec un grand nombre de branches. Pour éviter ce biais, C4.5 évalue l'information potentielle générée par la partition des exemples:

$$info\ partition(X) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times log_2\left(\frac{|T_i|}{|T|}\right) \text{bits.} \quad (05)$$

Le rapport entre le gain informationnel et l'information potentielle générée par la partition des exemples est utilisé pour départager les tests qui réalisent au moins la valeur moyenne du gain informationnel parmi l'ensemble des tests considérés.

Pour les attributs symboliques, le calcul du rapport de gain est trivial. Dans le cas des attributs numériques, C4.5 trie les exemples par rapport aux valeurs de l'attribut considéré et calcule le gain informationnel pour toutes les coupures entre deux exemples successifs, pour trouver la coupure optimale. Ce calcul peut s'effectuer en un seul passage mais demande l'examen de v-1 coupures, où v est le nombre de valeurs distinctes de l'attribut considéré (dans le cas le plus défavorable, v est égal au nombre total d'exemples).

Une optimisation possible de cette procédure passe par l'observation que la meilleure coupure du point de vue du rapport de gain surviendra toujours à la frontière entre deux exemples de classes différentes **[Fayyad et Irani, 1992].**

Une deuxième fonction d'évaluation, qui essaie de prédire le taux d'erreur qui sera réalisé par un sous-arbre sur des nouveaux exemples, est utilisée par C4.5 dans la recherche d'une représentation élaguée de l'arbre de décision. Le taux d'erreur est estimé à partir de la probabilité d'erreur a posteriori calculée pour les exemples d'apprentissage couverts par une feuille de l'arbre. Cette approximation génère une distribution binomiale, dont la limite supérieure de l'intervalle de confiance peut être évaluée pour un niveau de confiance donné. En tenant compte du fait que l'utilisation des exemples d'apprentissage pour l'évaluation du taux d'erreur va sûrement générer des prédictions trop optimistes, c'est cette limite supérieure qui sert d'estimation du taux d'erreur pour l'élagage. Dans l'implémentation de C4.5 proposée par son auteur, le niveau de confiance constitue un paramètre optionnel d'exécution. La valeur de 25% est utilisée par défaut pour l'élagage et s'avère performante dans la plupart des applications pratiques.

Après la description de l'algorithme C4.5 et le rapport de gain, nous allons proposer une hybridation entre l'algorithme C4.5 et l'algorithme de colonie de fourmis

4. Approche proposée

L'auteur Wassem Shahzad **[Shahzad, 2010]** a utilisé l'algorithme ID3. Dans ce chapitre nous allons utiliser C4.5 un successeur d'ID3 qui prend en compte les attributs numériques **[Alaoui et Belkadi, 2012].**

Chacun de ces algorithmes a des caractéristiques, parmi elles nous citons:

En ce qui concerne ID3 :

- Cet algorithme construit un arbre de Décisions dont les attributs doivent tous être qualitatifs et ils sont considérés comme nominaux.

- Il ne peut pas prendre en charge des exemples ou des données dans lequel il manque la valeur d'attribut.
- Il utilise les notions d'entropie et de gain pour déterminer l'attribut à tester dans chaque nœud.

En ce qui concerne C4.5 :

- Les attributs peuvent être qualitatifs ou quantatifs.
- On peut utiliser des exemples dans lequel la valeur de certains attributs est inconnue lors de la construction de l'arbre de décision.
- On peut prédire la classe de données dont la valeur de certains attributs est inconnue.
- On utilise le rapport de gain pour déterminer l'attribut à tester dans chaque nœud.

Nous proposons dans ce chapitre une stratégie de sélection d'attributs basée sur les algorithmes de colonies de fourmis.

4.1 Description de la stratégie

L'idée principale est de fournir un graphe bien connecté N*N (N est le nombre total d'attributs présents dans la base de données). Ce graphe se comporte comme un espace de recherche pour le déplacement des fourmis, où les liens représentent les connexions entre les attributs pour une base de données particulière et les nœuds sont les attributs. Chaque fourmi construit une solution candidate dans l'espace de recherche en traversant un chemin de nœuds et de liens, ce chemin est un sous ensemble d'attributs. Après qu'une fourmi termine son chemin, une fonction d'évaluation du chemin traversé (les attributs sélectionnés) est calculée par le classifieur C4.5 avec la procédure 10-validation croisée (On obtient une précision du modèle d'apprentissage).

On effectue la procédure 10-validation croisée (voir chapitre la Sélection d'Attributs) pour estimer le taux d'erreur moyen du classifieur. Ce dernier est la fitness d'un sous ensemble d'attributs particulier. Le processus de recherche continue jusqu'à ce que le critère d'arrêt soit vérifié. Si ce dernier est vérifié, on retient le meilleur sous ensemble d'attributs comme la meilleure solution.

Les principales caractéristiques (paramétrage de l'espace de recherche, initialisation des valeurs de phéromones, génération des solutions (sous ensembles), fonction d'évaluation « fitness » des solutions générées, mise à jour et évaporation de phéromone) sont détaillées comme suit :

4.1.1 Espace de recherche

L'espace de recherche dépend de la base de données étudiée, c'est un graphe de N*N dont N est le nombre d'attributs présents dans la base de données excluant l'attribut classe. Les nœuds représentent les attributs et les connexions entre les nœuds quand elles sont traversées par les fourmis indiquent le choix du nœud suivant (l'attribut suivant).

L'espace de recherche contient un nœud cible « Fin » **[Shahzad, 2010]**. Le nœud « Fin » est utilisé pour terminer la recherche, il est relié à tous les nœuds du graphe. Lorsqu'une fourmi choisit ce nœud dans son chemin, elle s'arrête d'ajouter des nœuds supplémentaires La **Figure IV.1** représente l'espace de recherche tels que A1, A2, A3, … sont les noms d'attributs présents dans la base de données. Il y a autant de couches que le nombre d'attributs. Chaque couche a tous les attributs. Chaque attribut est connecté à tous les autres attributs dans la couche suivante. Si une fourmi choisit un attribut, elle ne pourra pas en choisir à nouveau dans le chemin actuel. L'application de cette restriction évite l'ajout des attributs redondants dans l'ensemble formé.

Le sous ensemble obtenu contient des attributs distincts. Si la fourmi choisit le nœud « Fin », elle s'arrête d'ajouter de nouveaux attributs. Par exemple, à partir d'un ensemble d'attributs N=7 $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$, une fourmi a formé un sous ensemble d'attributs $\{A_2, A_4, A_5, A_7\}$.

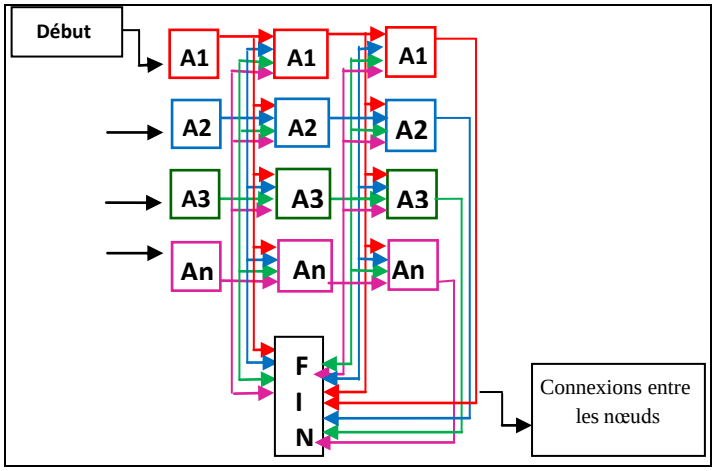

Figure IV.1: Espace de recherche N*N pour le chemin des fourmis

4.1.2 Initialisation des valeurs de phéromones

Les liens entre les nœuds sont associés avec des valeurs de phéromones. Ces valeurs sont initialisées avec une petite valeur aléatoire. Dans l'expérimentation, les valeurs de phéromones sont initialisées avec une même quantité pour tous les liens entre les nœuds pour éviter l'attraction d'un attribut par rapport à d'autres (aucune préférence existe initialement pour les attributs) **[Shahzad, 2010]**.

La valeur initiale de phéromone à l'itération 01 est calculée comme suit :

$$\tau_{ij}(t=1) = 1/N \qquad (06)$$

Dont N est le nombre total d'attributs présents dans la base de données excluant l'attribut classe

4.1.3 Sélection d'attributs

Pour choisir un attribut, une fourmi a besoin de deux composantes pour calculer la probabilité de déplacement d'un nœud actuel (attribut) vers un autre (nœud suivant).

Le premier composant dont elle a besoin est la valeur de phéromone. C'est une quantité de phéromone associée au lien entre le nœud i (présent) et le nœud j (suivant)

Le deuxième composant est la valeur heuristique de chaque nœud. Cette valeur décrit le mérite de chaque nœud (chaque attribut).

La probabilité de déplacement pour qu'une fourmi choisisse un nœud j à partir d'un nœud courant i est défini par l'équation (07) tel que le nœud j représente un nœud qui n'a pas été visité par la fourmi c'est-à-dire il appartient à l'ensemble S des nœuds (attributs) non visités.

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{k \in S}[\tau_{ik}]^{\alpha}[\eta_{ik}]^{\beta}} \qquad (07)$$

Où

τ_{ij} est la valeur de phéromone associée entre le nœud i et le nœud j.

η_{ij} est la valeur heuristique du nœud j.

S est l'ensemble des nœuds qui ne sont pas visités par le nœud actuel i.

Les paramètres α et β sont des facteurs qui ont une influence sur la valeur de phéromone et la valeur heuristique respectivement.

4.1.4 Valeur heuristique :

La valeur heuristique est utilisée pour indiquer la qualité d'un attribut. Elle constitue une des composantes nécessaires pour le déplacement et le choix du nœud suivant.

Dans cette expérimentation, nous allons utiliser le rapport de gain « Gain Ratio » de chaque attribut défini dans la section IV.3.1

Quand une fourmi décide de choisir le nœud suivant (un attribut), elle le fait avec un calcul de Rapport de Gain de ce nœud comme une valeur heuristique, ceci pour l'implémenter dans l'équation (07) en calculant la probabilité de déplacement.

4.1.5 Fonction d'évaluation fitness (taux d'erreur du chemin)

Pour voir si un sous ensemble d'attributs obtenu est le meilleur en termes de précision, nous évaluons par le classifieur C4.5 (arbre de décisions) avec la procédure 10-validation croisée. On divise aléatoirement la base de données en 10 sous ensembles de tailles équivalentes (sous ensembles mutuellement exclusive). Les neufs sous ensembles sont utilisés pour l'apprentissage et un sous ensemble est utilisé pour le test.

Le résultat de 10 exécutions est calculé en moyenne, on retient cette moyenne comme une fonction d'évaluation de sous ensemble d'attributs. Cette mesure est calculée par :

$$fitness = \sigma / N \quad (08)$$

Où

σ est le nombre d'exemples incorrectement classifiés par le classifieur.

N est le nombre total d'exemples.

4.1.6 Mise à jour de phéromone

Dans une itération, toutes les fourmis complètent leurs chemins. Chacune de ces fourmis retourne un sous ensemble. On l'évalue par le classifieur C4.5 avec une procédure 10-validation croisée, et on retourne un meilleur sous ensemble avec une meilleure fitness pour cette itération (un taux d'erreur inferieur), de sorte que dans les itérations suivantes les fourmis pourront utiliser ces valeurs (fitness, **[Shahzad, 2010]**) pour chercher le meilleur sous ensemble global.

Dans une itération, la quantité de phéromones est mise à jour pour le meilleur sous ensemble local (itération actuelle).

La mise à jour est calculée par la fonction suivante :

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + (1 - \frac{1}{1+ fitness})\tau_{ii}(t) \quad (09)$$

Tel que: $\tau_{ij}(t)$ est la valeur de phéromone entre le nœud i et le nœud j dans l'itération actuelle

ρ est le taux d'évaporation de phéromone.

fitness est la qualité du chemin actuel construit par une fourmi.

La mise à jour est faite pour augmenter la probabilité de choisir un attribut de meilleure qualité

4.2 Approche hybride ACO/C4.5 (Algorithme IV.2)

Nous commençons par le chargement du jeu de données. Ensuite la phase d'initialisation : l'espace de recherche est construit c'est un graphe N*N dont N est le nombre d'attributs dans la base de données excluant l'attribut classe. Les valeurs de phéromones sont initialisées sur tous les arcs par l'équation (06). La population des fourmis est créée et nous utilisons les valeurs des paramètres (α, β, ρ). Nous calculons les valeurs heuristiques (Gain Ratio) de chaque attribut dans la base de données (section IV.3.1).

Nous associons avec le nœud « Fin » une valeur heuristique (la moyenne des valeurs heuristiques de tous les attributs). Dans une itération de l'algorithme, initialement chaque fourmi choisit un attribut de façon aléatoire. Chacune complète son tour (choisir les autres attributs). Quand une fourmi commence la recherche, elle se déplace du nœud actuel vers le nœud suivant (attribut suivant) avec une probabilité équation (07). Chaque fourmi

termine son chemin (elle termine avec le nœud « Fin »), elle a un sous ensemble distinct d'attributs. Chaque sous ensemble d'attributs est utilisé pour construire l'arbre de décisions (classifieur C4.5). On retient la fitness par l'équation (08) de chaque sous ensemble généré par les fourmis. Nous cherchons le meilleur sous ensemble avec une meilleure fitness (un taux d'erreur inferieur de la classification) à partir de tous les sous ensembles générés dans cette itération.

Si la fitness de ce sous ensemble (itération actuelle) est meilleure que la fitness du meilleur sous ensemble d'attributs de l'itération précédente alors le sous ensemble d'attributs actuel (itération courante) devient le meilleur sous ensemble et sa fitness devient la meilleure fitness globale.

Dans l'itération suivante, la même procédure sera réalisée par toutes les fourmis.

Le processus continue jusqu'à ce que le critère d'arrêt soit vérifié. Dans notre approche, nous utilisons un nombre défini d'itérations et nous utilisons le taux d'erreur du classifieur comme fitness.

Finalement, le meilleur sous ensemble global est retenu comme le meilleur sous ensemble d'attributs de l'algorithme.

Algorithme IV.2: ACO/C4.5

Début
Charger le jeu de données
Calculer le rapport de gain (valeur heuristique) de chaque attribut **(**la section IV.3.1**)**
Générer une population de fourmis (k fourmis)
Initialiser les paramètres de l'ACO (phéromone **(équation (06))**, α, β, *p,* critère d'arrêt)
T= 1 (itération initiale)
Répéter
Pour chaque fourmi (i=1 à k) faire
Générer un sous ensemble S (chaque attribut de ce sous ensemble est déterminé suivant une probabilité **(équation (07))**
Evaluer le sous ensemble d'attributs (avec C4.5 et retenir fitness par **(équation (08))**)
Fin pour
Retenir le meilleur sous ensemble de l'itération t (avec meilleure fitness)
Si la fitness (itération courante) est la meilleure que l'itération précédente alors
Retenir la fitness de sous ensemble actuel comme une meilleure fitness globale
Fin si
Mise à jour de valeurs de phéromones **(équation09)**
Jusqu'à (le critère d'arrêt soit vérifié)
Retenir le meilleur sous ensemble d'attributs comme un sous ensemble d'attributs final
Fin

5. Expérimentation

L'implémentation de l'approche proposée (ACO/C4.5) est faite sous Java (un langage de programmation) et un logiciel libre Weka.

Weka est un logiciel libre qui met à la disposition d'un utilisateur une collection d'algorithmes d'apprentissage automatique. Parmi ces algorithmes J48 qui implante l'algorithme C4.5 conçue par l'auteur **[Quinlan, 1993]** dans le cadre de l'apprentissage supervisé. Cet algorithme permet la génération d'un arbre de décision, élagué ou non.

Nous avons choisi les bases de données pour l'expérimentation à partir de l'UCI[1] (University of California, Irvine) (**Tableau VI.1**).

Par la suite nous allons présenter le Logiciel Weka, les jeux de données utilisés de l'UCI et les résultats expérimentaux

5.1 Présentation de weka

Weka (Waikato Environment for Knowledge Analysis) est une suite populaire de logiciels d'apprentissage automatique. Écrite en Java, développée à l'université de Waikato, Nouvelle-Zélande. Weka est un Logiciel libre disponible sous la Licence publique générale GNU (GPL). L'espace de travail Weka contient une collection d'outils de visualisation et d'algorithmes pour l'analyse des données et la modélisation prédictive, allié à une interface graphique pour un accès facile de ses fonctionnalités. Elle est utilisée dans beaucoup de domaines d'application différents, en particulier pour l'éducation et la recherche. Les principaux points forts de Weka sont qu'il:

- est librement disponible sous la licence publique générale GNU,
- est très portable car il est entièrement implémenté en Java et donc fonctionne sur quasiment toutes les plateformes modernes.
- contient une collection de Prétraitements de données et de techniques de modélisation.

Weka supporte plusieurs tâches de la fouille de données (clustering, classifications,…). L'interface principale de Weka est « Explorer », mais à peu près les mêmes fonctionnalités peuvent être atteintes via l'interface Flux de Connaissance de chaque composant et depuis la ligne de commande. Il y a aussi « Experimenter », qui permet la comparaison systématique des performances prédictives des algorithmes d'apprentissage automatique de Weka sur une collection de jeux de données.

L'interface Explorer possède plusieurs onglets qui donnent accès aux principaux composants de l'espace de travail.

- L'onglet « Preprocess » a plusieurs fonctionnalités d'import de données depuis des bases de données, un fichier CSV, … et pour prétraiter ces données avec un algorithme

[1] http://archive.ics.uci.edu/ml/

appelé filtering. Ces filtres peuvent être utilisés pour transformer les données (ex : transformer des attributs numériques réels en attributs discrets) et rendre possible l'effacement d'instances et d'attributs selon des critères spécifiques.

- L'onglet « Classify » permet à l'utilisateur d'appliquer des classifications et des algorithmes de régression au jeu de données résultant, pour estimer la précision du modèle prédictif, et de visualiser les prédictions erronées, ….
- L'onglet « Associate » donne accès aux apprentissages par règles d'association qui essayent d'identifier toutes les relations importantes entre les attributs dans les données.
- L'onglet « Cluster » donne accès aux techniques de clustering de Weka, comme par exemple l'algorithme du k-means. Il y a aussi une implémentation d'algorithme espérance-maximisation pour l'apprentissage d'un mélange de distributions normales.
- L'onglet « Select attributes » fournit des algorithmes pour l'identification des attributs les plus prédictifs dans un jeu de donné.
- Le dernier onglet, « Visualize » montre une matrice de nuages de points, ou des nuages de points individuels peuvent être sélectionnés et élargis, et davantage analysés en utilisant divers opérateurs de sélection.

5.1.1 Utilisation de notre propre code sous weka

1. Nous ajoutons "notre_approche.jar " à classpath Java lors du démarrage de Weka. Cela se fait normalement en éditant la ligne commençant par "cp =" dans "RunWeka.ini".
2. Nous devons extraire le fichier "GenericPropertiesCreator.props" de weka.jar (les fichiers jar sont en fait des archives zip ordinaires, le fichier GenericPropertiesCreator.props est sous / weka / gui).
3. Nous plaçons le fichier que nous venons d'extraire dans le répertoire où nous avons installé Weka (sous Windows, il est généralement "C: Program Files \ Weka-3-6-5").
4. Sous l'entête: "Lists the Attribute Selection Search methods-Packages I want to "
   Nous ajoutons la ligne du chemin de classe "notre_approche"
   N'oublions pas d'ajouter une virgule et un anti-slash à la ligne précédente.
5. Nous utilisons la classe "notre_approche" qui se trouve dans le package
   "Select attributes" onglet **Figure IV.2**.

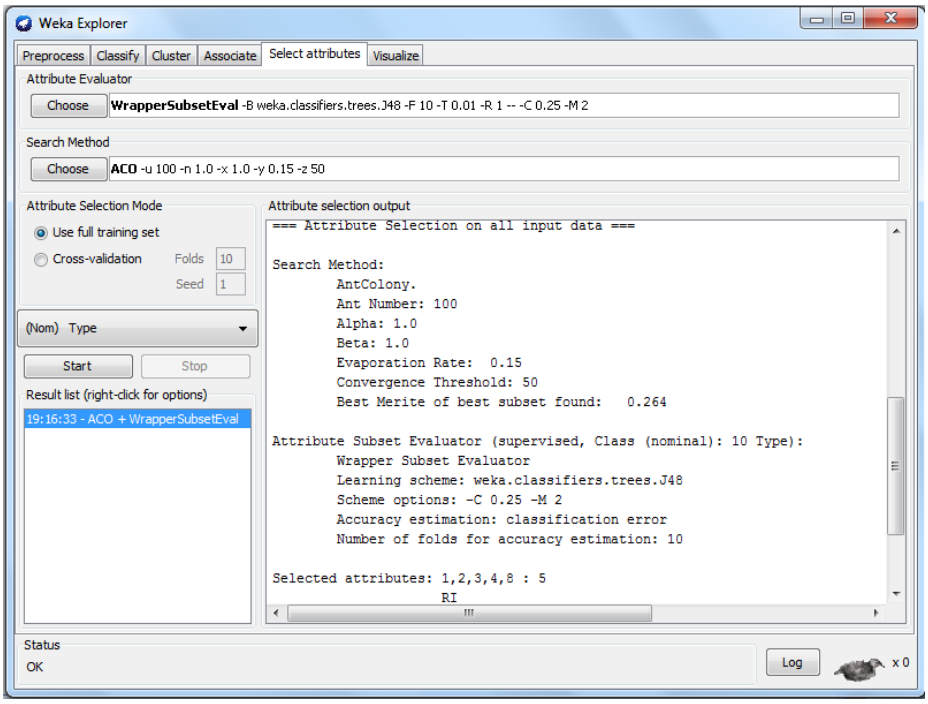

Figure IV.2: Interface de Weka

5.2 Jeux de données de l'UCI

UCI Machine Learning Repository est une collection de bases de données, des théories de domaine, et des générateurs de données qui sont utilisées par la communauté d'apprentissage automatique pour l'analyse empirique des algorithmes d'apprentissage automatique. L'archive a été créée comme une archive ftp en 1987 par David Aha et autres étudiants des cycles supérieurs à l'UC Irvine. Depuis ce temps, il a été largement utilisé par les étudiants, les éducateurs et les chercheurs du monde entier comme une source primaire d'apprentissage automatique des ensembles de données **[Blake et Merz, 1998]**.

Le détail des jeux de données est présenté dans le **Tableau IV.1**. Le choix de ces jeux de données repose sur les différentes caractéristiques (classe binaire ou multi classe, nombre d'attributs, nombre d'instances, nature des attributs). Certains jeux de données ont des classes binaires (Credit rating,…) et d'autres ont des multi classes (Zoo,….). Certains jeux de données ont un nombre inférieur d'attributs (Haberman, ...), et d'autres ont un nombre supérieur d'attributs (Sonar,…). Certains jeux de données ont un nombre inférieur d'instances (Yeast,…), et d'autres ont un nombre supérieur d'instances (Trains, ….). Certains jeux de données ont des attributs de type nominal, d'autres ont des attributs de type numérique.

Jeux de données	Nombre d'instances	Nombre d'attributs	Nombre de classes
Credit rating	690	15	2
Credit _g	1000	20	2
Diabetes	768	8	2
Ecoli	336	8	8
Glass	214	10	6
Haberman	306	3	2
hayes-roth_train	132	5	3
heart-statlog	270	13	2
Iris	150	4	3
Wine	178	13	3
Zoo	101	17	7
breast-cancer	286	9	2
breast-w	699	10	2
Ionosphere	351	34	2
Labor	57	16	2
Sonar	208	60	2
Trains	10	32	2
Vote	435	16	2
Yeast	1484	8	10

Tableau IV.1: Les Jeux de données

Nous décrivons brièvement quelques jeux de données :

« Glass » : L'étude de la classification des différents types de verres a été motivée par l'avancée des enquêtes criminologiques. En effet, sur la scène d'un crime, le verre présent peut être utilisé comme une preuve. C'est pourquoi les mesures de quantité des éléments constituants (Magnésium, Aluminum, Silicon, etc.) du verre, mais également les propriétés physiques ou optiques de celui-ci (comme l'indice de réfraction) sont importants.

Le jeu de données à disposition est donc composé de 214 observations décrites par 10 attributs (N =214, M =10), et six classes (K = 6) : "Verre 1" (70 objets), "Verre 2" (17 objets), "Verre 3" (76 objets), "Verre 4" (13 objets), "Verre 5" (9 objets) et "Verre 6" (29 objets).

« Ionosphere » : Des données radar ont été recueillies par un système composé d'un réseau de 16 antennes hautes-fréquences. Les cibles sont les électrons libres dans l'ionosphère. Les radars qualifiés de "corrects" sont ceux pour lesquels il existe un retour du signal permettant de mettre en évidence une structure de l'ionosphère. Ceux qualifiés de "incorrects" sont ceux dont les signaux émis passent à travers l'ionosphère. Les signaux reçus sont traités en utilisant une fonction d'auto-corrélation dont les arguments sont le temps et le nombre des impulsions. Pour le système utilisé, un signal est décrit par 17

impulsions. Ces derniers sont décrits par deux attributs chacun, correspondants aux valeurs complexes renvoyées par la fonction résultant du signal électromagnétique complexe. Le jeu de données à disposition est composé de 351 observations décrites par 34 attributs (N = 351, M = 34), et réparties en deux classes (K = 2) : "Correct" (225 objets) ou "Incorrect" (126 objets).

5.3 Résultats expérimentaux

Il n'y a pas des valeurs standards des paramètres de l'algorithme de colonies de fourmis. Ces valeurs dépendent des études empiriques et ses résultats. Pour notre approche hybride « optimisation par colonies de fourmis et les arbres de décision C4.5 », nous avons observé que les valeurs suivantes donnent en pratique des bons résultats:

- Nombre de fourmis (la population): 100
- α (indique l'importance des valeurs de phéromone): 1
- β (indique l'importance des valeurs heuristiques): 1
- Seuil de convergence (critère d'arrêt): 50
- Taux d'évaporation (*p*): 0.15

Le **Tableau IV.2** montre les résultats expérimentaux obtenus sans la sélection d'attributs (C4.5) et avec la sélection d'attributs (approche proposée), en se basant sur le nombre réduit des attributs et le taux d'erreur de la classification.

L'approche proposée montre son efficacité par une réduction remarquable du taux d'erreur de la classification (mentionnés en gras dans le Tableau IV.2) dans les jeux de données :

Credit rating, Credit _g, Diabetes, Glass, Haberman, hayes-roth_train, heart-statlog, wine, zoo, breast-cancer, breast-w, labor, sonar, trains, vote

L'approche hybride « optimisation par colonie de fourmis et les arbres de décisions C4.5 » réduit le nombre d'attributs comparant au nombre d'origine d'attributs des jeux de données, c'est une sélection d'attributs bénéficiaire puisque nous remarquons de meilleures précisions.

Notre approche nous permet une sélection des sous ensemble d'attributs pertinents avec une diminution du taux d'erreur de la classification (**Figure IV.3**), tout en se basant sur les valeurs heuristiques calculées par le rapport de gain de chaque attribut, en se basant sur les itérations de recherche par la population de fourmis et une fonction heuristique fitness qui est le taux d'erreur résultant de l'implémentation de l'arbre de décision C4.5. C'est une méthode basée sur l'approche Wrapper et la procédure 10- validation croisée.

Jeux de données	ACO/C4.5		C4.5	
	nombre d'attributs réduit	taux d'erreur de la classification	nombre d'attributs original	taux d'erreur de la classification
Credit rating	7	**0,137**	15	0,139
Credit _g	8	**0,253**	20	0,295
Diabetes	5	**0,258**	8	0,262
Ecoli	6	0,173	8	0,158
Glass	5	**0,264**	10	0,332
Haberman	1	**0,264**	3	0,271
hayes-roth_train	3	**0,193**	5	0,197
heart-statlog	9	**0,186**	13	0,233
iris	2	0,045	4	0,04
wine	5	**0,034**	13	0,062
zoo	10	**0,049**	17	0,079
breast-cancer	6	**0,24**	9	0,245
breast-w	6	**0,046**	10	0,054
ionosphere	17	0,096	34	0,085
labor	4	**0,178**	16	0,263
sonar	22	**0,204**	60	0,288
trains	14	**0,1**	32	0,2
vote	8	**0,034**	16	0,037
yeast	5	0,485	8	0,44

Tableau IV.2: Comparaison de notre approche hybride (ACO/C4.5) et C4.5

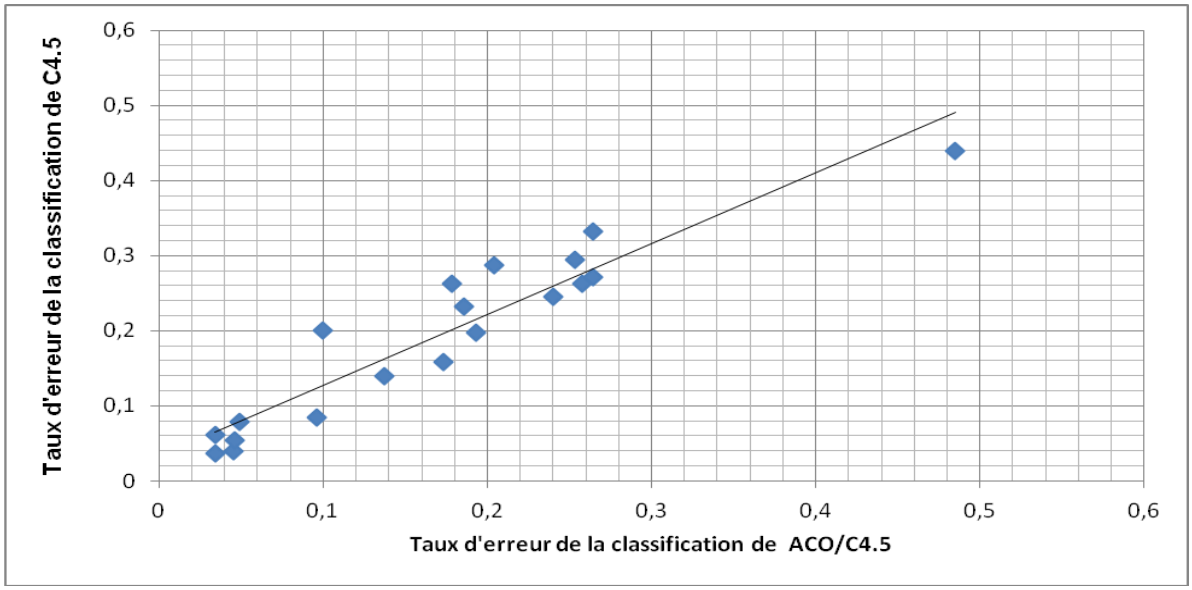

Figure IV.3: Comparaison du taux d'erreur entre ACO/C4.5 et C4.5

Afin de bien évaluer la qualité des solutions trouvées et pour voir si notre approche proposée est compétitive, nous avons implémenté deux autres approches hybrides sur l'ensemble d'attribut d'origine des jeux de données, en utilisant des métaheuristiques basées sur la population « Algorithme Génétique/C4.5 et Recherche par Dispersion/C4.5 » (nous obtenons des sous ensembles d'attributs qui peuvent augmenter ou diminuer le taux d'erreur de la classification supervisée).

5.3.1 Algorithme Génétique

Dans cet algorithme, les opérateurs de mutation et de croisement permettent d'explorer l'espace de recherche. Ces opérateurs agissent de manière aléatoire sur les individus sans nécessiter l'utilisation de la fonction « objectif ». Ainsi, la diversification peut être associée à ces deux types d'opérateurs. L'intensification est effectuée par la sélection des individus. Cette sélection est fondée sur la fonction de fitness des individus (évaluation des solutions) et consiste à favoriser la conservation des meilleurs individus.
Ce principe d'intensification par la sélection de solutions est connu sous le nom de pression sélective.

Certaines études ne partagent pas ce point de vue où l'opérateur de croisement est uniquement associé à l'exploration de l'espace de recherche. L'objectif de cet opérateur est parfois énoncé comme étant la combinaison des meilleures composantes des individus. À ce titre, le croisement peut être considéré comme étant un opérateur d'intensification. Ce point de vue est particulièrement soutenu dans le cadre des approches à base de « schèmes » ou « building-blocks » dans les algorithmes génétiques **[Goldberg, 1994]**.

L'ajustement de l'algorithme génétique est délicat, l'un des problèmes les plus caractéristiques est celui de la dérive génétique, qui fait qu'un bon individu se met, en espace de quelques générations, à envahir toute la population. On parle dans ce cas de convergence prématurée, qui revient à lancer à une recherche locale autour d'un minimum, qui n'est pas forcément l'optimum. Les méthodes de sélection proportionnelle peuvent en particulier favoriser ce genre de dérive. Un autre problème surgit lorsque les différents individus se mettent à avoir des performances similaires : les bons éléments ne sont alors plus sélectionnés, et l'algorithme ne progresse plus. Le choix d'une représentation intelligente pour permettre un remplacement générationnel efficace est un autre aspect de la question, et l'efficacité de l'algorithme génétique dépend beaucoup de la façon dont on opère le croisement des individus.

On utilise **GeneticSearch** disponible sous weka qui effectue une recherche en utilisant l'algorithme génétique simple décrite dans [**Goldberg, 1989]** avec les arbres de décisions C4.5 (J48 sous weka).
Les valeurs des paramètres définis pour **GeneticSearch** sont les suivantes:

- La probabilité de croisement: 0,6

- Le nombre de générations pour l'évaluation: 20
- La probabilité de mutation: 0,033
- La taille de la population (le nombre d'individus dans la population): 20

5.3.2 Recherche par dispersion

Elle se distingue des algorithmes évolutionnistes classiques par l'utilisation d'une procédure de recherche locale et par la généralisation de l'opérateur de croisement. Cette procédure de croisement peut être appliquée à un ensemble de solutions et pas uniquement à un couple de solutions **[Glover et al, 2003].** Elle construit des solutions par la combinaison d'autres solutions en se basant sur des principes qui capturent l'information qui n'existe pas dans les solutions originales. Elle prend l'avantage de quelques méthodes heuristiques pour sélectionner les éléments qui vont être combinés pour générer une nouvelle solution.

Elle utilise des stratégies pour diversifier et intensifier la recherche qui a prouvé son efficacité dans des variétés des problèmes d'optimisation. Cette méthode fait évoluer une population de solutions et s'appuie sur le principe suivant. Une population de solutions (assez importante au départ) est générée (en essayant de proposer des solutions diverses les unes des autres) et chaque individu est amélioré par l'application d'une recherche locale. De cette population on extrait un ensemble de références (Reference set, R) contenant les meilleures solutions de la population initiale. Ensuite ces solutions sont combinées entre elles (et avec les nouvelles solutions générées) puis améliorées jusqu' à ce qu'il n'y ait plus de nouvelles solutions générées par combinaison. Ensuite une moitié de la population est régénérée (remplacée par des solutions diverses) et le processus recommence jusqu'à satisfaction d'un critère d'arrêt.

On utilise **ScatterSearchV1** avec les arbres de décisions **C4.5.**

ScatterSearchV1 est disponible sous weka qui effectue une recherche par dispersion dans l'espace de sous-ensembles d'attributs. On commence avec une population des sous-ensembles divers et significatifs et on s'arrête lorsque le résultat est supérieur à un seuil donné où il n'y a pas plus d'amélioration, **[Garcia et al, 2004].**

Les valeurs des paramètres définis de Recherche par dispersion sont les suivantes:

- Combinaison (Définir le type de combinaison pour combiner des sous ensembles ReferenceSet) : combinaison gourmande
- Taille de la population (définir le nombre de sous-ensemble pour générer dans la population initiale) : -1

Le **Tableau IV.3** contient des descriptions générales des Métaheuristiques utilisées dans ce chapitre.

<table>
<tr><th></th><th>Intensification</th><th>Diversification</th><th>Mise à jour</th><th>Paramétrage</th></tr>
<tr><td>Algorithme génétique (AG)</td><td>Sélection</td><td>Mutation et croisement</td><td>Mise à jour de la population</td><td>Modification de la taille de la population et des taux de mutation et croisement ….</td></tr>
<tr><td>Recherche dispersée (RD)</td><td>Sélection des meilleures solutions et recherche locale</td><td>Combinaison</td><td>Mise à jour de l'ensemble de références</td><td></td></tr>
<tr><td rowspan="2">Optimisation par colonies de fourmis (ACO)</td><td colspan="2">Création de solutions par les fourmis</td><td rowspan="2">Mise à jour de phéromone (dépôt de phéromone et évaporation)</td><td rowspan="2">Modification du taux d'évaporation, de la taille de la colonie et des paramètres de décision des fourmis …</td></tr>
<tr><td>Tendance à suivre les traces de phéromone</td><td>Tendance aléatoire et à suivre les informations heuristiques</td></tr>
</table>

Tableau IV.3: Les trois algorithmes utilisés dans l'implémentation

Dans le but de tester et de prouver l'efficacité de l'approche proposée, la comparaison est basée sur le nombre réduit des attributs par rapport à l'ensemble d'attributs d'origine en réalisant une meilleure précision par le modèle d'apprentissage et en préservant les attributs pertinents. Les résultats expérimentaux sont présentés dans le **Tableau IV.4**

En se basant sur le taux d'erreur obtenu par le classifieur C4.5 (il est implémenté sur l'ensemble d'origine d'attributs), l'approche proposée a donné un taux d'erreur inférieur en comparant aux deux approches Algorithme Génétique/C4.5 et Recherche par Dispersion dans les cas suivants :

Avec l'approche proposée les jeux de données (Glass, Haberman, hayes-roth_train, Wine, breast-cancer, Vote) ont des taux d'erreur de classification inferieurs (0.264, 0.264, 0.193, 0.034, 0.24, 0.034) par rapport à l'approche Algorithme Génétique/C4.5 (0.269, 0.265, 0.194, 0.037, 0.241, 0.035) et à l'approche Recherche par Dispersion/C4.5 (0.297, 0.275, 0.194, 0.044, 0.255, 0.044).

D'autre part, nous remarquons que l'approche proposée converge vers des optimums locaux, ce qui nécessite d'améliorer l'algorithme en tenant compte du réglage des paramètres α, β, seuil de convergence,…

Les résultats obtenus indiquent que l'approche hybride (optimisation par Colonie de Fourmis et les Arbres de Décisions C4.5) est compétitive par rapport aux deux métaheuristiques l'Algorithme Génétique et la Recherche par Dispersion avec hybridation par les arbres de Décisions C4.5

Jeux de données	ACO/C4.5		Algorithme Génétique/C4.5		Recherche par Dispersion/C4.5	
	nombre d'attributs réduit	taux d'erreur de la classification	nombre d'attributs réduit	taux d'erreur de la classification	nombre d'attributs réduit	taux d'erreur de la classification
Credit rating	7	0,137	7	0,125	1	0,145
Credit _g	8	0,253	7	0,241	3	0,265
Diabetes	5	0,258	5	0,247	2	0,255
Ecoli	6	0,173	5	0,171	4	0,183
Glass	5	**0,264**	6	0,269	6	0,297
Haberman	1	**0,264**	2	0,265	1	0,275
hayes-roth_t	3	**0,193**	3	0,194	3	0,194
heart-statlog	9	0,186	3	0,142	3	0,206
iris	2	0,045	2	0,045	2	0,045
wine	5	**0,034**	4	0,037	4	0,044
zoo	10	0,049	7	0,03	6	0,02
breast-cancer	6	**0,24**	3	0,241	2	0,255
breast-w	6	0,046	5	0,044	3	0,044
ionosphere	17	0,096	16	0,065	4	0,063
labor	4	0,178	3	0,112	3	0,112
sonar	22	0,204	24	0,183	9	0,151
trains	14	0,1	2	0,1	2	0,1
vote	8	**0,034**	5	0,035	1	0,044
yeast	5	0,485	7	0,435	6	0,43

Tableau IV.4: Nombre d'attributs et Taux d'erreur de la classification des trois approches hybrides

6. Conclusion

Nous avons présenté et détaillé dans ce chapitre l'approche hybride (ACO/C4.5).
Les algorithmes de Colonies de Fourmis nécessitent une bonne définition de l'espace de recherche, l'initialisation de phéromones sur les arcs de déplacement, une valeur heuristique pour chaque attribut et une fonction d'évaluation. Cette approche hybride vise à sélectionner le meilleur sous ensemble d'attributs pertinents à partir d'un ensemble de données volumineux qui contient des données redondantes, bruitées ou non pertinentes.

Le but est d'améliorer une des tâches de la fouille de données qui est la classification supervisée. Avec les résultats expérimentaux, nous avons pu remarquer que l'approche proposée est compétitive comparant à d'autres métaheuristiques hybridées par le même classifieur C4.5

Conclusion Générale

Au cours de ce travail de magister, nous avons vu un des apports le plus important des Métaheuristiques dans l'amélioration des tâches de la fouille de données (DataMining). Nous avons proposé une approche hybride pour améliorer la tâche de la classification supervisée de la fouille de données.

Ces dernières années le volume de données de toutes sortes croît de plus en plus. Avec tant de données disponibles, il est nécessaire de développer des algorithmes qui peuvent extraire des informations significatives à partir de ce vaste volume. La recherche des pépites utiles d'information parmi des quantités énormes de données est devenue connue comme le champ de la fouille de données. Ce champ interdisciplinaire tire ces techniques et tâches de plusieurs autres domaines. Une tâche en particulier est la classification supervisée.

La sélection d'attributs consiste à choisir parmi un ensemble d'attributs de grande taille un sous ensemble d'attributs intéressants pour le problème étudié. Cette problématique peut concerner la fouille de données ou différentes tâches d'apprentissage. Elle réduit le nombre des attributs, élimine les données non pertinentes, redondantes ou bruitées. Elle accélère les algorithmes de DataMining, améliore la performance de recherche comme la précision de la prédiction, la compréhension des résultats et la réduction du temps de traitement des données. L'étude de ce problème se justifie par le fait qu'une recherche exacte a un cout exponentiel en temps de calcul et en espace mémoire en effet cela fait partie des problèmes NP-difficiles et qui peuvent être résolus par les métaheuristiques.

Nous avons proposé dans ce mémoire de magister la résolution d'un problème d'optimisation NP-difficile la sélection d'un sous ensemble d'attributs pertinents par la métaheuristique Optimisation par colonie de fourmis hybridée avec l'algorithme de la classification C4.5 (arbres de décision) pour l'amélioration de la tâche de la classification supervisée de la fouille de données en améliorant la précision et la compréhension du classifieur. Les expérimentations sont réalisées sur des bases de données de l'UCI (University of California, Irvine). Les résultats expérimentaux montrent que cette approche est compétitive.

En perspective, plusieurs voies de recherches prometteuses peuvent être suivies pour une continuité de ce travail:

- Eviter la convergence de notre approche par le réglage des paramètres de l'algorithme de colonies de fourmis (alpha, beta, *p*,…) par exemple avec l'intégration de la logique floue.
- Utiliser d'autres valeurs heuristiques qui peuvent améliorer les résultats.
- Utiliser d'autres algorithmes de la classification tels que les réseaux de neurones, le plus proche voisin, les séparateurs à vaste marge, …
- Utiliser d'autres métaheuristiques telles que l'optimisation par essaim particulaires,…

Bibliographie

[Aarts et al., 1997] E H L. Aarts, J H M. Korst, et P J M. Laarhoven: Simulated annealing. In E.H.L. Aarts et J.K. Lenstra (Eds.), Local search in combinatorial optimization (pp. 91-120). Chichester: Wiley-Interscience, 1997.

[Agrawal et al, 1993] R. Agrawal, T. Imielinski, et A. Swami: Mining association rules between sets of items in large databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, 207-216, 1993.

[Agrawal et Srikant, 1994] R. Agrawal et R. Srikant: Fast algorithms for mining association rules in large databases. Proc. VLDB conf., pp 478–499, September 1994.

[Alaoui et Belkadi, 2012] A. Alaoui et K. Belkadi : Feature Selection for Classification Using a hybrid approach based on Ant Colony Optimization and Decision Trees. ICISMT: International Conference on Information Systems, Management and Technology. Paris, France, 2012.

[Almuallim et Dietterich, 1991] H. Almuallim et T.G. Dietterich: Learning with many irrelevant features. In Proceedings of the Ninth National Conference on Artificial Intelligence, pages 547–552, 1991.

[Baluja, 1994] S. Baluja: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[Bayes, 1763] T. Bayes: An Essay towards solving a Problem in the Doctrine of Chances. Philosophical Transactions of the Royal Society of London, 1763.

[Bell et Wang, 2000] D. Bell et H. Wang: A formalism for relevance and its application in feature subset selection. Machine Learning, pages 175–195, 2000.

[Berkhin,2002] P. Berkhin: Survey of Clustering Data Mining Techniques. Technical report, Accrue software, San Jose,California, 2002.

[Blake et Merz, 1998] C. Blake, et C. Merz: UCI Repository of machine learning DataBases. University of California, Irvine, dept of Information and computer sciences (1998).

[Blum et Rivest, 1992] A.L. Blum et R.L. Rivest : Training a 3-node neural networks is NP-complete. Neural Networks, pages 117 – 127, 1992.

[Blum et Langely, 1997] A. L. Blum et P. Langley : Selection of relevant features and examples in machine learning. Artificial Intelligence, pages 245–271, 1997.

[blum et roli, 2003] C. Blum et A. Rol: Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Survey, 2003

[Brassard et Bratly, 1996] G. Brassard et P. Bratley: Fundamentals of Algorithms. Prentice Hall, New Jersey, 1996.

[Breiman et al., 1984] L. Breiman, L. Friedman, J. Olshen, et C. Stone: Classification and regression trees. CA : Wadsworth International Group, 1984.

[Breiman, 1996] L. Breiman. Bagging predictors. Machine Learning, 24:123–140, 1996.

[Candillier, 2004] L. candillier : la classification non supervisée, équipe GRAPPA, lille3, 2004

[Chen et Liu, 1999] K. Chen et H. Liu : Towards an evolutionary algorithm : A comparison of two feature selection algorithms. In 1999 Congress on Evolutionary Computation, pages 1309–1313, Piscataway, NJ, 1999.

[Clerc et Siarry, 2004] M. Clerc et P. Siarry : Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires ,2004

[Collette and Siarry, 2002] Y. Collette et P. Siarry: Optimisation multiobjectif. Eyrolles edition. 2002.

[Colorni et al., 1992] A. Colorni, M. Dorigo et V. Maniezzo: Distributed optimization by ants colonies. European Conf. ECAL'91, p. 134-142, France,1992.

[Cortes et Vapnik, 1995] C. Cortes et V. Vapnik: Support vector network. Learning machine, 20:1–25, 1995.

[Cotta et Moscato, 2003] C. Cotta et P. Moscato: The k-feature set problem is w[2]-complete. Journal of computer ans system sciences, pages 686–690, 2003.

[Cuntu-paz, 2004] E. Cantu-Paz," Feature Subset Selection, class separability, and Genetic Algorithms. Genetic and Evolutionary Computation Conference, Seattle, WA, June 26-30, 2004

[Das, 2001] S. Das. Filters, wrappers and a boosting-based hybrid for feature selection. In Proceedings of the Eighteenth International Conference on Machine Learning, 2001.

[Dash et Liu, 1997] M. Dash et H. Liu. Feature selection for classification. Intelligent Data Analysis, pages 131–156,1997.

[Dash et Liu, 1998] M. Dash et H. Liu: Hybrid search of feature subsets. In H.Y. Lee and H. Motoda, editors, Proceedings of the Fifth Pacific Rim International Conference on AI :PRICAI'98, pages 238–249, Singapore, 22–27 November 1998.

[De Castro et Von Zuben, 2000] L. De Castro et F. Von Zuben : Artificial Immune Systems : Part II - A Survey of Applications. Technical Report DCA-RT 02/00, State University of Campinas, Brazil. 2000.

[Deriche, 2009] M. Deriche, "Feature selection using ant colony optimization," in 6th International Multi-Conference on Systems, Signals and Devices", pp. 1-4, 2009.

[Doak, 1992] J. Doak. An evaluation of feature selection methods and their application to computer security. Technical report, Davis CA: University of California, Department of Computer Science,1992.

[Dorigo et al., 1996] M. Dorigo, V. Maniezzo, et A. Colorni: The Ant System : Optimization by a Colony of Cooperating Agents. IEEE Trans. Syst. Man Cybern, 1996

[Dorigo et Gambardella, 1997] M. Dorigo et L. Gambardella: Ant Colonies for the Traveling Salesman Problem. BioSystems, 1997.

[Dorigo, 2007] M. Dorigo: Ant colony optimization, http://www.scholarpedia.org /art-icle /Ant_colony_optimization, 2007.

[Dréo, 2006] J. Dréo : Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomedical, 2006.

[Fayyad et Irani, 1992] U.M. Fayyad et K.B. Irani: On the Handling of Continuous-Valued Attributes in Decision Tree Generation, *Machine Leaning*, Kluwer Academic Publishers, vol.8, p.87-102, 1992.

[Fayyad et al, 1996] U. Fayyad, S. Piatetsky, Gregory et S. Padhraic: The KDD process for extracting useful knowledge from volumes of data. ACM New York, NY, USA. Vol. Volume 39, 11. 1996.

[Fayyad,1998] U.Fayya: Data Mining and Knowledge Discovery. [ed.] Kluwer Academic Publishers., Vol. Volume 2 , Issue 1, pp. Pages: 5 – 7, Janvier 1998.

[FEO, 1989] T.A. FEO: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8, 1989.

[FEO, 1994] T.A. FEO: Greedy randomized adaptive search procedures. Journal of Global Optimization 6: 109-133, 1994.

[Frawley et al, 1992] W. Frawley, J. Piatetsky-Shapiro, Gregory et M. Christopher: Knowledge Discovery in Databases: An Overview, 1992.

[Gambardella et Dorigo, 1995] L.Gambardella, et M. Dorigo: A Reinforcement Learning Approach to the Travelling Salesman Problem. In Proceedings Twelfth International Conference on Machine Learning, Morgan Kaufmann, 1995.

[Garcia et al, 2004] F. Garcia, M. Garcia Torres, B. Melian Batista, A.Jose, M. Perez, J. Marcos: Solving feature subset selection problem by a Parallel Scatter Search. European Journal of Operational Research 169, 2004.

[Garey et Johnson, 1979] M. R. Garey et D. S. Johnson: Computers and intractability; a guide to the theory of NPcompleteness.W.H. Freeman, 1979.

[Gennari et Fisher, 1989] J. H. Gennari, P et D Fisher: Models of incremental concept formation. Artificial Intelligence, pages 11-61, 1989.

[Glover, 1977] F. Glover: Heuristics for integer programming using surrogate constraints. Decision Sciences, 8(1) :156 – 166, 1977.

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

[Glover et al, 2003] F. Glover, M. Laguna et R. Martí: Advances in Evolutionary Computation: Theory and Applications. Springer-Verlag, New York, pp. 519-537, 2003.

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

[Goldberg, 1994] D. E. Goldberg: Algorithmes génétiques. Addison-Wesley France, 1994.

[Guyon et Elisseeff, 2003] I. Guyon et A. Elisseeff : An introduction to variable and feature selection. Journal of Machine Learning Research, pages 1157–1182, 2003.

[Han et al, 1992] YC. Han et N. Cerconet : Knowledge Discovery in Databases: An Attribute-oriented approach. Proceedings of the 18th VLDB Conference, 1992.

[Hansen, 2000] Hansen: Tabu Search for Multiobjective Combinatorial Optimization, Control and Cybernetics, 29(3) :799-818, 2000.

[Holland, 1962] J. Holland: Outline for a logical theory of adaptive systems. Journal of the Association of Computing Machinery, 1962.

[Huang et Dun, 2008] Huang et Dun: A distributed PSO/SVM hybrid system with feature selection and parameter optimization Journal Applied Soft Computing archive, 2008

[Hun et al., 1996] E. Hun, J. Marin et P.J. Stone: Experiments in Induction, Academic Press, New York, E.U., 1996.

[John et al, 1994] G.H. John, R. Kohavi, et K. Pfleger: Irrelevant features and the subset selection problem. In: Proceedings of the Eleventh International Conference on Machine Learning, pages 121–129, 1994.

[John et Langley, 1995] G.H. John et P. Langley: Estimating Continuous Distributions in Bayesian Classifiers. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. pp. 338-345. Morgan Kaufmann, San Mateo,1995.

[Jollois,2003] F,X,Jollois: Contribution de la classification automatique à la fouille de données. Thèse de Doctorat,Université de Metz,2003.

[Jourdan, 2003] L. Jourdan: Métaheuristiques pour l'extraction de connaissances: Application à la génomique, Thèse de Doctorat de l'Université de Lille 1, 2003.

[Kennedy et Eberhart, 1995] J. Kennedy et R.C. Eberhart: Particle Swarm Optimisation.. Proc. IEEE Int. Conf. On Neural Networks, pp.1942-1948, Perth, Australia, 1995.

[Kohavi et John, 1997] R. Kohavi et G.H. John: Wrappers for feature subset selection. Artif. Intell., pages 273–324,1997.

[Koller et Sahami, 1996] D. Koller et M. Sahami: Toward optimal feature selection. 13th International conference on machines learning, pages 1–15, 1996.

[Kumar, 2000] V. Kumar:An Introduction to Cluster Analysis for Data Mining. Technical report, C.S. Dept. Univ. Minnesota, 2000.

[Langly et Iba, 1993] P. Langley et W. Iba: Average-case analysis of a nearest neighbor algorithm. In International Joint Conference on Artificial Intelligence (IJCAI), 1993.

[Larrañaga et Lozano, 2001] P. Larrañaga et J.A. Lozano: Estimation of Distribution Algorithms, a new tool for evolutionary computation., Kluwer Academic Publishers, 2001.

[Lauriere, 1978] J-L. LAURIERE: A language and a program for stating and solving combinatorial problems, Artificial Intelligence 10 : 29-127, 1978.

[Liu et Motoda, 1998] H. Liu et H. Motoda: Feature Selection for Knowledge Discovery and Data Mining. Boston: Kluwer Academic Publishers, 1998.

[Liu et Yu, 2005] H. Liu et L. Yu: Toward Integrating Feature Selection Algorithms for Classification and Clustering, Department of Computer Science and Engineering,Arizona State University, 2005

[Liu et Motoda, 2008] H. Liu et H. Motoda : Computational methods of feature selection. Taylor & Francis group, 2008.

[MacQueen, 1967] J. B. MacQueen: "Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, 1:281-297, 1967.

[Michael et al, 1997] J.A. Michael, Gordon et S. Linoff : Data Mining:Techniques appliquées au marketing, à la vente et aux services clients, 1997.

[Michalski, 1983] R. S. Michalski: A theory and methodology of inductive learning. Artidicial Intelligence, pages 111–116, 1983.

[Miller et al, 2001] Miller, Harvey. J and Jiawei, Han. Geographic data mining and knowledge discovery: an overview. New York : Taylor & Francis, 2001.

[Mingers, 1989] Mingers, J: An empirical comparison of pruning methods for decision tree induction. Machine Learning, 4(2):227_243, 1989.

[Mladenovic et Hansen, 1997] N. Mladenovic, et P. Hansen: Variable Neighborhood Search. Computers&Operations Research 24, 1097–1100. 1997.

[Molina et al, 2002] L. C. Molina, L. Belanche, et A. Nebot: Evaluating feature selection algorithms. CCIA-LNCS, pages 216–227, 2002.

[Mühlenbein et Paass, 1996] H. Mühlenbein et G. Paass: From recombination of genes to the estimation of distributions i. binary parameters. In PPSN 1996.

[Nemhauser et Wolsey, 1988] G. L. Nemhauser et A. L. Wolsey: Integer and Combinatorial Optimization. John Wiley & Sons,New York, 1988.

[Narendra et Fukunaga, 1977] P.M. Narendra et K. Fukunaga: A branch and bound algorithm for feature subset selection.IEEE Trans. on Computer, pages 917–922, 1977.

[Osman et Laporte, 1996] I. H. Osman et G. Laporte: Metaheuristics: A bibliography. Annals of Operations Research,63:513.623, 1996.

[Papadimitriou, 1976] C. H. Papadimitriou: The complexity of combinatorial optimization problems. PhD, 1976.

[Papadimitriou et Steiglitz, 1982] C. H. Papadimitriou et K. Steiglitz: Combinatorial Optimization - Algorithms and Complexity. Dover Publications, Inc., New York, 1982.

[Parsaye et al, 1996] Parsaye K: Surveying Decision Support: New Realms of Analysis. Information Discovery, 1996

[Perner,2001] Perner: Improving the accuracy of decision tree induction by feature pre-selection. Applied Artificial Intelligence, pages 747-760, 2001.

[Quinlan, 1986] J. R. Quinlan: Induction of decision trees. Machine Learning, 1986.

[Quinlan, 1993] J.R. Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann,San Diego, 1993.

[Russell et Norvig, 1995] S.J. Russell et P. Norvig: Artificial Intelligence: A Modern Approach Prentice Hall, Englewood Cliffs, NJ, 1995.

[Russell et Norvig, 2003] S.J. Russell et P. Norvig: Artificial Intelligence : A Modern Approach. Pearson Education, 2003.

[Schapire, 1990] R.E. Schapire: The strength of weak learning. Machine Learning, 5(2):197–227, 1990.

[Seban et Nock, 2001] M. Seban et R. Nock: Impact of learning set quality and size on decision tree performances. International Journal of Computers, Systems and Signals, pages 85-105, 2001.

[Shahzad, 2010] W. Shahzad: Classification and Associative Classification Rule Discovery Using Ant Colony Optimization. these, Pakistan. 2010.

[Shakhnarovich et al, 2006] Shakhnarovich, Gregory, T. Darrell et P. Indyk: Nearest-Neighbor Methods in Learning and Vision. The MIT Press, 262 p, 2006.

[Stutzle and Hoos, 1997] T. Stutzle et H. Hoos: Improvements on the Ant System : Introducing MAX-MIN Ant System. In Proceedings International Conference on Artificial Neural Networks and Genetic Algorithms, Vienna. Springer-Verlag. 1997

[Syari et al, 1983] K.P. Syari et C.D Gelatt: Optimization by Simulated Annealing, Vecchi M.P., SCIENCE, p671-680, 1983.

[Talbi, 2002] E.G. Talbi: A Taxonomy of Hybrid Metaheuristics. Journal of Heuristics, 8(5):541--564, 2002.

[Usama et al, 1996] M.Usama, Fayyad, P.S. Gregory et S. Padhraic: From data mining to knowledge discovery: An overview. In Advances in Knowledge Discovery and Data Mining, pages 1–34. 1996.

[Van dijck et al, 2005] G. Van Dijck, M. M. Van Hulle, et M. Wevers: Genetic Algorithm for feature subset selection with Exploitation of feature correlations from contiuous wavelet transform: A real case Application, World Academy of Science, Engineering and Technology, 2005

[Verel, 2008] S. Verel : Métaheuristiques pour l'Optimisation Difficile, équipe ScoBi, Université de Nice Antipolis 2008, cours.

[Wang et al, 2007] X. Wang, J. Yang, Xi. Teng, W. Xia et Richard Jensen: Feature selection based on Rought sets and particle Swarm Optimization. Journal Pattern Recognition Letters , 2007.

[Wijsen , 2001] J. Wijsen: Data Mining et Data Warehousing, 2001.

[Yahang et Honavar, 1997] J. Yahang, et V. Honavar: Feature subset selection using a genetic algorithm. IEEE Intelligent Systems, Vol. 13, No. 2, pp. 44-49, 1997.

[Ye et al, 2002] C.Z. Ye, J. Yang, D.Y. Geng: Fuzzy rules to predict degree of malignancy in brain glioma. Med Biol Eng Comput, 40(2): p. 145-52, 2002.

[Yu et Liu, 2004] L. Yu et H. Liu: Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research, pages 1205–1224, 2004.

Annexes

Annexe A

A.1 Les quatre phases d'extraction des itemsets fréquents

- préparation des données

Cette phase consiste à sélectionner les données (attributs et objets) de la base de données utiles à l'extraction des règles d'association et transformer ces données en un contexte d'extraction. Ce contexte, ou jeu de données est un triplet B= (O, A, R) dans lequel O est un ensemble d'objets, A est un ensemble d'attributs, également appelé items, R est une relation binaire entre O et A.

Un contexte d'extraction des règles d'association D constitue de six objets, chacun est identifié par son OID (identification d'objet), et cinq items. Ceci est représenté dans le **Tableau A.1**. Cette étape est essentielle pour qu'il soit possible d'appliquer les algorithmes d'extraction des règles d'association sur des données de natures différentes provenant de sources différentes, de concentrer la recherche sur les données utiles pour l'application et de minimiser le temps d'extraction.

OID	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE
6	BCE

Tableau A.1: Contexte d'extraction des Règles d'association D

- extraction des ensembles fréquents d'attributs

Dans cette phase, on extrait du contexte tous les ensembles d'attributs binaires $I \subset A$, appelé itemsets, qui sont fréquents dans le contexte B, un itemset I est fréquent si son support, qui correspond au nombre d'objets du contexte qui « contiennent » I, est supérieur ou égale au seuil minimal de support min support défini par l'utilisateur. L'ensemble des itemsets fréquents dans le contexte est noté F. le problème de l'extraction des itemsets fréquents est de complexité exponentielle dans la taille m de l'ensemble d'items puisque le nombre d'itemsets fréquents potentiels est 2^m. Ces itemsets forment un treillis dont la représentation sous forme de diagramme de Hasse (une représentation visuelle d'un ordre fini. Similaire à la représentation habituelle d'un graphe sur papier) pour le contexte D est présentée dans la **Figure A.2**

Des balayages du contexte doivent être réalisés lors de cette étape, il est donc important de développer des méthodes efficaces d'exploration de cet espace de recherche exponentielle.

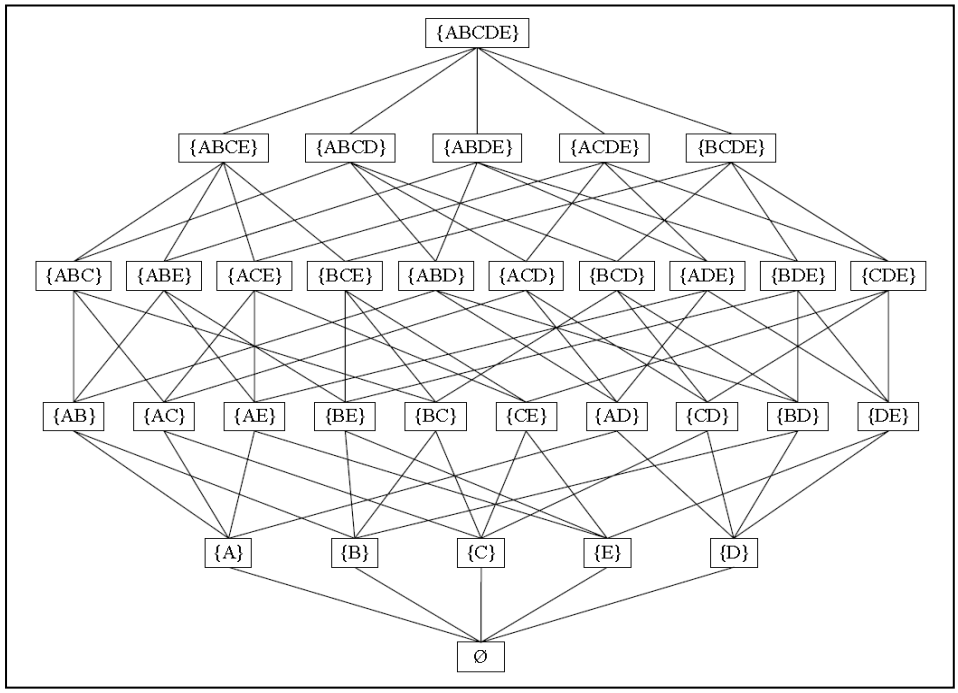

Figure A.1: Diagramme de Hasse représentant le treillis des itemsets

- Génération des règles d'association

Les itemsets fréquents extraits durant la phase précédente sont utilisés pour générer les règles d'association qui sont des implications entre deux itemsets fréquents I_1, $I_2 \in F$ tel que $I_1 \subset I_2$, de la forme r : $I_1 \rightarrow (I_2/I_1)$. Afin de limiter l'extraction aux règles d'association les plus informatives, seules celles qui possèdent une confiance supérieure ou égale au seuil minimal min confiance défini par l'utilisateur sont générés.

La confiance d'une règle r : $I_1 \rightarrow (I_2/I_1)$ est définie comme la proportion d'objets contenant la conséquence (I_2/I_1) de r parmi ceux qui contiennent l'antécédent I_1 de r. Cette valeur est égale au rapport entre le support de l'itemset I_2 et le support de l'itemset I_1.

- Interprétation des résultats

L'utilisateur visualise les règles d'association extraites du contexte et leur interprétation afin d'en déduire des connaissances utiles pour améliorer l'activité concernée. Le nombre important des règles d'association extraites en général impose le développement d'outils de classification des règles, de sélection par l'utilisateur de sous ensembles de règles et de leur visualisation sous une forme intelligible.

A.2 Clustering

Les propriétés des algorithmes de clustering qui nous intéressent dans le Datamining

Les caractéristiques désirables des algorithmes de clustering dépendent du problème particulier à étudier **[kumar 2000, berkhin 2002]**, elles incluent :

1. Scalabilité: Les techniques de clustering pour un ensemble de données volumineux doivent être scalable, en termes de temps et d'espace. Il est habituel qu'une base de données peut contenir des milliers d'enregistrements, n'importe quel algorithme de clustering utilisé devrait avoir une complexité temporelle linéaire ou proche pour gérer un ensemble volumineux de données.
2. Indépendance de l'ordre des entrées**:** Pour assurer la validité des clusters obtenus.
3. Moyens efficaces pour la détection et la lutte contre le bruit ou les points isolés**:** Eliminer le bruit et les points isolés, soit en cours de l'exécution du processus de clustering ou dans une étape de post traitement. Il faut s'assurer que l'algorithme de clustering n'est pas faux pour la majorité des points dans le cas où les points isolés ne sont pas rejetés.
4. Interprétation des résultats: Décrire un cluster comme une région peut être mieux compréhensible qu'une liste de points.
5. La robustesse en présence de différentes caractéristiques de données et du cluster: Les caractéristiques prises en compte sont : la confidentialité, les bruits, la distribution statistique, la forme du cluster (rectangulaire, convexes, …), la taille du cluster, la densité du cluster, la séparation du cluster, le type d'espace de données (euclidienne ou non, …), les types d'attributs (temporel, catégorie, …).
6. Estimation des paramètres**:** L'algorithme de clustering a la capacité d'estimer par exemple le nombre de clusters, la taille des clusters ou la densité des clusters.
7. La capacité de fonctionner d'une manière progressive: C'est pour ajouter des nouvelles données ou supprimer les anciennes données sans réexécuter le nouvel ensemble de données.

Le système de clustering [Candillier, 2004]

Le principe consiste à maximiser la similarité des observations à l'intérieur d'un cluster et à la minimiser entre clusters, les étapes à suivre sont:

1. la représentation des données (inclut éventuellement extraction et/ou la sélection d'attributs). C'est pour produire de nouveaux attributs ou utiliser un sous ensemble efficace pour le clustering.

2. la définition d'une mesure de proximité appropriée au domaine des données pour éviter la domination d'un attribut sur un autre.

3. le regroupement (la segmentation) selon la technique adoptée.

4. l'abstraction des données (si nécessaire), typiquement c'est la description de chaque cluster.

5. l'évaluation de la sortie (si nécessaire), il peut se faire de trois manières:

- évaluation externe: comparer à une solution à priori.

- évaluation interne: déterminer si la solution est intrinsèquement appropriée aux données.
- évaluation relative: comparer les différentes solutions possibles.

Annexe B

B.1 Algorithme Filter [Liu et Yu, 2005]

Algorithme B.1: Algorithme Filter

Entrées: $D(F_0, F_1, \ldots, F_{n-1})$ //l'ensemble des données d'apprentissage avec N attributs
S_0 // un sous ensemble initial pour commencer la recherche
δ // critère d'arrêt
Sorties: S_{best} // un sous ensemble optimal
Début
Initialiser : $S_{best} = S_0$;
γ_{best}=eval(S_0, D, M); //évaluer S_0 par une mesure d'indépendance M
Répéter
S=générer(D) ; // générer un sous ensemble pour l'évaluation
γ=eval(S, D, M); //évaluer le sous ensemble actuel S par M
si (γ est meilleur que γ_{best}) alors
$\gamma_{best} = \gamma$;
S_{best}=S ;
Jusqu'à (δ est atteinte) ;
Retourner S_{best};
Fin

Etant donnée D un ensemble des données, l'algorithme débute la recherche à partir du sous ensemble S (un ensemble vide ou complet ou choisi aléatoirement). Selon une stratégie donnée, l'algorithme recherche dans l'espace d'attributs. Chaque sous ensemble généré par S est évalué par un critère indépendant M et comparé avec le meilleur précédent, s'il est le meilleur alors il est considéré comme le meilleur sous ensemble courant. L'itération s'arrête tant que le critère d'arrêt J est satisfait.
Le résultat final : l'algorithme retourne le meilleur sous ensemble courant S_{best}.

B.2 Algorithme Hybride [Liu et Yu, 2005]

L'algorithme hybride commence la recherche à partir d'un sous ensemble S_0 (généralement un ensemble vide dans la sélection séquentielle « Forward » et faire l'itération pour trouver le meilleur sous ensemble à chaque augmentation de cardinalités.

Algorithme B.2: Algorithme Hybride

Entrées: $D(F_0, F_1, \ldots, F_{n-1})$ //l'ensemble des données d'apprentissage avec N Attributs
S_0 // un sous ensemble initial pour commencer la recherche
δ // critère d'arrêt
Sorties: S_{best} // un sous ensemble optimal
Début
Initialiser : $S_{best}= S_0$;
c_0= card (S_0) ; // calculer la cardinalité de S_0
γ_{best}=eval(S_0, D, M); //évaluer S_0 par une mesure d'indépendance M
θ_{best}=eval(S_0, D, A); //évaluer S_0 par un algorithme de recherche A
pour c=c_0 + 1 to N **debut**

pour i=0 to N-c **debut**
S= S_{best} U {F_j}; // générer un sous ensemble avec une cardinalité c pour l'évaluation
γ=eval(S, D, M); //évaluer le sous ensemble actuel S par M
si (γ est meilleur que γ_{best}) **alors**
$\gamma_{best}= \gamma$;
S'$_{best}$=S ;
Fin pour;

θ=eval(S'$_{best}$, D, A); //évaluer S'$_{best}$ par A

si (θ est meilleur que θ_{best}) **alors**
S_{best}= S'$_{best}$;
$\theta_{best}= \theta$;
sinon
retourne S_{best};
Fin si
Fin pour ;
retourner S_{best};
Fin

Pour un meilleur sous ensemble avec cardinalité c, il cherche les meilleurs sous ensembles de cardinalité c+1 par l'ajout d'un attribut du reste des attributs.

Chaque nouveau sous ensemble S avec c+1 est évalué par une mesure d'indépendance M et comparé avec le meilleur précédent. Si S est le meilleur, il devient le meilleur sous ensemble actuel S'$_{best}$ à niveau c+1. A la fin de chaque itération, l'algorithme de recherche A est appliqué sur S'$_{best}$ à niveau c+1,
La qualité du résultat recherché O est comparée avec ceux du meilleur sous ensemble

à niveau c. si S'_{best} est le meilleur, l'algorithme continue pour trouver le meilleur sous ensemble pour le niveau suivant.

Autrement, il s'arrête et sort avec un meilleur sous ensemble actuel comme un meilleur sous ensemble final.

La qualité des résultats à partir de l'algorithme de recherche fournissent un critère d'arrêt naturel dans un modèle hybride.

B.3 Exemple:

Cet exemple présente une catégorisation de certains algorithmes de sélection d'attributs en trois dimensions **[Liu et Yu, 2005]**

			Stratégies de recherche					
			Complète		Séquentiel		Aléatoire	
Critères d'évaluation	Filter	Distance	B&B BFF		Relief ReliefF ReliefS SFS Segen's			
		Information	MDLM		DTM Koller's SFG FCBF	Dash's SBUD		
		Dépendance	Bobrowski's		CFS RRESET POE+ACC DVMM	Mitra's		
		Consistance	Focus ABB MIFES1 Schlimmer's		SetCover		LVI QBB LVF	
	Wrapper	la prédiction d'exactitude ou les bons clusters	BS AMB&B FSLC FSBC		SBS- SLASH WSFG WSBG BDS PQSS RC SS QUEIROS	AICC FSSEM ELSA	SA RGSS LVW RMHC-PF GA RVE	
	Hybride	Filter + Wrapper			BBHFS XING'S	Dash-Liu's		
			Classification	Clustering	Classification	Clustering	Classification	Clustering
			Les tâches de La fouille de données					

Tableau B.3: Cadre de catégorisation