N° d'ordre: 02/2015-D/INF

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Électronique et de l'Informatique

THESE

Présentée pour l'obtention du **grade** de **DOCTEUR EN SCIENCE**

En: INFORMATIQUE
Spécialité: Intelligence artificielle

Par: Chemchem Amine

Sujet:

De la fouille de donnes vers la fouille de connaissances: Apprentissage incrémental et approches multi-niveaux

Soutenue publiquement, le 21/02/2015, devant le jury composé de:

M Guessoum Ahmed	Professeur	à l'USTHB	président
Mme Drias Habiba	Professeur	à l'USTHB	Directrice de thèse
M Azzoune Hamid	Mitre de Conférence/A	à l'USTHB	Examinateur 1
M Boukhalfa Kamel	Mitre de Conférence/A	à l'USTHB	Examinateur 2
Mme Kamel Nadjet	Mitre de Conférence/A	à l'univ de Setif	Examinatrice 3
M Moussaoui Abdelouahab	Professeur	à l'univ de Setif	Examinateur 4

"Et dis : Ô mon Seigneur, accrois mes connaissances"
Sourate TA-HA (20/114) Coran

Remerciements

Au terme de ce travail, je tiens tout d'abord à remercier ALLAH, le clément et le miséricordieux qui m'a permis d'aboutir à cette fin tant souhaitée, en me dotant de la patience et du courage nécessaires pour surmonter les moments difficiles que j'ai rencontrés tout au long de mon cursus.

Je tiens aussi à exprimer mes remerciements les plus chaleureux à ma directrice de thèse : PR Drias Habiba pour sa précieuse aide, pour son suivi rigoureux et sa disponibilité pendant notre travail.

Que les membres du jury trouvent ici le témoignage de ma reconnaissance pour avoir bien accepté d'évaluer mon travail.

Merci pour tous les professeurs et mes collègues du laboratoire LRIA, pour les conseils et les expériences passées ensemble, notamment lors des organisations des conférences et doctorials tout au long de mon cursus, que des bons souvenirs avec une ambiance chaleureuse : Mme Aouat, Mme Boughaci, Mr Guessoum, et mes amis : Ilyes, Bachir, Mourad, Hadia, Marwa, Ibrahim sans oublier Riadh et Noureddine.

Un très grand merci à ma chère copine Aida, pour son soutient moral tout au long de mon travail, elle a toujours su me poussé vers l'avant.

A la fin, je tiens aussi à remercier du fond du cœur tous les membres de ma famille :Maman, Papa et mon fréro Wassim qui, je le crains fort, n'ont toujours rien compris à ce que je fais, mais m'ont toujours soutenu de leur attention et de leur amour. Je ne sais pas trop comment je pourrais effacer une telle dette.

Résumé L'ère actuelle est caractérisée par de gigantesques volumes de connaissances qui sont extraites de façon permanente du Web grâce à la robustesse d'outils intelligents, et plus précisément aux techniques de la fouille de données. Pour accélérer le processus d'extraction de méta-connaissances de façon efficace, nous serons confrontés à deux problèmes majeurs, à savoir :

- Comment fouiller cette masse faramineuse de connaissances de façon rapide et continue.
- Comment gérer le flux de connaissances produit constamment dans le Web sans avoir à redémarrer le processus de fouille à chaque arrivée de nouvelles connaissances.

Pour résoudre le premier problème nous proposons deux approches de fouille de connaissances multi-niveaux, après avoir présenté tout un concept théorique pour l'extension de la fouille de données à celle des connaissances. Pour le second problème, nous avons développé trois approches de fouille de connaissances incrémentale, la première est l'extension d'algorithme de fouille de données incrémentale, et les deux autres sont des approches de traitement de paquets de connaissances d'une manière progressive. Par la suite, une étude expérimentale est élaborée sur ces différentes approches afin de les comparer et d'en extraire les plus performantes en matière de temps de calcul, et du taux de réussite de la segmentation.

Une application fort intéressante consiste à projeter les résultats de ces travaux sur la technologie des agents cognitifs et de l'étendre au concept d'agents super intelligents. Dans ce contexte il s'agit d'agents sensés recevoir des quantités faramineuses de connaissances en une courte période pendant laquelle il leur est impossible de les exploiter. La solution apportée est d'organiser leurs connaissances dans des clusters en exploitant les approches de fouille proposées, afin de n'utiliser que les connaissances concernées dans le but d'engendrer une nouvelle connaissance.

Mots clés : Fouille de données - extraction de connaissances à partir de bases de données - représentation des connaissances - règle d'induction - fouille de connaissances - paradigme incrémental - approches multi-niveaux - algorithmes génétiques.

Abstract The era in which we live today is characterized by huge amounts of knowledge that are extracted permanently from the web, tanks to the robustness of artificial intelligence and especially to the techniques of data mining. To speed up the process of extracting only the useful knowledge (called meta-knowledge), in an efficient and continuous manner, we face two major problems ; the first one will be how to mine this huge mass of knowledge quickly and effeciently, and the second is how to manage the flow of knowledge which is constantly produced on the web without having to restart the mining process from the beggining at each arrival of new knowledge.

To solve the first problem, we first present a theoretical concept for the extension of data mining to deal with knowledge, followed by the proposition of two approaches of multilevel knowledge mining. For the second problem, we will propose three incremental approaches of knowledge mining, the first is an extension of an incremental data mining algorithm, and the two other approaches are based on packet processing of knowledge in a progressive manner. Thereafter, an experimental study will be applied to these different approaches in order to compare them and select the most efficient in terms of computation time and success rate.

A very interesting application would be to project the results of this study on the technology of intelligent agents, and extend it to the concept of super intelligent agents. In this context these agents will receive a huge amounts of knowledge in a short period of time during which it is impossible to exploit it. A solution would be to organize their knowledge using the approaches proposed of the knowledge mining to use only the relevant knowledge in order to generate new knowledge.

Keywords : Data mining, knowledge extraction, knowledge representation, induction rules, knowledge mining, incremental paradigm, multilevel approaches, genetic algorithms.

Table des matières

Liste des tableaux

Table des figures

Introduction

Avec l'évolution du Web, le monde actuel est caractérisé par de gigantesques volumes de données. Grâce à l'intelligence artificielle et aux technologies de l'information, ce problème fait l'objet d'investigation avec notamment les techniques de fouille de données et de l'extraction de connaissances.

Actuellement, même avec les techniques de fouille de données, le volume des connaissances extraites explose de façon exponentielle, et de manière continue, sans même savoir si ces connaissances sont intéressantes et pertinentes ou non.

Pour répondre à cette problématique, il serait judicieux d'explorer un nouveau champ sur la fouille de connaissances afin d'en extraire des méta-connaissances (des connaissances de plus haut niveau).

A ce niveau, nous serons confrontés à deux problèmes majeurs à savoir :
- Comment fouiller cette masse phénoménale de connaissances d'une manière rapide et efficace afin d'en extraire uniquement des méta-connaissances plus importantes et pertinentes.
- Comment gérer la continuité du flux de connaissances constamment en production sur le web, sans relancer le processus de fouille à zero à chaque arrivée de nouvelles connaissances.

Pour résoudre le premier problème, on proposera d'abord un nouveau concept de fouille de connaissances, en présentant de nouvelles préliminaires mathématiques pour les connaissances telles qu'une nouvelle mesure de similarité et de nouvelles formules de calcul du centre de gravité dans l'ensemble des connaissances, dans le but d'étendre les approches de fouille de données à celle des connaissances. Ensuite, et en se basant sur le paradigme multi-niveaux, deux nouvelles approches de fouille de connaissances multi-niveaux sont proposées afin de maitriser la grande masse de connaissances. La première est une appli-

cation directe du principe multi-niveaux sur la fouille de connaissances, et la deuxième est une hybridation avec une méta-heuristique.

Pour le deuxième problème, on fera appel à la méthode incrémentale, et proposera trois approches de fouille de connaissances incrémentale. La première est inspirée directement de la fouille de données incrémentale, et les deux autres se basent sur le principe de traitement par paquet de manière progressive.

Par la suite, on réalisera des études comparatives sur les approches proposées, afin d'en extraire les meilleures en terme de performance (taux de réussite, temps d'exécution).

Une application fort intéressante serait de projeter les résultats de l'étude sur la technologie des agents cognitifs et de l'étendre au concept d'agents super intelligents. Dans ce contexte il s'agit d'agents sensés recevoir des quantités faramineuses de connaissances en une courte période pendant laquelle il leur est impossible de les exploiter. Une solution serait d'organiser ces connaissances en utilisant les approches de fouille proposées, afin de n'utiliser que les connaissances concernées dans le but d'engendrer une nouvelle connaissance. Pour cela, on a organisé ce travail en sept chapitres :

Le premier chapitre introduit les approches et les techniques de fouille de données les plus utilisées. Le deuxième chapitre présente le passage de la notion de données vers celles des connaissances, ainsi que les différentes représentations de connaissances. Dans le troisième chapitre, nous passons en revue les paradigmes de résolution qu'on a adoptés dans ce travail, à savoir le paradigme multi-niveaux, et l'approche incrémentale. Le support théorique conçu pour asseoir de la fouille de connaissances est présenté dans le quatrième chapitre, en proposant une nouvelle mesure de similarité entre les connaissances de types règles, ainsi que de nouvelles formules de calcul du centre de gravité. Les approches incrémentales et multi-niveaux pour la segmentation des connaissances de type règles d'induction sont décrites respectivement dans les chapitre cinq et six, avec des études d'évaluation et d'expérimentation pour chacune d'elles. Enfin, dans le septième chapitre, nous exposerons une application de la fouille de connaissances multi-niveaux et incrémentale pour la technologie des agents. L'idée est d'intégrer un module de fouille de connaissances sur l'architecture d'un agent basé sur les règles d'induction pour obtenir un agent plus performant et plus évolué appelé : agent super-intelligent. Ce dernier est comparé avec l'agent

cognitif classique afin de montrer ses performance. Nous terminerons cette thèse avec une conclusion et quelque perspectives.

Première partie

État de l'art

Chapitre 1

La fouille de données et l'extraction des connaissances

1.1 Introduction

Ces dernières années, les technologies de l'information et de la communication (TIC) ne cessent d'évoluer, tout en fournissant d'immenses vagues de données aux différentes entreprises. De ce fait, de nouvelles approches d'extraction de connaissances sont nées.
Ces nouvelles approches, connues aussi sous le nom d'outils de KDD (KDD : Knowledge Discovery in Data_bases) ont pour but d'extraire des informations cachées dans cette grande masse de données. Ces informations sont très utiles pour l'aide à la décision, l'optimisation des requêtes... et sont aussi facilement compréhensibles par l'utilisateur [Fay96].
En analysant par exemple des données d'organismes de vente par correspondance avec ces approches, il est possible de regrouper les clients selon certains critères, ce qui permettra ensuite de limiter les coûts des « mailing », et de définir de manière plus précise pour chaque produit ces clients potentiels.
Historiquement, les premières conférences internationales sur le concept de KDD ont été organisés en 1995. Puis, grâce à l'évolution très rapide de l'intelligence artificielle, plusieurs techniques de fouille de données dans le processus de KDD ont vu le jour notamment les techniques d'apprentissage automatique, et les techniques de résolution des problèmes.
Dans ce chapitre, nous allons définir les principales étapes du processus KDD avec son historique, suivi d'un passage en revue des principales techniques de fouille de données.

1.2 Définition de l'exploration de données

S.Tufféry définit l'exploration de données, connue aussi sous l'expression de "fouille de données", "prospection de données", "data mining", ou encore "extraction de connaissances à partir de données", « ECD » en français, « KDD » en anglais, comme étant un ensemble de méthodes et techniques qui ont pour objet l'extraction d'un savoir ou d'une connaissance à partir de grandes quantités de données, avec un processus automatiques ou semi-automatiques [Tuf10].

Les méthodes de la fouille de données sont appliquées dans plusieurs domaines, on peut citer par exemple : la gestion de la relation client, la maintenance préventive, sans oublier la détection de fraudes, le domaine du marketing (comme les systèmes de création des profils de clients pour cibler des clients potentiels), le domaine de la finance (minimisation des risques financiers), la bio-informatique (analyse des génomes), le monde du Web (e-commerce et détection d'intrusions .etc).

De plus, plusieurs logiciels de fouille d'exploration de données sont devenus populaires selon leurs intérêts : SPAD, SAS, SPSS Clementine ,STATISTICA DATA MINER, IBM, Inteligent Miner, TANAGRA, SIPINA, WEKA, et ORANGE [Kan11] [Rak03].

1.3 Histoire de l'exploration de données

Au cinquième siècle av. J.-C, le pharaon Amasis organise le recensement de sa population provoquant ainsi une des premières collecte de données dans l'histoire de l'humanité [JC10].

Au XVIIe siècle, en 1763, Thomas Bayes montre qu'on peut déterminer, non seulement des probabilités à partir des observations issues d'une expérience, mais aussi les paramètres relatifs à ces probabilités. En 1805, Legendre publie un essai sur la méthode des moindres carrés qui permet de comparer un ensemble de données à un modèle mathématique.

Les années 1950 voient l'apparition de calculateurs encore onéreux et des techniques de calcul par lots sur ces machines. Simultanément, des méthodes et des techniques voient le jour telles que la segmentation, classification (entre autres par la méthode des nuées dynamiques), une première version des futurs réseaux de neurones qui se nomme le "Per-

ceptron", et quelques algorithmes auto-évolutifs qui se nommeront plus tard génétiques.
Dans les années 1960 les arbres de décision et la méthode des centres mobiles voient le jour, ces techniques permettent aux chercheurs d'exploiter et de découvrir des modèles de plus en plus précis. En 1969, Myron Tribus publie sont ouvrage : "Rational descriptions, decisions and designs" [Tri69] qui généralise les méthodes bayésiennes dans le cadre du calcul automatique, il le traduit en français par la suite en 1973 sous le titre "Décisions rationnelles dans l'incertain". Une idée importante qui faut noter dans cet ouvrage est que, parmi toutes les distributions de probabilité satisfaisantes aux observations (leur nombre est infini), il faut choisir celle qui contient le moins d'arbitraire (donc le moins d'information ajoutée), et en conséquence celle d'entropie maximale.
L'expression « data mining » avait une connotation péjorative au début des années 1960, exprimant le mépris des statisticiens pour les démarches de recherche de corrélation sans hypothèses de départ. Elle tombe dans l'oubli, puis Rakesh Agrawal l'emploie à nouveau dans les années 1980 lorsqu'il entamait ses recherches sur des bases de données d'un volume de 1 Mo. Le concept d'exploration de données fait son apparition, d'après Pal et Jain, aux conférences de l'IJCAI en 1989 [PJ05].
Dans les années 80, « Data mining » étant sous la protection d ?un copyright, Gregory Piatetsky-Shapiro employa l'expression "Knowledge Discovery in Databases" (KDD)[PS00].
Ensuite, dans les années 1990, viennent les techniques d'apprentissage automatique telles que les SVM [Tuf10] en 1998, qui complètent les outils de l'analyste. Au début du XXIe siècle, les grandes entreprises comme Amazon.com se servent de tous ces outils pour proposer à leurs clients des produits susceptibles de les intéresser.

1.4 Les facteurs d'émergence du KDD

Dans le passé, le souci majeur des sociétés était de fournir une large gamme de produits, plus large que celle de leurs concurrents, sans prendre en considération si ces produit répondaient réellement aux besoin de leurs clients, à ce moment là, le modèle d'économie était plutôt "orienté produit".
De nos jours, la concurrence est plus forte et les clients sont devenus plus exigeants. Dans cet environnement moderne, les entreprises sont attendues à offrir les biens et/ou services

qui répondent au mieux aux besoins du client, même les plus spécifiques. Nous sommes ainsi passés vers une économie "orientée client". Pour mieux répondre à la demande du client, la connaissance de son comportement est décisive, on fait illusion ici à la gestion de la relation client connue par (Customer Relationship Management (CRM)).
D'un autre point de vue, et depuis l'apparition des bases de données dans les années 60, l'augmentation des capacités des solutions de sauvegarde fait exploser le volume des entrepôts de données dans le monde. Une étude à estimé que sa taille double tous les vingt mois [Kod96]. Cette évolution exponentielle a été confirmée dans une étude de l'université de Californie de Berkeley qui fait état d'une augmentation de 114% entre 1999 et 2002 de la quantité d'information produite annuellement sur disque dur dans le monde [LVD+00]. La question qui se pose naturellement à ce stade est de savoir quoi faire de ces données, car leur collecte et leur maintenance ont malgré tout un coût, même modeste.
La figure1.1 montre bien cette évolution en terme de technologies et aussi en terme de volume des données. Nous somme passés depuis les bases de données relationnelles dont la taille atteignait quelques dizaines de gigaoctets, vers les entrepôts de données dont la taille se calcule en plusieurs dizaines de téraoctets.
On remarque bien que l'étude schématisée à la figure s'arrête à l'année 2005. De nos jours (année 2014) les grandes sociétés internationales sont passées au data-centers de plusieurs dizaines de yotta-octets, sachant que :
1 yottaoctet (Yo) = 2^{80} octets = 1 024 Zo = 1 208 925 819 614 629 174 706 176 octets.

1.5 Le processus de découverte de connaissances

Le processus de découverte de connaissances (KDD) connu aussi sous la nomination du processus d'extraction de connaissances à partir d'un entrepôt de données est un processus complexe, itératif, et parfois interactif (communication entre KDD et un expert) ; il est découpé en cinq étapes : sélection de données, nettoyage et enrichissement de données, codage de données, fouille de données, validation [HKP06], [MR05].
La figure1.2 présente l'enchainement de ces différentes étapes.

Avant de détailler chaque étape du processus d'extraction de connaissances, on donne une petite définition sur les données.

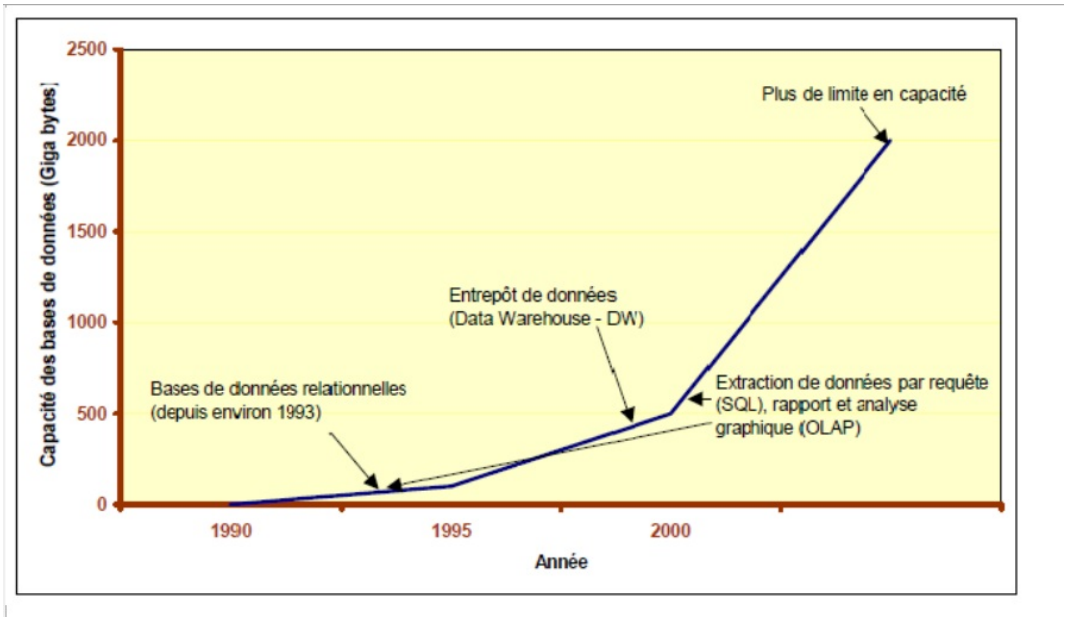

FIGURE 1.1 – *Évolution de la taille des bases de données [ZR02]*

Une donnée Les données sont représentées sous forme d'attributs et d'exemples, un attribut est un descripteur qui permet de représenter une caractéristique d'un exemple. Par contre un exemple est un ensemble d'attributs formant un individu, et l'ensemble d'individus est la population à explorer.

Par exemple : soit le problème de E-commerce :

La population est l'ensemble des clients.

Un individu est un client.

Types de données Les données à fouiller peuvent être représentées selon plusieurs types, on peut citer quelque uns : [Kan11].

- Numérique discrète : la valeur de la variable appartient à l'ensemble E (entier) ou N (Naturel) (par exemple : l'âge d'un étudiant x est 18).
- Numérique continue : la valeur de la variable peut prendre une valeur dans R (par exemple la moyenne d'un étudiant x est 11.36).
- Qualitative : avec ou sans relation d'ordre (par exemple :rouge, chaud,...).
- Binaire : la valeur peut prendre 0 ou 1 par exemple (0 signifie que l'étudiant est ajourné, 1 s'il est admis).

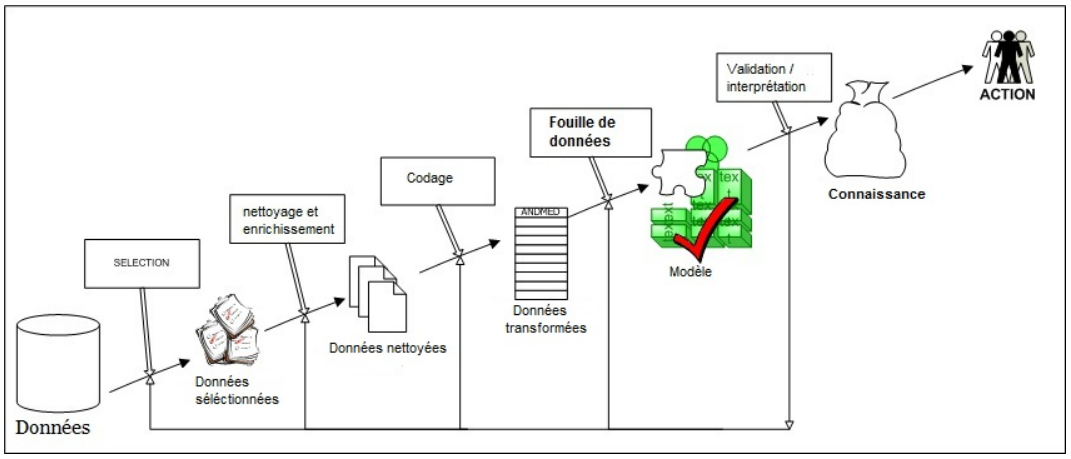

FIGURE 1.2 – *Les étapes du processus d'extraction de connaissances à partir des données[MR05]*

- Chaînes de caractères : la filière de l'étudiant X est : « informatique » par exemple.

1.5.1 La sélection / préparation de données

La préparation des données consiste dans un premier temps à obtenir des données en accord avec les objectifs que l'on s'impose. Ces données proviennent le plus souvent de bases de production ou d'entrepôts.
Après cette première étape, ces données sont copiées sur une machine adéquate, pour des questions de performance, mais surtout parce qu'elles seront exploitées (dans le cas des entrepôts de données).

1.5.2 Le nettoyage et l'enrichissement des données

Ce traitement consiste à corriger les erreurs contenues dans les données afin de les rendre plus fiables. La méthode la plus usuelle consiste à définir un espace compris entre la moyenne et l'écart type et à exclure toutes les données se trouvant à l'extérieur de l'espace. Souvent les données utilisées sont insuffisantes ; pour cela on est obligé de récupérer ou d'acheter d'autres bases de données produites dans un autre lieu.

1.5.3 Le codage de données

Il consiste à réaliser quelques traitements sur nos données afin de les rendre plus significatives. Cette partie prépare les données à la fouille et les choix de codage sont particulièrement guidés par le modèle de fouille utilisé. Il existe plusieurs techniques de codage telles que :

1.5.3.1 Le regroupement

Certains attributs prennent un très grand nombre de valeurs discrètes. C'est typiquement le cas des adresses. Lorsqu'il est important de considérer ces attributs pour la fouille de données, il est obligatoire d'opérer des regroupements et ainsi obtenir un nombre de valeurs raisonnable.

1.5.3.2 Le changement de type

Pour certaines manipulations, comme les calculs de distance et les calculs de moyenne, il est préférable de modifier les types de certains attributs. Par exemple, on peut convertir la date de naissance en âge.

1.5.3.3 L'uniformisation d'échelle

Certaines données sont très élevées par rapport à d'autres, pour ne pas avoir des perturbations en exploration, les échelles seront uniformisées par exemple la variable "revenu" est plus élevée que celle de l' "âge". L'idée consiste à diviser les valeurs du revenu par un entier x afin d'équilibrer les valeurs des attributs "âge" et "revenu".

1.5.4 La fouille de données

La fouille de données est le coeur du processus car elle permet d'extraire de l'information sur ces données. Néanmoins, c'est souvent une étape difficile à mettre en oeuvre, coûteuse et dont les résultats doivent être interprétés et relativisés.

1.5.5 La validation

Cette étape consiste à valider le modèle d'exploration. Il existe deux modes principaux de la validation : par expert ou automatique.
Pour certains domaines d'application (le diagnostic médical, par exemple), il est essentiel que le modèle produit soit compréhensible. Donc ici l'expert qui est le médecin est appelé à valider le modèle de fouille.
Dans le mode de validation automatique qui est utilisé souvent pour les problèmes d'apprentissage, on divise nos données en 3 ensembles : apprentissage, test et validation.
L'ensemble apprentissage peut réellement générer le modèle d'exploration. L'ensemble de validation permet de valider le résultat de l'ensemble apprentissage. Par ailleurs, l'ensemble test permet de valider le modèle mais après la phase de fouille.

1.6 Les tâches de la fouille de données

A des fins d'exploration des tâches de fouille de données, on se situe dans un environnement d'aide à la décision à partir de données. On suppose que de grandes quantités de données sont disponibles. En général, ces données sont structurées et correspondent aux enregistrements d'une table ou de plusieurs tables d'une base dédiée à la décision (Infocentre ou entrepôt de données) [HKP06]. Nous allons présenter, dans cette section, les tâches, i.e. les problèmes que l'on cherche à résoudre. Les tâches les plus connues de la fouille de données sont schématisées dans la figure1.3.

1.6.1 La classification

C'est un processus qui permet d'organiser un ensemble de données en classes. Le but de la classification est de former des groupes cohérents et isolés. Deux critères de classification de données sont définis comme suit :
Critère de cohérence : il faut que les données partagent les même similitudes au sein d'une classe.
Critère d'isolation : il faut que les classes soient aussi dissemblables que possible. Il existe principalement deux types de classification, que nous discuterons dans ce qui suit.

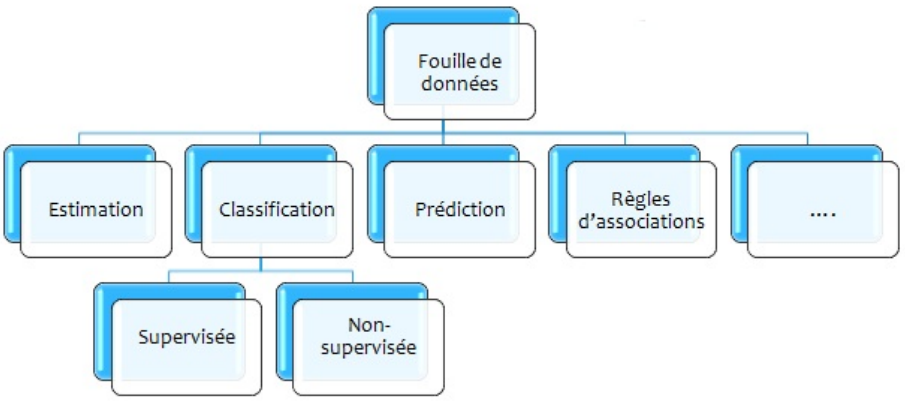

FIGURE 1.3 – *Les tâches de la fouille de données [HKP06]*

1.6.1.1 La classification supervisée

C'est une opération qui permet de placer chaque individu de la population dans une classe parmi l'ensemble de classes déjà établies en fonction des caractéristiques de cet individu. On peut la définir aussi comme étant l'opération qui consiste à examiner les caractéristiques d'un objet et lui attribuer une classe, la classe étant un champ particulier à valeurs discrètes. Des exemples de la tâche de la classification supervisée sont : attribuer ou non un prêt à un client, établir un diagnostic, accepter ou refuser un retrait dans un distributeur,...
Parmi les techniques utilisées : K plus proches voisins, réseaux de neurones, arbres de décision [Die98].

1.6.1.2 La classification non-supervisée (Clustering en anglais)

Connue aussi sous le nom de segmentation, cette opération consiste à former des groupes (clusters) homogènes à l'intérieur d'une population. Pour cette tâche, il n'y a pas de classes à expliquer ou de valeur à prédire définie a priori (contrairement à la classification supervisée). Ici il s'agit de créer des groupes homogènes dans la population (l'ensemble des enregistrements). Il appartient ensuite à un expert du domaine de déterminer l'intérêt et la signification des groupes ainsi constitués. Cette tâche est souvent effectuée avant les précédentes pour construire des groupes sur lesquels on applique des tâches de classification

ou d'estimation.
Il existe deux types de segmentation : le partitionnement réalisé souvent par l'algorithme très connu "K-means", et la segmentation hiérarchique dont l'algorithme le plus connu pour cette tache est appelé "HAC" (Hierarchical agglomerative clustering)[WKQ+08].

1.6.2 L'estimation

Cette tâche consiste à estimer la valeur d'un champ à partir des caractéristiques d'un objet. Le champ à estimer est un champ à valeurs continues. L'estimation peut être utilisée dans un but de classification. Il suffit d'attribuer une classe particulière pour un intervalle de valeurs du champ estimé. Des exemples de tâche d'estimation sont : noter un candidat à un prêt ; cette estimation peut être utilisée pour attribuer un prêt (classification), par exemple, en fixant un seuil d'attribution, estimer les revenus d'un client. La technique la plus utilisée ici étant les réseaux de neurones [Kan11].

1.6.3 La prédiction

La prédiction consiste à estimer une valeur future. En général, les valeurs connues sont archivées. On cherche à prédire la valeur future d'un champ. Cette tâche est proche des précédentes. Les méthodes de classification et d'estimation peuvent être utilisées en prédiction. Des exemples de tâche de prédiction sont : prédire les valeurs futures d'actions, prédire au vu de leurs actions passées les départs de clients [DWK05].

1.6.4 Les règles d'associations

L'extraction de règles d'association consiste à déterminer les attributs qui manifestent une sorte d'association par leurs valeurs. L'exemple type est la détermination des articles (le poisson et la boisson gazeuse ; la baguette, le camembert et de l'eau, ...) qui se retrouvent ensemble sur un même ticket de supermarché. Cette tâche peut être effectuée pour identifier des opportunités de vente croisée et concevoir des groupements attractifs de produits. C'est une des tâches qui nécessite de très grands jeux de données pour être effective. Cette tâche a engendré l'exemple (l'anecdote) suivant présent dans de nombreux articles sur le data mining : dans les supermarchés américains, il a été possible de mettre en évidence des

corrélations entre achat de bières et achat de couches-culottes pour bébé avant le week-end ! remarque justifiée par le comportement des jeunes pères américains qui préparent leur week-end en préparant leur provision de bière pour regarder la télévision et qui font les achats pour bébé au même moment. Une connaissance qui a été utilisée par la majorité des supermarchés, en éloignant le plus possible ces deux articles, afin que le client puisse visiter le maximum de marchandises possible. Les algorithmes les plus utilisés ici sont : l'algorithme a priori et l'algorithme d'arbre de modèle fréquent.

1.7 Les techniques de fouille de données

Comme présenté à la section précédente, la fouille de données connait plusieurs tâches, chacune d'elles peut se réaliser avec différents algorithmes et techniques, le tableau 1.1 résume chaque tache avec ses principaux algorithmes.

TABLE 1.1 – Tableau de techniques de fouille de données

La tâche de fouille de données	**Quelques techniques utilisées**
La classification supervisée	K-plus-proches-voisins, arbre de décision, réseaux de neurones, apprentissage bayésien...
La classification non supervisée (clustering)	K-means, CHA, fuzzy C-means, algorithmes génétiques basés sur K-means...
Estimation	Réseaux de neurones...
Prédiction	Arbre de décision, réseaux de neurones...
Règles d'associations	Algorithme a priori, Arbre de modèle fréquents...

Les algorithme KPPV, K-means, et apriori figurent dans le top 10 des algorithmes de fouille de données [WKQ$^+$08]. Dans ce qui va suivre nous apporterons les détails de chacune de ces techniques.

1.7.1 L'algorithme KPPV

L'algorithme des K plus proches voisins (kNN en anglais) est l'une des méthodes de classification supervisée des plus fondamentales et des plus simples. Il devrait être l'un des premiers choix pour une étude de classification quand il y a peu ou aucune connaissance préalable sur la répartition des données. L'algorithme K-plus-proches-voisins a été développé pour la première fois à l'école d'air force aux États Unis en 1951. Fix et Hodges ont introduit une méthode non paramétrique pour la classification de modèle qui a depuis connu la règle des k plus proches voisins [Che84]. Plus tard, en 1967, quelques nouvelles propriétés formelles de l'algorithme k-plus-proches-voisins ont été élaborées. En l'occurrence, il a été montré que pour $k = 1$ l'erreur de classification k-plus proches voisins est majorée par deux fois le taux d'erreur de Bayes [HIL68]. Une fois les propriétés formelles de la classification avec KPPV ont été établies, une longue lignée d'enquêtes s'ensuivit sur les autres approches de classification afin de les améliorer, notamment l'approche du raffinement avec amélioration du taux d'erreur de Bayes [FH75], les approches à distances pondérées [Dud76], [BJ78], les méthodes exactes et les méthodes floues [Jóź83].
En intelligence artificielle, l'algorithme de k plus proches voisins est une méthode d'apprentissage supervisé. Son principe est défini comme suit ; on dispose d'une base de données d'apprentissage constituée de N couples « entrée-sortie ». Pour estimer la sortie associée à une nouvelle entrée x, la méthode des k plus proches voisins consiste à prendre en compte (de façon identique) les k échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée x, selon une distance à définir. Par exemple, dans un problème de classification, on retiendra la classe la plus représentée parmi les k sorties associées aux k entrées les plus proches de la nouvelle entrée x.

1.7.1.1 Principe

L'algorithme KPPV

```
Debut
Paramètres:
- le nouvel individu «x».
```

```
- Un entier K entre 1 et n /* n le nombre d'individus déjà classés. */
     Pour chaque individu "indiv" déjà classé faire
        Calculer la distance D(x, indiv).
     fin_pour
  Extraire_les_kppv(x).
     pour chaque indiv appartient à kppv(x) faire
        Calculer le nombre d'occurrence de chaque classe.
     fin_pour
  Attribuer à "x" la classe dont le nombre d'occurrence est maximal
Fin
```

1.7.1.2 Calcul de Complexité

Soit n le nombre d'individus déjà classés. La complexité de l'algorithme k-plus-proche-voisins est $O(n)$ parce qu'on fait n itérations dans la première boucle et k itérations dans la deuxième boucle tel que k $\leq$ n.

1.7.1.3 Exemple

Soit un ensemble d'individus déjà classés comme montré dans le tableau 1.2.

Individu	age	Grade	Classe
E_1	24	5	2
E_2	22	5	2
E_3	21	5	3
E_4	21	4	2
E_5	27	4	1

TABLE 1.2 – Exemple de données pour l'algorithme KPPV

En utilisant l'algorithme k-plus-proches-voisins avec la distance de Manhattan, On veut classifier le nouvel individu "x" ayant un âge de 22 et un grade de 4. Pour K (le nombre de voisins) = 3 : On calcule les distances entre l'individu "x" avec tous les individus déjà classés comme suit :

D(x, E_1)=|24-22|+|5-4|=3, D(x, E_2)=1, D(x, E_3)=2, D(x, E_4)=1, D(x, E_5)=5.
Après les 3 plus proches voisins de l'individu "x" sont sélectionnés, à savoir E_2, E_3, E_4. Leurs classes sont respectivement : 2,3,2.
Classe(E_2) = classe(E_4)=2 alors fréquence de la classe 2 est « 2 ».
Classe (E_3)= 3 alors fréquence de la classe 3 est « 1 ». Donc la classe de l'individu "x" selon l'algorithme KPPV est la classe 2.

1.7.2 L'algorithme K-means

On rappelle que le terme "k-means" a été employé la première fois par James MacQueen en 1967 [M$^+$67], bien que l'idée de base revient à Hugo Steinhaus en 1957. En 1965, E.W.Forgy a édité essentiellement la même méthode pour l'améliorer, et c'est pourquoi on trouve sur quelques références la méthode de classification sous le nom de Lloyd-Forgy. Une version plus efficace a par la suite été proposée et publiée en Fortran par Hartigan et Wong en 1975/1979 [Har75], [HW79].
L'algorithme des k-moyennes (ou K-means en anglais) est un algorithme de partitionnement de données relevant des statistiques et de l'apprentissage automatique et plus précisément de l'apprentissage non supervisé. C'est une méthode dont le but est de diviser des observations en K partitions (clusters) dans lesquelles chaque observation appartient à la partition avec la moyenne la plus proche. Les nuées dynamiques sont une généralisation de ce principe, pour laquelle chaque partition est représentée par un noyau pouvant être plus complexe qu'une moyenne. Le principe de base de l'algorithme classique K-means est simple, il est défini comme suit.

1.7.2.1 Principe

L'algorithme K-means

```
Notation : Gi : centre de gravité du groupe i.
 1-choisir k centres initiaux C1,C2,...,Ck.
 2-pour chaque objet : l'affecter au groupe i dont le centre est le plus proche.
 3-si aucun élément ne change de groupe alors arrêt et sortir.
```

```
4-calculer les nouveaux centres Gi pour tous Ci tel que Gi est la moyenne
  des éléments du groupe Ci.
5-aller à 2.
```

1.7.2.2 Calcul de Complexité

Soit n le nombre d'individus dans une population et k le nombre de clusters. La complexité de l'algorithme K-means standard est O (n^k log (n)) par ce qu'au pire des cas chaque individu visite tous les clusters possibles, calculant ainsi toutes les combinaisons possibles de tous les groupes avec tous les individus.

1.7.2.3 Exemple

Soit l'ensemble des points suivants dans un espace euclidien A(1,3),B(2,2),C(2,3),D(2,4),E(4,2). En appliquant l'algorithme k-means avec k=2, la distance euclidienne comme mesure de similarité et comme centres initiaux les points D et B sont choisis aléatoirement.

C1=$\{B\}, C2 = \{D\}; g1 = (2, 2), g2 = (2, 4)$.

$d(A, g1) = \sqrt{2}$=d(A,g2)=$\sqrt{2}$ ⇒ on met A dans C1 ou C2, on choisit par exemple C1.
d(B,g1)=0 < d(B,g2)⇒ on met B dans C1.
d(C,g1)=1=d(C,g2)=1 ⇒ on met C dans C1 ou C2, soit C1.
d(D,g2)=0<d(D,g1)⇒ on met D dans C2. d(E,g1)=2<d(E,g2)=$\sqrt{8}$ ⇒ on met E dans C1.
A la fin de la première itération on aura C1= $\{B, A, C, E\}, C2 = \{D\}$.

Les groupes sont modifiés par rapport à leur état initial alors on continue, mais avant il faut calculer les nouveaux centres de gravité :
g1=((1+2+2+4)/4 , (3+2+3+2)/4)= (2.25 , 2.5).
g2= (2/1 , 4/1)= (2,4).
On répète ce processus jusqu'à ce qu'aucun élément ne change de groupe. A la fin on aura :
C1= {A, B, C, D}, C2={E}.

1.7.3 L'algorithme Apriori

L'extraction des règles d'association est le processus de découverte d'un ensemble de règles pertinentes entre attributs à partir d'une base de transactions. Chaque transaction est un ensemble d'items alors qu'un item est la donnée la plus élémentaire du problème rencontré. A titre d'exemple : un item peut être considéré comme un produit pour le problème d'analyse du panier de consommation (Market Basket Analysis) [OLW08], comme un terme pour le problème de recherche d'information [WWT99], ou comme un génome pour le problème de classification de génotypes [TD08].

Le processus d'extraction des règles consiste à trouver les règles les plus représentatives des données incluses dans la base de transactions. Apriori est l'algorithme le plus utilisé pour l'extraction des règles d'associations. Il est classé parmi les dix meilleurs algorithmes de fouille des données [WKQ+08]. Il est principalement composé des deux étapes suivantes :

La génération des itemsets fréquents : Les itemsets fréquents sont déterminés d'une manière récursive de telle sorte que les itemsets de taille "k" sont générés à partir des itemsets de taille (k-1). Ce processus permet la réduction proportionnelle de l'espace de recherche des itemsets fréquents. Cependant, on perd beaucoup de temps en terme de calcul des itemsets fréquents. En effet, à chaque fois qu'un itemset est généré, son support doit être calculé. S'il est supérieur à un support minimum (Minsup) alors, l'itemset sera inséré dans la liste des itemsets fréquents. Ce processus se répète jusqu'à ce qu'il n'y ait plus d'itemsets fréquents à produire.

La génération des règles d'association : La génération des règles se fait à partir des itemsets fréquents, déjà trouvés dans la première étape. Pour chaque itemset fréquent, on génère toutes les règles correspondantes. Par la suite, pour chaque règle, on calcule sa confiance. Si sa confiance est supérieure à MinConf alors la règle est acceptée, sinon elle est rejetée.

Par conséquent, de nombreux algorithmes basés sur Apriori ont été proposés, Park et_al. [PCY95] développent "the Direct Hashing Pruning (DHP)" qui est une extension de l'algorithme Apriori utilisant une mémoire supplémentaire afin de calculer à l'avance les 2-itemsets fréquents en cours de la première itération. Aussi, le DHP réduit progres-

sivement la taille de la base de transactions en éliminant à chaque fois les itemsets non fréquents.
L'algorithme DIC(Dynamic Itemsets Counting) proposé par S.Brin et d'autres [BMUT97], est la généralisation de l'algorithme apriori où la base de transactions est partitionnée en plusieurs parties de même taille. Initialement, les supports d'items sont calculés pour la première partition. Les items trouvés localement sont utilisés pour générer les candidats 2-itemsets. Ensuite, la deuxième partition est lue afin de trouver le support de tous les candidats actuels. Ce processus est répété pour les autres partitions. Après le traitement de la dernière partition, le même processus se répète jusqu'à ce qu'aucun candidat ne puisse être généré.
Par la suite, plusieurs travaux sur l'extension de l'algorithme Apriori ont vu le jour, notamment Ashok Savasere et d'autres [SON95] qui ont proposé l'algorithme de partition à deux passes, qui divise logiquement la base de transactions en partitions totalement disjointes.
Aussi, l'algorithme (SEAR) d'Andreas Mueller [Mue98] est identique à Apriori, sauf que SEAR stocke les candidats dans une structure appelée "prefix tree" au lieu de la structure "hash tree".

1.8 Conclusion

Depuis l'explosion des capacités de stockage informatique au moins à partir du début des années 1990, la question de l'analyse de grands volumes de données s'est imposée.
L'extraction de connaissances à partir de données (Knowledge Discovery in Databases (KDD) en anglais) est le domaine de recherche tentant de répondre à cette question en mettant au point des outils de fouille de données capables d'analyser les grands volumes de données.
Nous avons vu précédemment que le KDD est un processus constitué de plusieurs étapes allant de la sélection et la préparation des données jusqu'à l'interprétation des résultats, en passant par la phase de recherche des connaissances « data mining ». En effet, le data mining se réfère à une étape particulière dans le processus KDD pour l'application des algorithmes spécifiques pour l'extraction de motifs à partir des données.
Le principe du processus KDD est la découverte non triviale, à partir de données, d'une

information implicite, précédemment inconnue et potentiellement utile. Une telle « information » extraite d'une base de données devient alors une « connaissance », qui doit être actionnable c-à-d. une connaissance avec une bonne information, détectée au bon moment, et destinée à la bonne personne.

Le résultat du processus KDD est souvent une certaine quantité de connaissances dont la majorité ne sera pas utile pour le décideur. Ce que nous proposons dans notre projet de recherche, c'est un nouveau concept de fouille de connaissances, afin d'en extraire des meta-connaissances, qui seront de petite taille mais surtout avec une plus grande importance. Le problème majeur est la complexité et la diversification de la représentation des connaissances existantes. La modélisation de ces dernières sera plus évoluée et plus complexe que celle de la fouille de données. Dans le chapitre qui suit, nous allons présenter les différentes représentations de connaissances rapportées par la littérature.

Chapitre 2

La représentation des connaissances

2.1 Introduction

Le challenge d'aujourd'hui des grandes sociétés est de définir la façon de gestion de leurs connaissances, ce phénomène appelé en anglais « Knowledge Management » (KM) est devenu indispensable pour ces entreprises pour plusieurs raisons. D'une part, le développement continu des sociétés, et leurs objectifs de plus en plus audacieux obligent ces dernières à être beaucoup plus compétitives qu'avant, et de ne pas redémarrer depuis le départ à chaque passage à un nouveau projet. Et d'un autre coté, l'évolution rapide de la communauté de l'intelligence artificielle et plus précisément les outils du data mining, produisent de plus en plus de connaissances, la grande masse de ces derniers a rendu la tâche de sélection des connaissance intéressantes et utilisable très pénible.

Pour résoudre ces problèmes, nous essayons dans ce travail de concevoir un nouveau concept de fouille de connaissances afin d'en extraire des méta-connaissances. Ces méta-connaissances qui sont des connaissances sur les connaissances sont bien évidemment pertinentes et très intéressantes pour la prise de décision.

Dans la suite de ce chapitre, nous mettons en relief la différence entre les données et les connaissances, ainsi que les différents types de connaissances.

2.2 Des données aux connaissances

Plusieurs chercheurs se sont intéressés à la distinction entre données, informations, et connaissances dans le domaine des sciences cognitives, où l'exploitation d'informations diverses par un système complexe permet l'acquisition de connaissances capables de diriger la mise en oeuvre d'action. Nous nous limiterons aux définitions acceptées de manière générale dans le domaine de l'informatique exprimées par Tuomi de la façon suivante [Tuo99] : Une données est définie comme étant le résultat d'observation, les informations comme étant le résultat de l'interprétation de ces données. Tandis que les connaissances définissent la façon d'utiliser ces données et ces informations. Cette distinction a été reprise par Devlin, Schreiber, et Wille qui l'ont représentée d'une façon plus formelle de la manière suivante [Dev00] [SAA+99] [Wil02] : Les données sont égales à des signes plus une syntaxe. L'information est définie comme étant des données plus un sens (ou sémantique). Tandis que la connaissance est égale à une information assimilée et interprétée plus une possibilité de mise en action de cette information.

Si on prend un exemple du domaine de la génétique, présenté dans le travail de A. Coulet [Cou08] et qui étudie la séquence d'ADN constitutive d'un gène au c ?ur d'une cellule. La séquence d'ADN est représentée sous forme d'une suite de plusieurs milliers d' A, C, G, et T. Cette suite de lettre peut être considérée comme une donnée brute. Mais le fait qu'on sache que cette séquence représente un gène particulier est une information. Enfin, le code génétique de la cellule constitue les connaissances qui permettent d'interpréter ce gène comme une protéine, utilisée par la suite par différentes fonctions biologiques.

Les données, les informations, et les connaissances peuvent être représentées selon les formes suivantes :

Selon G. Simoni [Sim05] les données sont des signaux matériels externes produits par des événements, ce sont des faits brutes et objectifs.

- L'information : Contient un formalisme de sens pour ces données, grâce à une opération d'interprétation, donc il s'agit d'un ensemble de données porteur de sens.
- La connaissance est représentée sous la forme d'une base ordonnée et signifiante d'informations et de compréhensions, susceptibles d'être transformées en actions.

En d'autres termes :

- Les données : sont souvent représentés par un nombre, une chaîne de caractères,... par exemple la chaine de caractère suivante : "ATCGGCTAGCTTATATCGATCGAT".
- L'information : souvent représentée par des données dans une base de données, ou sous forme de tableau associé aux méta-données nécessaires à leur interprétation : généralement sous la forme d'un couple (attribut = valeur), comme par exemple :
"séquence_du_gène = ATCGGCTAGCTTATATCGATCGAT".
- Les connaissances : sont de nouvelles informations acquises par un processus intelligent. On peut les définir comme des contraintes, des règles, des axiomes logiques utilisables par des programmes pour exploiter les informations dans le cadre de la réalisation d'une action : par exemple l'aide à la décision, la découverte de nouvelles connaissances. Dans l'exemple précédent on aura la règle suivante :
Si "séquence_du_gène = ATCGGCTAGCTTATATCGATCGAT" alors "l'auteur = Marc".

La figure 2.1 schématise le processus de transformation de données en informations puis en connaissances. A gauche un processus en pyramide et à droite un autre en boucle. La lettre C représente les connaissances.
On remarque bien que le processus est représenté sous forme de pyramide où les connaissances occupent la place la plus haute pour souligner le fait que plusieurs données sont nécessaires pour l'obtention d'une seule connaissance [Tuo99] [Cou08].

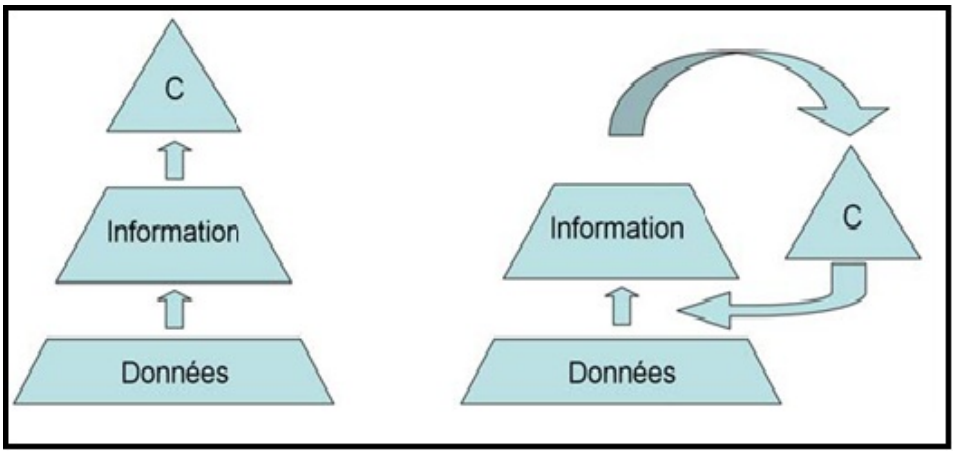

FIGURE 2.1 – *Des données aux connaissances [Cou08]*

2.3 Définition et types de connaissances

Dans la littérature, plusieurs définitions ont été attribuées au connaissances, selon le domaine, l'environnement et l'importance de ces dernières.
On peut citer quelques unes comme suit [Tit12] :

- Définition 1 : D'après le dictionnaire LAROUSSE : une connaissance est un nom féminin qui veut dire une maitrise dans un domaine.

- Définition 2 : Selon le dictionnaire le ROBERT : « ce qui est connu, est présent à l'esprit ; ce que l'on sait pour l'avoir appris ».

- Définition 3 : Selon le livre intitulé "Conceptual knowledge discovery in databases using formal concept analysis methods", « un ensemble de représentations, idées ou perceptions acquises par l'étude ou par l'expérience » [SWW98].

- Définition 4 : selon le livre de I.Nonaka et H.Takeuchi : « la connaissance est du savoir, du savoir-faire, de l'expérience, voir du savoir-être. La connaissance peut être tacite ou explicite » [NT97].

Il existe plusieurs manières de classifier les connaissances. On peut distinguer principalement deux grandes classes qui sont : les connaissances tacites et les connaissance explicites [NK98] [Wya01] [Tit12].
Le tableau 2.1 montre bien la différence entre les deux types.

2.3.1 La connaissance explicite

Les connaissances explicites sont des connaissance que l'on peut rendre explicite aux autres personnes au moyen d'une déclaration verbale, ou sous un langage formel et structuré ...etc.
Parmi les exemples des connaissances explicites, on peut citer : le code de la route, les règles de déplacement des pièces du jeux d'échecs...etc . Selon G. Stumme et al. « les connaissances explicites se réfèrent à la connaissance qui peut être exprimée (formalisée) sous forme de mots, de dessins, d'autres moyens "articulés" notamment les métaphores » [SWW98].

TABLE 2.1 – Type de connaissances

Connaissances tacites (subjectives)	**Connaissances explicites (objectives)**
- Idées, intuitions et pressentiments.	- Formelles et systématiques.
- Connaissances de l'expérience (corps).	- Connaissances de la rationalité (mental).
- Pas facilement visibles et exprimables.	- peuvent être exprimées en mots et en chiffres.
- Très personnel, difficile à formaliser, difficile à communiquer ou partager avec les autres	facilement communiquées et partagées forment des données brutes, formules, procédures codifiées...
- Enraciné dans les actions et les expériences de l'individu, y compris les idéaux, les valeurs, ou les émotions	- Peuvent être exprimées dans un code de calcul, formule chimique, ensembles de principes généraux.

2.3.2 La connaissance tacite

La connaissance tacite englobe une variété de phénomènes, comme la capacité à reconnaître quelque chose (par exemple le visage d'une personne), même si on ne peut pas décrire en termes de contexte indépendants comment on la reconnaît (par exemple sans dire, "Je sais que Amine ressemble à ceci"). Donc c'est une connaissance personnelle acquise par l'expérience et qui est très difficile voir quasi impossible à la représenter sous forme de code ou un langage formel.
Selon J. Senker "Le transfert de la connaissance tacite se réalise principalement par la collaboration et l'interaction des individus. C'est à travers cette interaction que les individus peuvent avoir de nouvelles idées et innover" [Sen95].

2.4 Interprétation et représentation des connaissances

L'interprétation des connaissances est un processus qui permet de transformer les résultats brutes du processus d'extraction des connaissances à partir des bases de données

KDD (vue précédemment en chapitre 1) en unités de connaissances.
La forme des unités extraites par le processus KDD est différente selon la méthode de fouille utilisée : motif fréquent, concept formel, règles d'associations, cluster ...etc.
Afin de faciliter l'interprétation, les résultats sont transformés pour faire l'objet d'une visualisation graphique par exemple sous la forme d'un arbre de décision, d'une hiérarchie de groupes (clusters), d'un réseau de neurones, d'un graphe de concepts...etc.
La représentation des connaissances revient à établir une correspondance entre le monde extérieur et un formalisme symbolique qui peut être traité par une machine. Le domaine de la connaissance est trop vaste et varié pour être représenté et exploité par un formalisme unique. De ce fait plusieurs représentations sont utilisées.
D'une façon générale, on peut dire que la représentation de la connaissance est une formalisation des croyances vraies au moyen de figures, d'écritures ou de langages. Il existe plusieurs approches de représentation de connaissances dont l'approche procédurale, et l'approche déclarative, comme montré à la figure2.2.

2.4.1 L'approche procédurale

Invoque la simplicité et la facilité de compréhension du raisonnement représenté par des algorithmes simulant le comportement réel.
La représentation procédurale permet, de traiter des problèmes de type algorithmique c'est à dire complètement analysés et entièrement connus.

2.4.2 L'approche déclarative

Cette approche est plus flexible car elle propose des heuristiques exprimées à l'aide d'énoncés. La représentation déclarative permet de spécifier un savoir indépendamment des contraintes et des méthodes d'utilisation.
Les connaissances sont classées par rapport à leurs natures de représentation : procédurales ou déclaratives [Tuo99].

La figure 2.2 présente les divers formalismes de représentation des connaissances allant du procédural (plus figé, plus structuré) au plus déclaratif (plus ouvert, plus libre).
Les représentations peuvent être regroupées aussi dans les trois grandes classes suivantes :

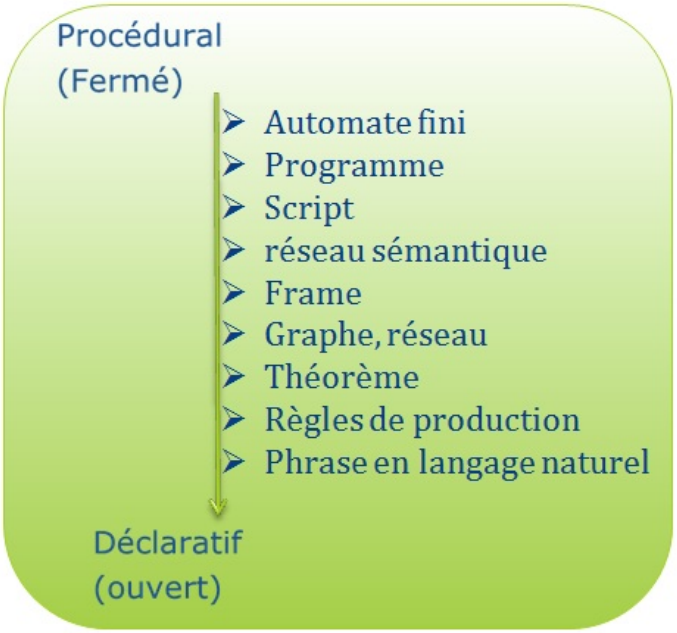

FIGURE 2.2 – *Les divers formalismes de la représentation des connaissances [Tuo99]*

- Procédurale : incluant les automates finis et les programmes.

- Déclarative : comprenant le calcul des prédicats et les règles de production.

- Structurée : contenant les réseaux sémantiques, frames, schémas, scripts, objets.

Certains modèles de représentation sont plus aptes à modéliser une connaissance fortement déductible ; d'autres conviennent mieux à une connaissance descriptive et enfin d'autres encore à une connaissance structurée.
Plusieurs approches générales sont donc utilisées pour représenter les connaissances, comme vu dans la figure2.2. Il s'agit notamment de : règles de production, hiérarchies d'inclusion, logiques mathématiques, cadres, scripts, réseaux sémantiques, contraintes et bases de données relationnelles.

- Les règles de production : c'est une méthode générale qui est particulièrement appropriée lorsque les connaissances sont orientées vers l'action, (appelées aussi règles d'induction ou règles d'inférence).
- Les hiérarchies d'inclusion : gèrent très bien un type particulier de connaissances : les connaissances sur les objets qui peuvent être regroupés en catégories, certaines ca-

tégories pouvant être divisées en sous-catégories. Les hiérarchies d'inclusion peuvent être utilisées comme un système d'organisation de relations avec d'autres méthodes, telles que le calcul des prédicats.

- Les logiques mathématiques : comme le calcul des prédicats fournissent une capacité générale et fondamentale qui supporte une inférence logique globale. Toutefois, ces logiques fournissent rarement un support organisationnel pour le groupement des faits de manière à ce que ces faits puissent être utilisés efficacement. En ce sens, les logiques mathématiques sont des systèmes de représentation de "bas niveau" qui fonctionnent bien pour les détails mais nécessitent un support supplémentaire pour être utilisés dans la construction de systèmes non triviaux.
- Les cadres : fournissent uniquement un système d'organisation de bases de connaissances. Par conséquent les représentations détaillées nécessitent d'autres méthodes.
- Les scripts : ont été utilisés dans certains systèmes expérimentaux pour la compréhension du langage naturel afin de représenter les scénarios avec la chronologie standard comme par exemple : lorsqu'une personne va au restaurant : elle obtient une table, attend pour le menu, passe la commande, mange, paye la facture, et s'en va. Les scripts sont comme des cadres avec un support supplémentaire pour décrire la chronologie.
- Les réseaux sémantiques : c'est une structure d'organisation générale, (pareil aux cadres) mais il n'y a pas nécessairement de type particulier de support de bas niveau dans un système de réseau sémantique. Tout système dans lequel les modules de connaissances peuvent être décrits comme des noeuds dans un graphe étiqueté, peut être appelé "réseau sémantique". Ils peuvent cependant être des systèmes qui tentent d'imiter la structure d'interconnexion des neurones du cerveau biologique en étant le plus souvent étiquetés par leurs créateurs comme des « réseaux sémantiques ».
- Les contraintes : c'est un type de connaissances qui est souvent décrit comme une méthode de représentation. Une contrainte est une relation entre un, deux ou plusieurs objets, et peut être considérée comme un prédicat. La contrainte doit être satisfaite par le système afin d'arriver à la solution d'un problème. En insistant sur l'utilisation de contraintes dans la représentation d'un ensemble d'objets et de leurs interrelations,

une approche basée sur les contraintes à la représentation des connaissances peut être utilisée.

- Les bases de données relationnelles : Peuvent parfois servir comme une méthode de représentation des connaissances, comme elles sont généralement mises en oeuvre. Elles permettent de manipuler de grandes quantités d'informations de structure régulière dans certaines formes largement prédéterminées. Au paravent, les bases de données relationnelles n'ont pas été concluantes pour les applications d'IA en raison de leur inefficacité à traiter un grand nombre de petites déductions tant sur de très petites relations que sur de petites parties de relation plus grandes. Il y a actuellement des recherches pour concevoir des bases de données relationnelles plus adaptées aux applications d'IA.

Dans notre étude, on s'intéresse à la représentation déclarative (la plus évoluée possible), plus précisément sous forme de règles de production, car elle permet de représenter n'importe quelle connaissance au monde de façon compréhensive, facile à manipuler, et qui se rapproche beaucoup plus des phrases du langage naturel ; ces dernières sont très difficiles voire impossible à structurer afin de les manipuler.
A noter aussi, que dans cette section nous avons ignoré la représentation de connaissances en concepts ontologiques, du fait que parmi les applications du présent travail, la découverte de relations entre les connaissances peut en être une.

2.5 Connaissances de type règles d'induction

Les règles d'induction sont une des approches les plus répandues pour la représentation des connaissances. Appelées aussi règles de production, règles d'inférence, ou encore règles d'expertise ; elle peuvent prendre plusieurs formes, comme par exemple :

- SI condition ALORS action
- SI prémisse ALORS conclusion
- SI proposition P1 et proposition P2 sont vraies ALORS Proposition P3 est vraie

Un des avantages des règles d'induction est qu'elles sont modulaires. Chacune d'elles définit (en principe, relativement) une pièce indépendante de la connaissance. De nouvelles règles peuvent être ajoutées et les anciennes supprimées généralement indépendamment

des autres règles [DM94].

Un système à base de règles contient des règles et des faits sur le domaine de la connaissance couverte. Lors d'une exécution particulière du système, une base de connaissances locale peut également être établie. Un des exemples des tutoriels les plus largement utilisés dans les systèmes de règles est Mycin, qui est un système expert conçu pour aider les médecins dans le diagnostic et le traitement de l'infection bactérienne. Il utilise l'approche à base de règles et démontre la manière dont l'incertitude (à la fois dans les observations et dans le processus de raisonnement) peut être traitée également .

Mycin a été conçu pour aider le médecin à décider si un patient a une infection bactérienne, quel organisme en est responsable, quel médicament peut être approprié pour cette infection, et doit être utilisé pour traiter spécifiquement le patient. Cette base de connaissances mondiale contient des faits, et des règles relatives à des symptômes d'infections. Citons une règle typique de Mycin :

```
IF   the identity of the germ is not known with certainty
     AND the germ is gram-positive
     AND the morphology of the organism is "rod"
     AND the germ is aerobic
  THEN there is a strong probability (0.8) that the germ is of type
  enterobacteriacae
```

Une connaissance de type règle d'induction peut être définie aussi comme une implication de la forme : $R : X \Rightarrow Y$. Où X et Y sont des clauses disjointes, c à d :$X \cap Y = \emptyset$. L'ensemble X est appelé la partie prémisse de la règle, l'ensemble Y est sa conséquence [DBS77], [CW94].

2.6 Conclusion

Dans ce chapitre nous avons montré la différence entre une donnée et une connaissance. Aussi, nous avons vu que pour obtenir une connaissance dans un domaine quelconque il faut plusieurs données au départ. Les connaissances sont orientées action et peuvent être représentées sous plusieurs formes. La représentation en règles d'induction est le plus proche

formalisme des phrases du langage naturel.

Notre travail consiste à proposer un concept de fouille de connaissances. Puis, dans la deuxième partie de cette thèse, nous montrerons comment adapter les techniques de fouille de données aux connaissances de type règle, en proposant des nouvelles préliminaires mathématiques, à savoir une nouvelle mesure de similarité, et une nouvelle formule de calcul du centre de gravité.

Chapitre 3

Approche multi-niveaux et fouille incrémentale de données

3.1 Introduction

Dans ce chapitre, nous allons présenter un aperçu sur les paradigmes incrément et multi-niveaux, ainsi que les principaux travaux qui les ont adoptés à travers l'évolution du domaine de l'intelligence artificielle pour la résolution des problèmes, sachant qu'a priori les approches multi-niveaux sont fiables pour les problèmes de grande échelle [JLT03], tandis que le paradigme incrémental est utilisé souvent pour les problèmes en exécution continue, tels que des applications sur le web constamment en activité [NXC+07].

3.2 L'approche incrémentale

Ces dernières années, avec l'évolution du Web, les tailles des bases de données sont devenues gigantesques, et le volume utilisé dans la fouille de ces données est constamment en augmentation. Les algorithmes de fouille exécutés à chaque mise à jour de ces bases de données sont excessivement couteux en matière de complexité. De ce fait, on est amené à penser à des algorithmes robustes et efficaces pour faire face à ce problème et diminuer sa complexité.

3.2.1 État de l'art

De la problématique précédente, l'approche incrémentale est née. Dans les algorithmes de fouille incrémentale, le taux de complexité est proportionnel à la taille de données de la mise à jour. Plusieurs chercheurs, dans le passé récent, ont développé des approches incrémentales pour les différentes tâches de fouille de données. Par exemple, des algorithmes d'extraction de règles d'associations d'une manière incrémentale dans les travaux [CHNW96],[LLC01],[GZ04], ou des algorithmes de classification incrémentale comme dans les travaux [XWZ00],[WJ04], [LHC04].

La segmentation incrémentale est le processus de mise à jours d'un ensemble de classes déjà existant, plutôt que de refaire la segmentation à partir du début à chaque mise à jour de la base de données. Un bref aperçu du travail effectué sur la segmentation incrémentale est résumé dans ce qui suit.

COBWEB proposé par Fisher [Fis87], est un algorithme de segmentation incrémentale qui construit une taxonomie de classes sans avoir un nombre prédéfini de ces classes. Les auteurs dans [GLF89] ont proposé CLASSIT, qui associe des distributions normales avec les noeuds d'un cluster. L'inconvénient majeur de COBWEB et CLASSIT est que les deux méthodes résultent en arbres très déséquilibrés [GGV$^+$11].

Dans [CCFM97a] de nouveaux algorithmes déterministes et aléatoires de segmentation sont introduits tout en essayant de réduire au minimum les diamètres maximaux des clusters. Le diamètre d'un cluster est la distance maximale entre ses points, il est utilisé dans le processus de restructuration des clusters. Quand un nouveau point arrive, il est soit affecté à l'un des groupes actuels, ou bien il initialise son propre cluster tandis que deux clusters existants sont combinés en un seul.

EN 1998 les auteurs de [EKS$^+$98a] ont présenté Incremental DBSCAN ; un algorithme approprié à l'analyse de données dans un environnement d'entrepôts de données. Incrémental DBSCAN est basé sur l'algorithme DBSCAN qui repose sur la segmentation à base de densité. Il utilise l'arbre R* comme une structure d'index afin d'exécuter des requêtes de région. En raison de son traitement à base de densité, dans Incremental DBSCAN les effets de l'insertion et la suppression d'objets ne se limitent qu'au voisinage de ces objets. Incremental DBSCAN nécessite seulement une fonction de distance et est applicable à n'importe

quel ensemble de données appartenant à un espace métrique. Cependant, la méthode proposée ne résoud pas le problème du changement de point des densités dans le temps, ce qui nécessiterait d'adapter les paramètres d'entrée pour incrémental DBSCAN au fil du temps. Une autre limitation de l'algorithme est qu'il n'ajoute ou ne supprime qu'un seul point de données à la fois.
La segmentation fractale (Fractal clustering (FC)) a été proposée par Barbara et Chen [BC00]. FC appartient à la classe des techniques de segmentation par grille. Le concept de HFD(Hausdorff Fractal Dimension) est fondamental pour l'algorithme FC pour les attributs numériques, qui utilise plusieurs couches de grille. FC commence par initialiser k clusters, où k est déterminée en utilisant un seuil d'initialisation et d'échantillon de données, ensuite FC essaie d'ajouter toutes les données de manière incrémentale.
Dans [Bar02], Barbara a fixé les exigences pour réussir une segmentation de données en streaming, à savoir des critères bien précis qui doit satisfaire l'algorithme de la segmentation incrémentale.

3.2.2 Le principe de l'approche incrémentale

Il existe plusieurs approches de la classification incrementale, comme par exemple pour la classification incrémentale de document sur le web dans [WF02], ou encore pour le partitionnement incrémental de graphes dans [ZCY10]. Ces approches sont différentes et varient selon la méthode de classification appliquée. Une façon d'uniformiser le principe de base d'un algorithme assurant le paradigme incrémental est résumée dans les travaux [AD13] et [JK12] : Un algorithme d'apprentissage incrémental doit assurer tous les critères suivants :

1. Il doit être capable d'assurer l'apprentissage et la mise à jour avec toutes les nouvelles données-étiquetées ou non-étiquetées.
2. Il doit permettre la préservation des connaissances déjà acquises.
3. Il ne doit pas modifier les données originales.
4. Il doit avoir la possibilité de générer une nouvelle classe (ou un cluster) en cas de besoin, ainsi qu'à diviser ou fusionner les classes (clusters) si nécessaire.

5. Il doit être de nature dynamique, et pouvoir s'adapter à un environnement changeable.

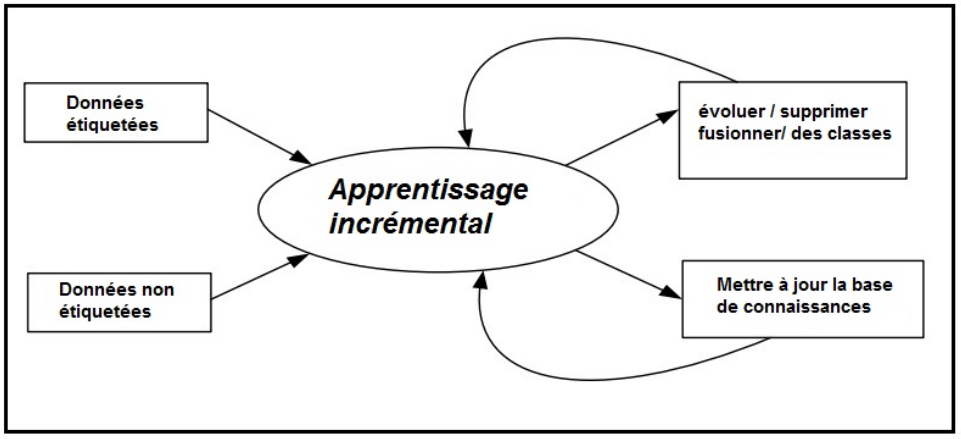

FIGURE 3.1 – *Le mécanisme de fonctionnement de l'approche incrémentale [JK12]*

La figure 3.1 montre le mécanisme de fonctionnement de l'approche incrémentale, qui doit assurer les critères présentés précédemment.

3.2.3 L'apprentissage incrémental non-supervisé

De manière générale, l'extraction des modèles qui appartiennent à une même catégorie ou sont probables à être dans un même groupe est la tâche principale de la segmentation (clustering). Lors d'une segmentation incrémentale, le processus d'apprentissage essaye de définir une fonction particulière afin que les nouvelles données soient attribuées à des classes sans re-classification.
Les algorithmes de segmentation comme k-means [FSS$^+$09] sont régis par le nombre d'analyses [ZRL96] [DA07].
D'autres facteurs qui influent sur le clustering sont la sélection des centroïdes et la forme de groupes formés. Le nombre de clusters à être formés est également un paramètre critique qui régit l'apprentissage.
Les algorithmes rapides et stables qui sont de nature incrémentale, peuvent remédier aux difficultés rencontrées par les méthodes de classification antérieures. Les différentes méthodes de classification visent à limiter la phase de re-segmentation, pouvant accueillir le

nouvel ensemble de données non-étiqueté, dans le même temps, elle offrent la fonctionnalité de mise à jour efficace des clusters [YAKR10].

La recherche dans le domaine de la segmentation incrémentale a commencé suite à plusieurs facteurs, dans plusieurs domaines. Un de ces domaines est la classification de documents, et la classification d'images. Dans le même domaine les auteurs respectifs de [CCFM97b] et [HK03] proposent le regroupement progressif où une segmentation de points dynamique est considérée. Composée de fusion et mise à jour pour maintenir les groupes, l'approche est basée principalement sur la mesure de la distance. En outre, les techniques de regroupement ont tendance à employer le calcul de mesure de similarité entre les clusters et les données, où les nouveaux échantillons sont regroupés progressivement.

Les auteurs de [SLWQ09] proposent une autre approche de segmentation incrémentale pour les mêmes motifs, où la valeur du seuil joue un rôle important pour la détermination des groupes. Avec cette valeur, une approche de division et d'agglomération supplémentaire est proposée dans [LA08] et est utilisée dans les ensembles de données relationnelles.

Dans certains cas, la combinaison des méthodes de Bayes avec la mesure de similarité rendent l'apprentissage plus efficace comme dans le travail [LLKH12]. Dans [CHO02], les auteurs proposent un algorithme de GRIN par rapport à BIRCH (où BIRCH est un algorithme de segmentation hiérarchique incrémentale, et est capable de traiter des grandes bases de données).

La gestion d'un entrepôt avec de nouvelles données est un défi qui est géré par les méthodes de l'apprentissage incrémental. Les approches existantes de DBSCAN sont encore améliorées pour être utilisées progressivement dans un environnement, où une partie du nouveau groupe affecté est examiné, et de façon à ce que les groupes soient mis à jour avec les insertions et les suppressions en prenant en considération leur densité [EKS$^+$98b].

"ART" (Adaptive Resonance Theory) est encore un concept populaire qui a été utilisé avec les réseaux de neurones pour l'apprentissage incrémental non-supervisé. Avec différentes variantes qui sont spécifiques au type de données, [HH08] propose une ART modifiée, qui gère les données d'attributs mixtes, en utilisant une nouvelle distance entre les données.

3.3 Le paradigme multi-niveaux

Durent ces 50 dernières années, un immense effort a été fourni pour l'étude de la résolution des problèmes NP-complets et de l'optimisation combinatoire [Wal08].
Il existe maintenant un tableau déconcertant de (méta) heuristiques pour résoudre ces problèmes. Aujourd'hui la question qui se pose chez chaque utilisateur est "*Quel est le meilleur algorithme pour résoudre mon problème ?*"
Cependant, en étudiant bien le paradigme multi-niveaux, c'est une autre question qui se pose, vu que le cadre multi-niveaux est collaboratif et qu'il agit toujours avec une autre technique, la question posée est :"*Sachant que j'ai utilisé l'algorithme X pour mon problème, est ce que ces performances peuvent être boostées en utilisant une version multi-niveaux de l'algorithme X* ?"
Comme nous allons le voir, très souvent, la réponse semble être catégoriquement oui, soit en termes de la qualité de la solution, ou en temps de calcul, ou les deux critères en même temps. En effet, dans certains cas, en termes de performances, la décision d'utiliser un système multi-niveaux est beaucoup plus importante que le choix réel de l'algorithme de résolution.
Encore plus encourageant, les preuves empiriques suggèrent que, dans l'élaboration d'un système multi-niveaux a part entière (qui, à sa base nécessite un algorithme de grossissement et un opérateur de projection / extension) est généralement beaucoup plus facile à mettre en oeuvre qu'une optimisation de haute qualité [Wal08].

3.3.1 État de l'art

Le paradigme multi-niveaux a émergé comme l'une des méthodes les plus efficaces pour résoudre les problèmes numériques et combinatoires. Il a été utilisé dans le multi-grille, le domaine de décomposition, la recherche des structures géométriques, ainsi que comme algorithme d'optimisation pour des problèmes tels que le partitionnement de graphes [Ten99].
Dans la littérature, le paradigme multi-niveaux était proposé pour la première fois par Bernard et al. [BS94] comme une méthode pour accélérer la dichotomie spectrale. Après, ce paradigme a été amélioré pour lui permettre d'englober un algorithme de raffinement

local par Hendrickson et al. [HL95].
Ensuite, cette approche a été rendue populaire par Karypi et al. dans [KK98], qui montre bien l'efficacité d'application du multi-niveaux dans le partitionnement de graphes irréguliers. Et depuis, ce paradigme est devenu couramment utilisé pour le partitionnement de graphes comme le prouvent le travail de Sanders et al. dans [SS11], et les travaux de Dhillon et al [DGK07] et [Dhi05].
L'approche multi-niveaux est aussi souvent utilisée en hybridation avec d'autres méta-heuristiques, comme dans le travail de Korosec, et al. [Š11] qui combine le multi-niveaux avec la meta-heuristique ACO (Ant Colony Optimization) pour le partitionnement de graphes.
Récemment, le paradigme multi-niveaux est adopté avec des méthodes pour la résolution de problèmes comme le travail de Bouhamla et al. [Bou12] qui combine le multi-niveaux avec les algorithmes mimétiques pour la résolution du problème SAT, ainsi que le travail de Djeffal et al. dans [DD13], qui combine le paradigme multi-niveaux avec la méta-heuristique BSO (Bees swarm Optimization) pour le même problème (SAT).

3.3.2 Principe de l'approche multi-niveaux

Le principe de base de l'approche multi-niveaux est techniquement très simple : il commence par assurer des grossissements récursifs afin de produire des sous problèmes plus petits et plus faciles à résoudre par rapport au problème originel.
Cette approche est constituée de trois phases : phase de grossissement, l'étape de la solution initiale, et enfin la phase de raffinement.

La figure3.2 montre le mécanisme de fonctionnement de l'approche multi-niveaux.

La phase de grossissement vise à fusionner les composants associés à ce problème pour former une hiérarchies de sous problèmes. De façon récursive, le problème originel est réduit en sous-problèmes de plus en plus petits, de telle sorte que chaque sous problème représente le problème original avec un degré réduit de liberté.
Le niveau le plus grossier peut ensuite être utilisé pour calculer une solution initiale. La solution trouvée au niveau le plus bas est étendue pour donner une solution initiale pour le niveau suivant, puis raffinée récursivement en utilisant un algorithme d'optimisation choisi

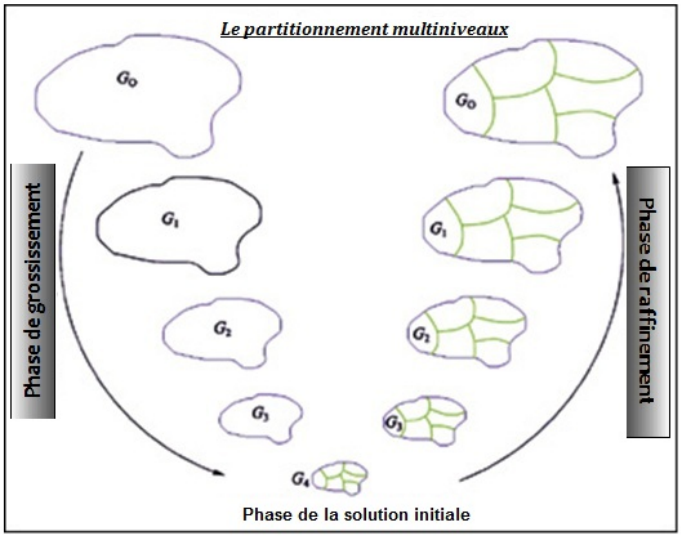

FIGURE 3.2 – *Principe de base de l'approche multi-niveaux[KK98]*

jusqu'à l'obtention de la solution finale au problème originel.

3.4 Conclusion

Dans ce chapitre, nous avons montré les principaux domaines et la majorité des méthodes inhérents aux approches incrémentales et les approches multi-niveaux. Considérant les différents domaines et le travail qui se fait place, le chercheur doit avoir une vue d'ensemble et choisir le domaine où le processus d'apprentissage aidera à prendre des décisions fortes.

D'après ce qui est présenté, nous pouvons remarquer que le point fort de l'approche incrémentale est sa capacité de progression et de se mettre à jour à chaque arrivée de nouvelles données sans redémarrer le processus à zéro, en d'autres termes, avoir une continuité de processus, et une stabilité de complexité de calcul. Par ailleurs, l'avantage majeur de l'approche multi-niveaux, est sa capacité à traiter de très grandes masses de données, de telle sorte que son principe consiste à d'abord réduire la taille du problème, afin de déterminer une solution initiale, et de la raffiner par la suite.

Dans la deuxième partie de cette thèse, nous allons montrer comment appliquer ces approches pour la fouille de règles d'induction, en proposant plusieurs approches et en les comparant sur plusieurs jeux de connaissances.

Deuxième partie

Contribution à la fouille des règles d'induction

Chapitre 4

Préliminaires mathématiques pour la fouille de connaissances

4.1 Introduction

Dans le domaine de la segmentation et de la classification de manière générale, la notion de mesure de similarité joue un rôle crucial pour le bon déroulement du processus. Dans ce chapitre nous proposons une nouvelle mesure de similarité entre les règles d'induction, et plusieurs nouvelles formules du calcul de centre de gravité.
Ces formules permettront d'étendre l'algorithme classique K-means pour traiter les connaissances de type règles d'induction (appelé K-means-IR). Par la suite, nous montrerons la comparaison des performances de ces formules sur une très grande base de règles en utilisant plusieurs formules d'évaluation.

4.2 Mesure de similarité

Dans la fouille de données, la notion de mesure de similarité est très importante. Nous pouvons citer quelques distances connues pour la fouille de données selon [Kan11] comme suit :
Pour les données dont les attributs sont de type numérique, la distance euclidienne est souvent utilisée, elle est définie comme suit :
$D(X_i, X_j) = \sqrt{\sum_{r=1}^{n}(X_{ir} - Xjr)^2}$.
Ou bien aussi la distance de Manhattan définie comme suit :

$D(X_i, X_j) = \sum_{r=1}^{n} |X_{ir} - X_{jr}|$.
Tel que, X_i et X_j représente deux données différentes, et que n représente le nombre d'attributs de ces données. D'autre part, pour les attributs qualitatifs, de nombreux indices de ressemblance ont été proposés : indice de Jaccard, indice de Russel et Rao, indice de Saporta (1990) etc.

Dans notre cas, qui est la fouille de connaissances, la notion la plus connue c'est la distance ontologique, appelée aussi la distance entre concepts.

Une ontologie peut être définie comme étant un ensemble structuré des termes et concepts représentant le sens d'un champ d'informations [ZS+04] ; que ce soit par les méta-données d'un espace de noms, ou les éléments d'un domaine de connaissances. L'ontologie constitue en soi un modèle de données représentatif d'un ensemble de concepts dans un domaine, ainsi que des relations entre ces concepts. Elle est employée pour raisonner à propos des objets du domaine concerné. Les concepts sont organisés dans un graphe dont les relations peuvent être de type sémantique, ou bien de type subsomption (inclusion).

L'objectif premier d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné, qui peut être réel ou imaginaire, ainsi que de pouvoir calculer les mesures de similarité entre ces connaissances.

Les ontologies sont employées dans la fouille de données, la recherche d'information, le Web sémantique, le génie logiciel, et l'informatique biomédicale comme une forme de représentation de la connaissance au sujet d'un monde ou d'une certaine partie de ce monde.

Pour le calcul de distance entre les concepts, appelée aussi « l'identification de la similarité sémantique », on distingue deux classes d'approches :

1- Les approches basées sur les noeuds : utilisant typiquement des mesures du contenu informationnel pour déterminer la similarité conceptuelle. En plus, la similarité entre les concepts est déterminée par le degré de partage de l'information.

2- la deuxième famille d'approches repose uniquement sur une hiérarchie ou sur les distances des arcs, le principe de calcul de similarité avec ces approches est basé sur l'idée suivante : "plus le chemin entre deux noeuds est court plus ils sont plus semblables". L'autre

notion qui caractérise cette deuxième famille est que les arcs d'une taxonomie représentent des distances uniformes. Par conséquent, dans cette famille, tous les liens sémantiques possèdent le même poids, ce qui impose des difficultés au niveau de la définition et du contrôle des distances des liens [SBM07].

Comme présenté précédemment, la distance ontologique entre connaissances qui se base sur la représentation graphique pour pouvoir calculer le plus court chemin n'est pas efficace dans notre cas, parce que d'un côté elle représente une très grande complexité si on dispose d'une grande masse de connaissances (ce qui est notre cas), et d'un autre coté l'un des objectifs de la fouille de connaissance est de pouvoir classer les connaissances similaires dans une même classe. Pour ces raisons, nous avons ignoré la distance ontologique dans notre travail. De plus, cette distance impose d'avoir des connaissances préalables sur les connaissances traitées.

Ceci nous a mené à développer une nouvelle formule de distance entre les connaissances de type règles, afin de l'utiliser par la suite dans les techniques de fouille de connaissances.

4.2.1 La présentation de la mesure de similarité Sim-C

Une mesure de similarité entre deux règles de production représente le degré de similitude entre elles. Dans un travail précédent [DAB12] une distance appelée (D-12) a été proposée entre deux règles comme suite :

$$D(r_1, r_2) = \frac{|C_1 \bigcup C_2| - |C_1 \bigcap C_2| + |V_1 \bigcup V_2| - |V_1 \bigcap V_2|}{|R_1 + R_2|}. \tag{4.1}$$

Tel que :

C_i : représente l'ensemble de clauses de R_i.

V_i : représente l'ensemble de variables de R_i.

$|R_1 + R_2|$: représente la longueur total des règles, à savoir le nombre total de leurs clauses.

La distance D-12 représente parfaitement la similarité entre deux règles. Néanmoins, il y a des opérations supplémentaires qui sont : $V_1 \bigcup V_2$ est $V_1 \bigcap V_2$, parce qu'on remarque bien que l'ensemble V (ensemble de variables) est inclus dans l'ensemble C (ensemble de clauses). Cependant les opérations $C_1 \bigcup C_2$ et $C_1 \bigcap C_2$ sont déjà calculées, et compte tenu

du nombre de fois où les algorithmes de fouille répètent la fonction de calcul de similarité, il serait judicieux de garder uniquement les opérations essentielles.

Dans cette partie nous proposons une nouvelle mesure de similarité appelée $Sim\_Clauses$, cette mesure reflète la façon de distinguer les objets chez l'être humain. Intuitivement, si on veut différencier un objet à un autre, il suffit de voir les caractéristiques partagées entre eux. On déduit alors que deux règles sont similaires si elles ont plusieurs clauses en commun, et le contraire est vrai, elles sont totalement différentes si elles ne partagent aucune clause. Sur ce raisonnement, nous proposons la mesure de similarité comme suit :

$$D(r_i, r_j) = (clauses(r_i) \cup clauses(r_j)) - (clauses(r_i) \cap clauses(r_j)). \tag{4.2}$$

tel que : clauses(r_i)= $\{(attribut_1, valeur_1)$,$(attribut_2, valeur_2)$,...,$(attribut_n, valeur_n)\}$.

Exemple : Si on a deux règles comme suite :
R_1 : if outlook=sunny and humidity=normal and temperature $<$ 18 then play=yes.
R_2 : if outlook=rain and windy=false then play=yes.
$Clauses_{R1}$= { (outlook , sunny),(humidity , normal),(temperature , 18),(play , yes)}.
$Clauses_{R2}$= { (outlook , rain),(windy , false),(play , yes)}.

$Sim\_Clauses(r_1, r_2)$= 6 - 1 = 5.

La mesure de similarité proposée (Sim_Clauses) est une valide métrique fonction de distance dans l'ensemble de règles, parce que nous avons pu démontrer mathématiquement les quatre propriétés d'une fonction de distance métrique standard qui sont :
1- La distance représente toujours un nombre réel :

$\forall(r_1, r_2) \in IR^2, Sim\_Clauses(r_1, r_2) \in R.$

2- La distance entre la règle est elle même est toujours égale à zéro :

$\forall r \in IR, Sim\_Clauses(r, r) = 0.$

3- La symétrie :

$\forall(r_1, r_2) \in IR^2, Sim\_Clauses(r_1, r_2) = Sim\_Clausse(r_2, r_1).$

4- L'inégalité triangulaire :

$\forall(r_1, r_2, r_3) \in IR^3, Sim\_Clauses(r_1, r_2) \leq Sim\_Clausse(r_1, r_3) + Sim\_Clauses(r_3, r_2).$

4.2.2 Analyse et Démonstration

Si on considère l'espace de règle IR l'ensemble qui contient toutes les règles possibles. Afin de prouver que la formule Sim_Clauses est une mesure de distance métrique et valide entre deux règles appartenant à l'ensemble IR, les quatre propriétés suivantes devront être démontrées :

- $\forall(r_1, r_2) \in IR^2, Sim\_Clauses(r_1, r_2) \in R.$
- $\forall r \in IR, Sim\_clauses(r, r) = 0.$
- $\forall(r_1, r_2) \in IR^2, Sim\_Clauses(r_1, r_2) = Sim\_Clauses(r_2, r_1).$
- $\forall(r_1, r_2, r_3) \in IR^3, Sim\_Clauses(r_1, r_2) \leq Sim\_Clauses(r_1, r_3) + Sim\_Clauses(r_3, r_2).$

Puisque, Clauses(r_i) est un ensemble :

$Clauses\_Total(r_1, r_2) = Clauses(r_1) \bigcup Clauses(r_2)$....(1)

$Clauses\_Commun(r_1, r_2) = Clauses(r_1) \bigcap Clauses(r_2)$....(2)

Les démonstrations suivantes sont basées sur équations(1)et(2).

Propriété 1 : Sim_Clause est décomposée en deux parties :

Nb_Clause_total(r_1,r_2) $\in$ R et Nb_Clause_commun(r_1,r_2) $\in$ R.

C'est évidant que Nb_Clauses_total - Nb_Clauses_commun $\in$ R, Donc Sim_Clauses $\in$ R. Ceci implique que la première propriété est vérifiée.

Propriété 2 : Considérant $r_1 \in$ IR :

Sim_Clauses$(r_1, r_1) = Nb_Clauses_total(r_1, r_1) - Nb_Clauses_commun(r_1, r_1)$.

$clauses\_total(r_1, r_1) = clauses(r_1) \bigcup clauses(r_1)$.

$\Rightarrow Nb\_Clauses\_total(r_1, r_1) = Nb\_Clauses(r_1)$.....(3)

Par ailleurs ;

Clauses_Commun$(r_1, r_1) = Clauses(r_1) \bigcap Clauses(r_1)$.

$\Rightarrow Nb\_clauses\_commun(r_1, r_1) = Nb\_Clauses(r_1)$.....(4)

avec (3) et (4), on aura :

Sim_Clauses$(r_1, r_1) = 0$.

Et donc, la deuxième propriété est vérifiée.

Propriété 3 : On considère deux règles $(r_1, r_2) \in IR^2$.

Sim_Clauses$(r_1, r_2) = Nbr_Clauses_total(R1, R2) - Nbr_Clauses_commun(R1, R2)$.

Clauses_total$(r_1, r_1) = Clauses(r_1) \bigcup Clauses(r_2) = Clauses(r_2) \bigcup Clauses(r_1)$.

=Clauses_total(r_2, r_1).

$\Rightarrow Nb\_clauses\_total(r_1, r_2) = Nb\_clauses\_total(r_2, r_1)$.....(5)

Par ailleurs ;

Clauses_commun$(r_1, r_2) = Clauses(r_1) \bigcap Clauses(r_2) = clauses(r_2) \bigcap Clauses(r_1)$.

$\Rightarrow Nb\_Clauses\_commun(r_1, r_2) = Nb\_clauses\_commun, (r_2, r_1)$.....(6)

avec (5) et (6), on aura :

Sim_Clauses$(r_1, r_2) = Sim_Clauses(r_2, r_1)$.

$\Rightarrow$ La troisième propriété est vérifiée.

Propriété 4 : on pose que :

$f(r_1, r_2) = Sim\_Clauses(r_1, r_2)$.

et $g(r_1, r_2, r_3) = Sim\_Clauses(r_1, r_2) + Sim\_Clauses(r_2, r_3)$.

Si on peut prouver que : $Max(f(r_1, r_2)) \leq Min(g(r_1, r_2, r_3))$ alors on conclura que :

$\forall (r_1, r_2, r_3) \in IR^3, Sim\_Clauses(r_1, r_2) \leq Sim\_Clausse(r_1, r_3) + Sim\_Clauses(r_3, r_2)$.

En d'autres termes, f sera maximisé et g sera minimisé, si on trouve que f reste toujours inférieure à g alors, on peut dire que :

$\forall (r_1, r_2, r_3) \in R^3, f(r_1, r_2) \leq g(r_1, r_2, r_3)$.

Selon le nombre total de clauses de r_1, r_2, et r_3, on aura six cas différents :

- 1er cas : $Nb\_Clauses(r_2) \geq Nb\_Clauses(r_3) \geq Nb\_Clauses(r_1)$.
- 2éme cas : $Nb\_Clauses(r_2) \geq Nb\_Clauses(r_1) \geq Nb\_Clauses(r_3)$.
- 3éme cas : $Nb\_Clauses(r_3) \geq Nb\_Clauses(r_2) \geq Nb\_Clauses(r_1)$.
- 4éme cas : $Nb\_Clauses(r_3) \geq Nb\_Clauses(r_1) \geq Nb\_Clauses(r_2)$.
- 5éme cas : $Nb\_Clauses(r_1) \geq Nb\_Clauses(r_3) \geq Nb\_Clauses(r_2)$.
- 6éme cas : $Nb\_Clauses(r_1) \geq Nb\_Clauses(r_2) \geq Nb\_Clauses(r_3)$.

Pour chaque cas, l'inégalité suivante devrait être démontrer :

$Max(f(r_1, r_2)) \leq Min(g(r_1, r_2, r_3))$.

1er cas : $Nb\_Clauses(r_2) \geq Nb\_Clauses(r_3) \geq Nb\_Clauses(r_1)$:

Dans un coté, Max(f) ⇒ Max(Nb_Total_Clauses(r_1,r_2) et Min(Nb_Clauses_Commun(r_1,r_2).

donc, Clauses(r_1)$\bigcap$ Clauses(r_2)=$\emptyset$....(7).

Ceci implique que :

Nb_Total_Clauses(r_1,r_2) = Nb_Clauses(r_1)+Nb_Clauses(r_2)...(8).

et

Nb_Clauses_commun(r_1,r_2)=0....(9).

(8) et (9) donne :

Max(f(r_1,r_2)= Nb_Clauses(r_1)+Nb_Clauses(r_2)...(10).

Dans un autre coté on a :

Min(g)⇒ Max(Nb_Clauses_commun(r_1,r_3),

Max(Nb_Clauses_commun(r_3,r_2) et Min(Nb_Total_Clauses(r_1,r_3),

Min(Nb_Total_Clauses(r_3,r_2)).

Min(Nb_Total_Clauses(r_1,r_3)) ⇒ Clauses(r_1)$\subset$ Clauses(r_3)....(11).

Donc, Nb_Clauses_Commmun(r_1,r_3)=Nb_Clauses(r_1)...(12).

Pour maximiser Nb_Clauses_commun(r_3,r_2) et avec (7),(11) on déduit :

Clauses (r_3) $\bigcap$ Clauses(r_2)=Clauses(r_3)$\setminus$ Clauses(r_1).

Ceci nous donne :

Nb_Clauses_commun(r_3,r_2)= Nb_Clauses(r_3) − Nb_Clauses(r_1).....(13)

Avec l'hypothèse du 1er cas :

Nb_Clauses(r_3) > Nb_Clauses(r_2) ⇒ Min(Nb_Clauses_commun(r_3,r_2))= Nb_Clauses(r_2).....(14).

avec (11),(12),(13),(14) on obtient :

Min(g)=2∗Nb_Clauses(r_3)+Nb_Clauses(r_2).....(15).

et

Nb_Clauses(r_1)+Nb_Clauses(r_2) ≤ Nb_Clauses(r_3)+Nb_Clauses(r_2)....(16).

En remplaçant (16) en (15) et (10) on obtient :

Max(f)≤ Min(g). Donc, le 1er cas est vérifié.

Les cinq cas restants, sont prouvés avec le même raisonnement.

4.3 Le calcul du centre de gravité

Pour grouper les règles d'induction en utilisant l'algorithme classique k-means (ou un algorithme de partitionnement en général), on a besoin d'une formule de calcul de centres de gravité. Les formules de centres de groupes conventionnelles ne peuvent pas êtres appliquées ici, sachant que l'algorithme K-means manipule généralement de simples données et non pas des règles. Par ailleurs, le calcul de la règle moyenne dans un groupe dépend de la mesure de similarité utilisée. Pour cette raison, on propose deux types de formules pour le calcul du centre de gravité.

4.3.1 Les formules basées sur la logique propositionnelle

En étudiant bien la mesure de similarité proposée $Sim\_Clauses$, on remarque qu'elle est constituée d'opérateurs de la logique propositionnelle. Pour cette raison, nous présentons trois formules différentes pour le calcul du centre de gravité basées sur la logique propositionnelle comme suit, (nous avons publié cette étude dans les travaux [CDD13a]et[CD14].)

1. Formule UCI (Union Over Intersection) : Consiste à calculer séparément l'union et l'intersection de toutes les clauses d'un groupe de règles C_i, ensuite à soustraire l'ensemble d'intersection de l'ensemble d'union.
   $clauses(rgle\_moyenne) = [clauses(r_1) \cup clauses(r_2) \cup \ldots \cup clauses(r_m)]$
   $-[clauses(r_1) \cap clauses(r_2) \cap \ldots \cap clauses(r_m)]$.

2. Formule IUC (Intersection Union Center) : Ici, on calcule l'intersection de toutes les clauses du groupe de règles C_i, ensuite on applique l'union entre le résultat obtenu et l'ancien centre du groupe C_i comme suit :
   $clauses(rgle\_moyenne) = [clauses(r_1) \cap clauses(r_2) \cap \ldots \cap clauses(r_m)] \cup clauses(gi)$.

3. Formule UIC (Union Intersection Center) : Son principe est de calculer l'union de toute les clauses du groupe de règles C_i, ensuite d'appliquer l'intersection entre le résultat obtenu avec l'ancien centre du groupe C_i comme suit :

   $clauses(rgle\_moyenne) = [clauses(r1) \cup \ldots \cup clauses(rm)] \cap clauses(gi)$.

4.3.2 Exemple

Si on considère qu'on a un groupe dans une itération i constitué des règles $R1$, $R2$,$R3$ suivantes :

- $R1$: if x= 2 and y=3 and $z = 5$ then $t = 8$
- $R2$: if x=2 and y=3 and $z = 6$ then $t = 8$
- $R3$: if y=3 and z=2 then $t = 8$.

  Et que le centre de gravité de ce groupe à l'itération $i - 1$ était $G1$ comme suit :

  $Clauses(G1) = \{(x, 2), (y, 3), (z, 6), (t, 8)\}$.

  Pour calculer le nouveau centre de gravité de l'itération i avec les trois formules présentées précédemment, il faut commencer par l'extraction des clauses pour chaque règle :

- $clauses(R1) = \{(x, 2), (y, 3), (z, 5), (t, 8)\}$
- $clauses(R2) = \{(x, 2), (y, 3), (z, 6), (t, 8)\}$
- $clauses(R3) = \{(y, 3), (z, 2), (t, 8)\}$
- $UOI(center\_rule) = [clauses(r_1) \cup clauses(r_2) \cup clauses(r_3)]$
  $- [clauses(r_1) \cap clauses(r_2) \cap clauses(r_3)] = \{(x, 2), (z, 5), (z, 2), (z, 6)\}$.
- $IUC(center\_rule) = [clauses(r_1) \cap clauses(r_2) \cap clauses(r_3)] \cup clauses(gi)$
  $= \{(y, 3), (t, 8), (x, 2), (z, 6)\}$.
- $UIC(center\_rule) = [clauses(r_1) \cup clauses(r_2) \cup clauses(r_3)] \cap clauses(gi)$
  $= \{(x, 2), (y, 3), (z, 6), (t, 8)\}$.

4.3.3 Les formules basées sur la fréquence de clauses

Une formule basée sur la fréquence de clauses a été présentée et utilisée dans des travaux précédents [DAB12] [CD13]. Le principe de cette formule est de construire la règle centre d'un cluster à partir des clauses les plus fréquentes, sachant que le nombre de clauses de cette règle est égal à la moyenne des nombres de clauses de toutes les règles appartenant à ce groupe, cela veut dire que le calcul du centroide avec cette formule doit respecter les points suivants :

- La taille du centroïde est égale à la taille moyenne de l'ensemble des règles de son groupe.

TABLE 4.1 – Table de fréquence de clauses

Clause	Fréquence
$(x, 1)$	01
$(y, 2)$	03
$(z, 3)$	02
$(x, 2)$	01
$(f, 3)$	01
$(h, 0)$	01
$(x, 3)$	01
$(f, 0)$	01
$(h, 1)$	01

- Si les différentes clauses ont la même fréquence, ceux contenant les variables les plus fréquentes dans l'ensemble de règles sont sélectionnés.
- La distance entre le centre de gravité et chaque règle est approximativement la même.
- Le déplacement des connaissances d'un cluster à un autre trouble la position du centre de gravité, qui par conséquent doit être mis à jour.

4.3.4 Exemple

Supposons qu'on a trois règles R1,R2,et R3 dans le même cluster, et on veut calculer le centroid :

R1 : if x=1 and y=2 then z=3.

R2 : if y=2 and x=2 and f=3 then h=0.

R3 : If i z=3 and y=2 and x=3 and f=0 then h=1.

Les tailles des règles (en nombre de clauses) R1, R2, et R3 sont respectivement : |R1|= 3, |R2|= 4, |R3|= 5.

La taille du centre de gravité (appelé G) est calculée comme suit :

$|G| = \frac{|R1|+|R2|+|R3|}{nombre\_de\_rgles} = \frac{3+4+5}{3} = 4.$

Maintenant, on calcule la fréquence de chaque clause, les fréquences sont notées dans le tableau4.1.

A partir du tableau4.1, on note que les clauses les plus fréquentes sont :(y,2),(z,3). En tenant compte que la taille du centroide devrait être construite de 4 clauses, on choisit deux autres clauses.

Pour cela, on cherche les clauses avec les variables les plus fréquentes, et remarquant que la variable x est répétée plusieurs fois, on choisit la clause (x,1) suivie d'une autre clause aléatoirement choisie, soit (f,3).
Donc, les clauses du centroide sont les clauses de G = {(y,2),(z,3),(x,1),(f,3)}.

4.4 L'extension de l'algorithme K-means pour la segmentation des règles d'induction

Dans les sections précédentes, nous avons proposé des nouvelles préliminaires mathématiques pour les connaissances de type règles d'induction, à savoir : une nouvelle mesure de similarité entre les règles, ainsi que des nouvelles formules du calcul de la règle centre d'un groupe.
Ces préliminaires mathématiques peuvent être utilisées pour étendre l'algorithme classique K-means, qui habituellement ne traite que des simples données, pour la segmentation de règles d'induction.
L'algorithme K-means pour les règles d'induction (noté K-means-IR) est présenté comme suit dans Algorithme1 :

Algorithm 1 Algorithme K-means-IR

1: g_i : le centre de gravité du groupe i
2: Choisir aléatoirement k centres initiaux $Cluster_1, Cluster_2 \ldots, Cluster_k$
3: **for all** règle r **do**
4: affecter r au groupe i le plus proche du centroide (en utilisant la distance)
5: **if** aucune règle ne change de groupe **then**
6: stop and exit
7: **else**
8: Recalculer les nouveaux centres de gravité g_i pour tous les $cluster_i$
9: **end if**
10: **end for**

Comme pour l'algorithme K-means classique, la compléxité de l'algorithme K-means-IR au pire des cas est égale à O($K \times n \times m \times IMAX$), telle que :

K est le nombre de groupes.

m est le nombre totale de règles.

n est le nombre de clauses des règles.

$IMAX$ est le nombre maximum d'itérations.

4.4.1 Exemple

Exemple : soit la petite base de règle suivante :

R1 : If (temperature=hot) and (humidity=low) and (outlook=sunny) then ($play\_tennis = yes$).

R2 : If (age=29years) and (income>=3000) and (married=yes) then (children=1).

R3 : If (name=Casillas) and (age=29years) and (children=1) then ($team = Real\_Mardrid$).

R4 : If (vehicle=classic) and (brake=true) then ($danger\_state = false$).

R5 : If (temperature=hot) and (outlook = sunny) then (fishing =true).

R6 : If (ESP=true) and (ABS=true) and (mark = audi) then ($danger\_state = false$).

En fixant le paramètre K à 3, le fonctionnement de l'algorithme K-means-IR sera comme suit :

Initialement, les 3 centres de gravité sont choisis aléatoirement :

Soit : C1 = R1, C2 = R3, et C3 = R4.

Maintenant, les distances qui séparent chaque règle à chaque centroide sont calculées, les résultats sont notés dans le tableau suivant (table 4.2).

Règle	c_1	c_2	c_3	Minimum distance
R_1	0	8	7	C1
R_2	8	4	7	C2
R_3	8	0	7	C2
R_4	7	7	0	C3
R_5	3	7	6	C1
R_6	8	8	5	C3

TABLE 4.2 – Tableau de distances dans la première itération

La table4.2 présente la distance entre les centroides et toutes les règles à la première itération. Après l'extraction des minimums des distances, une première classification est obtenue. $Cluster_1 = \{R_1, R_5\}$, $Cluster_2 = \{R_2, R_3\}$, $Cluster_3 = \{R_4, R_6\}$.

Les nouveaux centres de gravité sont calculés comme suit :

$|C_1| = |\frac{4+3}{2}| = 3$. La taille du centroide 1 est de 3 clauses, maintenant les 3 clauses les plus

fréquentes dans R1, et R5 constitueront le centre 1 comme suit :

C_1={(temperature , hot),(outlook, sunny),(humidity , low)}.

De la même façon, les deux autres centres sont calculés :

$|C_2| = |\frac{4+4}{2}| = 4$.

C_2={(age , 29),(children, 1),(income , 3000), (married , yes)}.

$|C_3| = |\frac{3+4}{2}| = 3$.

C_3={(danger_state = false),(mark , audi),(ABS, true)}.

Après cette étape, on recalcule toutes les distances qui séparent chaque règle aux nouveaux centres, les résultats sont notés dans le tableau 4.3.

Règle	c_1	c_2	c_3	Minimum distance
R_1	1	7	7	C1
R_2	7	0	7	C2
R_3	7	4	7	C2
R_4	6	7	4	C3
R_5	2	7	6	C1
R_6	7	8	1	C3

TABLE 4.3 – Tableau de distances à la deuxième itération

Après l'affectation de chaque règle à son cluster le plus proche (le groupe qui a le plus proche centre de gravité), on obtient la segmentation suivante :

$Cluster_1 = \{R_1, R_5\}$, $Cluster_2 = \{R_2, R_3\}$, $Cluster_3 = \{R_4, R_6\}$.

On remarque bien que le règles n'ont pas changé de cluster, donc on a atteint la stabilité de la segmentation, et le processus va s'arrête.

4.5 Évaluations et expérimentations des préliminaires mathématiques proposées

Dans cette partie, nous essayons de comparer les performances de chaque formule de centre de gravité proposée, et de comparer la mesure de similarité proposée avec celle qui existe déjà, à travers l'application de l'algorithme k-means-IR sur une base de règles générée aléatoirement, et qui contient 100,000 règles.

4.5.1 Construction du jeu de règles

Nom du dataset	Nombre d'attributs	Nombre d'instances
Chess (King-Rook vs. King) Data Set	07	28056
Abalone Data Set	09	4177
Statlog (Shuttle) Data Set	09	58000
Car Evaluation Data Set	07	1728
Letter Recognition Data Set	16	20000

TABLE 4.4 – Construction de la collection de connaissances

Notre collection de règles d'induction est construite à partir des cinq datasets publics collectés et publiés par le centre d'apprentissage automatique et des systèmes intelligents de l'université de Californie, Irvine, (UCI) [MM98]. La table4.4 montre les benchmarks inclus dans notre collection, qui contient en tout plus de 100,000 instances.
Toutes les instances de données sont transformées en règles d'induction comme expliqué dans l'exemple suivant :
Si $(attribut_1 = valeur_1)$ et $(attribut_2 = valeur_2)$ et ... alors $(attribut_n = valeur_n)$, où :
n est le dernier attribut du dataset.

4.5.2 Évaluation du taux de réussite de la segmentation

Dans un premier temps, et comme nous avons construit notre base de règles à partir de cinq bases différentes, nous fixons le paramètre k (le nombre de clusters en sortie) à 5. Ensuite, et après le lancement du processus de segmentation, les cinq clusters obtenus sont comparés avec les data-sets originels. Si le processus de la segmentation est efficace, les clusters en sortie doivent être identiques au data-sets en entrée. Le taux de réussite de cette méthode est calculé par la formule suivante :

$$\text{Taux de réussite} = \sum_{i=1}^{5} \frac{ncr_i}{npr_i}; ncr(BR_i) = max(\bigcap(BR_i, C_j))/\forall j = 1..5 \quad (4.3)$$

tel que :

ncr = nombre de règles correctement classifiées.

npr = nombre de règles pertinentes.

ncr_i= nombre de règles correctement classifiées de la base de règle i.

npr_i=nombre de règles pertinentes de la base de règle i.
$NCR(BR_i) = max$(nombre de règles communes $(BR_i, groupe_j)$
pour chaque i et j variants entre [1..5].
Le nombre de règles pertinentes de la BR_i = le nombre total de la base de règles BR_i.

Dans un second temps, nous fixons le nombre de règles d'induction et nous varions le nombre de clusters, pour cela on utilise la F-mesure comme formule d'évaluation du taux de réussite de segmentation. La F-mesure est une formule très connue et très utilisée pour l'évaluation dans le domaine de la recherche d'information, de la classification,...etc ; et elle est définie comme suit :

$$F\_mesure_\beta = \frac{(\beta^2+1)\cdot P \cdot R}{\beta^2 \cdot P + R}, P = \frac{TP}{TP+FP}, R = \frac{TP}{TP+FN} \tag{4.4}$$

où :
P représente le taux de précision et R le taux de rappel
TP (true positives en anglais) représente le nombre de vrais positifs.
TN (true negatives en anglais) représente le nombre de vrais négatifs.
FP (false positives en anglais) représente le nombre de faux positifs.
FN (false negatives en anglais) représente le nombre de faux négatifs.

La F-mesure peut être utilisée pour équilibrer la contribution de faux négatifs en pondérant le rappel par le paramètre $\beta \geq 0$.
Dans notre cas, β est égale à 0, $F_0 = P$. Dans d'autres termes, le rappel n'influence pas sur la F-mesure quand $\beta = 0$, parce que l'augmentation de β alloue une augmentation du poids du rappel dans la dernière F-mesure.

4.5.3 Évaluation et comparaison des mesures de similarité

La Figure 4.1 montre comment le temps d'exécution change en augmentant le nombre de règles d'induction à traiter pour les deux cas, l'algorithme K-means-IR basé sur notre mesure de similarité et K-means-IR basé sur la mesure $D-12$ présentée dans [DAB12]. Selon cette figure4.1, l'algorithme basé sur la distance proposée $Sim - C$ est plus rapide

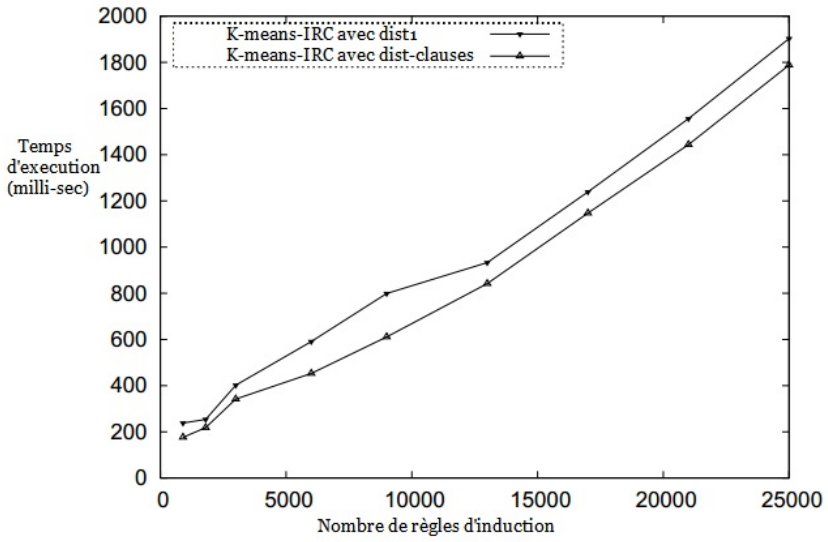

FIGURE 4.1 – *Comparaison du temps d'exécution des deux mesures de similarité*

que celui basé sur la mesure $D-12$. Ceci est du au fait que la mesure D-12 est constituée d'opérations de calcul supplémentaires entre les variables des règles, tandis qu'avec la mesure Sim_C ces opérations ont été ignorées. Le temps d'exécution du K-means-IR basé sur Sim_C ,ne dépasse pas 1,800 secondes pour traiter 25000 règles, contrairement à celui basé sur la $D-12$ qui prend 1,915 secondes pour traiter le même nombre de règles.

La figure4.2 montre la comparaison de la qualité de la classification de l'algorithme K-means-IR basé sur la distance $D-12$ à celui basé sur notre mesure de similarité Sim_C. Le taux de réussite ici est calculé avec l'équation présentée précédemment Eq4.3.Selon la figure, on remarque qu'il n'y a pas une grande différence entre le taux de réussite des deux approches, de telle sorte que lors de l'augmentation du nombre de règles de 1000 à 25000, le taux de réussite de K-means-IR basé sur $D-12$ est réduit de 99%, à 77%, en revanche, celui de K-means-IR basé sur notre mesure de similarité $Sim-C$ est réduit de 95% à 76%.

La figure4.3 montre la comparaison de la qualité du clustering des deux mesures de similarité. Le taux de réussite ici est calculé par la F-mesure présentée précédemment dans Eq4.4. Selon cette figure on note bien que l'algorithme K-means-IR basé sur notre distance $Sim-C$ est plus performant que celui basé sur la distance $D-12$. De telle sorte que lorsqu'on varie le nombre de cluster entre 3 et 40 groupes ; K-means-IR basé sur la

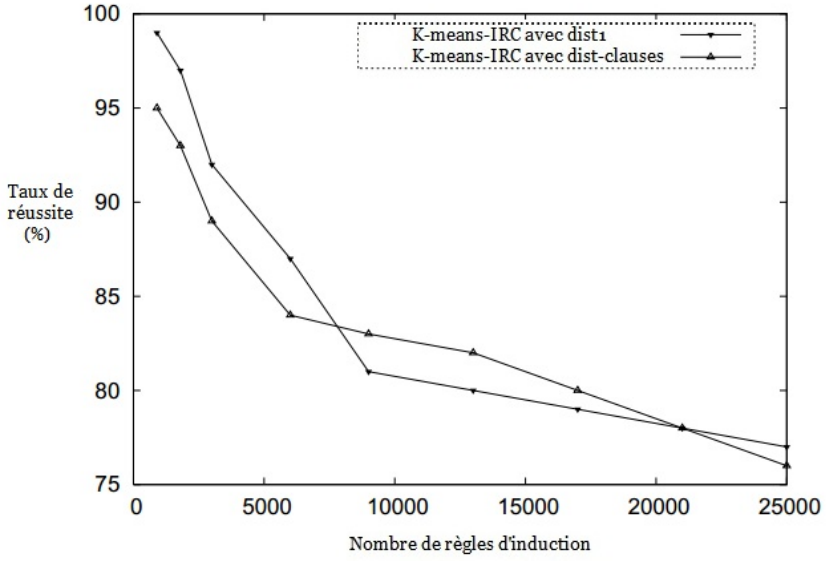

FIGURE 4.2 – *Comparaison du taux de rèussite de la segmentation des deux mesures de similarité*

distance $D-12$ est réduit de 77% à 72%, en revanche l'algorithme K-means-IR basé sur la mesure $Sim-C$ est réduit juste de 82% à 76%.

A partir de ces résultats, on note que l'algorithme K-means-IR basé sur notre mesure de similarité $Sim-C$ est plus performant en matière de temps d'exécution que celui basé sur la distance $D-12$ présentée dans un travail précédent. On explique cela par le fait que dans la formule de distance $D-12$ existe une redondance de calcul qui a été supprimé dans la formule de distance proposée dans ce travail $Sim-C$. Donc, pour les prochaines expérimentations on implémente l'algorithme K-means-IR en utilisant la distance proposée $Sim-C$.

4.5.4 Évaluation et Comparaison des formules de calcul des centres de gravité

La figure4.4 montre la comparaison du temps d'exécution des trois formules proposées pour le calcul des centres de gravité (UCI, IUC, et UIC). Selon la figure, on remarque que la formule UCI est plus lente que les deux autres. Ceci est du au fait que UCI utilise trois opérateurs (Union Intersection and Over) entre les règles, et ceci est très couteux en

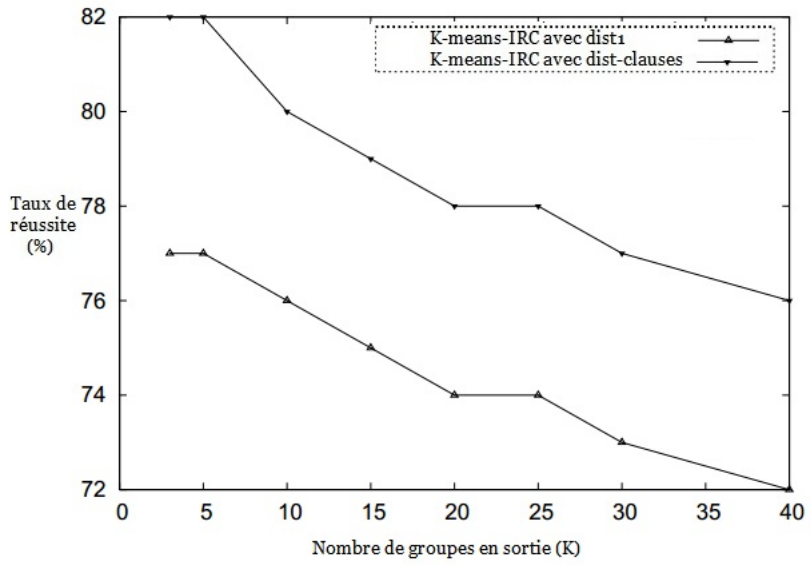

FIGURE 4.3 – *Comparaison des F-mesures des deux mesures de similarité*

temps de calcul. Par ailleurs, UIC est plus performante en temps de calcul que IUC. On explique ces résultats par le fait que l'opérateur de l'union est plus rapide que l'opérateur de l'intersection. dans la formule UIC, l'opérateur de l'union est appliqué aux règles du même groupe, ensuite l'intersection est appliquée seulement sur la règle de l'union avec le précédent centre de gravité. En revanche, dans la formule IUC, l'intersection est appliquée sur toutes les règles du même cluster. Le temps de calcul de la formule UIC ne dépasse pas 2200 millisecondes pour une base de 2500 règles, contre 2600 millisecondes avec la formule UCI.

La comparaison de la qualité de la segmentation des trois formules de centre de gravité (en utilisant l'équation présentée précédemment Équation4.3)est schématisée à la figure4.5. Selon cette figure, on remarque bien que la formule IUC est moins performante que les deux autres formules. Lors de l'augmentation du nombre de règles de 1000 à 25000 règles, le taux de réussite de la formule IUC est réduit de 93% à 69%, contrairement à celui de la formule UIC qui est réduit juste de 95% à 84%.

Selon ces résultats, on peut noter que la performance de l'algorithme K-means-IR (temps d'exécution, et taux de réussite) dépend de la formule du centre de gravité choisie.

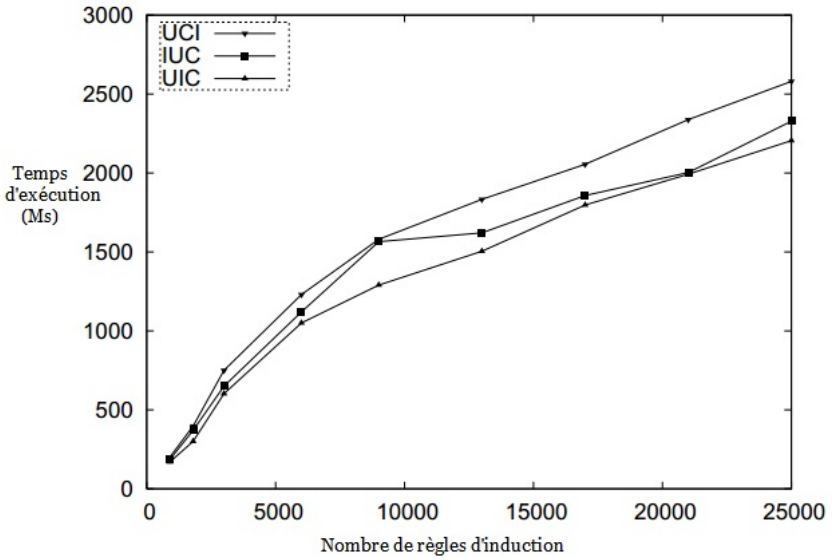

FIGURE 4.4 – *Comparaison du temps de calcul des trois formules de centres de gravité*

Si on veut obtenir la meilleure performance sur les deux critères (temps + qualité) la formule UIC devrait être appliquée, ou bien UCI aussi, parce qu'elles ont approximativement le même temps de calcul. Mais on doit ignorer IUC parce qu'elle est moins performante que les deux autres formules sur les deux critères. Pour cette raison, et pour les prochaines expérimentations nous implémenterons nos algorithmes de segmentation en utilisant la formule UIC.

4.6 Conclusion

Dans ce chapitre, nous avons réussit à présenter une nouvelle mesure de similarité entre les connaissances de type règle d'induction, nous avons prouvé mathématiquement que la formule proposée est une formule valide est classée parmi les distances métriques. Ensuite, de nouvelles formules du calcul de centre de gravité ont été proposées, implémentées à travers l'algorithme K-means-IR qui traite les règles d'induction.

Après l'implémentation et la comparaison de toutes ces formules à travers l'algorithme K-means-IR, nous avons noter que les performances de ce dernier dépendent des formules choisies, et qu'il existe un compromis entre les critères de temps de calcul et le taux de

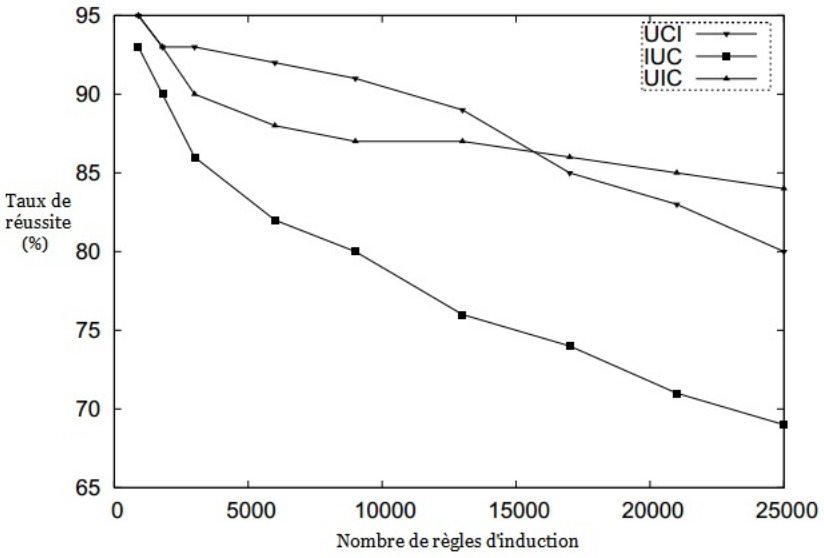

FIGURE 4.5 – *Comparaison de la qualité de la segmentation des trois formules de centres de gravité*

réussite de la segmentation. Si on veut obtenir la meilleure performance sur les deux critères (temps + qualité) la formule UIC devrait être appliquée, ou bien UCI aussi, parce qu'elles ont approximativement le même temps de calcul. Pour cette raison, et pour les prochaines expérimentations nous implémenterons nos algorithmes de segmentation en utilisant la formule UIC.

Chapitre 5

Les approches de segmentation multi-niveaux pour les règles d'induction

5.1 Introduction

La classification non-supervisée est l'une des principales tâches de l'analyse et l'exploration de données. Son but est d'extraire des nouveaux modèles ou de nouvelles connaissances sans avoir d'information préalables sur les données en entrée [HKP06] [CDD13b]. De plus, le traitement à grande échelle est le plus grand défi de la science actuellement. Pour cela, différentes approches ont vu le jour comme l'hybridation des méta-heuristiques avec les algorithmes de segmentations comme dans le travail [Wu08], ou bien en utilisant le parallélisme comme dans [ZMH09].

Dans ce chapitre, nous montrons comment on utilise le paradigme multi-niveaux (précédemment présenté), pour amoindrir la grande complexité de la segmentation à grande échelle, à travers la présentation de deux approches ; la première applique le paradigme multi-niveaux, et la deuxième en l'hybridant avec un algorithme génétique, ces approches sont appelées respectivement : multi-niveaux, et bio-inspirée multi-niveaux. Ces dernières sont implémentées, expérimentées, et comparées à travers une très grande base de règles.

5.2 L'approche multi-niveaux

Cette approche consiste à appliquer le paradigme multi-niveaux sur les connaissances de types règles d'induction, dans le but de les classifier. Le problème global est réduit au fur et à mesure en plusieurs niveaux, jusqu'à l'obtention d'un petit problème (en matière de taille) qui est plus facile à résoudre. Une fois une solution est trouvée pour ce problème de petite taille, elle sera raffinée par la suite en la propageant niveau par niveau jusqu'au dernier, qui affichera la solution finale du problème initial.
On peut imaginer l'architecture globale de cette approche comme la structure d'oignon comme illustré dans la figure 5.1 :

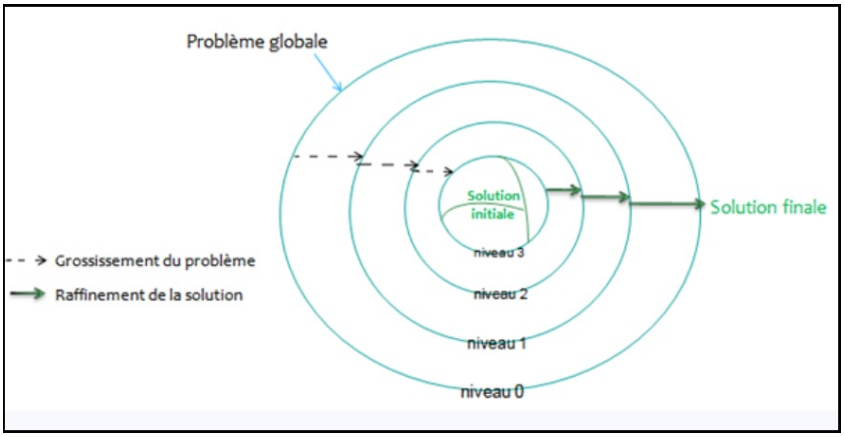

FIGURE 5.1 – *Architecture de l'approche multi-niveaux*

Le nombre de couches de l'oignon représente le nombre de niveaux dans notre approche, et la taille d'une couche d'oignon représente le nombre d'objets dans un niveau, et la résolution du problème de la classification se fait sur les trois étapes qui sont : l'étape de grossissement, la phase de la solution initiale, et l'étape de raffinement.

5.2.1 L'étape de grossissement (contraction)

Au début, il faut donner une valeur au paramètre NL qui représente le nombre de niveaux. Ensuite, en se basant sur la matrice des distances qui sépare chaque objet d'un autre, les objets les plus proches sont supprimés de la base et stockés dans une autre base (appelée base temporaire) sous forme de paquets, tel que chaque paquet représente un

niveau. Ce processus de suppression d'objets s'arrête lors de l'obtention d'une taille égale à la taille de la mini_base. L'algorithme 2 explique bien l'étape de contraction de cette approche :

Algorithm 2 Phase de grossissement de l'approche multi-niveaux

```
1: Fonction : Calculer_matrice_distance()
2: Begin
3: for i = 0 to n do
4:    for j = 0 to n do
5:       matrice_distance[i][j] ← Distance_clauses (dataset[i],dataset[j])
6:    end for
7: end for
8: End Function.
9: Entrée : NL /* Nombre de niveaux */
10: Sortie : mini_base : tableau[1...taille_mini_base] d'entier.
11: Lire (NL) /* parametre nbr de niveaux */
12: taille_niveau ← taille_dataset div NL
13: taille_mini_base ← taille_niveau + (taille_base mod NL)
14: Calculer_matrice_distance() ;
15: cpt = 0 ; cp2=0 ;
16: while cpt < NL do
17:    for i = 1 to NL /* pour chaque paquet*/ do
18:       indice_min[i] ← extraire(indice_min[i],matrice_distance)
19:       indice_min[j] ← extraire(indice_min[j],(matrice_distance)
20:       table_temporaire[cp2++]←dataset[indice_min_i]
21:       table_temporaire[cp2++]←dataset[indice_min_j]
22:       supprimer_dataset[indice_min[i]
23:       supprimer_dataset[indice_min[j]
24:       Calculer_matrice_distance() ;
25:    end for
26:    cpt++ ;cp2++ ;
27: end while
28: return mini_base[]
```

5.2.2 L'étape de la solution initiale

Après la phase de grossissement, on obtient une mini base qui maximise les distances entre les règles. Donc, il est prévisible que ces dernières balaient tout l'espace de règles de la base de règles initiale.

Dans cette étape, on applique le simple algorithme k-means-IR présenté précédemment (Algorithme1), sur la mini base, et il est prévu que les centres de gravité résultants seront

fiables. Le résultat de cette classification constitue la solution initiale au problème.

5.2.3 L'étape de raffinement (propagation de la solution)

Une fois la solution initiale obtenue, qui est représentée sous forme de classification des règles appartenant à la mini base, on applique l'algorithme K-plus-proches-voisins (KPPV) adapté au règles, pour la classification supervisée des règles restantes niveau par niveau. Le but d'utiliser l'algorithme KPPV-IR est de classifier toutes les règles de la base originale, en tenant compte de la premiere classification (de l'étape de solution initiale). L'algorithme 3 décrit KPPV-IR pour la phase de raffinement.

Algorithm 3 L'algorithme KPPV-IR pour la phase de raffinnement

```
Liste-Règle := Toutes les règles (La base de règles originale) - règles(la solution initiale
).
while Liste − Rgle n'est pas vide do
  règle_courante := extraire_règle (Liste-Règle).
  for chaque règle R_i déja classée do
    Calculer la distance Dist_Clauses(règle-courante,R_i).
  end for
  Calculer les k_Plus_proches_voisins(Rgle − courante)
  for chaque règle appartenant aux K plus proches voisins do
    Calculer le nombre de fréquence de chaque classe.
  end for
  Attribuer à "Règle-courante" la classe la plus fréquente.
end while
```

5.3 L'approche bio-inspirée multi-niveaux

Toujours en respectant le principe de base de l'approche multi-niveaux, cette approche se compose de trois parties essentielles qui sont la phase de grossissement, l'étape de la solution initiale, et en fin l'étape de raffinement. l'étape de grossissement ici est assurée grace à un algorithme génétique comme expliqué à la figure 5.2, suivi de l'algorithme k-means-IR appliqué à la mini base, et enfin la phase de raffinement est appliquée selon un algorithme classificateur qu'on expliquera par la suite.

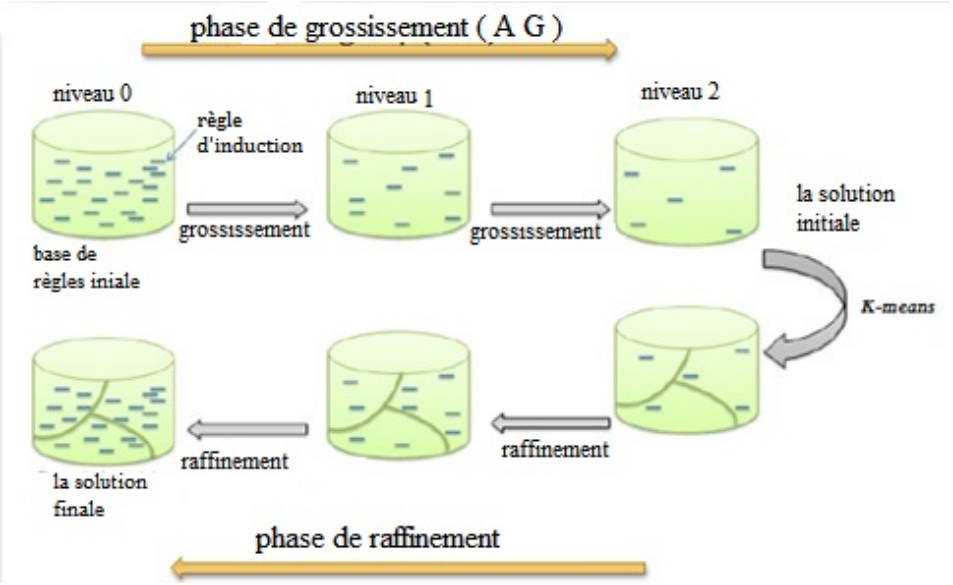

FIGURE 5.2 – *Le mécanisme de l'approche de classification bio-inspirée multi-niveaux*

5.3.1 L'étape de grossissement (contraction)

L'approche de segmentation multi-niveaux bio-inspirée commence comme pour l'approche multi-niveaux par l'élimination des plus proches règles afin de réduire la taille du problème initial, mais grâce à un algorithme génétique, contrairement à la première approche qui se base sur la matrice des distances.

L'objectif de cette phase comme déjà noté, est de réduire le nombre de règles à classifier. Pour cela un algorithme génétique est appliqué tout en assurant un balayage équitable sur toute la base de règles à traiter. Les règles les plus proches sont éliminées (temporairement) de façon à ne garder que celles qui sont dissimilaires entre elles.

Les différents composants de l'algorithme génétique appliqué sont expliqués comme suit :

- Un chromosome (une solution) est représenté par un tableau d'entiers. chaque case du tableau (représente un gène) qui varie entre 0 et la taille de la base de règles. La taille de la solution représente la taille choisie pour la mini base de règles. Par exemple, si on a le chromosome suivant (19,201,1987,2012,3033,4200), la mini base contiendra 6 règles, qui sont localisées comme suit : la première à la position 19 de la base de de règle originale, la seconde à la position 201, etc..
- La fonction Fitness : La notion de fitness est fondamentale pour l'application des

algorithmes génétiques. c'est une valeur numérique qui permet d'évaluer la performance d'un chromosome (une solution). Donc, pour comparer les différentes solutions nous proposons une fonction fitness égale à la somme de distances entre tous ses gènes comme suit :

$$fitnesse(chromosome) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Sim\_Clauses(i, j). \tag{5.1}$$

pour tout i différent de j, avec i et j appartenant au chromosome.

- La population initiale : avant le commencement de la phase de contraction, l'algorithme génétique choisit aléatoirement une population initiale qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global.
- Le croisement : le but du croisement est d'atteindre des régions de solutions plus performantes de l'espace de recherche. De nouvelles solutions sont créées par la combinaison de paires d'individus de la population en appliquant l'opérateur de croisement à chaque paire choisie. Une solution non-visitée i_k est visitée aléatoirement avec une solution non-visitée i_l. Ensuite, l'opérateur de croisement est appliqué à un point choisit aussi aléatoirement. Cette opération génère deux nouveaux individus appelés les fils de la paire d'individus i_k et il.
- La mutation : Le principe de la mutation est de générer des individus modifiés en introduisant des nouveaux attributs. Avec la mutation, les gènes du chromosome ont la chance d'être modifiés par un autre gène. L'opérateur de mutation prend en considération un paramètre Pm qui spécifie la probabilité d'appliquer la mutation. Dans notre implémentation, l'opérateur de mutation choisit un gène de façon aléatoire et le modifie en le flippant en une autre règle. La probabilité de la mutation assure théoriquement que chaque région de l'espace de recherche soit explorée.
- La sélection : après l'application du croisement et de la mutation, la sélection est appliquée à son tour sur la population courante. Dans cette étape, la recherche de la solution globale devient de plus en plus claire. Selon la fonction fitness présentée précédemment, les meilleurs individus sont sélectionnés pour la prochaine génération. La sélection est stochastique et biaisée pour les meilleurs individus.

Avec les opérateurs présentés précédemment, l'algorithme de l'étape de grossissement peut être appliqué comme le montre l'algorithme 4.

Algorithm 4 L'algorithme génétique pour la phase de grossissement

```
Fixer les paramètres N et max-iter /*N : la taille de la population.*/
Fixer les paramètres taux_de_mutation, taux_de_croisement /* un pourcentage entre [10%-80%]*/
Générer aléatoirement une population initiale P_0 (de solutions).
Calculer Fitness(S) pour chaque solution qui fait partie de la population initiale
while i < Max-Iter do
    Générer RC et RM : des entiers aléatoires entre[1 and 100]/*taux de croisement et de mutation */.
    if RC > taux de croisement then
        Appliquer l'opérateur de croisement sur 2 individus choisis aléatoirement.
    end if
    if RM > taux de mutation then
        Appliquer l'opérateur de mutation sur 1 individu choisi aléatoirement.
    end if
    Evaluer fitness(S') /* S' est la population P_i./
    Considérer les N meilleures solutions.
    i++ ;
end while
Choisir la meilleure solution.
```

5.3.2 L'étape de la solution initiale

A la fin de l'étape de grossissement, une (meilleure) solution est obtenue qui maximise la distance entre les règles de la base de règles originale. Cela signifie que cette dernière assure un très bon balayage de tout l'espace de règles. Dans cette étape, l'algorithme K-means-IR est appliqué sur cette solution pour le groupement de ses règles, et c'est prévisible que les centres des clusters obtenus soient fiables. Les clusters obtenus constituent la solution initiale.

5.3.3 L'étape de raffinement

Après l'obtention de la solution initiale (qui est une classification de la mini base de règles), et dans le but de reconstituer toute la base de connaissances initiale, on applique un algorithme classificateur appelé PPC (le Plus Proche Centre de gravité) (Algorithme 5). Le principe de cet algorithme est de calculer pour chaque règle non-classée les distances

qui les séparent de chaque centre de groupe, ensuite l'affecter au cluster dont le centre est le plus proche, contrairement à l'étape de raffinement de l'approche multi-niveaux qui était l'application directe de l'algorithme KPPV, qui calculait les distances qui séparent la règle non-classée à toutes les règles déjà classées d'une façon exhaustive.

Algorithm 5 L'algorithme PPC pour la phase de raffinement

```
Liste_Règles := toutes les règles (base de règles initiale) - règles(solution initiale).
while Liste_Rgles n'est pas vide do
  Règle_courante := extraire_règle (Liste_Règles).
  for chaque centre de gravité C_i do
    Calculer la distance Sim_C (Règle_courante,C_i).
  end for
  Extraire_le_plus_proche_centre (Règle_courante)
  Attribuer à Règle_courante le groupe du plus proche centre de gravité.
end while
```

5.4 Évaluations et expérimentations des approches multi-niveaux

Avant de comparer les performances des deux approches multi-niveaux, nous avons réalisé quelques expérimentations de chaque approche à part, avec différentes valeurs de paramètres afin d'en choisir les meilleures. Les résultats de nos expérimentations sont présentés dans les sections suivantes.

5.4.1 Expérimentation de l'approche multi-niveaux

Lorsqu'on fixe le nombre de règles à traiter à 700 avec un nombre de clauses pour chaque règle à 320 clauses. et en variant les valeurs de paramètres NL (nombre de niveaux) et K (nombre de clusters) de l'algorithme multi-niveaux, les résultat obtenus apparaissent sur la table 5.1.
Chaque valeur dans les tables et les figures suivantes est la moyenne des résultats de cinq exécutions successives.

Nous remarquons bien d'après les résultats de la table 5.1, que lors de l'augmentation de la valeur du nombre de niveaux (NL), le temps d'exécution se réduit, et nous expliquons

Nombre de niveaux (NL)	**Nombre de Clusters (K)**	**Qualité du clustering (MADP)**	**Temps d'exécution (milliseconde)**
3	5	54737	238
3	10	39445	299
3	15	34011	395
3	20	32087	413
3	35	27830	507
3	50	25274	790
3	75	37240	904
3	100	46192	1048
7	5	43223	176
7	10	23760	226
7	15	11485	287
7	20	12985	295
7	35	10887	354
7	50	11203	423
7	75	12822	501
7	100	10898	676
15	5	58759	165
15	10	35538	187
15	15	12264	239
15	20	10848	318
15	35	8945	357
15	50	3335	383
15	75	3031	489
15	100	4653	599

TABLE 5.1 – Expérimentations de l'approche multi-niveaux

ce phénomène par le fait que quand le nombre de niveaux augmente, la taille de ce dernier se réduit, ce qui implique que la taille de la matrice de distances entre les règles dans chaque niveau sera plus petite, et qui par conséquent prend moins de temps de calcul pour l'extraction des plus proches règles (à la phase de contraction).

De plus, nous faisons noter qu'il y a un compromis à faire lors du choix du couple de paramètres (temps d'exécution, qualité de segmentation), de façon à ce que lorsqu'on cherche d'accélérer le processus de segmentation sans faire attention à la qualité du clustering. On choisit un paramètre qui maximise le nombre de niveaux NL, et qui minimise le nombre de groupes K, par exemple avec ce jeu de règle le couple (NL= 15, K=5).

Par ailleurs, si on cherche une meilleure qualité de segmentation sans prendre en considération le temps d'exécution, on choisit un paramètre qui maximise le nombre de groupes K et qui minimise le nombre de niveaux comme dans notre exemple (NL= 15, k=75).

Dans notre cas, on essaie d'avoir des valeurs à ces paramètres satisfaisant approximativement les deux critères en même temps, et d'après la table 5.1 on remarque que les valeurs (NL= 15, K = 20) donnent des bons résultats.

5.4.2 Expérimentation de l'approche bio-inspirée multi-niveaux

Dans cette partie, on fait varier les valeurs des paramètres de l'approche de segmentation bio-inspirée multi-niveaux, afin de choisir les meilleures valeurs.

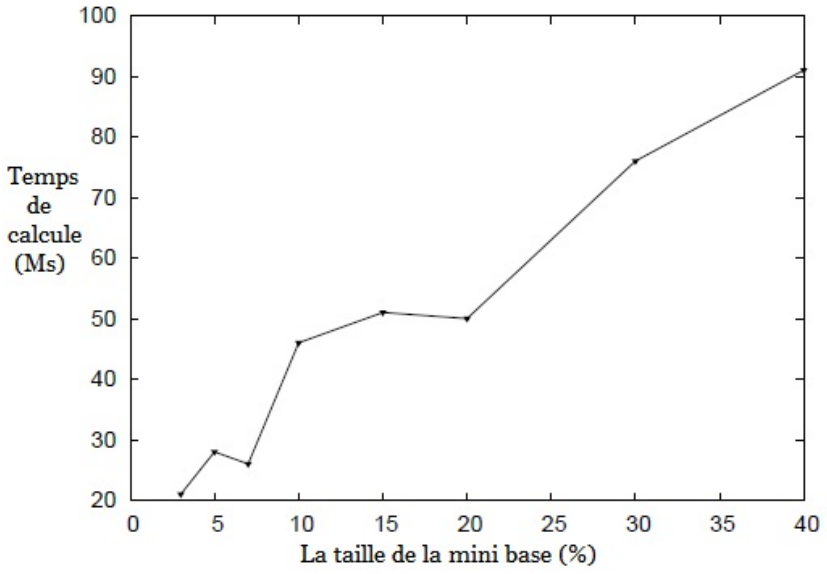

FIGURE 5.3 – *Temps d'exécution de l'approche B-I multi-niveaux*

La figure 5.3 montre comment le temps d'exécution de l'approche bio-inspirée multi-niveaux change, lors de l'augmentation de la valeur du paramètre de la taille de la mini-base de règle.

On remarque bien que lorsqu'on augmente la taille de la mini-base de 3% de la taille de la base de règles originale à 40%, le temps d'exécution augmente de 21 millisecondes à 91 millisecondes.

La figure 5.4 montre le changement du taux de réussite de la classification lors de l'augmentation de la valeur du paramètre $taille\_de\_la\_mini\_base$.

Le taux de réussite ici est calculé par la formule $MADP$ présentée comme suit à

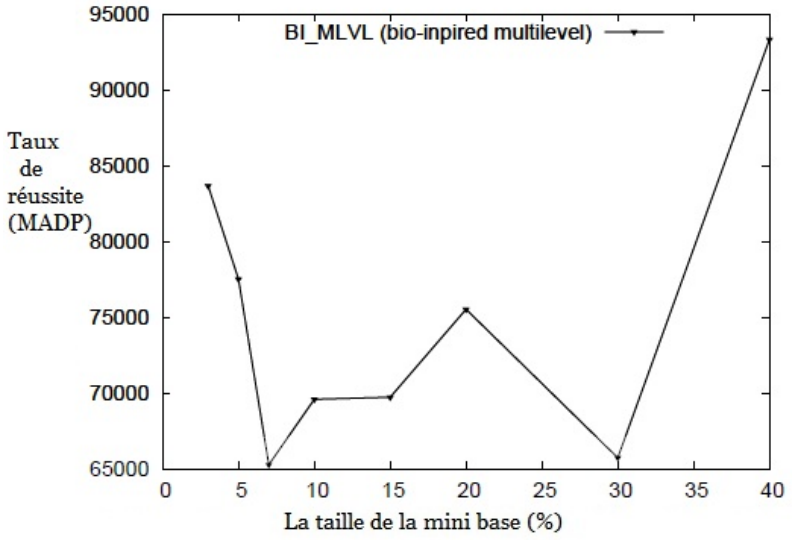

FIGURE 5.4 – *MADP de l'approche B-I multi-niveaux*

l'équation Eq 5.2. Le principe de cette formule est de minimiser la distance entre les règles du même groupe [DHD13] (minimiser la distance intra-groupes).

$$MADP = \sum_{l=1}^{K} \sum_{i=1}^{|Clusterl|} \sum_{j=1}^{|Clusterl|} Sim\_Clauses(R_i, R_j). \tag{5.2}$$

Où ; R_i et R_j des règles appartenant au même groupe.

On remarque dans la figure 5.4 que le meilleur résultat obtenu est lorsque la taille de la mini-base est égale à 7% et 30% de la taille de la base de règle originale.

D'après ces résultats, et pour la suite des expérimentations et comparaisons, on fixe le paramètre $taille\_de\_la\_mini\_base$ à 7%, parce qu'avec cette valeur, l'approche bio-inspirée prend un temps d'exécution plus faible qu'à celui avec la valeur 30%.

5.4.3 Comparaison des approches de segmentation multi-niveaux proposées

Dans cette partie, nous comparons les performances des trois approches de segmentation de règles proposées jusqu'ici et qui sont, le simple algorithme K-means-IR, l'approche multi-niveaux, et l'approche bio-inspirée multi-niveaux, sur les deux critères du temps

d'exécution, et du taux de réussite de la segmentation.

Pour cela, nous fixons les paramètres de chaque approche aux valeurs qui maximisent sa performance (comme montré dans les sections précédentes) et qui sont :

- Le paramètre K (nombre de groupes) à 15 pour toutes les approches.
- Le nombre de niveaux NL à 15 pour l'approche multi-niveaux.
- Le paramétre de la taille de la mini-base à 7% pour l'approche bio-inspirée multi-niveaux.

5.4.3.1 Comparaison du temps d'exécution

Lorsqu'on augmente le nombre de règles à classifier, tout en calculant le temps d'exécution de chaque approche, les résultats obtenus sont montrés à la figure 5.5.

On rappel ici que chaque point est le résultat de la moyenne de cinq essais successifs.

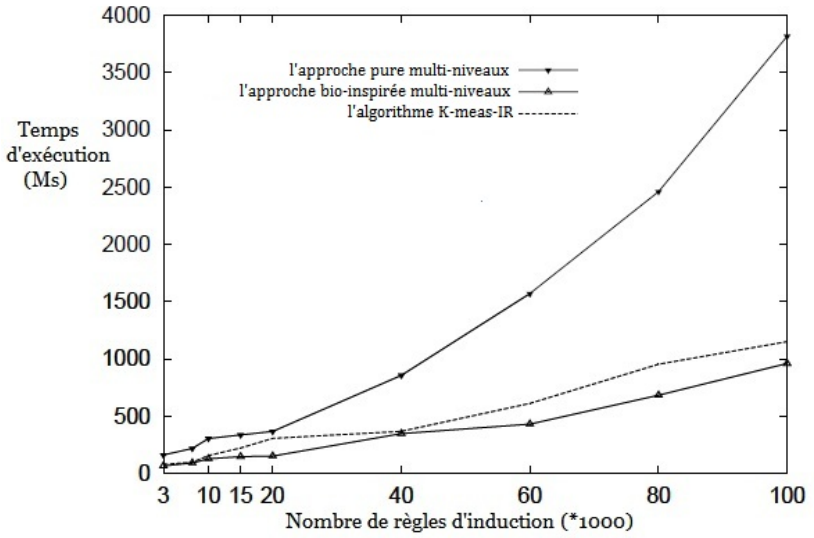

FIGURE 5.5 – *Comparaison de temps d'exécution des trois approches*

Fig 5.5 montre le changement du temps d'exécution des trois approches de segmentation en augmentant le nombre de règles à traiter. On remarque que les meilleurs résultats obtenus sont ceux de l'approche bio-inspirée. Quand le nombre de règles change de 3,000 à 100,000, le temps d'exécution ne dépasse pas 1 seconde, suivi de l'algorithme K-means-IR qui prend un peut plus de temps d'exécution par rapport a la première approche. Contrai-

rement à l'approche multi-niveaux, qui prend le plus de temps des trois approches, elle s'exécute en plus de 3500 millisecondes quand le nombre de règle est à 100,000.

5.4.3.2 Comparaison du taux de réussites de segmentation

La comparaison de la qualité de clustering est calculée dans un premier temps par la formule MADP 5.2 présentée précédemment divisée par le nombre de règles, afin d'éviter d'avoir des nombres excessifs lors du traitement avec un grand nombre de règles, et de pouvoir les schématiser correctement.
Pour cette raison, les résultats schématisés sur la figure 5.6 sont des valeurs obtenus par la formule MADP' que nous l'avons définis comme suit :

$$MADP' = \frac{MADP}{nombre\_de\_rgles}. \tag{5.3}$$

Les résultats obtenus sont schématisés sur la figure 5.6.

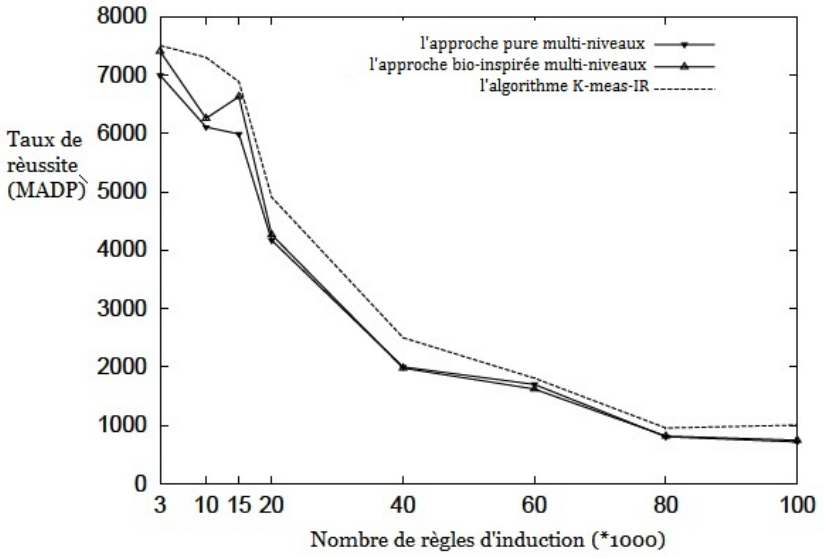

FIGURE 5.6 – *Comparaison de taux de réussite (MADP) des trois approches*

La figure 5.6 montre le changement de la qualité de la segmentation des trois approches lors de l'augmentation du nombre de règles. D'après la figure, on note que le taux de réussite de l'approche multi-niveaux est plus performant que les deux autres approches (bio-inspirée multi niveaux, et le simple K-means-IR), suivi par l'approche bio-inspirée qui donne des

résultats très proches de celles de l'approche multi-niveaux. On explique ces résultats par le fait que les approches de segmentation multi-niveaux sont basées sur un calcul spécifique et intelligent des centres de gravité des groupe dès la phase de grossissement, contrairement à l'algorithme simple K-means-IR qui se base sur une initialisation aléatoire des centres, ce qui provoque l'instabilité de son taux de réussite.

On remarque ici aussi, que les résultats obtenus par la formule MADP 5.6, peuvent être influencés par le nombre de règles traitées. Pour cette raison, on recompare le taux de réussite de la classification une seconde fois en utilisant la formule F-mesure très connue pour sa fiabilité dans le domaine de la fouille de données et la recherche d'information. La formule F-mesure est expliquée comme suit dans Eq 5.4 :

$$F\_mesure_{\beta} = \frac{(\beta^2+1)\cdot P \cdot R}{\beta^2 \cdot P + R}, P = \frac{TP}{TP+FP}, R = \frac{TP}{TP+FN} \tag{5.4}$$

où :

P représente le taux de précision et R le taux de rappel

TP (true positives en anglais) représente le nombre de vrais positifs.

TN (true negatives en anglais) représente le nombre de vrais négatifs.

FP (false positives en anglais) représente le nombre de faux positifs.

FN (false negatives en anglais) représente le nombre de faux négatifs.

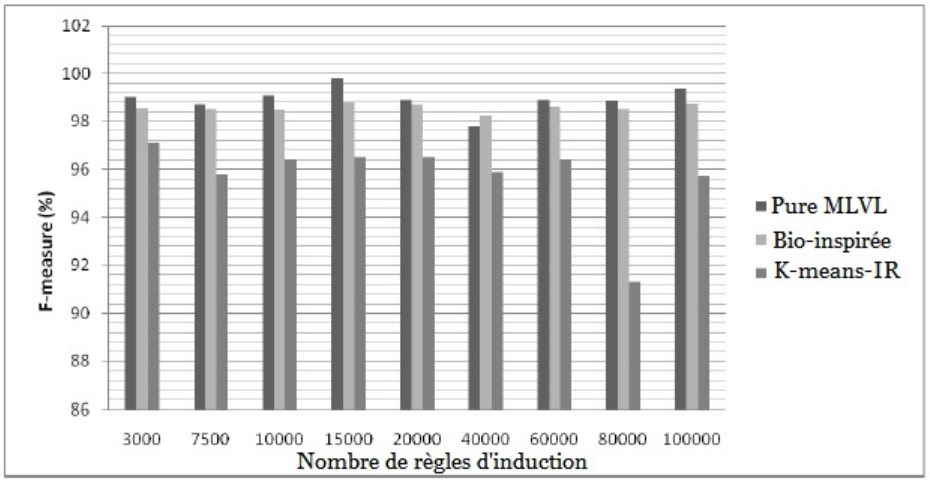

FIGURE 5.7 – *Comparaison de taux de réussite des trois approches (f-mesure)*

La figure 5.7 montre le changement du taux de réussite calculé par la F-mesure (Eq 5.4), avec l'augmentation du nombre de règles à classifier pour les trois approches proposées (multi-niveaux, bio-inspirée multi-niveaux, et simple K-means-IR).
Comme pour la première comparaison avec la formule de MADP, ici aussi les approches basées sur le paradigme multi-niveaux sont plus efficaces que la simple approche (k-means-IR), et ne descendent pas au dessous de 98 % lors de l'augmentation du nombre de règles de 3,000 à 100,000. Cela est dû au fait que les approches de segmentation multi-niveaux sont basées sur le calcul intelligent et spécifique des centres de gravité des groupes à l'étape de grossissement, contrairement à l'approche simple k-means-IR qui commence toujours par une affectation aléatoire des centres de gravité des groupes. Par ailleurs, on explique le fait que l'approche pure-MLTNV est plus performante en taux de réussite que l'approche Bio-inspirée-MLTNV, par la raison que la première approche se base sur le calcul exact des distances entre les règles grâce à une matrice lors de la phase de segmentation, contrairement à l'approche bio-inspirée qui se base sur une méta-heuristique qui aboutie au final à une solution approchée, mais en un temps d'exécution plus rapide que celui de l'approche pure-MLTNV.

5.5 Conclusion

Dans ce chapitre, nous avons présenté deux nouvelles approches de fouille de connaissances basées sur le paradigme multi-niveaux. La première est l'application directe du principe du multi-niveaux sur l'algorithme K-means-IR présenté au chapitre précédent. La deuxième approche, est une hybridation de la première avec les algorithmes génétiques. Son but est d'accélérer le processus de segmentation, et plus précisément, la phase de grossissement ou la première approche prenait un temps de calcul considérable pour réduire la base de règles de façon exacte.
Les résultats obtenus suite au différentes expérimentations montrent que l'approche multi-niveaux bio-inspirée est la plus performante en matière de temps d'exécution que les deux autres (multi-niveaux, et le simple K-meas-IR). En terme de qualité de segmentation, l'approche multi-niveaux est la plus performante suivie de près par l'approche bio-sinspirée.

Chapitre 6

Les approches de segmentation incrémentale pour les règles d'induction

6.1 Introduction

La grande masse de données, et la complexité de calcul des méthodes de fouille de données classique, sont des facteurs qui ont motivé le développement de nouvelles approches de fouille, tel que le parallélisme, le multi-niveaux,...etc.
De plus, le coût très élevé du calcul lors des mises à jour de ces bases a fait naitre l'approche incrémentale, qui se base principalement sur la réutilisation des connaissances acquises déjà, et d'éviter de refaire tout le processus de fouille depuis le départ à chaque mise à jour.
L'approche incrémentale a été adoptée récemment par des chercheurs dans plusieurs tâches de fouille de données telle que : la recherche des motifs séquentiels [YLC$^+$08] [EL09], l'extraction des règles d'association [LLH12], la classification non-supervisée[DLLL12],...etc.
Dans ce chapitre nous montrons l'adaptation du paradigme incrémental sur la classification non-supervisée des règles d'induction à travers trois nouvelles approches, la première est une extension de l'approche incrémentale classique pour le traitement de règles d'induction, et deux autres approches basées sur le traitement de paquets de règles.
Ces approches qui sont appelées respectivement SIRI-classique (Segmentation Incrémentale des Règles d'Induction Classique), SIRI par fusion de paquets, et SIRI par traitement simple de paquets sont implémentées et appliquées sur un large benchmark contenant plus

de 100,000 règles.

Le principe de chaque approche, ainsi que leurs implémentation et comparaison sont expliqués dans la suite de ce chapitre. Ces approches ont déjà été publiées dans une précédente conférence [CDD13a].

6.2 L'approche de segmentation incrémentale classique (SIRI-classique)

L'approche SIRI-classique (Segmentation Incrémentale des Règles d'Induction classique) est inspirée directement de la fouille de données incrémentale, initialement elle se base sur la segmentation simple de la base de règles. Après l'arrivée d'une nouvelle règle, celle ci est traitée en utilisant les métas-règles déjà acquises. La figure6.1 montre bien le mécanisme de cette approche.

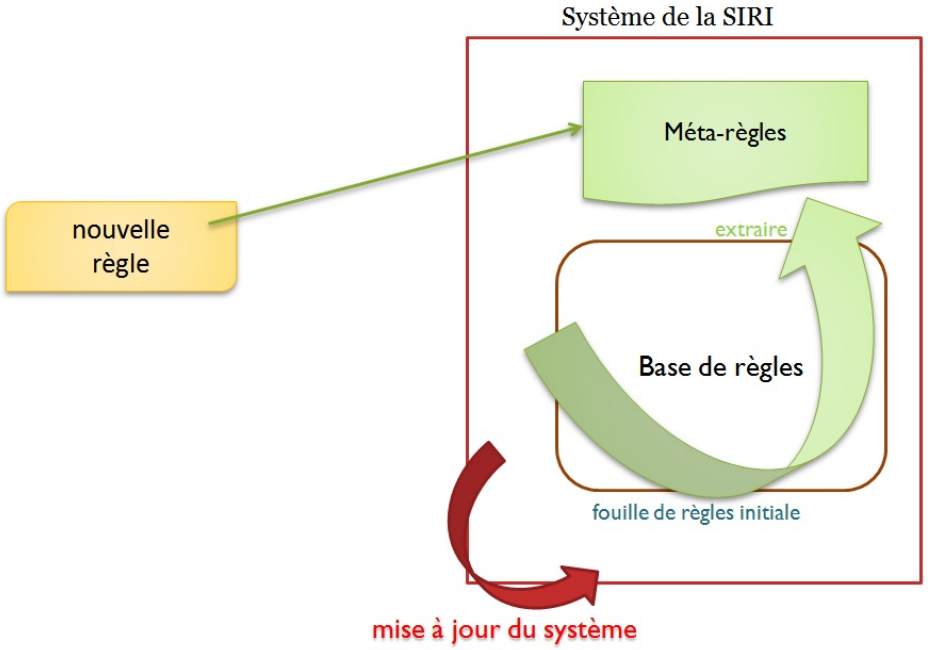

FIGURE 6.1 – *Mécanisme de l'approche de SIRI-Classique*

Les méta-règles dans notre conception représentent le centre de gravité de chaque groupe. Donc à chaque arrivée d'une nouvelle règle, on calcule les distances qui la séparent à chaque méta-règle, afin d'en choisir le plus proche. Ensuite, on vérifie si la distance qui la

sépare à la plus proche méta-règle est inférieure au paramètre "seuil-distance". On l'affecte dans ce cas au groupe de cette méta-règle, sinon on lui crée un nouveau groupe, et cette même règle en devient une méta-règle. Le principe de cette approche est bien résumé dans l'algorithme 6.

Algorithm 6 Algorithm de la SIRI-classique

/*Partie N°1 : traitement de la base de règles initiale */
Appliquer l'algorithme K-means-IR (la base de règles initiale) /* La Segmentation simple de la base de règles initiale */.
seuil_distance = max_distance_intra_cluster.
Extraction et sauvegarde des méta-règles (clusters de la base de règles initiale).
/* Partie N°2 : Traitement des nouvelles règles d'inductions */
while Liste_nouvelles_règles[] $\neq \emptyset$ **do**
  /* Calcul de distance entre règle[i] et chaque mèta-règle[j].*/
  Extraire la distance_min (règle[i],base_de_méta_rgles).
  if distance_min(régle[i],méta_régle[j]) < seuil_distance **then**
    Affecter régle[i] au groupe de la méta-règle[j].
  else
    Créer un nouveau groupe pour règle[i].
  end if
  Extraction et sauvegarde des méta-règles (nouveaux clusters de la base de règles).
end while

On note ici, que le paramètre seuil-distance représente la distance maximale qui sépare deux règles dans un même groupe. Ce paramètre est dynamique et capable de changer à chaque itération du processus, et son but est d'assurer le quatrième critère du principe du paradigme incrémental (les critères du paradigmes incrémental sont détaillés au troisième chapitre).

On remarque aussi que la complexité de calcul de cette approche n'est pas stable, entre le traitement initial de la base et les mises à jours répétitives à chaque traitement d'une nouvelle règle. Ceci nous a poussé à concevoir deux autres approches basées sur le traitement des règles par paquets pour essayer de garder une stabilité de complexité de calcul tout au long du processus.

6.3 L'approche de segmentation incrémentale par traitement simple de paquets (SIRI-TSP)

L'approche SIRI-TSP consiste à diviser la base de connaissances initiale en n paquets, et d'une manière incrémentale la base initiale est segmentée paquet après paquet tout en sauvegardant les méta-règles acquises au $paquet_i$ pour les utiliser au traitement du $paquet_{i+1}$. La mise à jour du système ne se fait qu'après le traitement de tous les paquets, contrairement à la première approche (SIRI-classique) qui répète la mise à jour à chaque traitement d'une règle. La Figure 6.2 résume le principe de fonctionnement de cette approche.

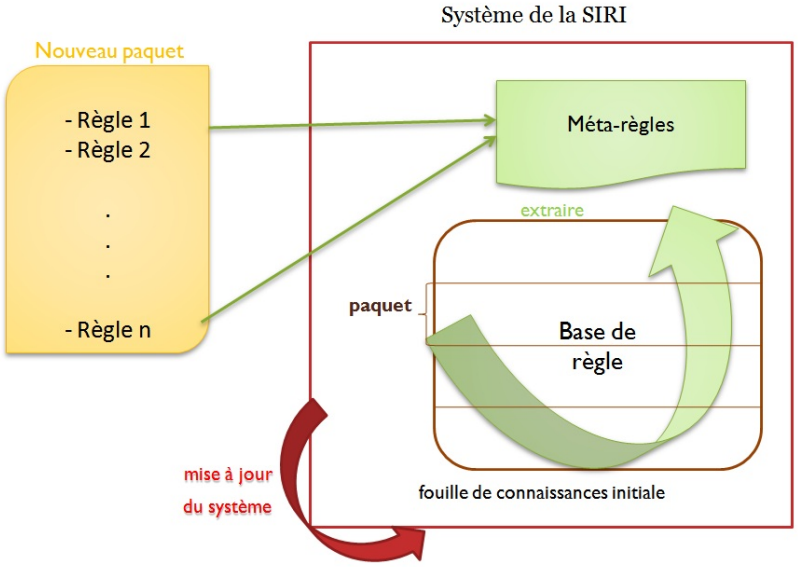

FIGURE 6.2 – *Mécanisme de l'approche SIRI-TSP*

L'algorithme 7 montre en détails le fonctionnement de l'approche de segmentation incrémentale de règles d'induction par traitement simple de paquets (SIRI-TSP).

6.4 L'approche de segmentation incrémentale par fusion de paquets (SIRI-FP)

Le principe de la fouille initiale de la base de connaissance dans l'approche SIRI-FP est le même que celui de la fouille par paquet simple (SIRI-TSP), c'est-à-dire diviser la base de connaissances en n paquets, et extraire les méta-règles à chaque fin de paquet, pour

Algorithm 7 Algorithme de la SIRI-TSP

1: Fixer le paramètre n /* le nombre de paquets */
2: Diviser la base de règles en n paquets de tailles égales.
3: **for** i = 1 à n **do**
4: segmenter le $paquet_i$ en utilisant les méta-règles acquises à l'étape du $paquet_{i-1}$.
5: Sauvegarder les méta-règles après le traitement du $paquets_i$.
6: **end for**
7: seuil_distance = max_intra_distance.
8: **while** liste_des_paquets $\neq \emptyset$ **do**
9: Affecter chaque règle du nouveau paquet au groupe le plus proche, en prenant en compte seuil_distance et les méta-règles déjà acquises.
10: Sauvegarder les nouvelles méta-règles.
11: **end while**

obtenir des nouvelles méta-règles et de les réutiliser lors de l'arrivée des nouveaux paquets de connaissances, et ainsi de suite.

La différence entre les deux approches réside lors de l'arrivée d'un nouveau paquet de règles, de telle sorte que l'approche SIRI-FP contrairement à l'approche précédente, traite l'ensemble des règles du nouveau paquet à part, et en extraire ses méta-règles. Après cela, la procédure de fusion est appliquée, pour fusionner les plus proches groupes ensemble en prenant en compte le paramètre seuil_distance, et les distances entre les nouvelles méta-règles et les anciennes. La figure 6.3 est un schéma explicatif de cette approche.

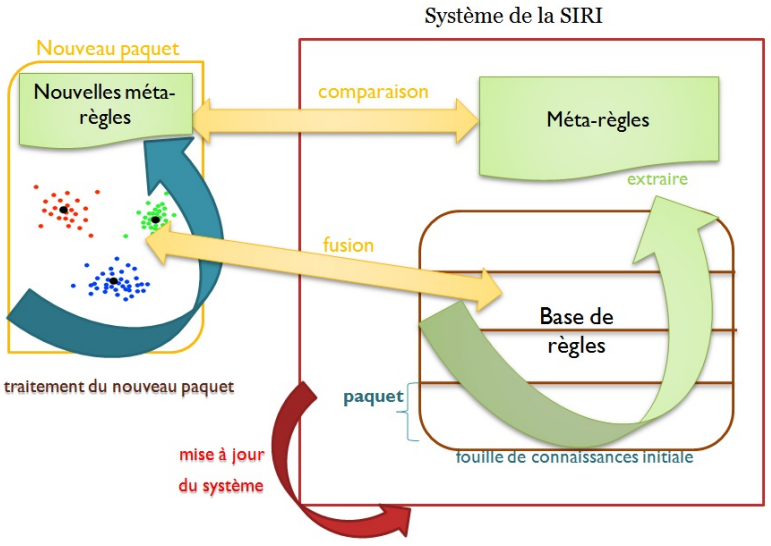

FIGURE 6.3 – *Mécanisme de l'approche de SIRI-FP*

L'algorithme 8 montre avec plus de détails le fonctionnement de l'approche de segmentation incrémentale de règles d'induction par fusion de paquets (SIRI-FP).

Algorithm 8 Algorithme de la SIRI-FP

1: Fixer le paramètre n /* le nombre de paquets */
2: Diviser la base de règle en n paquets égaux.
3: **for** i = 1 à n **do**
4: segmenter le $paquet\_i$ en utilisant les méta-règles acquises à l'étape du $paquet\_i - 1$.
5: Sauvegarder les méta-règles après le traitement du $paquets\_i$.
6: **end for**
7: seuil_distance = max_intra_distance.
8: **while** liste_des_paquets $\neq \emptyset$ **do**
9: Segmentation du nouveau paquet à part.
10: fusionner les nouveaux groupes avec les anciens /* en prenant en compte le paramètre seuil_distance, et les méta-règles de la base de règles initiale.*/
11: Sauvegarder l'ensemble de nouvelles méta-règles.
12: **end while**
13: mettre à jour le système (recalculer les centres de gravité).

6.5 Évaluations et expérimentations des approches incrémentales

Dans cette partie, nous allons comparer les différentes approches proposées pour la fouille incrémentale de règles d'induction, en utilisant toujours la même collection de connaissances présentée au chapitre 4 dans la table4.4, et aussi les mêmes formules de calcule du taux de réussite de segmentation à savoir la formule présentée dans l'équation 4.3 et la F-mesure dans l'équation 4.4 (chapitre 4).

La figure 6.4 nous montre la comparaison du temps d'exécution des trois approches de la SIRI. En analysant bien la figure, on peut remarquer qu'il n'y a pas une grande différence entre les temps d'exécution des trois approches lorsqu'on traite une base de règles inférieure à 20,000 règles d'induction. Par ailleurs, quand le nombre de règles est supérieur à 20,000 l'approche SIRI-classique est plus rapide que les deux autres, de sorte que lorsque la base de règles contient 100,000 instances, le temps d'exécution de la SIRI-classique est de 31,75 secondes, suivi par l'approche de la SIRI-TSP avec 40,77 secondes, et 48,69 secondes pour l'approche SIRI-FP.

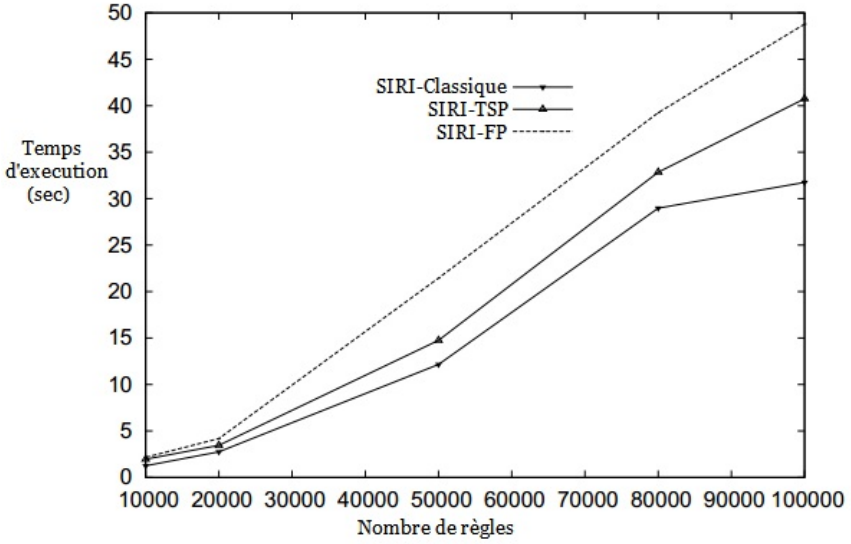

FIGURE 6.4 – *Comparaison des temps d'exécution des approches de la SIRI*

Nous expliquons ces résultats par le fait que l'approche SIRI-classique, n'applique la mise à jour (le recalcul du centre de gravité) que sur le cluster dont la nouvelle règle est affectée. Contrairement à l'approche SIRI-FP qui prend énormément de temps lors de l'étape de fusion à chaque arrivée d'un nouveau paquet de règles.

Lorsqu'on compare la qualité de la segmentation des trois approches en utilisant l'équation 4.3, on obtient les résultats schématisés à la figure 6.5.

D'après cette figure, on peut noter que les meilleurs résultats obtenus sont réalisés par l'approche de la SIRI par fusion de paquets, suivis par l'approche SIRI-TSP. Lors de l'augmentation du nombre de règles à traiter de 10,000 à 100,000 le taux de réussite de l'approche SIRI-FP est réduit de 92,35% à 88%, suivi de la SIRI-TSP de 89,75% à 86%, où le taux de réussite de l'approche SIRI simple est réduit de 88,35% à 84%.

De même, lors de la comparaison de la F-mesure des approches de segmentation incrémentale tout en augmentant le nombre de règles à traiter, comme montré à la figure 6.6, ou en augmentant le nombre de groupes (paramètre K) montré à la figure 6.7, pratiquement dans tous les cas, les résultats obtenus sont semblables, la meilleure qualité de segmentation

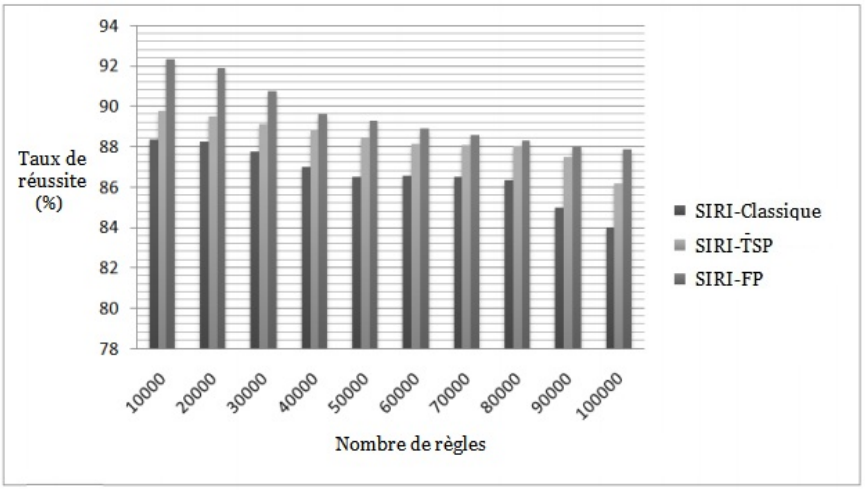

FIGURE 6.5 – *Comparaison de la qualité de la segmentation des approches de la SIRI*

est obtenue par l'approche SIRI-FP suivie par la SIRI-TSP.

D'après ces résultats, on peut noter que la performance des approches de segmentation incrémentale (temps d'exécution + qualité de segmentation) dépend de l'approche choisie parmi les trois présentées précédemment. De plus, il existe un compromis entre la qualité de la segmentation et le temps de calcul. Si on veut obtenir une meilleure qualité de segmentation, l'approche SIRI par fusion de paquets devrait être appliquée. Par contre, si on veut accélérer le processus de segmentation, l'approche SIRI classique devrait être appliquée.

6.6 Conclusion

Dans ce chapitre, on a présenté un nouveau concept de fouille de règles d'induction incrémentale. Tout en assurant une segmentation en ligne, trois approches on été proposées (SIRI-simple, SIRI-TSP, et la SIRI-FP).
Ces trois approches sont testées et évaluées sur cinq différents benchmarks sur les deux critères ; temps d'exécution et la qualité de la classification. Les résultats obtenus montrent qu'il existe un compromis, entre la qualité de segmentation, et le temps d'exécution. Pour plus de précision, si l'utilisateur souhaite obtenir une meilleure qualité de segmentation, il

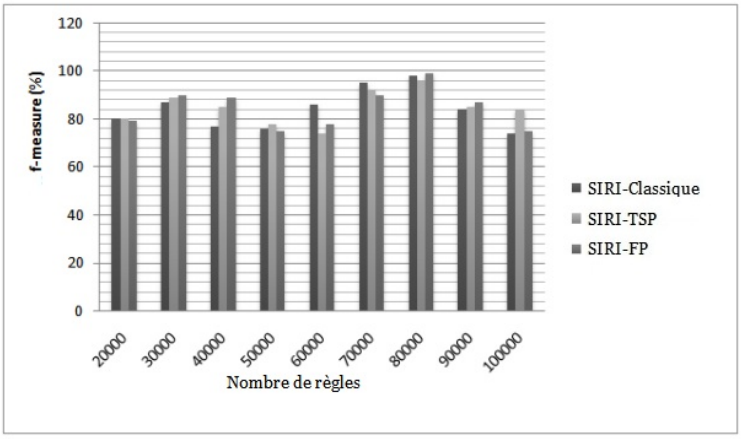

FIGURE 6.6 – *Comparaison de la F-mesure des approches de la SIRI*

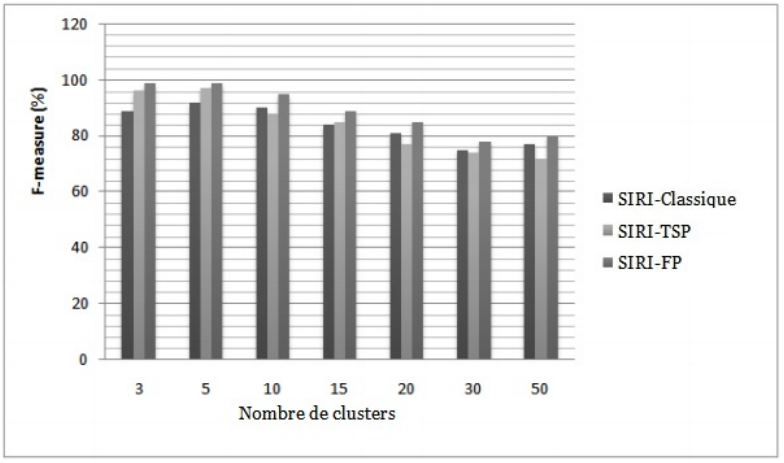

FIGURE 6.7 – *Comparaison de la F-mesure des approches de la SIRI*

aura à choisir l'algorithme par fusion de paquets. Par contre, s'il veut accélérer le processus de segmentation, l'approche SIRI simple devrait être appliquée.

Dans la prochaine partie, on va ajouter le module présenté de la fouille de règles d'induction dans l'architecture d'un agent cognitif, dans le but d'améliorer son raisonnement sur sa base de règle, et d'en faire un agent plus rapide et plus robuste lors de la phase d'inférence.

Troisième partie

Application de la fouille des règles d'induction à la technologie des agents

Chapitre 7

Proposition de la nouvelle architecture d'Agent Super Intelligent (ASI)

7.1 Introduction

Les problèmes de nos jours, sont devenus de plus en plus complexes. La résolution de ces problèmes avec l'intelligence artificielle classique nécessite un raisonnement délicat. Une issue à ce problème est l'intelligence artificielle distribuée. Rappelons ici que le concept de base de l'intelligence artificielle distribuée est l'agent cognitif [KJ98] [Lei09].
La principale caractéristique de l'agent est son degré d'autonomie. De plus, les agents sont capables de partager l'information afin d'atteindre leurs buts. Par conséquent, l'agent peut exprimer ses interactions à travers son comportement.
Ainsi, les agents peuvent être définis à partir d'autres caractéristiques comme leurs capacités d'apprentissage, leurs coopérations et leurs mobilités.
Il existe plusieurs types d'agents cognitifs selon un certain degré d'intelligence. Nous allons passer en revue dans ce chapitre les différents types d'agents, suivi d'une synthèse sur ces derniers afin de pouvoir extraire les insuffisances de la technologie des agents intelligents actuels. Par la suite, nous montrerons l'application du concept proposé de la fouille de règles d'induction à l'agent classique, qui fera naitre par la suite une nouvelle architecture d'agents super-intelligents. Enfin, nous conclurons ce chapitre avec quelque expérimentations et quelques comparaison des performances d'inférence de l'agent cognitif classique

avec la nouvelle architecture d'agent super-intelligent (ASI).

7.2 Le concept d'agent

De nos jours, le mot « agent » est utilisé dans plusieurs domaines, et de ce fait, plusieurs sens lui sont attribués. A noter que, même dans le domaine de l'informatique, plusieurs chercheurs ont défini le concept d'agent intelligent de manières différentes. Nous citons quelque uns parmi les définitions suivantes :

Définition 1 [PW05] « Un agent est un système informatique situé dans un environnement, capable d'appliquer des actions de manière autonome dans cet environnement afin d'atteindre son objectif de conception ».

Définition 2 [Sho93] « Un agent est une entité qui fonctionne continuellement et de manière autonome dans un environnement où d'autres processus se déroulent et d'autres agents existent ».

Définition 3 [WJ95] « Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre les objectifs (buts) pour lesquels il a été conçu ».

Définition 4 [RNC+95] « Un agent est une entité qui perçoit son environnement et agit sur celui-ci ».

Définition 5 [WJ99] « Les agents intelligents sont des entités logicielle qui réalisent des opérations à la place d'un utilisateur ou d'un autre programme, avec une sorte d'indépendance ou d'autonomie, et pour faire cela, ils utilisent une sorte de connaissances ou de représentations des buts ou des désirs de l'utilisateur ».

Définition 6 [WJK00] « Un agent est une entité logicielle autonome qui fonctionne continuellement afin d'atteindre un ensemble de buts d'une autre entité humaine ou bien d'un système logiciel. Cette entité logicielle est capable de percevoir son environnement et a un peu de connaissances et représentations sur les préférences des utilisateurs ».

D'après ces définitions on peut dire qu'un agent est une entité autonome, elle a un certain degré d'intelligence afin de développer ces connaissances en interagissant avec son

environnement pour la résolution des problèmes.

7.2.1 Les caractéristiques d'un agent

L'agent peut avoir quelques unes ou bien toutes les caractéristiques suivantes, selon le problème à confronter [SM06].

L'autonomie : cela signifie que pour exécuter ses différentes actions, un agent n'est pas dirigé par son environnement.

La situation : un agent est situé dans un environnement, il est capable de percevoir son environnement, et il peut aussi le modifier.

La flexibilité : un agent est capable de répondre à temps aux changements qui se produisent dans son environnement.

La pro-activité : un agent est capable d'avoir un comportement opportuniste, dirigé par ses buts ou sa fonction d'utilité, et de prendre des initiatives au moment approprié.

La sociabilité : un agent est capable d'interagir avec d'autres agents pour accomplir ses tâches ou aider les autres agents à accomplir leurs présentations.

La mobilité : la capacité de passer d'un environnement à un autre sans l'interruption de ses fonctions.

Le caractère : un agent peut avoir les caractéristiques des humains comme la personnalité et les états d'émotions.

L'apprentissage : un agent est capable d'apprendre de son environnement. Plus le processus d'apprentissage est efficace plus l'agent est intelligent.

La continuité : plus les fonctions d'un agent sont continues, plus l'agent est robuste.

L'adaptabilité : l'adaptation de son comportement à son environnement.

La coopération : l'agent peut être coopératif i.e il peut partager l'information avec d'autres agents.

7.2.2 Le comportement d'un agent

On peut représenter le comportement d'un agent selon ses trois tâches principales par le cycle de vie : Perception, Décision, Action.

L'agent perçoit les changements et les nouveaux faits de son environnement lors de la phase

de perception. Ensuite, et selon ses connaissances, ses expériences et son rôle, il choisit la ou les actions applicables sur son environnement.
Enfin, le cycle de vie de l'agent se termine en réalisant sur l'environnement les actions qu'il a sélectionné lors de la phase de décision.
Le comportement de l'agent est bien résumé dans la figure 7.1.

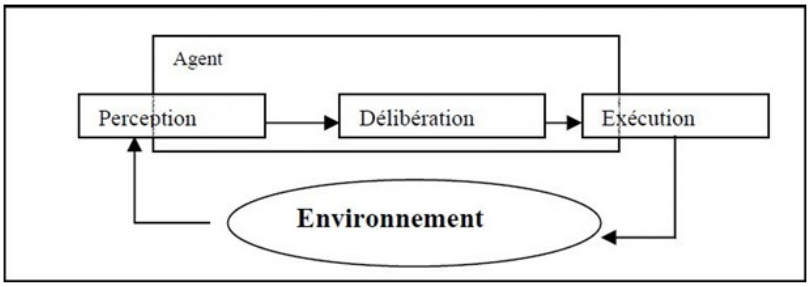

FIGURE 7.1 – *Cycle de vie : Perception, Décision, Action d'un agent[WJ95]*

7.3 Les types d'agents

Il existe plusieurs types d'agents selon leurs degré d'intelligence, ils se distinguent essentiellement sur deux grandes catégories :

7.3.1 Les agents réactifs

Les agents réactifs sont de plus bas niveau. Ils ne disposent que d'un protocole et d'un langage de communication réduit, leurs capacités répondent uniquement à la loi stimulus/réponse.

Les agents réactifs n'ont pas une capacité de raisonnement, ils reçoivent seulement les connaissances de leur environnement et ils les transfèrent [BB01]. La figure 7.2 représente l'architecture d'un agent réactif.

A noter ici que les agents réactifs ne sont pas considérés comme des agents intelligents. Par ailleurs, uniquement un système multi-agents réactifs est intelligent, grâce à la collectivité qui produit une certaine intelligence, comme par exemple le système des fourmis. Chaque fourmi a un comportement bien précis et elle ne peut pas trouver le plus court

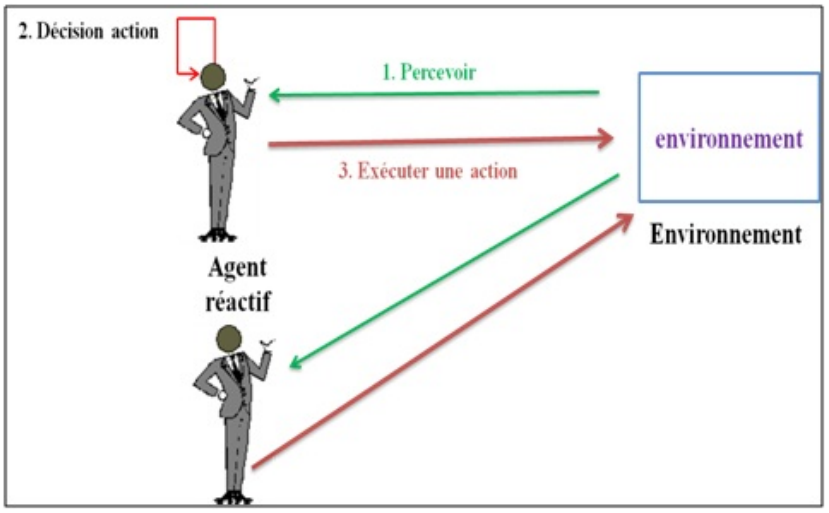

FIGURE 7.2 – *L'architecture d'un agent réactif*

chemin qui la dirige à la nourriture toute seule, mais grâce à la communication et le partage d'informations entre les fourmis, cette colonie est capable de trouver le plus court chemin qui mène à la nourriture (intelligence en essaim).

7.3.2 Les agents cognitifs

Un agent cognitif est un agent qui dispose d'une capacité de raisonnement sur une base de connaissances, d'une aptitude à traiter des informations diverses liées au domaine d'application et d'informations relatives à la gestion des interactions avec les autres agents et leur environnement.
Un agent cognitif peut tenir compte de son passé et ainsi anticiper sur l'avenir pour planifier ses actions. Il reçoit des connaissances de son environnement, il les développe à l'aide d'un système de développement de connaissances comme les systèmes experts, ou bien il reçoit seulement des données et il fait l'extraction de connaissances à partir d'un ensemble de données en utilisant soit l'apprentissage automatique soit la fouille de données, afin de produire de nouvelles connaissances ensuite les transmettre à son environnement. La figure 7.4 montre le fonctionnement d'un agent cognitif.

L'une des architectures les plus connues d'un agent cognitif est l'architecture "BDI" (Belief, Desire, Intentions) [BB01].

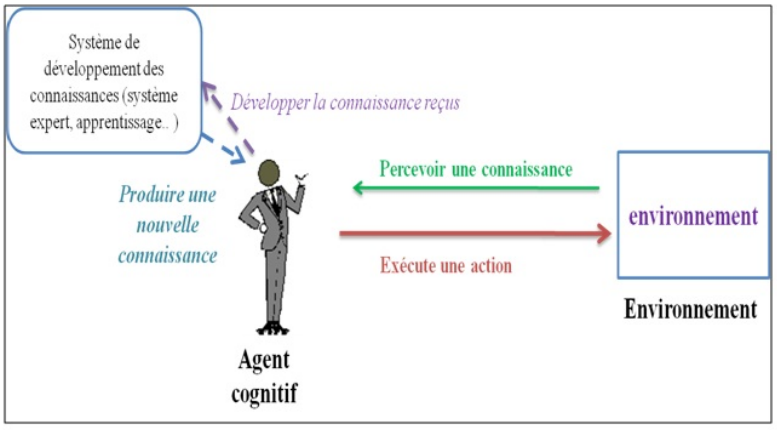

FIGURE 7.3 – *L'architecture d'un agent cognitif*

BDI = Beliefs (les croyances) + Desires (les désires) + Intentions(les intentions).

- La croyance : Un modèle symbolique de l'environnement de l'agent (connaissance de l'environnement). Un ensemble des croyances sur l'environnement et sur lui même.
  Par exemple : « Je crois que le bouton est appuyé » , « Je crois qu'il pleut ».
- Le désire : Une spécification symbolique que l'agent peut accomplir (sous forme de pré-condition-action-effet) représentant l'état de l'environnement avant que l'action soit posée ainsi que la conséquence de l'action.
  Par exemple : « Je voudrais qu'il fasse noir dans la pièce », « Je voudrais manger ».
  A noter ici qu'un ensemble de désirs peuvent être contradictoires.
  Par exemple : « Je voudrais qu'il fasse noir dans la pièce mais je veux lire le journal ».
- L'intention : Un algorithme de planification pouvant manipuler les symboles définis et un planning qui représente les actions qui doivent être réaliser par l'agent pour atteindre son but. Un ensemble d'intentions, buts, préférences (non-contradictoires).
  Par exemple : « J'ai l'intention d'appuyer sur le bouton », « J'ai l'intention de manger ».

On note ici que les systèmes multi-agents actuels ont une architecture hybride c'est-à-dire ils sont constitués des agents réactifs et cognitifs bien évidemment selon l'application désirée.

L'agent doit élaborer un plan d'actions en fonction des connaissances et des croyances dont il dispose, et des buts qu'il se fixe suite à une perception ou à une interaction avec le monde extérieur (l'environnement ou les autres agents). Pour cela, il doit décider du but à atteindre, planifier en fonction de ce but et passer à l'exécution. Le schéma présenté à la figure 7.4 montre en détails le fonctionnement d'un agent cognitif.

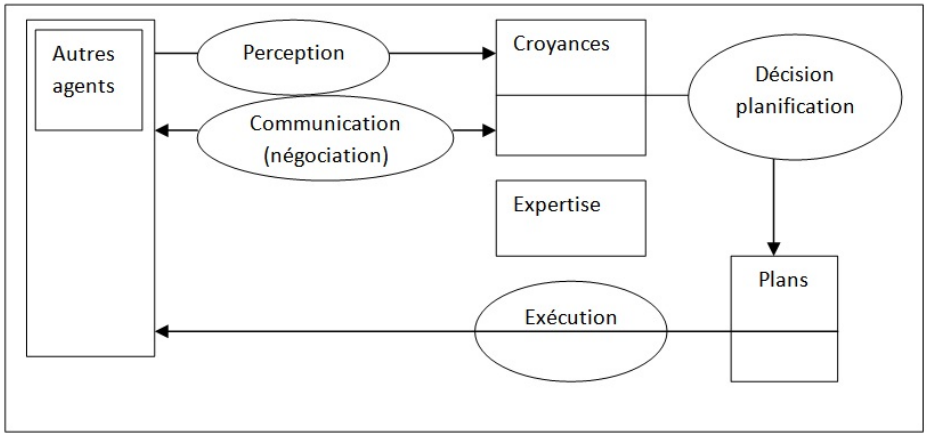

FIGURE 7.4 – *Le mecanisme de fonctionnement d'un agent cognitif*

7.3.3 La Comparaison entre agent réactif et agent cognitif

Le tableau 7.1 résume les principales différences entre les deux types d'agents :
Toutefois, pour profiter des avantages des deux modèles, à savoir l'architecture cognitive et réactive, les chercheurs ont mis en oeuvre des nouvelles architectures hybrides qui combinent les caractéristiques de ces deux architectures.
Cette découverte a permit par la suite de concevoir des systèmes hétérogènes comportant les deux types d'agents en même temps, et aussi d'avoir des agents cognitifs dotés de capacités de réactions aux événements.

Toutefois, pour avoir le meilleur des deux solutions, à savoir l'architecture cognitive et réactive, les chercheurs ont conçu des architectures hybrides qui combinent les caractéristiques de ces deux architectures. Il est possible de concevoir des systèmes hétérogènes comportant les deux types d'agents. Il est aussi possible de doter les agents cognitifs de

TABLE 7.1 – Différence entre l'agent cognitif et l'agent réactif[Tli07]

Agents cognitifs	**Agents réactifs**
- Représentation explicite de l'environnement.	- Pas de représentation de l'environnement.
- Peut tenir compte de son passé.	- Pas de mémoire de son historique.
- Connaissances individuelles	- Pas de connaissances individuelles.
- Agents complexes	- Fonctionnement Stimulus/Réponse
- Petit nombre d'agents suffit pour la résolution de problème.	- Grand nombre d'agents nécessaire pour la résolution de problème.

capacités de réactions aux événements : on parlera alors d'agents hybrides.

7.4 L'implémentation des agents

Il existe principalement deux approches pour implémenter les agents cognitifs : l'approche classique, et une nouvelle tendance d'implémentation, comme expliquer dans la figure 7.5.

7.4.1 L'approche classique

Dans l'approche classique on y trouve aussi deux types d'agents qui sont : les agents basés sur le systèmes experts, et les agents basés sur l'apprentissage automatique[SV00] [YYC00].

7.4.1.1 Les agents cognitifs basés sur les systèmes experts

Ce type d'agent développe ces connaissances en utilisant un système expert. Le système expert est une approche dominante de l'intelligence artificielle depuis les années 80 [BB01], il est capable de résoudre des problèmes en utilisant quelques connaissances d'un domaine, l'expérience de ce domaine est sauvegardé dans une base appelée base de connaissances.
Il y a des systèmes experts qui sont construits pour prendre la place de l'expert humain et d'autres qui sont construits pour aider les experts humains à prendre des décisions. Ainsi,

un système expert est un système déductif, c'est à dire à partir des connaissances à priori, on déduit des nouvelle connaissances.

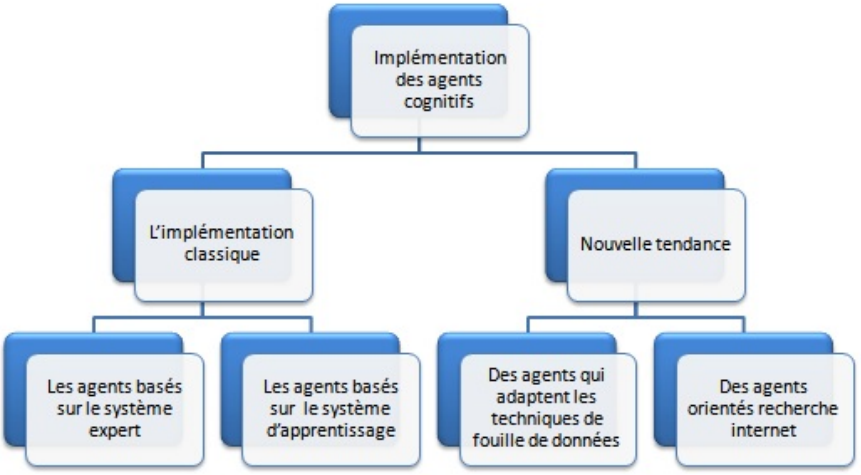

FIGURE 7.5 – *Les differentes implémentations des agents cognitifs*

Un système expert est constitué de deux parties :

- La partie représentation : cette partie consiste à représenter les connaissances à priori en un ensemble de faits et un ensemble de règles, le tout forme la base de connaissance de l'agent.

Chaque règle est composée comme suit : Si (condition) alors (action).

- La partie raisonnement : cette partie consiste à raisonner sur la base de connaissances pour déduire de nouvelles connaissances, on parle ici du moteur d'inférence.

Il existe principalement trois types de moteur d'inférence :

1. Chainage avant : on part de la base des faits en appliquant les règles en essayant de déduire la connaissance désirée.
2. Chainage arrière : on part de la connaissance désirée en appliquant les règles en essayant de déduire un fait parmi les faits qui existent dans la base des faits.
3. Chainage mixte : on combine les deux chainages en démarrant des faits et de la connaissance désirée de telle sorte qu'on obtienne un fait qui relie les deux derniers.

A noter aussi que l'idée de base d'un moteur d'inférence à chaînage avant est de déduire

tout ce qu'il est possible à partir d'un ensemble de faits initiaux et d'un ensemble de règles. A chaque fois qu'un nouveau fait est déduit, l'ensemble complet des règles (sauf celles déjà déclenchées) doit être ré-appliqué à la base des faits : le nouveau fait peut permettre le déclenchement d'une règle qui a été déjà essayée sans succès. Le processus d'inférence se termine lorsque plus aucun nouveau fait ne peut être déduit, ou que le but (s'il existe) est atteint.

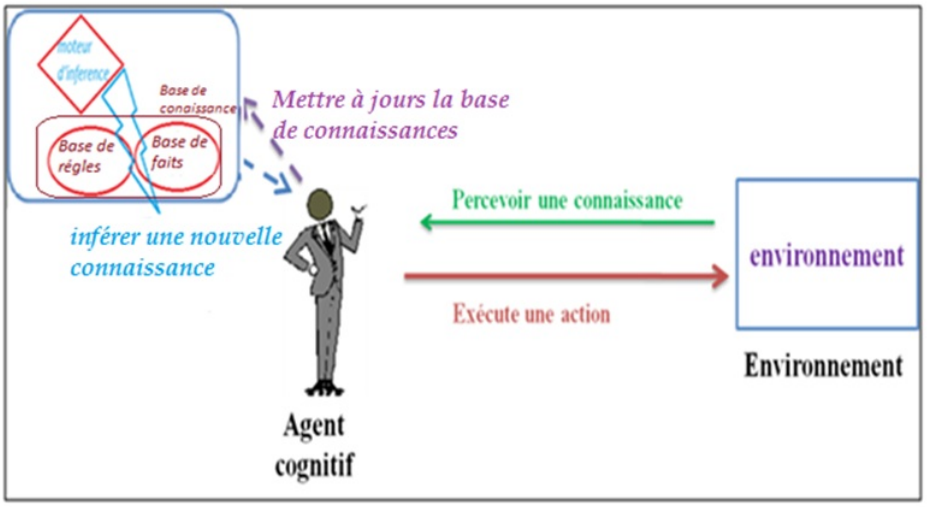

FIGURE 7.6 – *L'architecture d'un agent cognitif basé sur système expert*

La figure 7.6 montre un schéma explicatif de l'architecture d'un agent cognitif basé sur un système expert. A noter que pour ce type d'agents, les faits représentent les connaissances perçues par l'environnement, la connaissance désirée est la connaissance que l'agent veut obtenir pour la résolution de problème.

7.4.1.2 Les agents cognitifs basés sur le système d'apprentissage

Ce type d'agent développe ses connaissances en utilisant un système d'apprentissage. Un système d'apprentissage est un système inductif c'est-à-dire, des règles sont généralisées à partir d'un ensemble d'apprentissage. Un algorithme d'apprentissage est appliqué sur un ensemble d'apprentissage pour en apprendre des règles, qu'on utilisera pour déduire n'importe qu'elle connaissance.

Il existe plusieurs méthodes d'apprentissage tels que : les réseaux de neurones, les arbres de décisions,...etc.

7.4.2 La nouvelle tendance d'implémentation d'agents

7.4.2.1 Les agents fouilleurs de données

Ce type d'agents est plus général que les agents qui utilisent l'apprentissage automatique, de façon qu'à partir de données perçues de l'environnement, ces agents vont les explorer pour en extraire les connaissances voulues. Ici, la fouille de données intervient dans trois niveaux selon l'application désirée [SM06].

1- au niveau d'application : la fouille de données permet à l'agent l'extraction des connaissances à partir de l'ensemble de données perçues de son environnement.

2- au niveau du comportement : avec la fouille de données, on peut prédire le prochain comportement d'un agent.

3- au niveau d'évolution : en utilisant la fouille de données, la connaissance est extraite par l'agent, la diffusion de cette connaissance permet l'évolution des agents qui se situent dans l'environnement de l'agent précédent.

7.4.2.2 Les agents de recherche internet

Cette nouvelle tendance de concevoir des agents chercheurs sur internet fait partie de la technologie des agents mobiles est très développée et souvent utilisée par les grandes entreprises .
Des applications très intéressantes sont mises en oeuvre grâce à cette technologie dans plusieurs domaines tel que la recherche d'information, la recherche de web services, le domaine de la sécurité informatique...etc.

On note que dans la conception d'un agent cognitif, on peut combiner plusieurs architectures dans un seul agent, par exemple, on peut avoir un agent mobile et basé sur l'apprentissage automatique.

7.5 L'architecture de l'agent super intelligent

Notre contribution à la technologie des agents se situe au niveau du raisonnement des agents les plus connus c.à.d les agents basés sur un système expert. Plus précisément sur le mécanisme de chainage du moteur d'inférence.
En d'autres termes, on propose une extension de l'architecture d'un agent cognitif que l'on appelle l'Agent Super Intelligent (ASI). Ce dernier est implémenté principalement par l'intégration du module de fouille de règles d'induction dans l'architecture de base de l'agent cognitif. Cette extension est le résultat conséquent du besoin d'amélioration du mécanisme de raisonnement de l'agent cognitif classique.
On définit l'ASI comme un agent cognitif qui a la capacité de raisonner sur de très larges bases de connaissances et de l'auto-organisation, c.à.d qui est capable de fouiller ses règles d'induction pour en extraire des méta-règles par l'intégration du module de fouille de règles (présenté dans la deuxième partie de cette thèse).
l'ASI est plus intelligent que l'agent cognitif classique, de telle sorte que le premier intègre le module de fouille de règle d'induction afin de grouper ses règles en utilisant l'approche de segmentation multi-niveaux. Après, à chaque arrivée (ou découverte) d'une nouvelle connaissance, cette dernière est classifiée en l'affectant à son groupe correspondant en utilisant les méta-règles et en se basant sur les approches de segmentation incrémentale présentées précédemment. Tout cela pour inférer uniquement le cluster de la nouvelle règle ;i.e les règles qui sont susceptibles d'être inférées, au lieu de tester toutes les règles de la base de connaissances (une par une) de manière exhaustive, comme dans le cas de l'agent cognitif classique.

7.5.1 Les composants de l'agent super intelligent (ASI)

La figure 7.7 présente le mécanisme de fonctionnement de l'ASI. Les composants de l'architecture d'ASI peuvent être présentés comme suit :

1. Une très large base de connaissances, incluant les règles d'induction et les faits.
2. Un module de fouille multi-niveaux et incrémentale de règles d'induction, pour traiter le problème de la grande échelle de la base de règles et d'extraire des méta-règles.

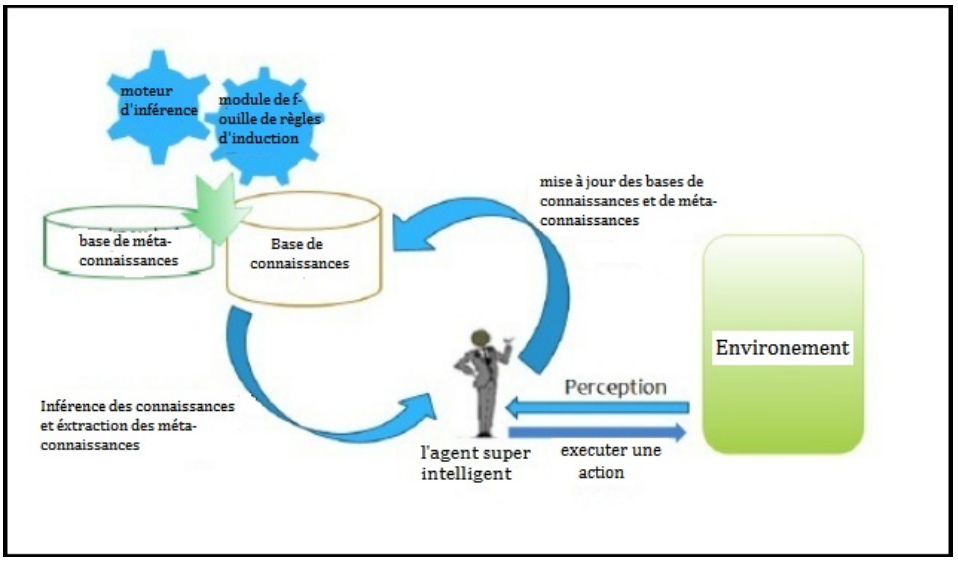

FIGURE 7.7 – *La proposition de l'architecture de l'agent super intelligent*

3. Une base de méta-règles, dans notre cas est représentée par les centres de gravité de chaque cluster.

4. Un moteur d'inférence, qui permet à l'agent le raisonnement dans la base de connaissance et de méta-connaissances.

5. Une interface, qui permet à l'agent la communication avec son environnement.

La base de règles La base de règles contient toutes les règles perçues et développées par l'agent. Elle contient des règles de la forme : "Si condition alors action", elle contient aussi les faits interprétés depuis l'environnement ou déduits par l'agent par le biais de son moteur d'inférence.

La base de méta-règles La base de méta-règles est constituée d'un ensemble de connaissances (appelées méta-connaissance) de taille remarquablement réduite par rapport à la taille de la base de connaissance. Ces méta-connaissances sont des connaissances sur les connaissance de la base. Dans notre cas, les méta-règles sont les centres de clusters existants dans la base de règles.

Le module de fouille (segmentation) de règles d'induction Le rôle principal de ce module est de maintenir les règles de la base de règles bien classifiées, de manière continue, à chaque arrivée d'une nouvelle connaissance, ou bien d'un nouveau fait. Ces derniers sont reformulés et traités en leur trouvant le bon groupe de règles qui sont en relation avec, afin de l'inférer, et donc accélérer le processus d'inférence. Ce module est constitué par les deux composants suivants :

1. L'outil de segmentation multi-niveaux : Cet outil permet de segmenter la base de règles de l'agent, et de communiquer les résultats à la base de méta-connaissances.
2. L'outil de segmentation incrémentale : Cet outil prend en charge les nouvelles connaissances inférées ainsi que les nouveaux faits, pour les attribuer au groupe de règles correspondant, et de faire appel au moteur d'inférence pour inférer uniquement ce groupe là.

Le moteur d'inférence Le moteur d'inférence applique un chainage avant sur la base de règles pour assurer le processus de raisonnement. Avant de lancer la phase d'inférence, l'ASI commence par localiser le fait courant (la nouvelle connaissance) en sélectionnant le bon cluster parmi ceux déjà existants.

Comme expliqué précédemment, la tâche de trouver la classe de règle de cette nouvelle connaissance (nouveau fait) est prise en charge par le module de fouille de règles. Premièrement, il commence par reformuler ce nouveau fait en règle d'induction sans la partie conséquence. Puis ce fait est classifié par une approche incrémentale de segmentation en se basant sur les groupes déjà existants et sur la base des méta-connaissances. Ensuite, uniquement le groupe concerné des règles est sélectionné afin de l'inférer et de trouver une réponse à la question en entrée.

Exemple : Considérant la petite base de connaissance de l'ASI qui est composée des règles suivantes :

- R1 : IF (temperature = 20) And (outlook = sunny) Then (practice sport = yes).
- R2 : IF (outlook = overcast) And (humidity = 80) Then (practice sport = yes).
- R3 : IF (temperature = hot) And (wind = light) Then (practice sport = no).

- R4 : IF (wind = light) And (outlook = rainy) Then (umbrella = yes).

- R5 : IF (engine = diesel) And (wheel = 4) Then (vehicle = yes).

- R6 : IF (wheel = 4) And (mark = audi) Then (vehicle = yes).

En appliquant l'outil de la segmentation multi-niveaux sur cette petite base de règles, on obtient des clusters comme suit :

- Cluster 1 = (R1, R2).

- Cluster 2 = (R3, R4).

- Cluster 3 = (R5, R6).

la base de méta-règles contient juste les centres de gravité de chaque cluster, à savoir :

- méta-règle 1 : R1.

- méta-règle 2 : R4.

- méta-règle 3 : R5.

Maintenant, on suppose que l'ASI veut déduire est ce qu'on pratiquera le sport aujourd'hui sachant que la température est 30 et l'humidité est de 13.
Ce nouveau fait est restructuré comme suit :

"new-fact" = if (temperature = 30) and (humidity = 13).

Pour répondre à cette question, l'ASI utilise l'outil de la segmentation incrémentale et en se basant sur les méta-règles il déduit la classe de "new-fact".
Après le calcul de distances, on trouve que la méta-règle 1 est la plus proche. De ce fait, "new-fact" est attribué au cluster 1 (celui de la méta-règle 1). Ensuite, l'inférence des règles appartenant au cluster1 est appliquée, donc on a uniquement R1 et R2 qui sont inférées, contrairement à l'agent classique qui consulte toute la base de règles.

7.6 Comparaison entre l'ASI et l'agent cognitif classique

Dans cette section, la performance de l'architecture proposée de l'agent super intelligent est comparée avec celle de l'agent cognitif classique. Quand l'agent cognitif classique infère toutes ces règles à chaque arrivée d'un nouveau fait par exemple avec un chainage avant,

l'ASI commence par l'application de la segmentation de ses règles, ensuite attribuer ce fait à la classe correspondante, et finalement inférer uniquement le cluster dont le nouveau fait appartient. Le temps d'exécution de ces deux processus est calculé et comparé dans la figure 7.8.

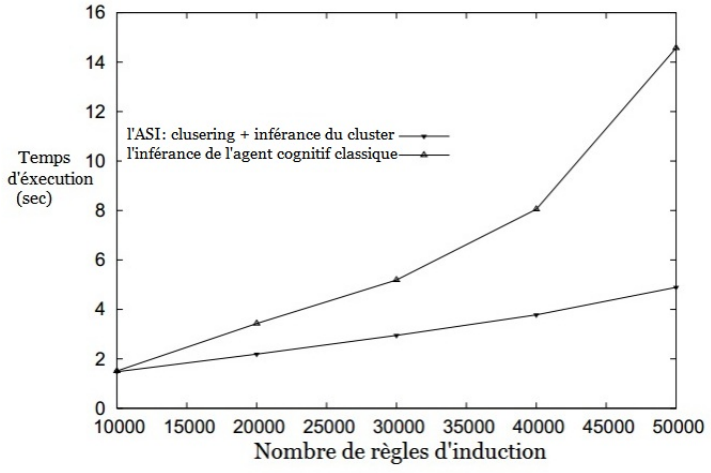

FIGURE 7.8 – *La comparaison des temps d'exécution de l'ASI et de l'agent cognitif classique*

Lors de l'augmentation du nombre de règles de la base de règle des agents, le temps d'exécution des deux types d'agents est calculé. Dans un coté, la somme des temps d'exécution des trois taches qui sont : la segmentation des règles (approche multi-niveaux)+ l'affectation à la classe correspondante (approche incrémentale) + inférence du cluster correspondant (chainage avant) pour l'ASI, et de l'autre coté l'inférence de toute la base de règles avec un chainage avant pour l'agent cognitif classique.
On remarque bien sur la figure 7.8 qu'il n'y a pas une grande différence entre les deux temps de raisonnement des deux types d'agents quand le nombre de règles à traiter est inférieur à 15000 règles. On explique ce résultat par le fait que l'ASI prend un temps d'exécution additionnel par rapport à l'agent classique et qui est le temps de segmentation + le temps de l'affectation. Cependant, lors du traitement des grandes base de règles, on remarque que l'ASI est beaucoup plus rapide que l'agent cognitif classique. Lors du traitement d'une base de règles qui varie entre 10,000 et 50,000 règles, l'exécution de l'agent cognitif classique augmente de 1,51 secondes à 14,57 secondes, contrairement à la somme des temps

d'exécution des trois tâches réalisées par l'ASI qui augmente juste de 1,48 à 4,89 secondes.

7.7 Conclusion

Dans ce chapitre, nous avons présenté un aperçu sur la technologie des agents, suivi par des définitions des différents types d'agents et la façon de les implémenter. Par la suite, nous avons montré la modélisation de la nouvelle architecture proposée appelée ASI. Cette dernière est plus performante que l'architecture de l'agent cognitif classique comme montré dans la section d'expérimentation. Cette architecture a aussi un avantage d'auto organiser l'agent et de lui permettre d'avoir des méta-connaissances, qui sont des connaissances sur ses connaissances.

Conclusion

Lors de cette étude, nous avons développé un nouveau concept et son application qui est la fouille de connaissances (Knowledge Mining). Pour cela, nous avons étudié quelques techniques de fouille de données, puis le passage de données aux connaissances, d'où la possibilité d'adapter les techniques de fouille de données pour la fouille de connaissances.
La complexité de l'approche de fouille de connaissances nous a conduit à fouiller des connaissances qui ont une même représentation, à savoir les connaissances de type règles d'induction.
La plupart des techniques de fouille de données sont basées sur la notion de similarité entre ces dernières. Par conséquent nous avons proposé une mesure de similarité entre les règles ainsi que de nouvelles formules de calcul des centres de gravité pour la fouille de connaissances. De plus, pour pouvoir fouiller les nouvelles connaissances acquises avec les connaissances déjà existantes, sans relancer le processus de fouille, et afin d'augmenter la performance du système lorsqu'on fouille une base de connaissances volumineuse, nous avons proposé des approches de fouille basées sur les paradigmes multi-niveaux et incrémentals.
A la fin nous avons réalisé une étude expérimentale et comparative de la performance des différentes approches de fouille, pour choisir la meilleure, afin de l'intégrer sur une application pratique qui est la technologie des agents intelligents.
L'analyse de l'application du module de fouille de règles d'induction sur la technologie des agent a aboutie à la création d'une nouvelle architecture d'agents, qui sont plus intelligents et mieux organisés que les agents cognitifs classiques. Cette nouvelle architecture est appelée ASI (Agent Super Intelligent).
Comme futures perspectives, nous prévoyons l'implémentation d'une nouvelle plateforme

des agents super-intelligents que l'on appellera méta-jade. Cette dernière va inclure le module méta-Jess qui permet d'une part d'accélérer le processus de découverte de nouvelles connaissances et d'autre part d'améliorer la performance du moteur d'inférence tout en ajoutant le module de fouille de connaissances.

Bibliographie

[AD13] Ms RR Ade and Dr PR Deshmukh. Methods for incremental learning : A survey. *International Journal of Data Mining & Knowledge Management Process*, 3(4), 2013. 55

[Bar02] Daniel Barbará. Requirements for clustering data streams. *ACM SIGKDD Explorations Newsletter*, 3(2) :23–27, 2002. 55

[BB01] Joseph P Bigus and Jennifer Bigus. Constructing intelligent agents using java. 2001. 114, 115, 118

[BC00] Daniel Barbará and Ping Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 260–264. ACM, 2000. 55

[BJ78] T Bailey and AK Jain. A note on distance-weighted k-nearest neighbor rules. *IEEE Transactions on Systems, Man, and Cybernetics*, (4) :311–313, 1978. 34

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD Record*, volume 26, pages 255–264. ACM, 1997. 39

[Bou12] Noureddine Bouhmala. A multilevel memetic algorithm for large sat-encoded problems. *Evol. Comput.*, 20(4) :641–664, December 2012. 59

[BS94] Stephen T. Barnard and Horst D. Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency : Practice and Experience*, 6(2) :101–117, 1994. 58

[CCFM97a] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 626–635. ACM, 1997. 54

[CCFM97b] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 626–635. ACM, 1997. 57

[CD13] Amine Chemchem and Habiba Drias. Multilevel clustering on very large scale of web data. In *Management Intelligent Systems*, pages 9–16. Springer International Publishing, 2013. 71

[CD14] Amine Chemchem and Habiba Drias. From data mining to knowledge mining : Application to intelligent agents. *Expert Systems with Applications*, 2014. 70

[CDD13a] A Chemchem, Y Djenouri, and H Drias. Incremental induction rules clustering. In *Systems, Signal Processing and their Applications (WoSSPA), 2013 8th International Workshop on*, pages 492–497. IEEE, 2013. 70, 100

[CDD13b] Amine Chemchem, Habiba Drias, and Youcef Djenouri. Multilevel clustering of induction rules for web meta-knowledge. In *Advances in Information Systems and Technologies*, pages 43–54. Springer, 2013. 83

[Che84] Philip E Cheng. Strong consistency of nearest neighbor regression function estimators. *Journal of Multivariate Analysis*, 15(1) :63–72, 1984. 34

[CHNW96] David W Cheung, Jiawei Han, Vincent T Ng, and CY Wong. Maintenance of discovered association rules in large databases : An incremental updating technique. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, pages 106–114. IEEE, 1996. 54

[CHO02] Chien-Yu Chen, Shien-Ching Hwang, and Yen-Jen Oyang. An incremental hierarchical data clustering algorithm based on gravity theory. In *Advances in Knowledge Discovery and Data Mining*, pages 237–250. Springer, 2002. 57

[Cou08] Adrien Coulet. *Construction et utilisation d'une base de connaissances pharmacogénomique pour l'intégration de données et la découverte de connaissances.* PhD thesis, Université Henri Poincaré-Nancy I, 2008. 15, 42, 43

[CW94] Stefano Ceri and Jennifer Widom. Deriving incremental production rules for deductive data. *Information Systems*, 19(6) :467–490, 1994. 50

[DA07] S Deelers and S Auwatanamongkol. Enhancing k-means algorithm with initial cluster centers derived from data partitioning along the data axis with the highest variance. *Proceedings of World Academy of Science : Engineering & Technology*, 36, 2007. 56

[DAB12] Habiba Drias, Asma Aouichat, and Aicha Boutorh. Towards incremental knowledge warehousing and mining. In *Distributed Computing and Artificial Intelligence*, volume 151 of *Advances in Intelligent and Soft Computing*, pages 501–510. Springer Berlin Heidelberg, 2012. 65, 71, 77

[DBS77] Randall Davis, Bruce Buchanan, and Edward Shortliffe. Production rules as a representation for a knowledge-based consultation program. *Artificial intelligence*, 8(1) :15–45, 1977. 50

[DD13] Marwa Djeffal and Habiba Drias. Multilevel bee swarm optimization for large satisfiability problem instances. In *Intelligent Data Engineering and Automated Learning–IDEAL 2013*, pages 594–602. Springer, 2013. 59

[Dev00] Keith J Devlin. Infosense : Turning information into knowledge, henry holt and co. *Inc., New York, NY*, 2000. 42

[DGK07] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors : A multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell*, 29 :2007, 2007. 59

[DHD13] Habiba Drias, Celia Hireche, and Ameur Douib. Datamining techniques and swarm intelligence for problem solving : Application to sat. In *Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on*, pages 200–206. IEEE, 2013. 93

[Dhi05] Inderjit Dhillon. A fast kernel-based multilevel algorithm for graph clustering. In *In Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 629–634, 2005. 59

[Die98] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7) :1895–1923, 1998. 31

[DLLL12] Dongsheng Duan, Yuhua Li, Ruixuan Li, and Zhengding Lu. Incremental k-clique clustering in dynamic social networks. *Artificial Intelligence Review*, 38(2) :129–147, 2012. 99

[DM94] Nikos Drakos and H Mann. Computer based learning unit, university of leeds. *Parallel Computing Works*, 1994. 50

[Dud76] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*, (4) :325–327, 1976. 34

[DWK05] Dursun Delen, Glenn Walker, and Amit Kadam. Predicting breast cancer survivability : a comparison of three data mining methods. *Artificial intelligence in medicine*, 34(2) :113–127, 2005. 32

[EKS+98a] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB*, volume 98, pages 323–333, 1998. 54

[EKS+98b] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Michael Wimmer, and Xiaowei Xu. Incremental clustering for mining in a data warehousing environment. In *VLDB*, volume 98, pages 323–333, 1998. 57

[EL09] Christie I Ezeife and Yi Liu. Fast incremental mining of web sequential patterns with plwap tree. *Data mining and knowledge discovery*, 19(3) :376–416, 2009. 99

[Fay96] Usama M Fayyad. Data mining and knowledge discovery : Making sense out of data. *IEEE Intelligent Systems*, 11(5) :20–25, 1996. 23

[FH75] Keinosuke Fukunaga and L Hostetler. K-nearest-neighbor bayes-risk estimation. *Information Theory, IEEE Transactions on*, 21(3) :285–293, 1975. 34

[Fis87] Douglas H Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2) :139–172, 1987. 54

[FSS+09] AM Fahim, G Saake, AM Salem, FA Torkey, and MA Ramadan. K-means for spherical clusters with large variance in sizes. *International Journal of Computer Science*, 4(3), 2009. 56

[GGV+11] Navneet Goyal, Poonam Goyal, K Venkatramaiah, PC Deepak, and PS SANNOP. An efficient density based incremental clustering algorithm in data warehousing environment. In *2009 International Conference on Computer Engineering and Applications, IPCSIT*, volume 2, 2011. 54

[GLF89] John H Gennari, Pat Langley, and Doug Fisher. Models of incremental concept formation. *Artificial intelligence*, 40(1) :11–61, 1989. 54

[GZ04] Zhiqiang Geng and Qunxiong Zhu. Incremental rules mining for information compression matrix algorithm. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 5, pages 4309–4313. IEEE, 2004. 54

[Har75] John A Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., 1975. 36

[HH08] Chung-Chian Hsu and Yan-Ping Huang. Incremental clustering of mixed data based on distance hierarchy. *Expert Systems with Applications*, 35(3) :1177–1185, 2008. 57

[HIL68] CG HILBORN. Dg lainiotis. *IEEE Transactions on Information theory*, 1968. 34

[HK03] Khaled M Hammouda and Mohamed S Kamel. Incremental document clustering using cluster similarity histograms. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 597–601. IEEE, 2003. 57

[HKP06] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining : concepts and techniques*. Morgan kaufmann, 2006. 15, 26, 30, 31, 83

[HL95] Bruce Hendrickson and Robert W Leland. A multi-level algorithm for partitioning graphs. *SC*, 95 :28, 1995. 59

[HW79] John A Hartigan and Manchek A Wong. Algorithm as 136 : A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979. 36

[JC10] Oriol J-C. *Une approche historique de la statistique*. 2010. 24

[JK12] Prachi Joshi and Parag Kulkarni. Incremental learning : Areas and methods-a survey. *International Journal of Data Mining & Knowledge Management Process*, 2(5), 2012. 15, 55, 56

[JLT03] P Jenny, SH Lee, and HA Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *Journal of Computational Physics*, 187(1) :47–67, 2003. 53

[Jóź83] Adam Jóźwik. A learning scheme for a fuzzy k-nn rule. *Pattern Recognition Letters*, 1(5) :287–289, 1983. 34

[Kan11] Mehmed Kantardzic. *Data mining : concepts, models, methods, and algorithms*. John Wiley & Sons, 2011. 24, 27, 32, 63

[KJ98] Michael Knapik and Jay Johnson. *Developing intelligent agents for distributed systems : exploring architecture, technologies, & applications*. McGraw-Hill, Inc., 1998. 111

[KK98] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1) :359–392, 1998. 15, 59, 60

[Kod96] Y Kodratoff. L'extraction de connaissances à partir des données : Un nouveau sujet pour la recherche scientifique. In *INFORSID. Congrès*, pages 3–25, 1996. 26

[LA08] Tao Li and Sarabjot S Anand. Hirel : An incremental clustering algorithm for relational datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 887–892. IEEE, 2008. 57

[Lei09] Paulo Leitão. Agent-based distributed manufacturing control : A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7) :979–991, 2009. 111

[LHC04] Yangguang Liu, Qinming He, and Qi Chen. Incremental batch learning with support vector machines. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 2, pages 1857–1861. IEEE, 2004. 54

[LLC01] Chang-Hung Lee, Cheng-Ru Lin, and Ming-Syan Chen. Sliding-window filtering : an efficient algorithm for incremental mining. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 263–270. ACM, 2001. 54

[LLH12] Chun-Wei Lin, Guo-Cheng Lan, and Tzung-Pei Hong. An incremental mining algorithm for high utility itemsets. *Expert Systems with Applications*, 39(8) :7173–7180, 2012. 99

[LLKH12] Piyuan Lin, Zijian Lin, Bingqia Kuang, and Peijie Huang. A short chinese text incremental clustering algorithm based on weighted semantics and naive bayes. *Journal of Computational Information Systems*, 8(10) :4257–4268, 2012. 57

[LVD+00] Peter Lyman, Hal R Varian, J Dunn, A Strygin, and K Searingen. How much information ? 2000, 2000. 26

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967. 36

[MM98] Christopher J Merz and Patrick M Murphy. {UCI} repository of machine learning databases. 1998. 76

[MR05] Oded Z Maimon and Lior Rokach. *Data mining and knowledge discovery handbook*, volume 1. Springer, 2005. 15, 26, 28

[Mue98] Andreas Mueller. Fast sequential and parallel algorithms for association rule mining : A comparison. 1998. 39

[NK98] Ikujiro Nonaka and Noboru Konno. The concept of" ba" : Building a foundation for knowledge creation. *California management review*, 40(3), 1998. 44

[NT97] Ikujiro Nonaka and H Takeuchi. The knowledge-creating company. *1995*, 1997. 44

[NXC+07] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S Huang. Incremental spectral clustering with application to monitoring of evolving blog communities. In *SDM*, pages 261–272. SIAM, 2007. 53

[OLW08] Sigurdur Olafsson, Xiaonan Li, and Shuning Wu. Operations research and data mining. *European Journal of Operational Research*, 187(3) :1429–1448, 2008. 38

[PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S Yu. *An effective hash-based algorithm for mining association rules*, volume 24. ACM, 1995. 38

[PJ05] Nikhil R Pal and Lakhmi Jain. *Advanced techniques in data mining and knowledge discovery*. Springer, 2005. 25

[PS00] Gregory Piatetsky-Shapiro. Knowledge discovery in databases : 10 years after. *ACM SIGKDD Explorations Newsletter*, 1(2) :59–61, 2000. 25

[PW05] Lin Padgham and Michael Winikoff. *Developing intelligent agent systems : A practical guide*, volume 13. John Wiley & Sons, 2005. 112

[Rak03] Ricco Rakotomalala. Tanagra. *TANAGRA : a free software for research and academic purposes*, 2 :697–702, 2003. 24

[RNC+95] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence : a modern approach*, volume 2. Prentice hall Englewood Cliffs, 1995. 112

[SAA+99] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Shadbolt, Walter Van de Velde, and Bob Wielinga. Knowledge engineering and management : the commonkads methodology. a bradford book, 1999. 42

[SBM07] Thabet Slimani, B BenYaghlane, and Khaled Mellouli. Une extension de mesure de similarité entre les concepts dâune ontologie. *the Proceedings of SETIT*, pages 1–10, 2007. 65

[Sen95] Jacqueline Senker. Networks and tacit knowledge in innovation. *Economies et societes*, 29(9) :99–118, 1995. 45

[Sho93] Yoav Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1) :51–92, 1993. 112

[Sim05] Gilda Simoni. *Capitaliser les connaissances générées dans les projets de R&D : pour un leadership intégratif et situationnel.* PhD thesis, Aix Marseille 2, 2005. 42

[SLWQ09] Xiaoke Su, Yang Lan, Renxia Wan, and Yuming Qin. A fast incremental clustering algorithm. In *International Symposium on Information Processing*, pages 175–178, 2009. 57

[SM06] Andreas L Symeonidis and Pericles A Mitkas. *Agent intelligence through data mining*, volume 14. Springer, 2006. 113, 121

[SON95] Ashok Savasere, Edward Robert Omiecinski, and Shamkant B Navathe. An efficient algorithm for mining association rules in large databases. 1995. 39

[SS11] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In Camil Demetrescu and MagnusM. Halldorsson, editors, *Algorithms â ESA 2011*, volume 6942 of *Lecture Notes in Computer Science*, pages 469–480. Springer Berlin Heidelberg, 2011. 59

[SV00] Peter Stone and Manuela Veloso. Multiagent systems : A survey from a machine learning perspective. *Autonomous Robots*, 8(3) :345–383, 2000. 118

[SWW98] Gerd Stumme, Rudolf Wille, and Uta Wille. *Conceptual knowledge discovery in databases using formal concept analysis methods.* Springer, 1998. 44

[TD08] Makio Tamura and Patrik D'haeseleer. Microbial genotype–phenotype mapping by class association rule mining. *Bioinformatics*, 24(13) :1523–1529, 2008. 38

[Ten99] Shang-Hua Teng. Coarsening, sampling, and smoothing : Elements of the multilevel method. In MichaelT. Heath, Abhiram Ranade, and RobertS. Schreiber, editors, *Algorithms for Parallel Processing*, volume 105 of *The IMA Volumes in Mathematics and its Applications*, pages 247–276. Springer New York, 1999. 58

[Tit12] Mawloud Titah. *Externalisation des connaissances tacites en connaissances explicites : cas diagnostic industriel.* PhD thesis, Université El Hadj Lakhdar de Batna, 2012. 44

[Tli07] Ahmed Tlili. *Etude et réalisation d'une plate forme multi-agents.* PhD thesis, Université El Hadj Lakhdar de Batna, 2007. 13, 118

[Tri69] Myron Tribus. *RATIONAL DESCRIPTIONS, DECISIONS, AND DESIGNS..* 1969. 25

[Tuf10] Stéphane Tufféry. *Titre : Data mining et statistique décisionnelle. L'intelligence des données. Format : 17 cm x 24 cm, 725 p. Bibliogr. p. 689-699.* Editions Technip Paris, 2010. 24, 25

[Tuo99] Ilkka Tuomi. Data is more than knowledge : implications of the reversed knowledge hierarchy for knowledge management and organizational memory. In *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*, pages 12–pp. IEEE, 1999. 15, 42, 43, 46, 47

[Š11] K. Tashkova ; P. Koroec ; J. Šilc. A distributed multilevel ant colony algorithm for the multiway graph partitioning. *Int. J. Bio-Inspired Comput.*, 3(5) :286–296, September 2011. 59

[Wal08] Chris Walshaw. Multilevel refinement for combinatorial optimisation : Boosting metaheuristic performance. In Christian Blum, MariaJose Blesa Aguilera, Andrea Roli, and Michael Sampels, editors, *Hybrid Metaheuristics*, volume 114 of *Studies in Computational Intelligence*, pages 261–289. Springer Berlin Heidelberg, 2008. 58

[WF02] Wai-chiu Wong and AdaWai-chee Fu. Incremental document clustering for web page classification. In Qun Jin, Jie Li, Nan Zhang, Jingde Cheng, Clement Yu, and Shoichi Noguchi, editors, *Enabling Society with Information Technology*, pages 101–110. Springer Japan, 2002. 55

[Wil02] Rudolf Wille. Why can concept lattices support knowledge discovery in databases ? *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3) :81–92, 2002. 42

[WJ95] Michael Wooldridge and Nicholas R Jennings. Intelligent agents : Theory and practice. *The knowledge engineering review*, 10(02) :115–152, 1995. 16, 112, 114

[WJ99] Michael J Wooldridge and Nicholas R Jennings. Software engineering with agents : Pitfalls and pratfalls. *Internet Computing, IEEE*, 3(3) :20–27, 1999. 112

[WJ04] Zeng Wenhua and Ma Jian. A novel incremental svm learning algorithm. In *Computer Supported Cooperative Work in Design, 2004. Proceedings. The 8th International Conference on*, volume 1, pages 658–662. IEEE, 2004. 54

[WJK00] Michael Wooldridge, Nicholas R Jennings, and David Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3) :285–312, 2000. 112

[WKQ+08] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1) :1–37, 2008. 32, 33, 38

[Wu08] Fang-Xiang Wu. Genetic weighted k-means algorithm for clustering large-scale gene expression data. *BMC bioinformatics*, 9(Suppl 6) :S12, 2008. 83

[WWT99] Pak Chung Wong, Paul Whitney, and Jim Thomas. Visualizing association rules for text mining. In *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on*, pages 120–123. IEEE, 1999. 38

[Wya01] Jeremy C Wyatt. Management of explicit and tacit knowledge. *Journal of the Royal Society of Medicine*, 94(1) :6, 2001. 44

[XWZ00] Rong Xiao, Jicheng Wang, and Fayan Zhang. An approach to incremental svm learning algorithm. In *Tools with Artificial Intelligence, 2000. ICTAI 2000. Proceedings. 12th IEEE International Conference on*, pages 268–273. IEEE, 2000. 54

[YAKR10] Steven Young, Itamar Arel, Thomas P Karnowski, and Derek Rose. A fast and stable incremental clustering algorithm. In *Information Technology : New Generations (ITNG), 2010 Seventh International Conference on*, pages 204–209. IEEE, 2010. 57

[YLC+08] Ding Yuan, Kyuhyung Lee, Hong Cheng, Gopal Krishna, Zhenmin Li, Xiao Ma, Yuanyuan Zhou, and Jiawei Han. Cispan : Comprehensive incremental mining algorithms of closed sequential patterns for multi-versional software mining. In *SDM*, volume 8, pages 84–95. SIAM, 2008. 99

[YYC00] Christopher C Yang, Jerome Yen, and Hsinchun Chen. Intelligent internet searching agent based on hybrid simulated annealing. *Decision Support Systems*, 28(3) :269–277, 2000. 118

[ZCY10] Yang Zhou, Hong Cheng, and J.X. Yu. Clustering large attributed graphs :

An efficient incremental approach. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 689–698, Dec 2010. 55

[ZMH09] Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-means clustering based on mapreduce. In *Cloud Computing*, pages 674–679. Springer, 2009. 83

[ZR02] Djamel Abdelkader Zighed and Ricco Rakotomalala. Extraction de connaissances à partir de données (ecd). *Techniques de l'Ingénieur, volume HA*, 2002. 15, 27

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch : an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM, 1996. 56

[ZS+04] Haïfa Zargayouna, Sylvie Salotti, et al. Mesure de similarité dans une ontologie pour l'indexation sémantique de documents xml. *IC 2004*, pages 249–260, 2004. 64

Chemchem Amine

De la fouille de données vers la fouille de connaissances : Apprentissage incrémental et approches multi-niveaux

Résumé :
L'ère actuelle est caractérisée par de gigantesques volumes de connaissances qui sont extraites de façon permanente du Web grâce à la robustesse d'outils intelligents, et plus précisément aux techniques de la fouille de données. Pour accélérer le processus d'extraction de méta-connaissances de façon efficace, nous serons confrontés à deux problèmes majeurs, à savoir :
Comment fouiller cette masse faramineuse de connaissances de façon rapide et continue.
Comment gérer le flux de connaissances produit constamment dans le Web sans avoir à redémarrer le processus de fouille à chaque arrivée de nouvelles connaissances.
Pour résoudre le premier problème nous proposons deux approches de fouille de connaissances multi-niveaux, après avoir présenté tout un concept théorique pour l'extension de la fouille de données à celle des connaissances. Pour le second problème, nous avons développé trois approches de fouille de connaissances incrémentale, la première est l'extension d'algorithme de fouille de données incrémentale, et les deux autres sont des approches de traitement de paquets de connaissances d'une manière progressive. Par la suite, une étude expérimentale est élaborée sur ces différentes approches afin de les comparer et d'en extraire les plus performantes en matière de temps de calcul, et du taux de réussite de la segmentation.
Une application fort intéressante consiste à projeter les résultats de ces travaux sur la technologie des agents cognitifs et de l'étendre au concept d'agents super intelligents. Dans ce contexte il s'agit d'agents sensés recevoir des quantités faramineuses de connaissances en une courte période pendant laquelle il leur est impossible de les exploiter. La solution apportée est d'organiser ses connaissances dans des clusters en exploitant les approches de fouille proposées, afin de n'utiliser que les connaissances concernées dans le but d'engendrer une nouvelle connaissance.

Mots clés :
Fouille de données - extraction de connaissances à partir de bases de données - représentation des connaissances - règle d'induction - fouille de connaissances - paradigme incrémental - approches multi-niveaux - algorithmes génétiques.