# Recherche de motifs graduels et application aux données médicales

Lisa Di Jorio

**U**niversité **M**ontpellier **II**
— **S**ciences et **T**echniques du **L**anguedoc —

# **T**hèse

pour obtenir le grade de
Docteur de l'Université Montpellier II

Discipline : Informatique
*Spécialité Doctorale* : *Informatique*
*Ecole Doctorale* : *Information, Structure, Systèmes*

présentée et soutenue publiquement par

Lisa Di Jorio
le 05 Octobre 2010

# Recherche de motifs graduels et application aux données médicales

Jury

Marianne Huchard, Professeur, LIRMM – Université Montpellier 2, ..... Présidente
Amedeo Napoli, Directeur de Recherche CNRS, LORIA Nancy, ..... Rapporteur
Sylvie Galichet, Professeur, LISTIC – Université de Savoie, ..... Rapporteur
Maria Rifqi, Maître de Conférence, LIP6 – Université Pierre et Marie Curie, ..... Examinateur
Sophie Martin, Maître de Conférence, LAMECO – Université Montpellier 3, ..... Examinateur
Béatrice Orsetti, Ingénieur de Recherche, IRCM, ..... Invitée
Maguelonne Teisseire, Directrice de recherche, CEMAGREF, ..... Co-Directrice de thèse
Anne Laurent, Maître de Conférence, LIRMM – Université Montpellier 2, ..... Co-Directrice de thèse

# Remerciements

La réalisation d'un travail de thèse n'est pas une tâche aisée. C'est le résultat de différentes collaborations, de nombreuses discussions, ainsi que d'un aller-retour permanent entre enthousiasme et scepticisme. Je n'aurais jamais pu réaliser ce travail sans l'aide de certaines personnes, que je tiens à remercier ici.

Tout d'abord, je souhaite remercier Maguelonne Teisseire pour m'avoir donné l'opportunité de réaliser cette thèse. Je tiens à remercier Anne Laurent, qui a dirigé cette thèse conjointement à Maguelonne Teisseire, pour les différentes directions qu'elle a su donner à ce travail. Ces derniers mois ont été pour moi très épanouissants et même productifs, et elle en est l'actrice principale.

J'adresse également tous mes remerciements à Amedeo Napoli et Sylvie Galichet qui ont accepté de rapporter cette thèse, et qui ont passé un temps conséquent à disséquer et critiquer mes propositions. De par leur retour enrichissant, ils ont contribué à l'amélioration de ce manuscrit.

Cette thèse s'inscrit dans un contexte médical. Afin de valider ce travail, j'ai eu la chance de travailler avec des experts non informaticiens. Béatrice Osetti, de l'IRCM, a su m'expliquer avec patience tous les concepts de biologie intervenant dans ses recherches, ainsi que pour le temps qu'elle m'a consacré ces trois dernières années. J'ai également travaillé avec Sophie Martin, du laboratoire LAMECO, qui a également pris le temps de m'expliquer les concepts généraux des tests psychologiques composant ses bases de données. La disponibilité de ces deux personnes a fortement contribué aux expérimentations présentées dans ce mémoire.

De plus, ces deux expertes ont accepté de faire partie de mon jury lors de ma soutenance. Elles ont consacré un temps précieux à cette soutenance, et je les remercie profondement d'avoir accepté ce rôle, d'autant plus qu'elles n'étaient pas dans leur domaine.

Je tiens également à remercier Marianne Huchard pour avoir présidé mon jury de thèse, ainsi que pour son écoute lors des moments difficiles.

Merci à tous les thésard passés qui ont partagé leur expérience avec moi : Céline Fiot, Marc Plantevit, Chedy Raissi et Nadia El M'rabet. Merci à mes amis pour tous les moments de rire : Delphine, Philippe, Julien, Hassan, Guillaume, Madalina, Didier, Valek, Xavier, Gilles, Zeina, Arnaud, Doomy, Paul, Marie, Christel et encore bien d'autres...

Paola et Cécile, pour tous les moments privilégiés méritent un remerciement particulier.

J'adresse un remerciement à toutes les personnes qui m'ont entourée à Polytech'Montpellier, où j'ai effectué mon monitorat : Alain, Claudine, Isabelle, Hervé et Tiberiu.

Merci à ma famille pour leur indéfectible soutien : ma mère, mon père, mon frère et ma sœur.

Enfin, mes derniers remerciements vont à mon amour, qui se reconnaîtra.

*A mes parents*

*"Toute certitude est par essence contradictoire avec la philosophie de la recherche"*
Pierre Juliot – La recherche passionnément

# Sommaire

# Chapitre 1

# Introduction

## 1.1 Motivations

Avec le développement des nouvelles technologies d'analyse (comme par exemple les puces à ADN) et d'informations (augmentation des capacités de stockage), le domaine de la santé a particulièrement évolué ces dernières années. En effet, des techniques de plus en plus poussées et efficaces sont mises à disposition des chercheurs, et permettent une étude approfondie des paramètres génomiques intervenant dans des problèmes de santé divers (cancer, maladie d'Alzheimer ...) ainsi que leur mise en relation avec les paramètres cliniques. Parallèlement, l'évolution des capacités de stockage permet désormais d'accumuler la masse d'information générée par les diverses expériences menées. Ainsi, les avancées en terme de médecine et de prévention passent par l'analyse complète et pertinente de cette quantité de données, en prenant en compte les spécificités de chaque patient. Cette nouvelle problématique met en jeu plusieurs acteurs : d'un côté les professionnels de la santé, qui expriment leurs besoins et analysent les connaissances extraites, et d'un autre côté la communauté *"Extraction de connaissances"*, dont l'axe de recherche principal s'articule autour de la fouille de données.

L'objectif de cette thèse est de proposer des solutions adaptées permettant de traiter des données médicales. Le domaine de la santé étant large, nous devrons prendre en compte des données intervenant à divers niveaux. Tout d'abord, au *niveau micro* qui consiste en l'analyse de diverses particularités physiques/biologiques telles que les séquences d'ADN, de protéines ou encore d'expression de gène. Ensuite au *niveau macro*, qui définit les données plus générales (données cliniques) sur l'état du patient, son suivi, et consiste à discerner les causes et les conséquences de certaines maladies. De plus, chacun de ces niveaux de formats hétérogènes étant étroitement liés, il nous faut à terme proposer des solutions d'extraction de connaissances intégrant toutes ces données. Ce travail passe par la modélisation des données médicales, puis par les types de connaissances que nous souhaitons extraire. Chacune des solutions théoriques proposées est testée sur des bases de données réelles, et les résultats fournis aux professionnels de la santé.

Ce chapitre est organisé de la manière suivante : tout d'abord, nous introduisons le processus d'extraction de connaissances dans la section 1.2, qui constitue le contexte général de notre travail. Nous insistons plus particulièrement sur l'étape de fouille de données, car c'est à ce niveau que se situent nos

contributions. Dans la section 1.3, nous détaillons les différents aspects des bases de données médicales, qui caractérisent le cadre tout à fait particulier pour la mise en œuvre des algorithmes de fouille de données et donc de nos propositions. Enfin, dans la section 1.4, nous détaillons nos objectifs ainsi que l'organisation de ce mémoire.

## 1.2 Processus d'extraction de connaissance et fouille de données

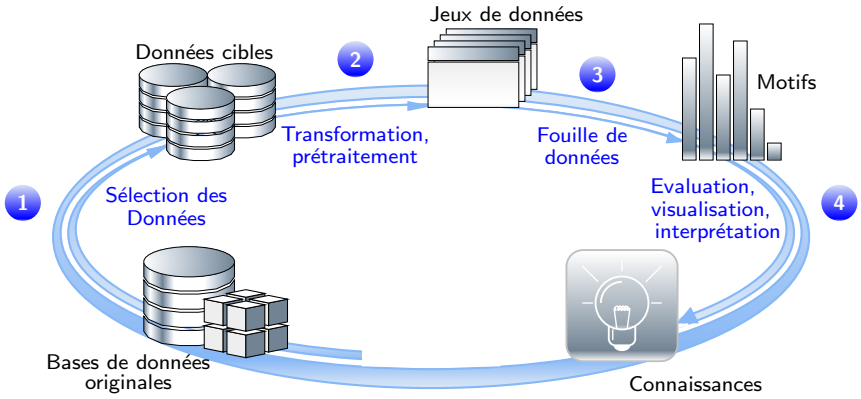



Fig 1.1 – Processus d'extraction de connaissances

Le processus d'extraction de connaissances dans les bases de données, (ECD, ou Knowledge Discovery from Databases en anglais) est un processus complet et complexe de découverte de connaissances au préalable inconnues à partir de grandes bases. Apparu au début des années 90, ce processus suscite un fort intérêt industriel, notamment pour son champ d'application très large, pour son coût de mise en œuvre relativement faible, et surtout pour l'aide qu'il peut apporter à la prise à la décision. Ce processus peut être découpé en quatre grandes étapes illustrées par la figure 1.1.

La fouille de données est l'étape centrale du processus d'extraction de connaissances. Elle consiste à découvrir de nouveaux modèles au sein de grandes quantités de données. Cependant, il est rarement possible d'appliquer directement la fouille de données sur les données brutes. Les premières opérations du processus ECD correspondent donc à la transformation des données avant de pouvoir appliquer des algorithmes de fouille de données.

Dans un premier temps, les bases de données qui serviront à l'extraction sont sélectionnées. En effet, il est courant que les données ne proviennent pas des mêmes sources et soient enregistrées sous divers formats. Cette phase d'acquisition consiste alors en diverses tâches d'intégration et de nettoyage : repérer lors de la sélection les inconsistances, les données trop bruitées, les nettoyer avant de les stocker dans les bases de données ciblées. Par exemple, il est courant dans le contexte médical de vouloir intégrer aux données brutes issues d'expérimentations les connaissances relatives au domaine. Ainsi, l'expert pourra, en plus de ses données expérimentales, sélectionner les annotations sémantiques s'y rapportant, souvent disponibles sous la forme d'ontologies. Les annotations apportent une information supplémentaire, qui peut être utile lors du processus de fouille. Par exemple, on peut annoter les gènes d'une base par leur

rôle fonctionnel. Durant cette étape, les données mal renseignées au cours de l'expérimentation ou encore dupliquées seront supprimées, et les sources de données sémantiques seront stockées sous la forme d'une base de données.

La seconde étape est une étape de prétraitement en vue de fabriquer les jeux de données adéquats à l'étape de la fouille. Il s'agira dans ce cas de sélectionner les items appropriés au processus décisionnel en cours, normaliser les données, les agréger, réduire le nombre de dimensions etc... Par exemple, dans un contexte médical, si l'expert souhaite étudier les données de certains gènes, il n'est pas nécessaire de conserver tous ceux présents dans la base originale. Cela aura pour conséquence de réduire la taille de la base de données, et donc d'augmenter les chances de succès de l'algorithme de fouille.

La fouille de données permet alors d'extraire des schémas qui modélisent ou synthétisent l'information contenue dans les données préalablement traitées. Selon les besoins et objectifs de la fouille, les schémas sont extraits par différentes techniques :

- la **classification**, dont le but est d'affecter des données à des classes préalablement définies ;
- le **clustering** (ou *segmentation*) qui permet de partitionner les données en sous-ensembles (ou groupes) de telle manière que la similarité entre les données d'un même cluster et la dissimilarité entre différents clusters soient les plus grandes possibles ;
- la **description des données** qui peut être réalisée à l'aide des règles d'association ou des motifs séquentiels, qui permettent d'extraire des corrélations tenant compte ou non d'une notion d'ordre.

Enfin, les schémas extraits sont ensuite analysés, interprétés et validés par l'expert. Durant cette étape, il est possible d'utiliser des techniques de visualisation de données, qui regroupent tous ou une partie des résultats, et permettent à l'expert de raffiner manuellement le résultat des fouilles. L'expert dispose également de mesures de qualité afin d'évaluer la pertinence des schémas découverts.

## 1.3 Fouille de données complexes et données médicales

Dans ce mémoire, nous nous intéressons à la fouille de données appliquée aux bases de données médicales. A l'origine, les algorithmes de découverte de motifs ont été proposés et appliqués dans un cadre d'analyse de données issues de supermarchés. Les transactions concernées ont donc un format spécifique : beaucoup plus de clients que d'attributs, hiérarchisation possible des produits, ou encore possibilité d'obtenir des informations temporelles sur les achats d'un même client (base de séquences). Peu à peu, l'intérêt de la communauté se tourne vers des domaines très variés : détection de fraudes et d'attaques pirates (finance), veille technologique (industrielle, militaire), comportement des utilisateurs web (internet)... Dans tous ces cas, le problème présente des spécificités particulières :

- **Fort nombre de clients** (objets) comparé au nombre d'attributs (items). Dans les exemples d'application cités ci-dessus, l'analyse recherchée est une analyse de masse. Il est donc très fréquent de trouver des bases retraçant le comportement d'un très grand nombre d'utilisateurs sur un faible nombre d'actions. Un exemple parlant est celui des supermarchés : les décideurs ne prennent pas en compte tous les produits, mais un ensemble très ciblé de produits.
- **Valeurs manquantes ou nulles** : dans ce cas, les bases sont dites *éparses* (à l'opposé des bases

*denses*). Cela signifie que l'espace de recherche sera plus facile à élaguer et donc que de très grandes bases peuvent être traitées en un seul passage, et dans des temps raisonnables. Par exemple, dans le cas d'un supermarché, il est irréaliste que tous les clients aient acheté tous les produits. Cela produira donc des bases avec de nombreux "trous"

- **Données binaires** : le plus souvent, les données sont présentées sous la forme présence / absence. Il existe peu de méthodes adaptées à la prise en compte des données quantitatives. Par exemple, il est possible de savoir qu'un client a acheté du pain, mais pas combien de baguettes de pain il a acheté.

Si la plupart des algorithmes de découverte de motifs ont montré leur efficacité pour ce type de bases de données, il en va différemment pour les bases médicales. En effet, ce n'est que récemment (début des années 2000) que les chercheurs se sont intéressés à l'adaptation de la fouille de données aux bases médicales. Cependant, les bases médicales ont des spécificités différentes, et rendent de fait l'application directe des algorithmes existants difficile. Globalement, les bases présentent les caractéristiques suivantes :

- **Format des bases inversées :** les algorithmes de fouille de données originalement proposés permettent l'extraction de connaissances sur des bases contenant un grand ensemble d'objets, et proportionnellement moins d'attributs. Certaines bases médicales au contraire contiennent un faible nombre de patients (objets) et un grand nombre d'attributs. Cela est notamment dû au coût de fabrication des bases : les expérimentations se font donc sur un nombre très limité d'individus, et les chercheurs en profitent pour noter un maximum d'information, ce qui produit un très grand nombre d'attributs par objet.
- **Bases denses :** dans la grande majorité des cas, les expériences menées permettent de remplir tous ou du moins une grande partie des renseignements. Les bases de données génomiques en sont un parfait exemple : le génome humain contient près de trois milliards de nucléotides représentés par les lettres A,T G et C (voir annexe A) répartis sur environ 30 000 gènes. Ainsi, une puce à ADN permet de renseigner pour un seul individu plusieurs milliers d'expressions de gènes.
- **Données numériques :** Les données médicales mélangent régulièrement quantitatif et qualitatif. De nombreuses méthodes ont été proposées afin de traiter les données binaires : approches par intervalle [SA96], approches floues [HLW03, FLT07] ou encore approche statistique [AL99]. Cependant, ces approches ne sont pas efficaces sur des bases ayant de telles spécificités.

Ainsi, les bases médicales constituent un véritable défi pour la fouille de données. Parmi les nombreuses problématiques posées par ces bases, nous recensons principalement :

- **Prise en compte des différents types de données :** actuellement, il existe peu de méthodes permettant de corréler les données patients (antécédents familiaux, sexe du patient, récidive de la maladie, etc...) aux données biologiques pures. En effet, il est difficile de prendre en compte des données *qualitatives* et des données *quantitatives* durant le processus de fouille.
- **Format des données extraites :** il existe actuellement divers types de connaissances, chacun ayant été validé par différentes communautés. Parmi les plus connus, on retrouve les clusters, les règles d'association, les motifs séquentiels ou encore les motifs ensemblistes. Cependant, ces formats ne suffisent plus, car soit ils n'apportent pas assez de connaissances (le clustering sur

données biologiques par exemple), soit ils sont difficiles à adapter aux bases médicales. C'est le cas des motifs séquentiels qui mettent en évidence des corrélations entre ensembles ordonnés. Il devient alors nécessaire d'enrichir les formats de données existants, tout en conservant une structure compréhensible par l'expert.

- **Typicité et atypicité :** si la découverte de comportements typiques s'avère primordiale pour la bonne compréhension des mécanismes du corps humain, les experts s'intéressent également de plus en plus à la découverte de comportements dit "atypiques", et qui peuvent aider à la compréhension de certaines maladies. Cependant, si la notion de "typicité" est généralement associée à la notion de support, il n'en est pas de même pour les comportements atypiques. Ainsi, il s'agit dans un premier temps de définir de manière formelle la notion d'atypicité, puis de proposer des algorithmes d'extraction efficaces adaptés à de telles connaissances.

## 1.4 Objectifs et contributions

Dans ce travail, nous avons tenté de répondre aux problématiques exposées ci-dessus. Tout d'abord, les méthodes de fouilles de données proposées sont étudiées afin de prendre en compte l'aspect quantitatif de la plupart des bases médicales. C'est pourquoi nous nous focalisons ici sur la recherche de co-variations de valeurs. En effet, ce type de connaissance désigné sous le terme de *motif graduel* a récemment émergé dans la communauté fouille de données [Hül02, BCS+07]. Cependant, les méthodes proposées se heurtent encore au verrou du passage à l'échelle d'une part, et du traitement de différents formats de données d'autre part. Afin de lever ces verrous, nos contributions sont les suivantes :

- **Formalisation des itemsets graduels et définitions des algorithmes d'extraction efficaces associés.** Les deux approches citées ci-dessus sont en effet les seuls travaux s'appuyant sur les techniques de fouille de données afin de traiter les motifs graduels. Cependant, ces travaux se positionnent dans le contexte particulier de la logique floue, alors que les motifs graduels peuvent être pensés dans un cadre de fouille non floue. De plus, il n'existe pas d'études expérimentales sur données synthétiques et réelles à grande échelle démontrant leur efficacité. Nous avons donc proposé un nouveau cadre formel pour les motifs graduels non flous nous permettant de mettre en évidence certaines propriétés et par là-même d'utiliser des structures de données optimisées.
- **Étude approfondie de la sémantique de la fréquence graduelle.** En fouille de données, la mesure de fréquence reflète la proportion des objets de la base validant un motif. Cette mesure est au cœur de tous les algorithmes d'extraction de connaissances puisque ses propriétés permettent un élagage efficace de l'espace de recherche. Cependant, si la fréquence est intuitive dans le contexte de la fouille de données classique, il n'en est pas de même lorsque l'on parle de gradualité. Ainsi, dans ce mémoire, nous discutons différentes définitions possibles et étudions leurs propriétés ainsi que leurs impacts sur les algorithmes et sur les résultats obtenus.
- **Découverte d'exceptions.** Si la découverte de tendances, basée sur la définition de mesures de fréquence, est importante, il est souvent également primordial d'extraire les objets ne suivant pas ces tendances. On parle alors d'exceptions. Dans ce mémoire, nous montrons que nos propositions sont adaptées à l'extraction d'exceptions et discutons des problèmes associés à la définition du concept d'exception dans le contexte graduel.
- **Multidimension, séquences et skylines** le domaine médical recèle différents types de bases de

données. Par exemple, des chercheurs se sont récemment intéressés à construire des entrepôts de données sur le VIH ou encore sur des patients en obstétrique. Ces bases possèdent des caractéristiques propres, comme par exemple le fait d'être représentées sur plusieurs dimensions, ou encore de posséder différents niveaux de granularité. Dans ce mémoire, nous définissons les itemsets graduels multidimensionnels ainsi que les algorithmes associés. En outre, nous proposons une réflexion sur d'autres types de format d'extraction tels que les skylines ou les motifs séquentiels, et montrons ce que les règles graduelles peuvent apporter à ces types de motifs.

## 1.5 Organisation du mémoire

Ce mémoire est organisé de la manière suivante : tout d'abord, dans le chapitre 2, nous dressons le panorama des travaux existants en fouille de données appliqués au milieu médical. Nous nous intéressons plus particulièrement aux techniques d'extraction de motifs dans le contexte des séquences biologiques et des expressions de gènes.

Le chapitre 3 s'intéresse à l'extraction d'itemsets graduels à partir de bases de données relationnelles. Nous analysons et comparons les travaux en relation avec l'extraction de connaissances graduelles, puis nous proposons deux approches d'extraction automatique. La première est une heuristique qui offre un type de structure particulier, et qui permet d'extraire des motifs plus longs sous certaines conditions. La seconde est une approche complète utilisant des structures adéquates. Ces deux approches sont expérimentées sur des jeux de données réels (données biologiques et issues de tests psychologiques) et synthétiques.

Le chapitre 4 s'intéresse à la sémantique du comptage. Nous définissons diverses mesures et en étudions les propriétés ainsi que la possibilité de les mettre en œuvre au sein d'un algorithme par niveau. De plus, nous nous intéressons dans ce chapitre à la définition des comportements atypiques, et proposons une méthode de mise en évidence des objets dont la variation de valeur est moins commune que les autres. Nous détaillons nos expérimentations sur deux jeux de données réels.

Le chapitre 5 s'intéresse à l'extraction d'itemsets graduels dans le contexte multidimensionnel. Nos contributions sont doubles : nous proposons une définition des itemsets graduels multidimensionnels basés sur les blocs de données, et un nouvel algorithme d'extraction de blocs de données multidimensionnels. Ce chapitre est également fourni en expérimentations, et plus particulièrement des expérimentations synthétiques afin de mesurer les performances de notre méthode.

Le chapitre 6 s'intéresse à l'extraction de motifs séquentiels graduels. Nous verrons que le concept de gradualité dans les bases de séquences (comme par exemple les bases temporelles) peut s'avérer très complexe. Nous identifions deux types de motifs séquentiels graduels, et proposons un algorithme d'extraction naïf pour l'un de ces deux types. En outre, nous nous intéressons aux skylines. Nous démontrons comment les itemsets graduels permettent une extraction efficace des skylines ainsi que les problématiques associées aux skylines. Ce chapitre est plus conceptuel que pratique, c'est pourquoi nous ne proposons pas d'expérimentations.

Enfin, dans le chapitre 7, nous concluons et présentons les différentes perspectives levées lors de notre travail.

# Chapitre 2

# Fouille de données et bases médicales

## 2.1 Introduction

Ce chapitre a pour but de présenter dans un premier temps les définitions préliminaires à la fouille de données ainsi que les principaux algorithmes d'extraction de motifs. Dans un second temps, nous nous intéressons aux différentes tentatives d'extraction à partir de bases médicales, et plus particulièrement à partir d'une base de séquences ou de protéines.

L'extraction de motifs consiste à mettre en évidence des schémas récurrents. Dans le domaine de la biologie, les algorithmes classiques d'extraction d'itemsets ne peuvent être appliqués directement, notamment à cause de l'extrême disparité des bases biologiques et de leur forme particulière : peu de clients, peu d'items, mais des transactions extrêmement denses. Afin de faire face à ces nouvelles conditions, les algorithmes doivent incorporer d'autres types de mesures que la fréquence, comme par exemple la similarité entre séquences. La mesure de similarité est d'autant plus importante qu'elle permet d'extraire des motifs intéressants d'une part, et participe à l'élagage de l'espace de recherche d'autre part.

Dans cette section, nous présentons certains de ces algorithmes, choisis pour la problématique abordée ou encore pour la structure proposée. Tout d'abord, nous présenterons des algorithmes qui permettent l'extraction de motifs de séquences récurrentes avec un certain taux d'erreur. Puis nous décrirons les algorithmes ciblés sur l'extraction des séquences dont la taille n'est pas limitée, mais qui sont décrites par des expressions régulières.

## 2.2 Extraction de motifs

Soit une table transactionnelle $\mathcal{DB}$. Cette base est composée d'un ensemble d'items noté $\mathcal{I}$ et d'un ensemble d'objets noté $\mathcal{O}$. Un itemset, noté $(i_1 i_2 ... i_n)$ est un sous ensemble non vide de $\mathcal{I}$. Afin d'illustrer notre propos, nous utilisons la base du tableau 2.1 qui décrit différents artistes, leur style de musique, s'ils forment un groupe, s'ils composent et leur instrument principal. Dans cette base, $\mathcal{I}_1 = (Rock,\ Guitare)$ est un itemset.

| **Artiste(s)** | **Style** | **Groupe** | **Compose** | **Instrument principal** |
|---|---|---|---|---|
| Keny Arkana | Rap | Non | Non | Batterie, Guitare |
| Demago | Rock | Oui | Oui | Guitare |
| Pink | Pop Rock | Non | Oui | Guitare, Synthétiseur |
| IAm | Rap | Oui | Non | Batterie |
| Mademoiselle K | Rock | Non | Oui | Guitare |

Tab 2.1 – Base de données exemple : artiste et catégorie de musique

Une transaction supporte un itemset si elle contient tous les items le composant. Par exemple, les transactions *Demago*, *Pink* et *Mademoiselle K* supportent l'itemset $\mathcal{I}_1$. La fréquence d'un itemset représente le pourcentage de transactions de la base qui supportent cet itemset. Dans le cas de $\mathcal{I}_1$, la fréquence est $Freq(\mathcal{I}_1) = \frac{3}{5} = 0.6$, soit 60%.

Une règle d'association est de la forme $X \rightarrow Y$ où $X \subseteq \mathcal{I}$ est appelé la *condition* de la règle et

$Y \subseteq \mathcal{I}$ est appelé la *conséquence*. De plus, $X \cap Y = \emptyset$. Une règle d'association est principalement évaluée par deux mesures :

- La **fréquence** de l'itemset $(XY)$, qui permet de connaître le pourcentage d'objets de la base contenant les itemsets $X$ et $Y$
- La **confiance** permet de connaître la probabilité qu'un objet contenant $X$ contienne aussi $Y$. Cette mesure est calculée par $Conf(X \rightarrow Y) = \frac{Freq(XY)}{Freq(X)}$

Si l'on reprend notre exemple précédent, on trouve $Conf(Guitare \rightarrow Rock) = \frac{3}{4} = 0.75$. Ainsi, l'utilisateur sait que 60% des artistes de la base contiennent le motif $(Guitare,\ Rock)$ et que dans 75% des cas, un artiste dont l'instrument principal est la guitare fait de la musique rock.

La problématique d'**extraction de règles d'associations** consiste à trouver à partir d'une base de données l'ensemble de toutes les règles d'association dont la fréquence et la confiance respectent des seuils de fréquence minimale ($minFreq$) et de confiance minimale ($minConf$) fixés par l'utilisateur. Pour ce faire, tous les algorithmes fonctionnent en deux étapes : dans un premier temps, l'ensemble des itemsets fréquents sont extraits, puis dans un second temps les règles d'associations sont générées à partir de cet ensemble.

L'espace de recherche que les algorithmes doivent explorer lors de la recherche d'itemsets fréquents est de très grande taille (de l'ordre de $2^{|\mathcal{I}|}$ itemsets candidats). Afin de limiter cet espace de recherche, les algorithmes reposent sur la propriété d'anti-monotonie :

**Propriété 1.** *(anti-monotonie) Soit $\mathcal{I}_1$ et $\mathcal{I}_2$ deux itemsets. Si $\mathcal{I}_1 \subseteq \mathcal{I}_2$ alors $Freq(\mathcal{I}_1) \geq Freq(\mathcal{I}_2)$.*

La propriété 1 est particulièrement importante dans les algorithmes d'extraction de connaissance, puisqu'elle permet d'affirmer que si un motif de taille $n$ n'est pas fréquent, alors aucun de ses sur-motifs ne le seront. Cela permet de ne pas tester ou même générer les sur-motifs d'un motif non fréquent.

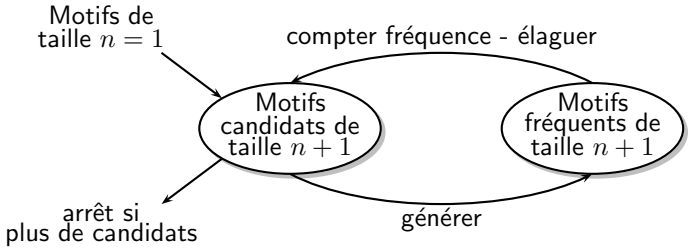



Fig 2.1 – Méthode générer-élaguer

Il existe deux paradigmes d'extraction d'itemsets fréquents : *générer-élaguer* (illustré par la figure 2.1) et *pattern-growth*. Le paradigme pattern-growth, dit sans génération de candidat adopte une technique divide-and-conquer. Dans ce mémoire, nous nous intéressons plus particulièrement à la méthode générer-élaguer, dont le principe est le suivant :

1. Génération des itemsets de taille 1 fréquent. Cette étape est réalisée à partir d'une lecture de la base
2. Génération des itemsets de taille n+1 par jointure sur les itemsets de taille n

3. Comptage de la fréquence, et élagage des itemsets non fréquents
4. Arrêt lorsqu'il n'y a plus d'itemsets candidats

Parmi les algorithmes fondateurs d'extraction de règles d'associations, on retrouve [AS94, AMS+96, HPY00, Zak01].

Les motifs séquentiels peuvent être vus comme une extension des itemsets. Ils sont particulièrement adaptés aux bases de données contenant un ordre, comme par exemple le temps. L'idée est de fouiller non plus les corrélations entre sous-ensembles d'itemsets, mais les ordres récurrents entre itemsets. Un motif séquentiel est défini de la manière suivante :

**Définition 1.** *Un motif séquentiel est une liste ordonnée et non vide d'itemsets noté* $s =< \mathcal{I}_1\mathcal{I}_2...\mathcal{I}_n >$.

| Id | Date | Achats |
|---|---|---|
| 1 | 1 | Pain, Beurre |
| 1 | 2 | Chocolat |
| 2 | 2 | Pain, Beurre, Chocolat |
| 2 | 5 | Beurre, Chocolat |
| 3 | 1 | Pain, Beurre |

Tab 2.2 – Base de données exemple : achats dans un supermarché

Par exemple, un motif séquentiel du tableau 2.2 est $< (Pain, Beurre)(Chocolat) >$. Sa fréquence est de $\frac{2}{3}$, puisque les clients 1 et 2 ont acheté du pain et du beurre, puis plus tard du chocolat. Lors de l'extraction de tels motifs, la principale difficulté réside dans le fait que l'espace de recherche est exponentiel. Depuis leur introduction, de nombreux algorithmes efficaces ont été proposés tels que [MCP98, PHMa+04, ZH05].

## 2.3 Extraction de motifs à partir de bases de données biologiques

Historiquement, les règles d'association et motifs séquentiels ont été proposés afin de répondre à la problématique du panier de la ménagère (analyse des données de supermarchés). Ce n'est que plus récemment, au début des années 2000, que les chercheurs s'intéressent aux bases de données médicales. Dans cette section, nous présentons les travaux leaders dans ce domaine. Notons que de nombreux autres travaux ont été menés dans le cadre de la fouille de données médicale en dehors de l'extraction de motifs. Le lecteur pourra se référer à [CL09].

La plupart de ces méthodes ont été proposées dans un cadre de fouille de séquences d'ADN, codées par les quatre lettres A,T,G et C, et ont pour but de découvrir des motifs d'intérêts, souvent à l'origine d'une fonction du corps humain. Dans ce contexte, la difficulté provient du fait qu'une sous séquence de lettres réalisant la même fonction n'est pas forcement exactement la même d'un individu à l'autre. Ainsi, les auteurs s'orientent vers la recherche de motifs contenant des caractères joker (un caractère jocker peut être remplacé par n'importe quel autre caractère), ou encore de motifs inexacts.

L'ADN étant à la base de toute fonction du corps, le champ d'application de ces méthodes est divers : `WEEDER` [PMP01] et `MITRA` [EP02] ont été introduits dans le cadre de la découverte de signaux, `MAMF`

[HJ06] a été expérimenté sur la découverte de facteur de transcription, tandis que TEIRESIAS [RF98] et SPLASH [Cal00] ont prouvé leur efficacité sur de nombreuses problématiques, telles que l'alignement de séquences multiples, la découverte de fonctions de gènes ou la découverte de similarité de structures au sein d'une famille de protéines.

### 2.3.1 Exploration exhaustive de l'espace de recherche

Les premières méthodes fonctionnent d'une manière inverse à la philosophie de la fouille de données : plutôt que d'élaguer les motifs non fréquents au fur et à mesure de leur découverte, l'ensemble des motifs présents dans la base est projeté en mémoire et rangé dans une structure adéquate. Les motifs inintéressants sont élagués dans une seconde phase. Ainsi, [Sag98] propose d'utiliser des arbres suffixés afin d'extraire des (l-d)-motifs (motifs de longueur $l$ et contenant au plus $d$ erreurs). Habituellement utilisés dans le domaine du traitement automatique de la langue, ce type d'arbre permet de compter aisément les récurrences de mots ou d'enchaînements de caractères.

Cet arbre suffixé est également exploité dans [PMP01], où les auteurs proposent d'élaguer rapidement l'espace de recherche à partir d'un taux d'erreur maximal fourni. Le remplacement d'une lettre par une autre est considéré comme une erreur. Ainsi, à partir d'un ensemble de séquences ADN, des paramètres $d$ (taux d'erreur maximal), d'un seuil minimal de fréquence k ainsi que de la longueur $l$ de motif maximal, un arbre suffixé est construit de manière récursive, en conservant le long du chemin le nombre d'erreurs commises : à chaque remplacement, le chemin voit son taux d'erreur incrémenté. La recherche s'arrête lorsque plus aucune séquence ne peut être générée, ou bien si le taux d'erreur maximal est atteint.

Cette méthode est exhaustive car elle nécessite une énumération successive de tous les motifs possibles n'ayant pas atteint le taux d'erreur maximal et vérifiant la contrainte de fréquence. Constatant que la recherche devient trop lente pour un nombre d'erreurs élevé ou pour des motifs d'une grande taille, les auteurs proposent une heuristique afin de partitionner l'espace de recherche. Cependant, comme cette méthode peut laisser de coté certaines séquences, il faut alors relancer l'algorithme en prenant en compte les résultats obtenus lors de la première passe.

Dans [EP02], l'ensemble des motifs est recherché à partir d'une seule séquence S. Ainsi, la stratégie de recherche diffère au moment de l'énumération, puisqu'il est possible de découper S en plusieurs sous séquences de longueur $l$. L'algorithme se déroule en deux étapes : construction de l'arbre suffixé et d'un graphe des cliques correspondant. Dans un graphe des cliques, chaque sommet contient l'une des l-séquences de l'arbre suffixé, et un arc relie deux sommets similaires. La similarité entre deux séquences est une distance de Hamming inférieure ou égale à deux. Une clique est un ensemble de nœuds connectés : chaque nœd est connecté à l'ensemble des autres nœuds de la clique. L'élagage des l-séquences non fréquentes au niveau de l'arbre permet de supprimer les arcs ne faisant pas partie de l'ensemble maximal des cliques. Malgré la forte complexité de l'algorithme, les auteurs montrent que les résultats en recherchant les (28-8)22-séquences sur une base d'environ 7000 séquences sont concluants.

[HJ06] propose de découper l'espace de recherche selon des motifs de longueur $l$ afin de créer un index sur la base des séquences. Ces motifs de petite taille servent de point de comparaison entre les séquences de la base. Dans un second temps, les séquences de la base sont comparées deux à deux

en fonction des l-motifs, qui servent à aligner les deux séquences comparées. Le cœur de l'algorithme réside dans la fonction de score de similarité entre deux séquences. Celle-ci est basée sur les morceaux de séquences communes, ainsi que sur l'unicité du motif par rapport au génome, qui est estimé à partir d'un modèle basé sur un calcul statistique réalisé en prétraitement et utilisant les chaînes de Markov cachées du troisième ordre. Ainsi, la mesure de score garantit de favoriser les couples de séquences similaires et uniques par rapport au génome. Cependant, les auteurs constatent qu'avec cette méthode, les séquences ayant un fort score sont des séquences hautement similaires. Ils proposent alors un raffinement des 1000 séquences ayant le plus fort score afin de n'extraire que les 30 séquences les plus scorées et les moins similaires.

Les auteurs proposent également d'utiliser des connaissances a priori fournies par les expressions de gènes. En effet, les auteurs cherchent à identifier les séquences promotrices responsables de la transcription de gènes. Sachant qu'un gène s'exprime rarement seul, une liste des gènes co-exprimés apporte une information utile. [HJ06] calculent donc un taux d'enrichissement basé sur la corrélation de Pearson et utilisent ce taux afin de déterminer si un motif extrait est une séquence promotrice ou non.

### 2.3.2 Combinaison de motifs

Une méthode alternative à l'énumération de toutes les possibilités de la base consiste à combiner les motifs, comme le proposent `TEIRESIAS` [RF98] et `SPLASH` [Cal00]. Ces méthodes se décomposent en deux étapes : extraction de tous les motifs fréquents de faible taille avec des paramètres fournis par l'utilisateur, puis combinaison de ces motifs afin de trouver les motifs de plus grande taille. Les deux algorithmes diffèrent principalement dans leur traitement de la seconde étape. Ces algorithmes représentent les séquences comme des expressions régulières, ce qui permet la prise en compte des caractères jokers, procurant ainsi un fort pouvoir d'expression et intègrent la notion de motifs similaires.

`TEIRESIAS` découpe la base de données en motifs élémentaires correspondant au critère de taille *W*, ayant au moins *L* items différents et de fréquence *K* (*L*, *W* et *K* sont des paramètres fournis par l'utilisateur). Cette étape appelée scanning peut être réalisée simplement durant une phase de lecture de la base. Le seuil de fréquence minimum sert à élaguer les motifs non fréquents. L'exemple 1 illustre cette première étape.

**Exemple 1.** *Soit S = ABCDEFG une séquence, W=4 et L=3. Un premier sur-ensemble des motifs élémentaires prend en compte la contrainte de taille et renvoie {ABCD, BCDE, CDEF, DEFG}. Ensuite, pour chacun des éléments de cet ensemble, la contrainte L=3 est appliquée. Par exemple nous redécoupons DEFG en quatre éléments : {DEF, DE.G, D.FG, EFG}. Dans cette notation, le '.' signifie "n'importe quel caractère". Ainsi, la séquence S va produire* $4 \times 4 = 16$ *motifs élémentaires.*

**Exemple 2.** *Soit* $S_1 = AS..L.JKP$ *et* $S_2 = AUB.K.LP$*. Dans la liste des préfixes,* $S_1$ *sera plus spécifique que* $S_2$*, car* $S_2$ *a un caractère en troisième position alors que* $S_1$ *a un '.'. En revanche,* $S_2$ *est plus spécifique que* $S_1$ *dans la liste des suffixes, car il possède un '.' en troisième position en partant de la fin, alors que* $S_1$ *possède un caractère à cette même position.*

Les motifs sont ensuite ordonnés selon leurs spécificités dans deux listes : une liste des préfixes, et une liste des suffixes. L'exemple 2 illustre la méthode de classement des motifs en fonction des listes.

L'algorithme entre ensuite dans l'étape de *convolution*, qui consiste à itérativement coller les motifs dont le préfixe de l'un est identique au suffixe de l'autre. Dès que l'algorithme trouve un motif maximal, il le renvoie en sortie. L'exemple 3 montre l'étape de convolution :

| | |
|---|---|
| $S_1$ | BCD |
| $S_2$ | CD..EF |
| $S_3$ | A.BC |

(a) Motifs élémentaires

| | |
|---|---|
| $S_1$ | BCD |
| $S_2$ | CD..EF |
| $S_{new_1}$ | BCD..EF |

(b) $S_1 \bigoplus S_2$

| | |
|---|---|
| $S_{new_1}$ | BCD..EF |
| $S_1$ | BCD |
| $S_3$ | A.BC |
| $S_{new_2}$ | A.BCD..EF |
| $S_{new_3}$ | A.BCD |

(c) $S_3 \bigoplus S_1$ et $S_3 \bigoplus S_{new_1}$

Tab 2.3 – Étape de convolution de l'algorithme `TEIRESIAS`

**Exemple 3.** *Le tableau 2.3a représente l'ensemble des motifs élémentaires générés. Soit l'opération $M_1 \bigoplus M_2$, qui permet de coller le préfixe de $M_2$ au suffixe de $M_1$. L'étape de convolution effectuera alors les opérations suivantes : $S_1 \bigoplus S_2$ (cf tableau 2.3b), $S_3 \bigoplus S_1$ et $S_3 \bigoplus S_{new_1}$ (cf tableau 2.3b). Notons que $S_{new_3}$ est un sous-motif de $S_{new_2}$. Il n'est pas maximal, et ne sera par conséquent pas retourné par l'algorithme.*

`SPLASH` extrait les "motifs élémentaires" de la même manière que `TEIRESIAS`. Cependant, `SPLASH` recherche les motifs maximaux par une politique d'extension vers la gauche, puis vers la droite. Dans un premier temps, `SPLASH` vérifie que tous les motifs élémentaires ne peuvent être étendus par la gauche en conservant la même fréquence, et qu'ils sont les plus spécifiques. Ces contraintes sont dites de maximalité gauche et de composition. Ensuite `SPLASH` étend itérativement chaque motif élémentaire par la droite, en assurant également les contraintes de maximalité décrites précédemment. L'exemple 4 illustre le déroulement de cette étape.

**Exemple 4.** *Soit la base composée de $S_1 = ZABBBCD$ et $S_2 = ABBBCDH$, et soit le motif élémentaire $M_1 = BB.C$. Tout d'abord, $M_1$ n'est pas maximal à gauche car il peut être étendu par la gauche en ajoutant un 'A'. Cependant $M_1 = ABB.C$ ne satisfait toujours pas les conditions de maximalité de composition puisqu'il n'est pas le plus spécifique possible : le '.' peut être remplacé par le caractère 'B'. Dans un second temps, `SPLASH` étend $M_1 = ABBBC$ par la droite en ajoutant le caractère 'D'. Ainsi, $M_1 = ABBBCD$ est maximal à droite et à gauche et peut être retourné.*

La nouveauté introduite dans `SPLASH` est la recherche de séquences "similaires". Les auteurs définissent les séquences similaires comme des classes d'équivalence. Ainsi, la recherche pourra être conduite en remplaçant chaque occurrence similaire par le nom de sa classe d'équivalence, comme le montre l'exemple 5.

**Exemple 5.** *Soit EQ = {BB, CA} et une base composée de $S_1 = ATBBC$ et $S_2 = ABCAD$. Alors un motif similaire à $S_1$ et $S_2$ peut être extrait : $A.[BB][CA]$.*

### 2.3.3 Limitations

Les algorithmes présentés ci-dessus recquièrent tous un certain nombre de paramètres comme par exemple la taille maximale du motif, ou encore le nombre de dissimilarités maximales autorisées. Ces paramètres supposent que l'utilisateur sait ce qu'il cherche, et ces méthodes, par leur aspect limitatif, laissent peu de place à la découverte de séquences inattendues, même s'il est possible pour l'utilisateur de définir les séquences comme des expressions régulières. Le seul algorithme ne demandant pas de paramètres est `MAMF`. Cependant, certains choix ne sont pas justifiés et laissent une certaine subjectivité critiquable, comme le fait de ne sélectionner que les 1000 premières séquences similaires, ou encore simplement de ne fournir que les 30 séquences restantes après filtrage.

Ces algorithmes sont également extrêmement sensibles aux variations de paramètres telles que l'augmentation de la taille maximale des séquences ou encore du nombre de dissimilarités. Cela est principalement vrai pour les algorithmes qui énumèrent et comptent l'intégralité des motifs potentiellement valides (recherche exhaustive), car l'espace de recherche est particulièrement grand. Ce problème, bien connu en fouille de données sous le nom d'explosion combinatoire rend le temps d'exécution des algorithmes important. De plus, certaines structures mémoires, telles que les arbres préfixés et suffixés ou encore les graphes (`WEEDER` et `MITRA`) recquièrent une importante capacité mémoire. Ainsi, certains algorithmes ne fonctionnent que sur des bases ayant un faible nombre de séquences (de l'ordre d'une vingtaine), comme `MAMF` ou encore `WEEDER`.

Les algorithmes d'extraction de motifs sont extrêmement sensibles au seuil de fréquence minimale : soit celui-ci est trop haut et un faible nombre de séquences sera extrait, soit la fréquence est trop basse et au contraire ce sera dans ce cas un trop grand nombre de séquences qui sera extrait. Afin de résoudre ce problème, il est fréquemment proposé d'utiliser une mesure de pertinence en plus de la fréquence. Cependant, mis à part `MAMF`, aucune méthode présentée ci dessus ne propose l'utilisation d'une telle mesure. Ainsi, la similarité par classe d'équivalence ou caractères joker (`SPLASH` et `TEIRESIAS`), ou encore d'un nombre de différence maximal (`WEEDER` ou `MITRA`) sera privilégié.

D'autre part, `SPLASH` et `TEIRESIAS` montrent un inconvénient majeur : ils supposent qu'une instance de chaque motif est présente dans chaque séquence. Ce problème est bien connu sous le nom de "*One Occurrence Per Sequence*" (OOPS) et se retrouve dans des algorithmes plus anciens desquels découlent `SPLASH` et `TEIRESIAS` : `MEME` et `MOTIF` [BE94, SAC90].

Pour finir, la dernière limitation de ces méthodes est leur manque de généricité. En effet, ils sont tous prévus et même implémentés pour un alphabet d'une taille comprise entre 4 (le nombre de bases azotées de l'ADN) et 22 (nombre d'acides aminés reconnus dans le code génétique). Ces algorithmes ne seront donc jamais applicables à des données telles que l'expression des gènes ou encore les structures des protéines, où le nombre d'attributs est de l'ordre de plusieurs dizaines de milliers. Or, avec l'évolution des technologies de puces à ADN, de nombreuses bases d'expressions de gènes ont vu le jour. Ces bases ont un format particulier sur lesquels des algorithmes classiques de fouille ne peuvent s'appliquer : beaucoup moins de clients que d'items, un faible nombre de transactions par client et beaucoup d'items par transaction (densité). Nous présentons dans la section suivante des méthodes d'extraction de connaissances dédiées à de telles bases, et plus particulièrement dans le cadre de l'extraction de règles

d'association.

## 2.4 Extraction de règles d'association par transposition des données

Les techniques de puces à ADN ont pour but l'étude du fonctionnement des gènes. Une analyse pertinente des bases de données produites peut contribuer à la création de réseaux de régulation des gènes, au profilage de gènes ou encore à la découverte de nouvelles fonctions véhiculées par les gènes. La découverte de telles connaissances est à l'origine de tant de traitements et d'avancées médicales qu'il est devenu indispensable de proposer des outils d'analyses adéquats et performants.

Les règles d'association s'avèrent particulièrement adaptées aux bases d'expressions de gènes. Cependant, les spécificités détaillées précédemment obligent les chercheurs à proposer de nouvelles méthodes. Dans cette section, nous présentons les principaux travaux associés à ces problématiques.

### 2.4.1 Une stratégie Divide & Conquer : CARPENTER

Constatant que les algorithmes classiques perdent trop de temps à énumérer incrémentalement l'ensemble des itemsets potentiellement fréquents, [PCT+03] propose de compter l'ensemble des items communs à un ensemble donné de clients. Ainsi, au lieu de construire l'arbre des items fréquents, `CAPENTER` construit l'arbre des clients, puis ajoute de manière récursive à chaque noeud l'ensemble des items supportés. La technique de recherche adoptée est du type "*divide and conquer*" car les auteurs partent de l'ensemble des clients de la base et divisent récursivement cet ensemble. L'algorithme s'arrête lorsque le nombre de clients de l'ensemble du niveau courant est inférieur au support minimal. Enfin, les auteurs proposent deux techniques d'élagage permettant de ne pas fouiller un ensemble qui sera non fréquent, et de se limiter à l'extraction de motifs fermés.

Cette proposition reste limitée car les auteurs continuent de fouiller dans les données originales et n'exploitent pas totalement l'idée d'une fouille transposée (transposer les objets et les attributs), ni de la relation existante entre l'ensemble des clients et des items. C'est pourquoi de nombreux autres travaux proposent une réelle transposition des données, notamment en utilisant des techniques provenant de l'analyse formelle de concepts (AFC).

### 2.4.2 Treillis de Galois transposés : DMINER et autres

[BRBR05] propose l'algorithme `DMINER`, basé sur les treillis de concepts. Introduits par [Wil82], les treillis de concepts sont des treillis dont les sommets sont des concepts. Un concept est un regroupement maximal d'objets possédant des items en commun. Ces concepts sont reliés par inclusion, et fournissent une représentation très efficace des données, notamment pour la fouille de données. La problématique principale devient alors la construction de ce treillis, ce à quoi répond `DMINER` en introduisant la notion de cutter. Un cutter est un élément qui va conduire à la construction de deux concepts. Notons que de nombreux algorithmes de fouille de règles d'association proposent des solutions efficaces basées sur les treillis des concepts, comme par exemple CHARM [ZH05], ou encore Touch et Talky-G [SVNG09].

`DMINER` réalise une véritable transposition des données, puisque l'ensemble des concepts est le même que la base soit transposée ou non. Cela permet alors de fouiller sur l'ensemble des objets s'il s'avère

plus petit que celui des items et par conséquent d'augmenter significativement les performances des algorithmes. De plus, la méthode permet l'incorporation de contraintes monotones, telle que la fréquence et permet donc d'extraire rapidement des règles d'associations. DMINER a été testé et a prouvé son efficacité sur des jeux de données réels. A partir de ces propositions, de nouvelles recherches ont émergées concernant principalement l'inclusion d'autres types de contraintes et la transposition de contraintes complexes [Rio05].

Dans [RBCB03], les auteurs montrent qu'il est possible d'appliquer des algorithmes existants efficaces (comme par exemple Apriori) sur une base transposée. La méthode proposée se déroule en trois étapes : dans un premier temps, transposer la base et la contrainte associée (le plus souvent, une contrainte de fréquence). Ensuite, appliquer un algorithme classique de recherche de motifs clos avec la contrainte transposée. Finalement, utiliser les opérateurs de Galois pour passer de l'ensemble des objets à l'ensemble des attributs. La transposition de la base est une étape triviale, puisqu'il s'agit simplement d'inverser les lignes et les colonnes. La transposition de la contrainte de fréquence revient à calculer les concepts dont le nombre d'attributs concernés est également supérieur au seuil de fréquence minimal. Or, la transposée d'une contrainte anti-monotone est une contrainte monotone. Les algorithmes d'extraction de règles classiques se basent sur l'anti-monotonie de la contrainte de fréquence afin d'élaguer l'espace de recherche. Les auteurs montrent donc qu'il faudra utiliser l'inverse de la transposée de la contrainte de fréquence pour préserver la propriété d'anti-monotonie.

Les auteurs montrent que l'idée fonctionne de manière spectaculairement efficace sur des bases réelles. L'idée de la transposition semble donc la plus adaptée et reste la plus utilisée dans le cas de bases d'expression de gènes.

Dans [PB05], les auteurs proposent un scénario complet d'étude de base d'expression de gènes, allant du prétraitement de la base à l'enrichissement de concepts extraits. Les auteurs ne proposent pas de nouvel algorithme, mais une stratégie pour faire émerger les motifs pertinents. Cela passe par l'intégration des informations disponibles dans la littérature afin d'ajouter une connaissance supplémentaire à la base de départ. Ces informations peuvent être de diverses natures, comme les facteurs de transcriptions associés aux gènes, ou les fonctions connues des gènes, ou même les motifs extraits lors d'une première passe de l'algorithme sur les données.

### 2.4.3 Limitations

Les techniques présentées ci-dessus ont été pensées et proposées dans le cadre d'extraction sur des bases d'expression de gènes. Leur principal inconvénient est qu'elles ne peuvent fonctionner que sur des données binaires. Cependant, l'expression d'un gène est quantifié par des valeurs allant de -1 à 1, ce qui impose une étape de discrétisation. Dans ce contexte, celle ci consiste à remplacer la quantification de l'expression du gène par un '1' pour "le gène se sur-exprime ou se sous-exprime" et par un '0' pour "le gène s'exprime normalement". Or, la sur ou sous expression d'un gène est une donnée importante, puisqu'elle signifie que le gène concerné va produire ou non une protéine, et donc révéler une absence anormale de production, ou au contraire une surproduction anormale. De plus, cela conduit à produire des règles du style "si le gène Z est souvent à 1, alors le Gène W aussi". Or rien ne dit que cette règle ne signifie pas simplement "lorsque le gène Z ne s'exprime pas, le gène W s'exprime", ce qui apporte une

information plus fine pour les biologistes.

Ces techniques ne permettent pas d'extraire des connaissances faisant intervenir des contraintes de temps. De fait, elles ne permettent pas de s'inscrire dans le cadre d'une découverte de relations dans un contexte biologique structurel, comme par exemple celui des protéines. Ainsi, il est difficilement envisageable d'appliquer ces méthodes dans le cadre de découverte de fonction de gène. De plus, il existe des puces à ADN donnant l'évolution du gène au fil du temps. Il n'est donc actuellement pas possible de fouiller ce type de base de manière efficace.

Pour finir, ces méthodes ne permettent pas d'inclure les données cliniques associées aux données d'expressions. Or, dans le cadre d'une analyse de profil de gène, ces données sont primordiales. Il n'est donc actuellement pas possible de recouper des co-expressions fréquentes avec un symptôme ou un contexte médical donné. Cependant, la découverte de telles corrélations devient de plus en plus nécessaire pour la recherche en biologie.

## 2.5 Motifs séquentiels et bases médicales

Dans le domaine médical, il existe actuellement peu de travaux qui s'appuient sur l'utilisation de motifs séquentiels. Cette tendance s'explique principalement par la forme des données, qui sont souvent décrites sur deux dimensions. Cette section a pour but de traiter l'apport de l'utilisation de motifs séquentiels dans le cadre de bases à caractère biologique. Elle est organisée de la manière suivante : premièrement, nous présentons deux méthodes basées sur les motifs séquentiels. Après avoir discuté des limitations de ces approches, nous donnerons les principaux apports des motifs séquentiels.

[CMB02] propose d'extraire tous les motifs séquentiels similaires à un motif de référence donné par l'utilisateur afin de pratiquer l'alignement de séquences ADN. L'originalité de cette méthode est la satisfaction de la contrainte de similarité, et l'adaptation de cette nouvelle contrainte aux algorithmes classiques. Pour ce faire, les auteurs proposent de relacher la contrainte de similarité afin de la convertir en une contrainte anti-monotone. [CMB02], démontre ainsi que les mesures de similarités basées sur la distance d'édition (mesure du nombre d'insertions, suppressions et remplacements nécessaires afin de transformer une séquence en une autre) peuvent être converties en contraintes anti-monotones.

[EPF05] utilise les motifs séquentiels afin de construire un classifieur de protéines. La méthode se déroule en trois étapes : dans un premier temps, les protéines du jeu d'entraînement sont classés selon leurs familles. L'information concernant l'appartenance d'une protéine à une famille est connue par avance. Ensuite, les motifs séquentiels sont extraits pour chaque ensemble de protéines. Dans ce contexte, chaque protéine est un objet, la dimension d'ordre est la position de l'acide aminé sur la protéine et un acide aminé est un item. Ainsi, les motifs séquentiels extraits ne contiennent qu'un item par itemset, une protéine ne pouvant contenir à la fois qu'un acide aminé par position. Enfin, la troisième étape consiste à "scorer" chaque protéine : si elle contient un motif extrait, alors son score est augmenté par le ratio de la longueur du motif sur le nombre de motifs par famille (formule 1). Cela permet d'affecter un poids à chaque famille, afin de construire une matrice de confusion, qui servira de

classifieur.

Malgré l'originalité de la méthode, les résultats obtenus lors d'expérimentations sur jeu de données réelles ne sont pas concluants : seulement 40% de protéines sont classées correctement. Plusieurs facteurs expliquent ce résultat, tels que le support minimum utilisé pour extraire les motifs, ou encore la mesure de score.

Les motifs séquentiels sont une forme évoluée des règles d'associations, puisqu'ils permettent d'extraire des connaissances contenant plusieurs niveaux d'information. Cependant, si l'application de règles d'association est évidente dans le domaine médical, il n'en va pas de même pour les motifs séquentiels, puisque ceux-ci imposent la prise en compte d'une dimension d'ordre. Pour les bases structurelles telles que les protéines, une première approche peut être naïvement réalisée en considérant la position de l'acide aminé sur une protéine, comme le proposent [EPF05]. Cette approche mène alors à l'extraction de séquences ne contenant qu'un item par itemset. Or, il serait intéressant de définir de nouvelles contraintes de temps de type windowsize afin de regrouper des morceaux de séquence particuliers. Cela permettrait d'incorporer des connaissances supplémentaires et donc d'extraire des informations utiles aux biologistes.

## 2.6 Vers la prise en compte des données numériques

Dans ce chapitre, nous avons présenté un ensemble de méthodes relativement récentes d'extraction de motifs appliquées au contexte médical. Cependant, aucune ne propose d'intégrer des données patients au sein de l'algorithme afin d'extraire des corrélations mélangeant niveau micro et niveau macro. Cependant, ces connaissances s'avèrent primordiales, et plus particulièrement dans le cas de l'étude de maladies. En effet, les chercheurs cherchent bien souvent à expliquer les différences entre séquences micro (ADN, gène ou protéine) en fontion d'une caractéristique clinique. Par exemple, il est fréquent de différencier les motifs obtenus pour les différents grades de tumeurs.

D'autre part, les motifs séquentiels pourraient être la base de plusieurs types d'applications, comme par exemple la recherche de gradualité, ou encore l'évolution des maladies. Cependant, la tâche de l'extraction reste difficile : les données sont denses, et souvent sous un format peu exploré : un nombre d'objets est proportionnellement faible par rapport au nombre d'attributs. Des propositions telles que la transposition des données sont actuellement difficilement envisageables, puisque nous ne sommes pas dans un contexte bi-dimensionnel. En effet, les motifs séquentiels contiennent une troisième dimension : l'ordre.

Suite à ces différents constats, notre objectif est double. Tout d'abord, il s'agira de ne pas perdre les informations numériques fournies par les bases, telles que les valeurs d'expressions, ou encore les données cliniques (grade d'une tumeur, nombre de récidives...). Nous pensons que les motifs apportés par la fouille ne doivent pas complètement segmenter les données. C'est pour cela que nous proposons d'extraire des *motifs graduels*. De plus, ces motifs sont suffisamment génériques pour être utilisés dans de nombreux contextes : base d'expression de gène, base de protéines, ou même base de questionnaires médicaux.

Enfin, il s'agit de mélanger données cliniques et données biologiques. Là encore, les motifs graduels

sont adaptés, du moment que les données cliniques sont quantifiées. Cela permet alors de caractériser les motifs en fonction de ces données. Par exemple, nous pourront extraire des règles de la forme “plus le gène Z s'exprime et moins le gène W s'exprime, alors plus le grade de la tumeur augmente”.

# Chapitre 3

# Gradualité et bases de données relationnelles

## 3.1 Introduction

L'objectif de ce chapitre est d'introduire la notion de règle d'association graduelle. Dans un premier temps, nous dressons un panorama des travaux associés (section 3.2). Nous présentons ensuite deux approches : une heuristique, qui est la première méthode que nous avons proposée dans le cadre de cette thèse (section 3.3), puis une méthode complète (section 3.4). Pour chacune de ces approches, nous détaillons pas à pas les algorithmes implémentés et effectuons les expérimentations nécessaires à leur évaluation. Nous comparons également ces deux méthodes, et en pointons les points forts et les points faibles. Ce chapitre se termine sur une discussion.

## 3.2 État de l'art

### 3.2.1 Modélisation et utilisation de motifs graduels

Pendant de nombreuses années, la notion de gradualité a été principalement abordée dans le contexte de la logique floue. Formalisée en 1965 par Lotfi Zadeh, la logique floue est une généralisation de la logique binaire (ou logique booléenne) visant à modéliser les concepts plus ou moins vagues du langage humain. Dans le cadre de la théorie classique des ensembles, un élément appartient ou n'appartient pas à un ensemble. Le principe de la logique floue est de relâcher ces contraintes, et donc de permettre à un élément d'appartenir à la fois à un ensemble et à son complément. Pour ce faire, l'univers est découpé (de manière automatique ou manuelle) en sous-ensembles appelés *sous-ensembles flous*. Par exemple, les expressions de gènes sont fréquemment associées à trois ensembles : sous-exprimé, normalement exprimé et sur-exprimé. Dans le cas de la théorie classique des ensembles, nous obtenons par exemple la base du tableau 3.1.

| Gène | Expression | Sous-Exprimé | Normalement Exprimé | Sur-Exprimé |
|---|---|---|---|---|
| $g_1$ | -1.5 | 1 | 0 | 0 |
| $g_2$ | 0.6 | 0 | 1 | 0 |
| $g_3$ | 2.2 | 0 | 1 | 0 |
| $g_4$ | 2.5 | 0 | 0 | 1 |

Tab 3.1 – Base de données exemple : valeur d'expression de quatre gènes

Dans notre exemple, nous considérons qu'un gène commence à être normalement exprimé à partir de 0.3 et qu'il commence à être sur-exprimé à partir de 1.7. Cela nous conduit à remplacer la base 3.1 par la base du tableau 3.2.

En logique floue, chacun de ces trois sous-ensembles d'expressions peut être décrit à l'aide d'une *fonction d'appartenance* (généralement notée $\mu$) telle que l'une de celle décrite à la figure 3.1. Dans ce cas, le gène $g_2$, exprimé à 0.6, appartient au sous-ensemble flou "sous-exprimé" avec un degré de 0.3 et au sous-ensemble flou "normalement exprimé" avec un degré de 0.7. Ce fait peut être sémantiquement décrit par "le gène $g_2$ se sous-exprime faiblement, et il s'exprime presqu'entièrement normalement".

Ainsi, en logique floue, un élément peut *graduellement* appartenir à plusieurs ensembles. Partant de ce constat, [DP08] définit les *éléments graduels*, éléments dont le choix dépend d'un paramètre et

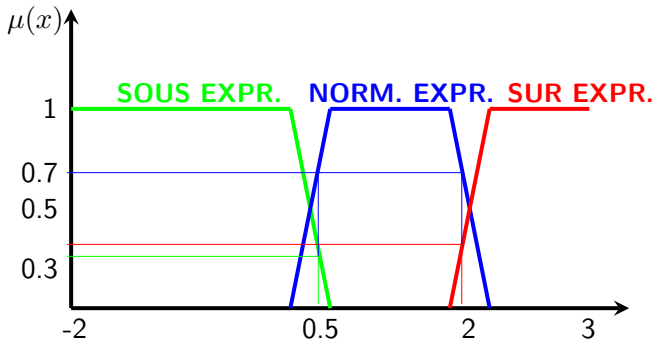



Fig 3.1 – Sous-ensembles flous définis pour l'expression

| Gène | Expression | Sous-Exprimé | Normalement Exprimé | Sur-Exprimé |
|---|---|---|---|---|
| $g_1$ | -1.5 | 1 | 0 | 0 |
| $g_2$ | 0.6 | 0.3 | 0.7 | 0 |
| $g_3$ | 2.2 | 0 | 0.6 | 0.4 |
| $g_4$ | 2.5 | 0 | 0 | 1 |

Tab 3.2 – Expression des gènes et leur degrés d'appartenance

permettent de décrire des concepts.

Cependant, les éléments graduels en eux-mêmes ne suffisent pas à modéliser les co-variations entre ensembles. Ainsi, de nombreux travaux s'orientent vers la notion de **règle graduelle**. Dans ce contexte, la gradualité n'est plus placée sur l'élément, mais sur l'*implication* entre ensembles d'éléments. Dans la littérature, différentes formes de règles graduelles sont discernées :

- Les **règles floues graduelles** sont de la forme "*plus* X *est* A, alors *plus* Y *est* B"
- Les **règles floues de certitude** sont de la forme "plus X est A, alors *plus il est certain* que Y est B"

Dans [DCL93], les auteurs désignent les règles graduelles sous le terme de "*topoi*" et "*règles d'inférences graduelles*". Les auteurs démontrent la présence de gradualité dans de nombreux contextes tels que l'extraction de connaissances ou encore le raisonnement automatique, et proposent deux solutions pour l'acquisition de ces règles. La première proposition consiste à étudier le discours sémantique de l'expert. Les auteurs notent que des termes précis identifient une inférence graduelle, comme "*d'autant plus*", "*augmente*", "*diminue*", "*plus*", "*au plus*", "*moins*", "*au moins*"... L'ingénieur peut alors manuellement construire les règles graduelles à partir de discussions avec l'expert. La seconde consiste à construire automatiquement des arbres de décisions, puis à s'appuyer sur ces arbres afin de définir manuellement les règles graduelles. Dans [DCL93], aucune formalisation de la gradualité n'est fournie. En revanche, les auteurs étudient de manière remarquable l'impact de l'utilisation de règles graduelles à plusieurs niveaux des méthodes de modélisation des connaissances stratégiques et des connaissances du domaine (KAD et KOD [WSB92]).

[BMD90] étudie l'impact des différents opérateurs d'implications sur les règles graduelles. En logique floue, l'implication d'une règle floue $R: X \Rightarrow Y$ est définie à l'aide d'un opérateur d'implication souvent associé à une t-norme. Une t-norme est notée $\top$. Il existe de nombreuses t-normes, les plus connues étant listées dans le tableau 3.3. [BMD90] note que l'opérateur d'implication utilisé conditionne l'information graduelle véhiculée par la règle. Par exemple, pour la règle floue graduelle $R_1$ : "*moins le vent est fort, moins la mer est houleuse*", l'utilisation de l'implication floue $R^L$ affaiblit la liaison graduelle, et transforme la règle en "*il est presque certain que moins le vent est fort, moins la mer est houleuse*". En revanche, l'utilisation de l'implication floue $R^{RG}$ vient renforcer la liaison graduelle de la règle $R_1$. Les auteurs étudient ainsi les cas de renforcement de la prémisse et de la conséquence de la règle, ainsi que la notion de certitude apportée par ces différentes implications floues.

| | **Nom** | $\top$ |
|---|---|---|
| $R^Z$ | Zadeh | $\min(x, y)$ |
| $R^P$ | Probabiliste | $xy$ |
| $R^L$ | Lukasiewicz | $\max(x + y - 1, 0)$ |
| $R^W$ | Weber | $\begin{cases} x & si\ y = 1 \\ y & si\ x = 1 \\ 0 & sinon \end{cases}$ |
| $R^{RG}$ | Rescher-Gaines | $\begin{cases} 1 & si\ x \leq y \\ 0 & sinon \end{cases}$ |

Tab 3.3 – Principales t-normes

[BMD90] mettent en évidence, sans les formaliser, les premières propriétés des règles graduelles, et notamment celle de l'antonymie : il est possible de passer d'une règle à son contraire. Par exemple la règle $R1$ peut être transformée de manière évidente en "*plus le vent est fort, plus la mer est houleuse*". Cependant, les auteurs ne proposent pas de solution d'extraction automatique de telles règles, mais un système d'aide à la formulation de ces règles pour les experts.

Une première formalisation des règles d'inférences graduelles est proposée par [DP92]. Il s'agit ici de définir un contexte formel de règles floues graduelles de la forme "plus $x$ est $A$, alors plus $y$ est $B$" en vue de leur traitement automatique dans le cadre d'un système de raisonnement.

Dans les années qui suivirent l'apparition des règles graduelles, celles-ci furent principalement utilisées comme connaissances dans de nombreux domaines tels que la modélisation de systèmes (contrôleurs flous)[DPG95, DR04], le domaine du traitement de l'imprécision [DGS02, GDP03, GDP04], le domaine du traitement de la langue [BLP97, BLP98, BPU99], ou encore en médecine [iMAS$^+$01, APS07]. Récemment, les règles graduelles ont été utilisées afin de construire des règles de classification [CDL$^+$09, Dâr10]. On note également une proposition d'extraction de règles graduelles closes [ABLP10].

Selon la complexité du système que l'on veut modéliser et la taille de la base de connaissances disponible à priori, il peut s'avérer extrêmement coûteux et source d'erreurs d'extraire les règles graduelles manuellement. De plus, depuis les premières modélisations de ces règles, les technologies de l'informa-

tion ont grandement évolué. Il est désormais possible pour les experts d'acquérir automatiquement les connaissances de leur domaine par l'intermédiaire de relevés de capteurs ou de techniques de numérisation et de les stocker au sein de bases de données. Dans la plupart des cas, ces bases sont denses : il y a peu de valeurs manquantes. Ainsi, à partir des années 2000, la recherche sur les règles graduelles s'oriente non plus sur comment les modéliser (leur modélisation est consensuelle en logique floue), mais sur les méthodes d'extraction automatique de telles règles.

### 3.2.2 Motifs graduels et fouille de données

A notre connaissance, la première proposition d'une méthode d'extraction automatique de règles floues graduelles revient à [Hül02]. Deux types de règles sont clairement identifiés :

- les **règles dérivables** (notées $A \rightarrow^d B$) expriment un écart significatif de la moyenne conditionnelle
- les **règles graduelles** (notées $A \rightarrow^t B$) expriment la dépendance graduelle entre deux itemsets $A$ et $B$.

| | $B(y) = 0$ | $B(Y) = 1$ | |
|---|---|---|---|
| $A(x) = 0$ | $n_{00}$ | $n_{01}$ | $n_{0\bullet}$ |
| $A(x) = 1$ | $n_{10}$ | $n_{11}$ | $n_{1\bullet}$ |
| | $n_{\bullet 0}$ | $n_{\bullet 1}$ | $n$ |

Tab 3.4 – Table de contingence pour la règle classique $A \Rightarrow B$

La génération de telles règles est réalisée en trois étapes :

1. *Construction d'un diagramme de contingence*. Dans le cas de règles d'associations classiques, les mesures de fréquence et confiance peuvent être obtenues à partir de tables de contingence comme celle du tableau 3.4. Cependant, de telles tables ne peuvent être utilisées avec des sous-ensembles flous, car chaque case est valuée. [Hül02] propose l'utilisation de diagrammes de contingence, qui sont des représentations des objets de la base en $n$ dimensions, $n$ étant la longueur de l'itemset graduel. La figure 3.2 montre un diagramme de contingence pour un itemset graduel de longueur 2.
2. *Génération d'informations sur le diagramme de contingence*. Pour ce faire, [Hül02] utilise une régression linéaire (la méthode des moindres carrés est utilisée, mais une autre méthode est tout à fait appliquable). Les coefficients $\alpha$ et $\beta$ permettent de mesurer le "degré de variation" entre les items composant l'itemset. La qualité de la régression est également prise en compte via l'utilisation du coefficient de corrélation $Q$ ($R^2$).
3. *Génération des règles floues graduelles*. Selon des seuils minimaux $Q_{\min}$ et $\alpha_{\min}$ fixés par l'utilisateur. Si les coefficients satisfont ces seuils, alors la règle $A \rightarrow^t B[\alpha, \beta]$ est générée. Notons que les règles dont la longueur est supérieure à 2 ne contiennent qu'un item dans la conséquence.

La première utilisation de la fouille de données afin d'extraire des règles floues graduelles est proposée par [BCS+07]. Les auteurs utilisent l'algorithme Apriori [AS94] et posent ainsi la première définition d'**item graduel** :

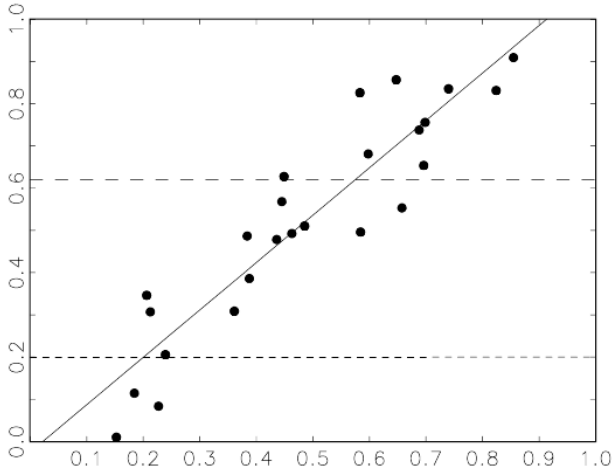



Fig 3.2 – Exemple de diagramme de contingence à deux dimensions extrait de [Hül02]

**Définition 2.** *Un item graduel est un triplet de la forme* $[*, X, A]$ *avec*

- $* \in \{<, >\}$
- *X un item*
- *A un sous-ensemble flou défini sur un item*

Dans cette définition, la notion de gradualité est réalisée au travers de l'utilisation des opérateurs de comparaison $\{<, >\}$. Cette définition implique la comparaison directe des degrés d'appartenance des objets, ce qui diffère totalement des règles d'associations classiques où un seul objet suffit à augmenter la fréquence. Un item graduel n'a alors de sens que si au moins deux objets de la base l'instancient. Par exemple, en reprenant l'exemple du tableau 3.2, six items graduels peuvent être générés : $[<$ , Gène, Sous-Exprimé] pour "moins le gène est sous-exprimé", $[<$, Gène, Normalement Exprimé] , $[<$, Gène, Sur-Exprimé], et $[>$, Gène, Sous-Exprimé] pour "plus le gène est sous-exprimé", etc...

La notion de support d'une règle graduelle est pour la première fois introduite. Dans le cas des itemsets classiques, la fréquence est la proportion d'objets de la base contenant l'itemset sur le nombre total d'objets de la base, alors que dans le cas de la gradualité, c'est la co-variation qui est mesurée. En réalité, cela revient à ordonner les valeurs de la base en fonction de l'itemset graduel, ce qui se traduit dans [BCS$^+$07] par le calcul de tous les couples d'objets de la base. Plus formellement [BCS$^+$07], définit la fréquence de la manière suivante :

**Définition 3.** *Soit* $(*, X, A)$ *un item graduel et* $\mathcal{D}$ *une base de données, et* $GT^{\mathcal{D}}$ *tel que* $\forall o = (x, y), o' = (x', y') \in \mathcal{D}, gt_{oo'} \in GT^{\mathcal{D}}$ *si* $A(x) * A(x')$. *Alors* $Freq([*, X, A]) = \frac{|\{gt_{oo'} \in GT^{\mathcal{D}} | A(x) * A(x')\}|}{|GT^{\mathcal{D}}|}$

Par exemple, pour l'item graduel $R_1 : [>$, Gène, Sous-Exprimé], la base $gt_{oo'}$ construite est représentée par le tableau 3.5a. Le support de cet item graduel sera alors $\frac{5}{12} \simeq 0.41$. De la même manière, le tableau 3.5b montre la base $gt_{oo'}$ pour $R_2 : [<$, Gène, Sous-Exprimé].

[BCS$^+$07] formalisent également la propriété de complémentarité, désignée par le terme "antonymie" par [BMD90] :

| Couples de gènes (x,y) | [Ss-Expr.(x), Ss-Expr.(y)] |
|---|---|
| $(g_1, g_2)$ | (1, 0.3) |
| $(g_1, g_3)$ | (1, 0) |
| $(g_1, g_4)$ | (1, 0) |
| $(g_2, g_3)$ | (0.3, 0) |
| $(g_2, g_4)$ | (0.3, 0) |

(a)

| Couples de gènes (x,y) | [Ss-Expr.(x), Ss-Expr.(y)] |
|---|---|
| $(g_2, g_1)$ | (0.3, 1) |
| $(g_3, g_1)$ | (0, 1) |
| $(g_3, g_2)$ | (0, 0.3) |
| $(g_4, g_1)$ | (0, 1) |
| $(g_4, g_2)$ | (0, 0.3) |

(b)

Tab 3.5 – Bases $gt_{oo'}$ pour l'item graduel (a) $[>, \text{Gène}, \text{Sous-Exprimé}]$ et (b) $[<, \text{Gène}, \text{Sous-Exprimé}]$

**Propriété 2.** *Soit $c$ un opérateur de $\{<, >\}$ tel que $c(>) =<$ et $c(<) =>$. Alors $Supp([*, X, A]) = Supp[c(*), X, A]$*

La propriété 2 permet de diviser par deux l'espace de recherche, car une moitié des règles graduelles peut être automatiquement déduite à partir de l'autre moitié.

[BCS+07] utilise Apriori afin d'extraire l'ensemble des itemsets graduels. Le calcul du support impose la construction de la base des couples $GT^{\mathcal{D}}$ soit pour chaque itemset (chaque noeud de l'arbre des préfixes), soit dans sa totalité pour être simplement scannée à chaque passe. Cependant, l'utilisation de cette base rend la complexité de l'algorithme trop élevée pour être appliqué. Les auteurs proposent donc de partitionner les sous-ensembles flous en $k$ partitions equi-depth [1], avec $k$ fixé par l'utilisateur. Cela permet d'associer à chaque itemset un vecteur de longueur $k+1$, dont chaque indice contient le nombre d'objets de la base tel que $A(x) = j/k$. Ainsi, la complexité du calcul de la fréquence d'un itemset de longueur $p$ passe de $\mathcal{O}(n^2)$ à $\mathcal{O}(n + k^p)$.

Dans [FMLT08a, FMLT08b] les auteurs proposent d'extraire des motifs séquentiels graduels. Rappelons qu'un motif séquentiel, contrairement aux règles d'associations, permet de mesurer la fréquence de certains comportements *dans le temps*. La notion de gradualité peut alors s'appliquer à deux niveaux :

- comme précédemment, au niveau des items. Cela traduit une co-variation dans le même sens entre plusieurs items ;
- au niveau de la liaison entre les itemsets. Cela introduit les notions de "puis rapidement", "longtemps après"...

Les motifs séquentiels flous graduels obtenus sont de la forme "Plus (moins) $X_{11}$ est $A_{11}$ et plus (moins) $X_{12}$ est $A_{12}$ précède une longue (courte) période de plus (moins) $X_{23}$ est $A_{23}$ ... précède une longue (courte) période de plus (moins) $X_{ij}$ est $A_{ij}$". Dans ce contexte, l'espace de recherche est infini. Ainsi, [FMLT08b] procède en trois étapes :

- *Construction d'une base* avec sous-ensembles flous et explicitant les variations entre toutes les dates. Par exemple, pour un objet ayant des valeurs renseignées aux dates $d_1$, $d_2$ et $d_3$, la base contiendra les différences de valeur entre $(d_1, d_2)$, $(d_1, d_3)$ ainsi que $(d_2, d_3)$. Cette base est appelée *base de tendance* et est construite à l'aide de l'algorithme `TED`.

1. les partitions equi-depth contiennent le même nombre d'objets

- *Fouille de données*, avec deux étapes distinctes afin de contourner le problème de l'espace de recherche exponentiel :
  - *Construction d'un graphe des séquences* sur le modèle de [MPT04]. Chaque entrée de la base de tendance constitue un noeud, et un arc est présent si pour un même objet, une date est inférieure à une autre. Un tel graphe permet par la suite de traiter les séquences se chevauchant temporellement.
  - *Les motifs séquentiels graduels sont extraits* à partir de ce graphe en utilisant l'algorithme d'extraction de motifs séquentiels flous TotallyFuzzy [FLT07].

Avec cette méthode, les auteurs extraient des motifs flous graduels tels que "*une augmentation lente du nombre de connexions à la page KBM précède une longue période d'augmentation de connexions à KOML, après une courte période. Puis vient une augmentation lente du nombre de connexions à DJAVA.*"

### 3.2.3 Discussion

Le tableau 3.6 synthétise les travaux précédemment cités selon différents critères :

- Les méthodes utilisent-elles des sous-ensembles flous afin de gérer les valeurs numériques ?
- Les méthodes utilisent-elles un opérateur d'implication floue afin d'évaluer des règles n-aires ?
- Les méthodes tirent-elles parti de la propriété de complémentarité (antonymie) de la gradualité ?
- Les méthodes permettent-elles d'extraire des règles de longueur n, et mélangeant des variations ascendantes et descendantes ?
- Les méthodes utilisent-elles des méthodes de fouille de données ?

| **Référence** | Sous-ensembles flous | Opérateurs d'implication floue | Complémentarité | Règles de variation composées | Fouille de données |
|---|---|---|---|---|---|
| [DCL93] | ✗ | | | ✗ | |
| [BMD90] | ✗ | ✗ | ✗ | | |
| [Hül02] | ✗ | ✗ | | ✗ | ✗ |
| [BCS+07] | ✗ | | ✗ | ✗ | ✗ |
| [FMLT08a] | ✗ | | | ✗ | ✗ |

Tab 3.6 – Synthèse des travaux existant

Toutes les approches présentées utilisent des sous-ensembles flous. Nous pouvons l'expliquer par le fait que très tôt dans la discipline, Zadeh constate que "la transition d'un état à un autre ne se fait pas de manière abrupte, mais graduellement". Dans un premier temps, les travaux vont donc dans le sens de la modélisation [DP08] et de l'utilisation de règles graduelles [DCL93, BMD90].

La possibilité d'utiliser des modificateurs sémantiques est également l'une des idées fondatrices de la logique floue, et l'une des raisons de son utilisation dans le cadre de la gradualité. En effet, afin de modéliser un comportement à partir des paroles d'un expert, il est appréciable de pouvoir juger que la

gradualité augmente "*rapidement*" ou encore que "le vent est *plutôt* houleux".

Cependant, l'utilisation de la logique floue peut poser deux problèmes. Tout d'abord, dans le cadre de la fouille de données, les sous-ensembles flous démultiplient le nombre d'items à considérer. En effet, chaque sous-ensemble flou est considéré comme un item à part entière. Par exemple, plutôt qu'avoir un item "expression du gène", nous aurons les trois items "sous-exprimé", "normalement exprimé" et "sur-exprimé". Plus formellement, soit $X = \{X_1, X_2, ..., X_m\}$ l'ensemble des attributs de $\mathcal{D}$, et pour chaque attribut $X_i \in X$ les ensembles $n_i$ des restrictions floues définies par les ensembles flous $A_{i1}, ..., A_{in_i}$. Alors la cardinalité de l'ensemble des items graduels potentiels $GI^{\mathcal{D}}$ est :

$$|GI^{\mathcal{D}}| = 2 \times \sum_{i=0}^{m} n_i$$

Cela influe donc directement sur la complexité des algorithmes proposés et ralentit en conséquence le processus d'extraction automatique. Ensuite, il s'avère difficile d'utiliser les opérateurs d'implication floue dans le cadre de règles contenant plusieurs items dans la condition de la règle [JDGC07]. D'ailleurs, aucune méthode utilisant un opérateur d'implication floue ne permet d'extraire des règles avec plusieurs items dans la conséquence.

Toutes les propositions ayant des conditions composées utilisent l'opérateur d'implication floue de Rescher-Gaines (voir tableau 3.3). Cependant, cet opérateur est très restrictif car la valeur du degré d'appartenance de $x$ est contrainte par la valeur du degré d'appartenance de $y$. Ainsi, une augmentation de la valeur $y$ constitue un relâchement de la contrainte, et permet d'avoir une valeur $x$ plus élevée. Cet opérateur ne mesure donc pas la co-variation de $x$ et $y$ en tant que telle, mais plutôt autorise $x$ à augmenter lorsque $y$ augmente.

Afin d'automatiser le processus d'extraction, [Hül02] utilise des outils de statistique tels que la régression linéaire. Cependant, cette méthode oblige l'utilisateur à présupposer la forme de la courbe afin d'appliquer la régression linéaire la plus adaptée. De plus, le coût d'une régression linéaire et du coefficient de détermination à chaque calcul de fréquence pose le problème du passage à l'échelle. Enfin, l'utilisation d'Apriori dans ce contexte nécessite deux lectures complètes de la base de données pour chaque itemset généré. Plus de détails sur les problèmes de conception algorithmique de la méthode sont fournis dans [Hül07].

Les autres approches d'extraction automatique citées n'utilisent pas d'outils statistiques. En revanche, elles nécessitent une reconstruction de la base de données afin de générer les couples d'objets dans le cas de [BCS+07] et les couples de date dans le cas de [FMLT08a]. Afin de pallier le problème de complexité mémoire, les deux méthodes proposent des optimisations précises et adaptées à une partie des cas. Par exemple, selon la base de données, il ne sera pas intéressant de considérer le même nombre de sous-ensembles flous pour chaque attribut. Cependant, il est important de noter que ces deux dernières méthodes sont les seules permettant de mélanger au sein d'un même itemset des variations ascendantes et descendantes, comme par exemple "plus le gène $g_1$ est sur-exprimé et moins le gène $g_2$ est sur-exprimé, alors plus le gène $g_3$ est sur-exprimé". Cela est possible grâce à l'utilisation d'algorithmes de fouille de données, qui permettent de balayer efficacement l'espace de recherche.

L'évolution des technologies de la médecine génère de très grandes quantités de données numériques. Ces bases sont denses et ont la particularité de contenir un très grand nombre d'items comparé

au nombre de patients. De plus, dans la plupart des cas, les bornes des domaines de valeurs ne justifient pas forcement l'utilisation de sous-ensembles flous. Cependant, il s'avère que la gradualité est particulièrement adaptée aux types de connaissances que les experts cherchent à obtenir à partir de ce type de bases. Néanmoins, les outils actuels d'extraction automatiques sont trop complexes pour être directement utilisés sur ces bases.

Nous pensons qu'il n'est pas utile de conserver les sous-ensembles flous afin d'extraire des règles graduelles. Nous ne parlerons alors plus de règles floues graduelles, mais simplement de règles graduelles. Dans les sections suivantes, nous détaillons deux méthodes d'extraction automatique de règles graduelles (non floue) : une heuristique qui n'est pas complète, puis une méthode exhaustive qui présente l'avantage de la complétude.

## 3.3 Une heuristique pour l'extraction d'itemsets graduels

Nous présentons ci-dessous l'une de nos contributions fondée sur l'utilisation d'une heuristique pour l'extraction d'itemsets graduels.

### 3.3.1 Définitions

Dans ce chapitre, nous considérons une base de données transactionnelle $\mathcal{DB}$ contenant un ensemble d'objets $\mathcal{O}$ et un ensemble d'items $\mathcal{I}$. Chaque ligne représente une transaction $t$ pour un objet $o$ et nous désignons la valeur associée à l'item $i$ pour l'objet $o$ $t_o[i]$. Afin d'illustrer notre propos, nous utiliserons la base 3.7, qui décrit huit patients atteints de cancer. Pour chacun, nous connaissons la valeur d'expression des gènes $g_1$ et $g_2$, compris entre $[-2, 3]$, ainsi que le grade de leur tumeur (allant de 1 à 3).

| **Patient (P)** | **Gène 1 ($g_1$)** | **Gène 2 ($g_2$)** | **Grade Tumeur (G)** |
|---|---|---|---|
| $p_1$ | -2 | -1.5 | 2 |
| $p_2$ | -0.5 | -1 | 2 |
| $p_3$ | -1 | -1.5 | 1 |
| $p_4$ | 0.5 | -0.5 | 2 |
| $p_5$ | 1 | -0.8 | 2 |
| $p_6$ | 1.5 | 1 | 2 |
| $p_7$ | 3 | 1 | 3 |
| $p_8$ | 1.2 | 3 | 3 |

Tab 3.7 – Base $\mathcal{DB}$ décrivant les expressions des gènes $g_1$ et $g_2$ pour 8 patients

Nous pensons que les algorithmes d'extraction de règles d'association classiques peuvent être utilisés dans le cadre de l'extraction de règles graduelles. Il existe deux paradigmes pour l'extraction de telles règles : *pattern-growth* et *générer-élaguer*. Dans le présent chapitre, nous adoptons une approche "générer-élaguer" et utilisons un algorithme de génération basé sur Apriori. Cependant, cela nécessite la définition d'item et d'itemset graduels. Nous considérons deux types de variations pour un item $i$ :

- D'un objet à l'autre, la valeur *augmente*. Dans ce cas, nous avons un item graduel qui est sémantiquement identifié comme "plus $i$". Nous noterons cet item graduel $i^{\geq}$ et utiliserons l'opérateur de

comparaison $\geq$.

- D'un objet à l'autre, la valeur *diminue*. Dans ce cas, nous avons un item graduel qui est sémantiquement identifié comme "moins $i$". Nous noterons cet item graduel $i^{\leq}$ et utiliserons l'opérateur de comparaison $\leq$.

Un item graduel est formellement défini de la manière suivante :

**Définition 4.** *(item graduel) Soit $i \in \mathcal{I}$ un item et $* \in \{\geq, \leq\}$ un opérateur de comparaison. L'item graduel $i^*$ est défini comme un couple associant l'item $i$ à l'opérateur $*$.*

**Exemple 6.** *A partir du tableau 3.7, nous pouvons extraire six items graduels : $g_1^{\geq}$, $g_1^{\leq}$, $g_2^{\geq}$, $g_2^{\leq}$, $G^{\geq}$ et $G^{\leq}$.*

La définition d'un itemset graduel est alors :

**Définition 5.** *(itemset graduel) Un itemset graduel est une liste non vide d'items graduels. Il est noté $s = (i_1^{*_1}, ..., i_k^{*_k})$. Un $k$-itemset est un itemset de longueur $k$, c'est-à-dire contenant $k$ items graduels.*

**Exemple 7.** *A partir du tableau 3.7, nous pouvons extraire les itemsets graduels suivants : ($g_1^{\geq}$ $g_2^{\geq}$ $G^{\geq}$), ($g_1^{\geq}$ $g_2^{\geq}$ $G^{\leq}$), ($g_1^{\geq}$ $g_2^{\leq}$ $G^{\geq}$), ($g_2^{\geq}$ $G^{\geq}$), ($g_2^{\geq}$ $G^{\leq}$), etc...*

Dans ce chapitre, nous utilisons les opérateurs de comparaison $\{\geq, \leq\}$. Dans le cas des itemsets classiques, un ensemble d'objets respectant cet itemset est construit. Dans le contexte de la gradualité, c'est un ensemble d'objets ordonné selon les co-variations de l'itemset graduel qui doit être construit. Cela passe par la comparaison des valeurs entre les différentes transactions. Nous choisissons ici de conserver les transactions dont les valeurs sont égales afin de ne pas construire, comme [BCS+07, FMLT08a] une base dérivée de $|\mathcal{O}|.(|\mathcal{O}| - 1)$ transactions. Cela nous évite ainsi une complexité mémoire trop élevée. Le chapitre 4 est consacré à la prise en compte de l'égalité.

Nous formalisons l'ordre entre deux objets de la manière suivante :

**Définition 6.** *(Ordre entre deux objets) Soit $X$ et $X'$ deux objets de $\mathcal{DB}$, et $s = (i_1^{*_1}, ..., i_k^{*_k})$ un itemset graduel. $X$ précède $X'$ si $\forall l \in [1, k]$ $X[i_l] *_l X'[i_l]$. Nous notons cette relation d'ordre $X \lhd_s X'$.*

**Définition 7.** *$X$ et $X'$ sont comparables selon l'itemset $s$ si $X \lhd_s X'$ ou $X' \lhd_s X$. Sinon, ils sont dits incomparables.*

**Exemple 8.** *Considérons l'itemset graduel $s_1 = (g_1^{\geq}\ g_2^{\geq})$ pour "plus l'expression du gène $g_1$ augmente, plus l'expression du gène $g_2$ augmente". Nous avons $p_1 \lhd_{s_1} p_3$, car $t_{p_1}[g_1] \leq t_{p_3}[g_1]$ et $t_{p_1}[g_2] \leq t_{p_3}[g_2]$. $p_1$ et $p_3$ sont donc comparables. En revanche, $p_4$ et $p_5$ sont incomparables car $t_{p_4}[g_1] > t_{p_5}[g_1]$.*

**Définition 8.** *(Liste ordonnée d'objets) Soit $s = (i_1^{*_1}, ..., i_k^{*_k})$ un itemset graduel. Une liste d'objets $\mathcal{L} =<_L X_1, ..., X_n >_L$ respecte $s$ si $\forall p \in [1, n-1], \forall l \in [1, k]$ $X_p[i_l] *_l X_{p+1}[i_l]$.*

**Propriété 3.** *Il peut exister plus d'une liste ordonnée d'objets respectant $s$.*

Par exemple, nous savons que $p_4$ et $p_5$ sont incomparables. Cela va créer au moins deux listes : l'une contenant $p_4$ mais pas $p_5$ et l'autre contenant $p_5$ mais pas $p_4$. Il existe en tout quatre listes ordonnées respectant l'itemset représentées par les tableaux 3.8a, 3.8b, 3.8c et 3.8d :

| P | $g_1$ | $g_2$ |
|---|---|---|
| $p_1$ | -2 | -1.5 |
| $p_3$ | -1 | -1.5 |
| $p_2$ | -0.5 | -1 |
| $p_4$ | 0.5 | -0.5 |
| $p_6$ | 1.5 | 1 |
| $p_7$ | 3 | 1 |

(a)

| P | $g_1$ | $g_2$ |
|---|---|---|
| $p_1$ | -2 | -1.5 |
| $p_3$ | -1 | -1.5 |
| $p_2$ | -0.5 | -1 |
| $p_4$ | 0.5 | -0.5 |
| $p_8$ | 1.2 | 3 |

(b)

| P | $g_1$ | $g_2$ |
|---|---|---|
| $p_1$ | -2 | -1.5 |
| $p_3$ | -1 | -1.5 |
| $p_2$ | -0.5 | -1 |
| $p_5$ | 1 | -0.8 |
| $p_6$ | 1.5 | 1 |
| $p_7$ | 3 | 1 |

(c)

| P | $g_1$ | $g_2$ |
|---|---|---|
| $p_1$ | -2 | -1.5 |
| $p_3$ | -1 | -1.5 |
| $p_2$ | -0.5 | -1 |
| $p_5$ | 1 | -0.8 |
| $p_8$ | 1.2 | 3 |

(d)

Tab 3.8 – Différentes listes obtenues pour $(g_1^{\geq}\ g_2^{\geq})$ : (a) $\mathcal{L}_1$, (b) $\mathcal{L}_2$, (c) $\mathcal{L}_3$, (d) $\mathcal{L}_4$

La fréquence d'un itemset graduel est calculée à partir de toutes ces listes. Nous notons $G_s^{\mathcal{D}}$ l'ensemble des listes ordonnées de tuples respectant l'itemset $s$. Parmi toutes ces listes, certaines contiennent plus d'objets que d'autres. Dans ce chapitre, nous avons une vision optimiste de la fréquence, qui consiste à considérer la plus grande liste pour le calcul de la fréquence. La fréquence d'un itemset graduel est défini de la manière suivante :

**Définition 9.** *Soit $s = (i_1^{*1} i_2^{*2} ... i_n^{*n})$ un itemset graduel et $G_s^{\mathcal{D}}$ l'ensemble de toutes les listes ordonnées d'objets respectant $s$. La fréquence de $s$ est :*

$$Freq(s) = \frac{\max(|\mathcal{L} \in G_s^{\mathcal{D}}|)}{|\mathcal{O}|}$$

**Exemple 9.** *Calculons la fréquence de l'itemset graduel $(g_1^{\geq}\ g_2^{\geq})$. Nous avons $\max(|\mathcal{L} \in G_s^{i\mathcal{D}}|) = 6$, car les listes $\mathcal{L}_1$ et $\mathcal{L}_4$, de tailles égales, contiennent six objets. Ainsi, $Freq(g_1^{\geq}\ g_2^{\geq}) = \frac{6}{8} = 0.75$, ce qui signifie que 75% des objets de la base ont une co-variation croissante pour l'expression des gènes $g_1$ et $g_2$.*

**Proposition 1.** *(Anti-monotonie des itemsets graduels) i) Soit $s$ et $s'$ deux itemsets graduels, nous avons : $s \subseteq s' \Rightarrow Freq(s) \geq Freq(s')$. ii) Soit $s$ et $s'$ deux itemsets graduels tels que $s \subseteq s'$, alors $s$ n'est pas fréquente implique $s'$ n'est pas fréquente.*

*Démonstration.* Soit deux itemsets graduels $s_k$ et $s_{k+1}$ tels que $s_k \subseteq s_{k+1}$, avec $k$ et $k+1$ la longueur de ces itemsets. Soit $ml$ une liste maximale à partir de $\mathcal{G}_{s_k}$. Cela signifie que $\forall X, X' \in ml$ :

- si $\neg(X \lhd_{s_{k+1}} X')$ alors $Freq(s_k) > Freq(s_{k+1})$
- si $(X \lhd_{s_{k+1}} X')$, alors $Freq(s_k) = Freq(s_{k+1})$

Donc $Freq(s_k) \geq Freq(s_{k+1})$. □

Pour le moment, nous parlons d'itemsets graduels, et non de règles d'association graduelles. La notion d'implication est définie au travers de la définition classique de la confiance de la manière suivante :

**Définition 10.** *Soit $s_1$ et $s_2$ deux itemsets graduels tels que $s_1 \cap s_2 = \emptyset$. Une règle d'association graduelle est de la forme $R : s_1 \Rightarrow s_2$ et deux mesures lui sont associées :*

- **la fréquence** *qui est la fréquence de l'union de la condition et de la conséquence, soit* $Freq(R) = Freq(s_1 \cup s_2)$
- **la confiance** *mesure la probabilité d'avoir* $s_2$ *sachant* $s_1$ *:* $Conf(R) = \dfrac{Freq(s_1 \cup s_2)}{Freq(s_1)}$

Les règles d'associations sont calculées comme post-traitement à partir des itemsets graduels, en suivant l'algorithme de [AS94]. Dans ce chapitre, nous nous concentrons donc uniquement sur l'extraction d'itemsets graduels.

Un itemset graduel est dit fréquent si sa fréquence est supérieure ou égale à un seuil minimal fixé par l'utilisateur. La problématique **d'extraction d'itemsets graduels fréquents** consiste à trouver l'ensemble **complet** de tous les itemsets graduels fréquents à partir d'une base de données $\mathcal{DB}$ contenant des valeurs numériques, selon un seuil de fréquence minimal $minFreq$.

Dans un premier temps, nous nous sommes intéressés au calcul de la liste ordonnée de tuples la plus grande. Nous sommes restés dans le paradigme générer-élaguer, en ayant l'intuition que ces listes maximales restaient les candidates les plus probables lors de la jointure des itemsets par niveau. L'idée ici est de ne conserver qu'une liste, et non l'ensemble de toutes les solutions, afin de permettre le passage à l'échelle. Cependant, lors de la jointure de deux listes candidates, des objets en conflit peuvent apparaître. Nous devons alors choisir entre deux objets celui qui sera conservé pour le calcul de la fréquence. Il s'avère que ce choix n'est pas une tâche triviale, puisqu'il va influer sur les fréquences des itemsets de longueur supérieure. Ainsi, *comment choisir l'objet qui satisfera le plus de contraintes lors des futures jointures ?* Nous répondons à cette question en proposant une heuristique.

### 3.3.2 Algorithmes

Notre proposition est basée sur l'observation suivante : lorsque l'on considère un itemset graduel $s$, certains objets de la base sont en conflit avec d'autres, et les conserver revient à éliminer les autres. Ainsi, il est aisé d'établir pour chaque objet $o$ entrant en conflit la liste des objets qui seront écartés si nous conservons $o$. Nous maintenons ainsi une liste d'**ensembles de conflits**, et basons nos choix sur cette liste. A partir de celle-ci, il sera possible de s'approcher de l'ensemble $\mathcal{L} \in G_s^{i\mathcal{D}}$ maximal.

En faisant ce choix, nous nous apercevons que la génération des itemsets de longueur 2 est un cas légèrement différent, qui se généralisera facilement à la génération des itemsets de longueur $n$. Nous décrivons donc dans un premier temps notre méthode de génération des 2-itemsets graduels.

#### Génération des 2-itemset graduels

Nous définissons un ensemble de conflits pour un 2-itemset de la manière suivante :

**Définition 11.** *Soit* $s = i_1^{*1} i_2^{*2}$ *un 2-itemset graduel, et* $\mathcal{O}$ *l'ensemble des objets de la base* $\mathcal{DB}$ *ordonnés sur* $i_1$ *selon l'opérateur de comparaison* $*_1$ *puis sur* $i_2$ *selon l'opérateur de comparaison* $*_2$*. Pour un objet* $o_i \in \mathcal{O}$*, nous conservons l'ensemble des objets qui contredisent la gradualité de* $s$ *dans un ensemble de conflits noté* $\mathcal{C}_i$*. En d'autres termes,* $\forall o_j \in \mathcal{C}_i, t_{o_i}[i_2] \neg *_2 t_{o_j}[i_2]$.

**Propriété 4.** *La structure des ensembles de conflits est symétrique : si* $o_i \in \mathcal{C}_j$ *alors* $o_j \in \mathcal{C}_i$.

| **Objet** | $g_1$ | $g_2$ | $\mathcal{C}_i$ |
|---|---|---|---|
| $p_1$ | -2 | -1.5 | $\emptyset$ |
| $p_3$ | -1 | -1.5 | $\emptyset$ |
| $p_2$ | -0.5 | -1 | $\emptyset$ |
| $p_4$ | 0.5 | -0.5 | $\{p_5\}$ |
| $p_5$ | 1 | -0.8 | $\{p_4\}$ |
| $p_8$ | 1.2 | 3 | $\{p_6, p_7\}$ |
| $p_6$ | 1.5 | 1 | $\{p_8\}$ |
| $p_7$ | 3 | 1 | $\{p_8\}$ |

Tab 3.9 – Ensembles des conflits pour $(g_1^{\geq}\ g_2^{\geq})$

**Exemple 10.** *Reprenons l'itemset graduel $(g_1^{\geq}\ g_2^{\geq})$. Le tableau 3.9 montre les objets du tableau 3.7 ordonnés sur $g_1$, puis sur $g_2$. La troisième colonne du tableau 3.9 montre les ensembles de conflits associés à chaque objet. Ainsi, les objets $p_1$, $p_2$ et $p_3$ ne posent pas de problème vis à vis de l'ordre imposé par $s$. En revanche, comme souligné dans l'exemple du tableau 3.8, l'objet $p_4$ est en conflit avec l'objet $p_5$. Nous créons donc un ensemble de conflits contenant $p_5$ associé à $p_4$ et un ensemble de conflits contenant $p_4$ associé à $p_5$. De la même manière, $p_8$ est en conflit avec les objets $p_6$ et $p_7$. Cela a pour conséquence la création de trois nouveaux ensembles de conflits.*

Les ensembles de conflits nous permettent de rapidement identifier les objets qui ne posent pas de problème à la gradualité, puisque leur ensemble $\mathcal{C}_i$ associé est vide. Ainsi, ces objets participeront d'office à la fréquence de $s$. En revanche, plus un ensemble de conflits $\mathcal{C}_i$ contiendra d'objets, plus nous devrons écarter d'autres objets afin de conserver $o_i$ lors du comptage de la fréquence. Dans le but de construire l'ensemble d'objets représentant $s$ le plus représentatif, nous écartons dans un premier temps les objets ayant le plus grand ensemble de conflit.

Dans un premier temps, nous conservons donc tous les objets ayant un ensemble de conflits associé vide : $t_0 = G^{\mathcal{D}} \leftarrow f_{emp}(\mathcal{C})$ (où $f_{emp}$ retourne tous les objets ayant un ensemble de conflits vide). Ensuite, nous écartons les objets ayant les plus gros ensembles de conflits à l'aide de la fonction $f_{max}$ : $t_1 = \mathcal{O} \setminus f_{max}(\mathcal{C})$. Effacer un objet de l'ensemble des objets candidats revient à le supprimer de tous les ensembles de conflits dans lequel il apparaissait. La symétrie de notre structure facilite cette tâche, puisqu'il suffit de considérer les objets contenus dans ensemble de conflit que l'on supprime. Ces deux sous-étapes sont notées de la manière suivante :

$$t_{01} = G^{\mathcal{D}} \leftarrow f_{emp}(\mathcal{O} \setminus f_{max}(\mathcal{C}))$$

Ensuite, nous répétons ce processus tant que nous avons des ensembles de conflits non vides. Cela conduit à la proposition suivante :

**Proposition 2.** *La fonction $t_n = G^{\mathcal{D}} \leftarrow t_{n-1}$ avec $t_0 = G^{\mathcal{D}} \leftarrow f_{emp}(\mathcal{O} \setminus f_{max}(\mathcal{C}))$ calcule un ensemble objets représentatif local maximal.*

*Démonstration.* Soit $|G^{\mathcal{D}}| = n$. Supposons qu'il existe un autre ensemble représentatif local $\mathcal{F}$ tel que $|\mathcal{F}| = m | m > n$. Cela signifie qu'il existe un objet $o_i \in \mathcal{F}$ tel que $o_i \notin G^{\mathcal{D}}$. Alors $\mathcal{C}_i = \emptyset$. Cependant, par construction, si $\mathcal{C}_i = \emptyset$, alors $o_i \in G^{\mathcal{D}}$. Il est donc impossible que $o_i$ existe. □

Nous illustrons la proposition 2 en calculant l'ensemble des objets représentatifs pour $(g_1^{\geq}\ g_2^{\geq})$. Dans cet exemple, le plus grand ensemble de conflits est celui de l'objet $p_8$. Dans un premier temps, l'opération $G^{\mathcal{D}} \leftarrow f_{emp}(\mathcal{O} \setminus p_8) \equiv G^{\mathcal{D}} \leftarrow \{p_1, p_3, p_2, p_6, p_7\}$ est effectuée. Le tableau 3.10 montre le résultat de cette première opération : l'objet $p_8$ est retiré de l'ensemble représentatif, et les ensembles de conflit de $p_6$ et $p_7$ sont mis à jour. Ils sont maintenant vides, $p_6$ et $p_7$ sont donc ajoutés à l'ensemble représentatif.

| **Objet** | $g_1$ | $g_2$ | $\mathcal{O}_{\|i}$ |
|---|---|---|---|
| $p_1$ | -2 | -1.5 | $\emptyset$ |
| $p_3$ | -1 | -1.5 | $\emptyset$ |
| $p_2$ | -0.5 | -1 | $\emptyset$ |
| $p_4$ | 0.5 | -0.5 | $\{p_5\}$ |
| $p_5$ | 1 | -0.8 | $\{p_4\}$ |
| ~~$p_8$~~ | ~~1.2~~ | ~~3~~ | ~~$\{p_6, p_7\}$~~ |
| $p_6$ | 1.5 | 1 | ~~$\{p_8\}$~~ $= \emptyset$ |
| $p_7$ | 3 | 1 | ~~$\{p_8\}$~~ $= \emptyset$ |

Tab 3.10 – Opération $t_{01}$ pour $(g_1^{\geq}\ g_2^{\geq})$

Dans l'étape suivante, nous avons le choix entre écarter $p_4$ ou écarter $p_5$, puisque la cardinalité de leur ensemble de conflits est la même. Quel que soit l'objet écarté, la cardinalité de l'ensemble représentatif local sera la même. Cependant, ce choix peut avoir des conséquences sur les ensembles représentatifs des sur-itemsets. Nous discutons de ces conséquences dans la section 3.3.5. Dans cet exemple, nous choisissons de supprimer le premier objet rencontré, soit $p_4$. L'ensemble représentatif maximal finalement obtenu est $G_s^{\mathcal{D}} = \{p_1, p_3, p_2, p_5, p_6, p_7\}$.

#### Génération des n-itemsets

Dans le cas des algorithmes générer-élaguer, l'ensemble des objets supportant un $n$-itemset est obtenu par intersection des ensembles de deux $(n-1)$-itemsets. Dans le contexte de la gradualité, cette méthode n'est pas applicable, car elle élague trop drastiquement les objets qui auraient pu participer à des ensembles représentatifs plus grands.

Afin d'illustrer notre propos, considérons l'itemset graduel $(g_1^{\geq}\ g_2^{\leq}\ G^{\leq})$. Celui-ci sera généré à partir des ensembles représentatifs de $(g_1^{\geq}\ g_2^{\leq})$ du tableau 3.11a et de $(g_1^{\geq}\ G^{\leq})$ du tableau 3.11b. Une intersection de ces deux ensembles représentatifs résulte en un ensemble constitué de l'objet $p_6$. Cependant, il existe trois ensembles représentatifs de cardinalité plus grande qui répondent à l'itemset $(g_1^{\geq}\ g_2^{\leq}\ G^{\leq})$ : $\{p_1, p_3\}$, $\{p_8, p_7\}$ et $\{p_8, p_6\}$.

Pour pallier ce problème, nous proposons de gérer deux ensembles de conflits lors de la génération d'itemsets de longueur supérieure à 2. Cela nous permet de maximiser l'ensemble représentatif local. Notons qu'à ce stade, le problème d'ordonnancement des valeurs ne se pose plus : cet ordre est calculé lors de la génération des itemsets de taille 2, et reporté lors de la génération des itemsets de taille supérieure.

**Définition 12.** *Soit $s = i_1^{*_1}...i_n^{*_n}$ un $n$-itemset, et $\mathcal{O}$ l'ensemble des objets de la base $\mathcal{DB}$ ordonné sur $i_1$ selon l'opérateur $*_1$ puis sur $i_2$ selon l'opérateur $*_2$... puis sur $i_n$ selon l'opérateur $*_n$. Pour*

| Objet | $g_1$ | $g_2$ | $\mathcal{O}_{\|i}$ |
|---|---|---|---|
| ~~$p_1$~~ | ~~-2~~ | ~~-1.5~~ | ~~$\{p_2,p_4,p_5,p_6,p_7,p_8\}$~~ |
| ~~$p_3$~~ | ~~-1~~ | ~~-1.5~~ | ~~$\{p_2,p_4,p_5,p_8,p_6,p_7\}$~~ |
| ~~$p_2$~~ | ~~-0.5~~ | ~~-1~~ | ~~$\{p_1,p_3,p_4,p_5,p_6,p_7,p_8\}$~~ |
| ~~$p_4$~~ | ~~0.5~~ | ~~-0.5~~ | ~~$\{p_1,p_2,p_3,p_6,p_7,p_8\}$~~ |
| ~~$p_5$~~ | ~~1~~ | ~~-0.8~~ | ~~$\{p_1,p_2,p_3,p_6,p_7,p_8\}$~~ |
| $p_8$ | 1.2 | 3 | $\{p_1,p_2,p_3,p_4,p_5\}$ |
| $p_6$ | 1.5 | 1 | $\{p_1,p_2,p_3,p_4,p_5\}$ |
| $p_7$ | 3 | 1 | $\{p_1,p_2,p_3,p_4,p_5\}$ |

(a) Ensemble représentatif pour ($g_1^{\geq}$ $g_2^{\leq}$)

| Objet | $g_1$ | $G$ | $\mathcal{O}_{\|i}$ |
|---|---|---|---|
| $p_1$ | -2 | 2 | $\{p_3,p_7,p_8\}$ |
| ~~$p_3$~~ | ~~-1~~ | ~~1~~ | ~~$\{p_2,p_4,p_5,p_6,p_7,p_8\}$~~ |
| $p_2$ | -0.5 | 2 | $\{p_3,p_7,p_8\}$ |
| $p_4$ | 0.5 | 2 | $\{p_3,p_7,p_8\}$ |
| $p_5$ | 1 | 2 | $\{p_3,p_7,p_8\}$ |
| ~~$p_8$~~ | ~~1.2~~ | ~~3~~ | ~~$\{p_1,p_2,p_3,p_4,p_5\}$~~ |
| $p_6$ | 1.5 | 2 | $\{p_3,p_7\}$ |
| ~~$p_7$~~ | ~~3~~ | ~~3~~ | ~~$\{p_1,p_2,p_3,p_4,p_5,p_6\}$~~ |

(b) Ensemble représentatif pour ($g_1^{\geq}$ $G^{\leq}$)

Tab 3.11 – Différents ensembles représentatifs de la base

*chaque objet $o_j \in \mathcal{O}$, nous conservons les objets écartés dans deux ensembles de conflits : le premier concernant l'item $i_{n-1}$ et noté $\mathcal{C}_{i_{n-1}}$, et le second concernant $i_n$ et noté $\mathcal{C}^{i_n}$. En d'autres termes, $\forall o_k \in \mathcal{C}^{i_{n-1}}, t_{o_j}[i_{n-1}] \neg *_{n-1} t_{o_k}[i_{n-1}]$ et $\forall o_k \in \mathcal{C}_{i_n}, t_{o_j}[i_n] \neg *_n t_{o_k}[i_n]$.*

**Algorithme 1** : Comp-Representative-Set

**Données** : Un n-itemset $s = (i_1^{*_1} ... i_n^{*_n})$,
Ensemble des objets $\mathcal{O}$ triés selon les $n-1$ items,
Ensembles de conflits $\mathcal{C}^n$ et $\mathcal{C}^{n-1}$

**Résultat** : Ensemble représentatif $G^{\mathcal{D}}$ pour $s$

1 $G^{\mathcal{D}} \longleftarrow \emptyset$
2 **tant que** $\mathcal{O} \neq \emptyset$ **faire**
3 $\quad o = f_{max}(\mathcal{C}^{i_n}, \mathcal{C}^{i_{n-1}})$
4 $\quad \mathcal{O} \longleftarrow \mathcal{O} \setminus \{o\}$
5 $\quad$ **pour chaque** $o_j \in \mathcal{O}$ **faire**
6 $\quad\quad f_{cnf}(o_j, \mathcal{C}^{i_n}) \longleftarrow f_{cnf}(o_j, \mathcal{C}^{i_n}) \setminus \{o\}$
7 $\quad\quad f_{cnf}(o_j, \mathcal{C}_{i^{n-1}}) \longleftarrow f_{cnf}(o_j, \mathcal{C}^{i_{n-1}}) \setminus \{o\}$
8 $\quad\quad$ **si** $f_{cnf}(o_j, \mathcal{C}_{i_n}) = \emptyset$ *and* $f_{cnf}(o_j, \mathcal{C}^{i_{n-1}}) = \emptyset$ **alors**
9 $\quad\quad\quad G^{\mathcal{D}} \longleftarrow G^{\mathcal{D}} + \{o_j\}$
10 $\quad\quad\quad \mathcal{O} \longleftarrow \mathcal{O} \setminus o_j$
11 $\quad\quad$ **fin**
12 $\quad$ **fin**
13 **fin**
14 **retourner** $\mathcal{O}_R$

La méthode de construction de l'ensemble représentatif est alors la même que précédemment, mais en mettant à jour les listes des deux items graduels à chaque suppression. L'algorithme de jointure est donné par l'Algorithme 1. Il implémente la fonction récursive utilisée par la propriété 2. Dans un souci de performance, nous utilisons une boucle `tant que` (ligne 2) plutôt qu'une fonction récursive. $G_s^{\mathcal{D}}$ est

construit dans la condition `si` (lignes 8 à 11). La fonction $f_{cnf} : \mathcal{O} \rightarrow \mathcal{C}$ permet d'accéder à l'ensemble de conflits associé à un objet.

Le tableau 3.12 montre le déroulement de l'algorithme 1 pour l'itemset graduel ($g_1^{\geq}\ g_2^{\leq}\ G^{\leq}$). Dans un premier temps, l'objet $p_1$ est supprimé car il a l'ensemble de conflits maximal. Cet ensemble de conflits est associé à l'item ($g_2^{\leq}$). Sa suppression entraîne la mise à jour de tous les ensembles de conflits, y compris ceux associés à l'item ($G^{\leq}$). Puis les objets $p_3, p_2, p_4, p_5$ et $p_6$ sont séquentiellement supprimés. L'ensemble représentatif final obtenu est $\{p_8, p_7\}$, et non uniquement $p_6$ qui aurait été obtenu par intersection entre les deux ensembles représentatifs de départ.

### 3.3.3 Propriétés des itemsets graduels

#### Complémentarité

Dans cette section, nous discutons des propriétés des itemsets graduels, et plus particulièrement celles qui sont utilisées dans un but d'optimisation. Dans un premier temps, nous retrouvons la propriété d'antonymie, évoquée dans [BMD90] et formalisée par [BCS+07].

Nous notons la *négation* (ou *complémentarité*) de la manière suivante :

$$c(*) = \begin{cases} \geq & si\ * = \leq \\ \leq & si\ * = \geq \end{cases}$$

La négation d'un itemset graduel est alors définie par :

**Définition 13.** *(itemset graduel complémentaire) Soit $s = (i_1^{*_1} ... i_n^{*_n})$ un itemset graduel. Alors la négation de $s$, notée $c(s)$, est $(i_1^{c(*_1)} ... i_n^{c(*_n)})$.*

**Exemple 11.** *Le complémentaire de ($g_1^{\geq}$) est ($g_1^{\leq}$). Le complémentaire de ($g_1^{\geq}\ g_2^{\leq}\ G^{\geq}$) est ($g_1^{\leq}\ g_2^{\geq}\ G^{\leq}$)*

Nous obtenons la proposition suivante :

**Proposition 3.** *(ensemble représentatif d'un itemset graduel complémentaire) Soit $s = (i_1^{*_1} ... i_n^{*_n})$ un itemset graduel. Si un ensemble d'objets $G^{\mathcal{D}}$ respecte cet itemset, alors l'ordre inversé des éléments de $G^{\mathcal{D}}$ respecte également $c(s) = (i_1^{c(*_1)} ... i_n^{c(*_n)})$.*

*Démonstration.* $\forall o, p \in \mathcal{O}, o * p \Leftrightarrow p\ c(*)\ o$. Cela implique que tous les objets de $G^{\mathcal{D}}$ respectent son complémentaire. □

**Corollaire 1.** $Freq(s) = Freq(c(s))$

Cela signifie que nous pourrons déduire une moitié des itemsets à partir de l'autre moitié. Cette optimisation divise l'espace de recherche par deux, et donc mène à une importante réduction de la consommation en temps et en mémoire. La figure 3.3 montre l'espace de recherche généré par cette optimisation. Nous choisissons ici arbitrairement de générer les items selon l'opérateur $\geq$ au premier niveau. L'utilisation de l'opérateur $\leq$ à la place est tout à fait possible. Ici, l'itemset ($g_1^{\leq}\ g_2^{\geq}\ G^{\leq}$) n'est jamais généré, puisque sa fréquence est la même que celle de ($g_1^{\geq}\ g_2^{\leq}\ G^{\geq}$).

| **Objet** | $g_1^{\geq}$ | $g_2^{\leq}$ | $\mathcal{O}_{\|g_2}$ | $G^{\leq}$ | $\mathcal{O}_{\|G}$ |
|---|---|---|---|---|---|
| $p_1$ | -2 | -1.5 | $\{p_2, p_4, p_5, p_6, p_7, p_8\}$ | 2 | $\{p_3, p_7, p_8\}$ |
| $p_3$ | -1 | -1.5 | $\{p_2, p_4, p_5, p_6, p_7, p_8\}$ | 1 | $\{p_2, p_4, p_5, p_6, p_7, p_8\}$ |
| $p_2$ | -0.5 | -1 | $\{p_1, p_3, p_4, p_5, p_6, p_7, p_8\}$ | 2 | $\{p_3, p_7, p_8\}$ |
| $p_4$ | 0.5 | -0.5 | $\{p_1, p_2, p_3, p_6, p_7, p_8\}$ | 2 | $\{p_3, p_7, p_8\}$ |
| $p_5$ | 1 | -0.8 | $\{p_1, p_2, p_3, p_6, p_7, p_8\}$ | 2 | $\{p_3, p_7, p_8\}$ |
| $p_8$ | 1.2 | 3 | $\{p_1, p_2, p_3, p_4, p_5\}$ | 3 | $\{p_1, p_2, p_3, p_4, p_5\}$ |
| $p_6$ | 1.5 | 1 | $\{p_1, p_2, p_3, p_4, p_5\}$ | 2 | $\{p_3, p_7\}$ |
| $p_7$ | 3 | 1 | $\{p_1, p_2, p_3, p_4, p_5\}$ | 3 | $\{p_1, p_2, p_3, p_4, p_5, p_6\}$ |
| **Pas 1 : suppression de** $p_1$ | | | | | |
| $p_3$ | -1 | -1.5 | $\{p_2, p_4, p_5, p_6, p_7, p_8\}$ | 1 | $\{p_2, p_4, p_5, p_6, p_7, p_8\}$ |
| $p_2$ | -0.5 | -1 | $\{p_3, p_4, p_5, p_6, p_7, p_8\}$ | 2 | $\{p_3, p_7, p_8\}$ |
| $p_4$ | 0.5 | -0.5 | $\{p_3, p_2, p_6, p_7, p_8\}$ | 2 | $\{p_3, p_7, p_8\}$ |
| $p_5$ | 1 | -0.8 | $\{p_3, p_2, p_6, p_7, p_8\}$ | 2 | $\{p_3, p_7, p_8\}$ |
| $p_8$ | 1.2 | 3 | $\{p_3, p_2, p_4, p_5\}$ | 3 | $\{p_2, p_3, p_4, p_5\}$ |
| $p_6$ | 1.5 | 1 | $\{p_3, p_2, p_4, p_5\}$ | 2 | $\{p_3, p_7\}$ |
| $p_7$ | 3 | 1 | $\{p_3, p_2, p_4, p_5\}$ | 3 | $\{p_2, p_3, p_4, p_5, p_6\}$ |
| **Pas 2 : suppression de** $p_3$ | | | | | |
| $p_2$ | -0.5 | -1 | $\{p_4, p_5, p_6, p_7, p_8\}$ | 2 | $\{p_7, p_8\}$ |
| $p_4$ | 0.5 | -0.5 | $\{p_2, p_6, p_7, p_8\}$ | 2 | $\{p_7, p_8\}$ |
| $p_5$ | 1 | -0.8 | $\{p_2, p_6, p_7, p_8\}$ | 2 | $\{p_7, p_8\}$ |
| $p_8$ | 1.2 | 3 | $\{p_2, p_4, p_5\}$ | 3 | $\{p_2, p_4, p_5\}$ |
| $p_6$ | 1.5 | 1 | $\{p_2, p_4, p_5\}$ | 2 | $\{p_7\}$ |
| $p_7$ | 3 | 1 | $\{p_2, p_4, p_5\}$ | 3 | $\{p_2, p_4, p_5, p_6\}$ |
| **Pas 3 : suppression de** $p_2$ | | | | | |
| $p_4$ | 0.5 | -0.5 | $\{p_6, p_7, p_8\}$ | 2 | $\{p_7, p_8\}$ |
| $p_5$ | 1 | -0.8 | $\{p_6, p_7, p_8\}$ | 2 | $\{p_7, p_8\}$ |
| $p_8$ | 1.2 | 3 | $\{p_4, p_5\}$ | 3 | $\{p_4, p_5\}$ |
| $p_6$ | 1.5 | 1 | $\{p_4, p_5\}$ | 2 | $\{p_7\}$ |
| $p_7$ | 3 | 1 | $\{p_4, p_5\}$ | 3 | $\{p_4, p_5, p_6\}$ |
| **Pas 4 et 5 : suppression de** $p_4$ **et** $p_5$ | | | | | |
| $p_8$ | 1.2 | 3 | $\emptyset$ | 3 | $\emptyset$ |
| $p_6$ | 1.5 | 1 | $\emptyset$ | 2 | $\{p_7\}$ |
| $p_7$ | 3 | 1 | $\emptyset$ | 3 | $\{p_6\}$ |
| **Pas 6 : suppression de** $p_6$ | | | | | |
| $p_8$ | 1.2 | 3 | $\emptyset$ | 3 | $\emptyset$ |
| $p_7$ | 3 | 1 | $\emptyset$ | 3 | $\emptyset$ |

Tab 3.12 – Déroulement de l'algorithme 1 pour la construction de l'ensemble représentatif de l'itemset graduel ($g_1^{\geq}$ $g_2^{\leq}$ $G^{\leq}$)

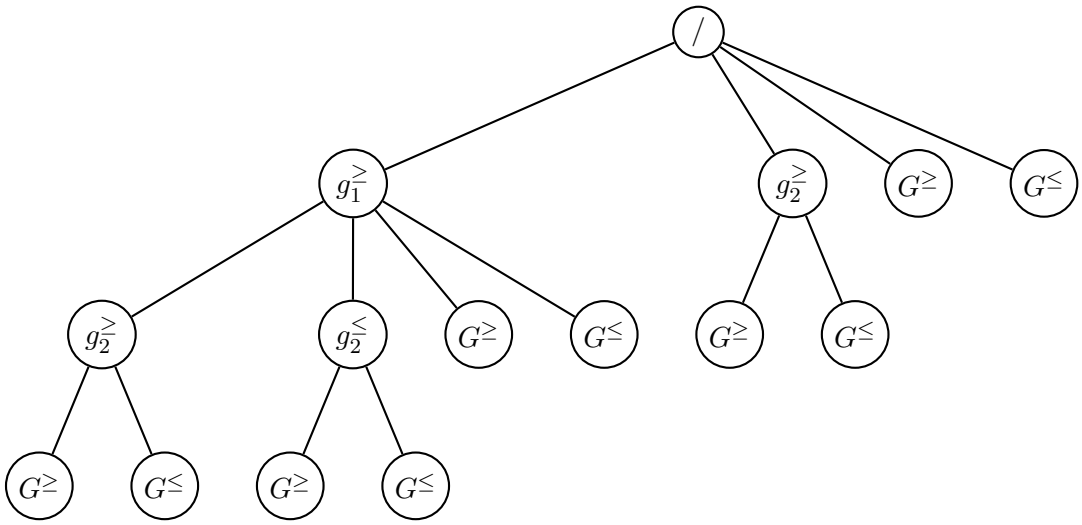



Fig 3.3 – Espace de recherche parcouru à partir de la base exemple

**Fréquence et confiance : cas des 1 et 2-itemsets**

Dans ce chapitre, nous exprimons la gradualité en utilisant une **relation d'ordre total**. Ainsi, quelque soit l'itemset de taille 1 considéré, tous les objets de la base de données (dans le cas d'une base ne contenant aucune valeur nulle) participent à l'ensemble représentatif de cet itemset, puisque tous les objets sont comparables. La fréquence des 1-itemsets est donc systématiquement de 100%. Pour cette raison, notre algorithme principal génère dès le départ les itemsets de taille 2. Cela permet d'ordonner sur deux dimensions, et une seule fois, puisque l'ordre obtenu sera reporté sur les sur-itemsets. De plus, un itemset de taille 1 n'a pas de signification graduelle : savoir que "$g_1$ augmente" et que "$g_2$ diminue" pour 100% des objets de la base n'apporte pas de connaissance utile aux experts.

Le calcul de la confiance d'une règle graduelle est basé sur la fréquence des deux itemsets la composant. Nous savons que $\forall i \in \mathcal{I}, Freq(i^+) = Freq(i^-) = 1$. Ainsi, pour une règle graduelle ne contenant qu'un item dans la condition et la conséquence, nous obtenons :

- $Conf(i_1^{*1} \Rightarrow i_2^{*2}) = \dfrac{Freq(i_1^{*1} i_2^{*2})}{Freq(i_1^{*1})}$
- $Conf(i_2^{*2} \Rightarrow i_1^{*1}) = \dfrac{Freq(i_1^{*1} i_2^{*2})}{Freq(i_2^{*2})}$

Comme $Freq(i_1^{*1}) = Freq(i_2^{*2})$, nous avons :

$$Conf(i_1^{*1} \Rightarrow i_2^{*2}) = Conf(i_2^{*2} \Rightarrow i_1^{*1}) = Freq(i_1^{*1} i_2^{*2})$$

Il est donc impossible d'établir l'implication la plus significative d'une règle de taille 2. Nous démarrons donc la génération de règles graduelles à partir des itemsets de taille 3 (troisième niveau de l'arbre des préfixes).

### 3.3.4 Expérimentations

Afin d'étudier les performances de notre algorithme, nous souhaitons réaliser des expérimentations sur des jeux de données synthétiques. Cependant, nous verrons dans un premier temps que les générateurs de jeux de données en vue d'extraction de connaissances graduelles ne sont pas les plus adaptés. Nous décrivons donc un générateur naïf dans le sens où il ne vise pas à reproduire des phénomènes réels, mais qui présente le fort avantage de permettre une étude approfondie des performances de notre algorithme. Nous répondons ensuite aux questions suivantes : *comment notre heuristique se comporte-t-elle face à des jeux de données au format inversé (plus d'items que de clients) ? Combien de motifs réels cette heuristique écarte-t-elle, et dans quels cas ?*

#### Vers un générateur de données pour l'extraction de motifs graduels

La génération de jeux de données synthétiques adaptés à des contextes spécifiques est un problème récurrent en fouille de données. Il s'agit de reproduire des phénomènes récurrents dans la réalité, mais qui sont souvent difficiles à modéliser. Dans ce contexte, l'un des premiers générateurs proposés pour l'extraction de règles classiques d'association est celui du groupe de recherche d'IBM d'Almaden [2]. Initialement proposé dans un but de mesure de performances de l'algorithme Apriori, ce générateur ne produit pas de jeux réalistes. Ce constat a motivé d'autres équipes [RMZ03, DD05, BFP07] à creuser la piste des bordures négatives et positives afin de proposer des générateurs pour l'extraction de motifs reproduisant des schémas réalistes.

Dans notre contexte, ces générateurs ne sont pas adaptés : quel que soit le générateur utilisé, nous obtenons pour chaque transaction une liste d'items présents. Les valeurs des items sont donc binaires (sous forme de présence /absence) et ne sont en aucun cas numériques. Il manque donc la partie graduelle, qui consiste à faire augmenter ou diminuer la valeur attribuée à ces items d'une transaction à l'autre. Dans un premier temps, nous avons affecté aléatoirement des valeurs à chacun des items présents dans une transaction. Cependant, pour un même item et d'une transaction à l'autre, aucune direction de variation (augmente ou diminue) n'est fournie. Ainsi, pour obtenir des itemsets graduels fréquents, il est nécessaire de baisser fortement le support (moins de 2%). La forme des itemsets fréquents finalement obtenus est affichée à la figure 3.4 :

```
(73+ 96+ 97+) : 0.3% (3/1000)
(73+ 96+ 97-) : 0.3% (3/1000)
(73+ 96- 97+) : 0.3% (3/1000)
(73+ 96- 97-) : 0.3% (3/1000)
(73+ 96- 98+) : 0.2% (2/1000)
(73+ 97+ 98-) : 0.2% (2/1000)
(73+ 97-) : 1.2% (12/1000)
(73+ 98+) : 0.3% (3/1000)
(73+ 98-) : 0.5% (5/1000)
```

Fig 3.4 – Extrait de résultats obtenus avec IBM Quest

2. www.almaden.ibm.com/software/projects/hdb/resources.shtml

La base utilisée dans la figue 3.4 contient 1000 transactions et 100 attributs. Avec une fréquence minimale fixée à 2%, environ la moitié des résultats obtenus sont "symétriques" (de la forme $A^+B^+ : X\%$, $A^+B^- : X\%$). et possèdent le même support. Cela signifie que pour chaque itemset, l'algorithme ordonne un très faible nombre de transactions. Notons également que dans le cas de cette base l'algorithme ne trouve aucun itemset fréquent pour un seuil supérieur à 2%. Ces phénomènes se sont répétés pour chacune des bases générées avec IBM Quest. Nous expliquons ce type de résultats par trois facteurs :

- Beaucoup de valeurs nulles. En effet, même si l'on fixe des paramètres maximaux concernant la taille des transactions de chaque objet de la base, le générateur produit une base contenant beaucoup de valeurs nulles (base peu dense).
- Aucune co-variation allant dans le même sens n'est générée pour un ensemble d'items : la génération de nos valeurs de manière aléatoire ne permet pas de faire varier ces valeurs dans le même sens d'une transaction à l'autre.
- Les supports, extrêmement faibles, indiquent qu'il est toujours possible de trouver des ensembles de deux ou trois objets respectant un itemset graduel. La consultation de ces ensembles montre que toutes les valeurs des items d'une transaction à une autre sont égales.

Afin d'évaluer convenablement les performances en terme de mémoire et de temps, nous avons mis en place un mini-générateur, dont la méthode est illustrée par l'algorithme 2. Le but n'est pas ici de coller à la réalité des bases extraites, mais plutôt de générer des transactions où les co-variations vont dans le même sens. Pour ce faire, nous avons mis en place un système de coefficientss. L'utilisateur fixe un nombre de transactions, un nombre d'items et le pourcentage de bruit qu'il souhaite obtenir. Dans un premier temps, une liste de coefficients de valeur -1 ou 1 est aléatoirement générée pour chaque item (ligne 2). Ces coefficients nous donneront, pour l'item concerné, son sens de variation principal. Afin de baisser la fréquence des itemsets les plus longs, la fréquence des derniers items de la base est divisée (ligne 4). Nous introduisons également un peu de "bruit", afin de ne pas déduire des fréquents trop uniformes. Enfin, la génération de la base est faite dans les lignes 6 à 14, et pour chaque item $i$ se base sur la précédente transaction générée et sur le coefficient attribué (ligne 11).

Un tel générateur produit des bases dont les premiers itemsets sont fréquents à 90%. Avec ce type de base, notre heuristique sera très sensible à la baisse du support. Cependant, au vu du nombre d'itemsets extraits avec des supports considérés habituellement comme relativement élevés (au dessus de 50%), nous pouvons affirmer que ce générateur convient parfaitement aux mesures de temps et de mémoire recherchées.

### Évaluation des performances

Notre approche a été implémentée en C++. Les avantages de ce langage sont doubles : d'une part ce langage compilé est très performant en terme de rapidité d'exécution, et d'autre part la gestion mémoire est facilement contrôlée par le programmeur. Afin de mettre en œuvre efficacement notre heuristique, nous avons codé des ensembles de conflits sous forme binaire. Les expérimentations ont été menées sur un ordinateur Precision WorkStation R5400 Xeon(R) CPU E5450 3.00GHz doté de 16Go de RAM. A l'aide du générateur, nous avons généré trois bases dont les caractéristiques sont résumées dans le tableau 3.13. Dans la suite de cette section, nous illustrons notre approche avec ces bases.

Les résultats des expérimentations sur la base C100I1000 sont présentés dans le tableau 3.14. Dans un premier temps, nous constatons qu'il est impossible de baisser le seuil de fréquence minimal en-

**Algorithme 2** : Génération de base graduelle

**Données** : Un nombre d'items $nbItems$
Un nombre de transactions $nbTrans$,
Un pourcentage de bruit $b$

**Résultat** : Une base de données numérique

**pour** $i = 0; i < nbItems; i++$ **faire**
  $coeff[i] \leftarrow GenerationAleatoire()$
  $pourcentageAbs = nbTrans \times (1 - \frac{nbItems-i}{nbItems})$
  $TransactionsVides[i] \leftarrow GenerationNull(pourcentageAbs, b)$
**fin**
**pour** $i = 0; i < nbTrans; i++$ **faire**
  **pour** $j = 0; j < nbItems; j++$ **faire**
    **si** $i \in TransactionsVides[j]$ **alors**
      $BD[i][j] = -1$
    **sinon**
      $BD[i][j] = BD[i-1][j] + random() \times coeff[j]$
    **fin**
  **fin**
**fin**
**retourner** $BD$

| Nom | #Transactions | #Items | #Valeurs |
|---|---|---|---|
| C100I10 | 100 | 10 | 100 |
| C100I1000 | 100 | 1000 | 5000 |
| C5000I100 | 5000 | 100 | 5000 |

Tab 3.13 – Noms et caractéristiques des bases synthétiques utilisées

dessous de 88%. Cependant, cela suffit pour caractériser le comportement de l'heuristique sur une base contenant plus d'items que d'objets. En effet, il faut environ deux heures pour extraire plus de deux millions d'itemsets graduels. Ces temps d'extraction sont acceptables, puisque la base est suffisamment dense pour contenir des itemsets de longueur 13 et de fréquence minimale 0.88, comme le montre la figure 3.5a.

| minFreq | #Itemsets | Temps (Min) |
|---|---|---|
| 90% | 381985 | 23 |
| 88% | 2007505 | 127 |

Tab 3.14 – Nombre d'itemsets et temps d'extraction pour la base C100I000

De plus, nous remarquons que la consommation mémoire est linéaire (figure 3.5b). En revanche, l'approche est très gourmande en mémoire à partir d'un certain nombre de motifs. Par exemple, pour

deux millions de motifs, environ 8 Go de mémoire vive sont nécessaires.

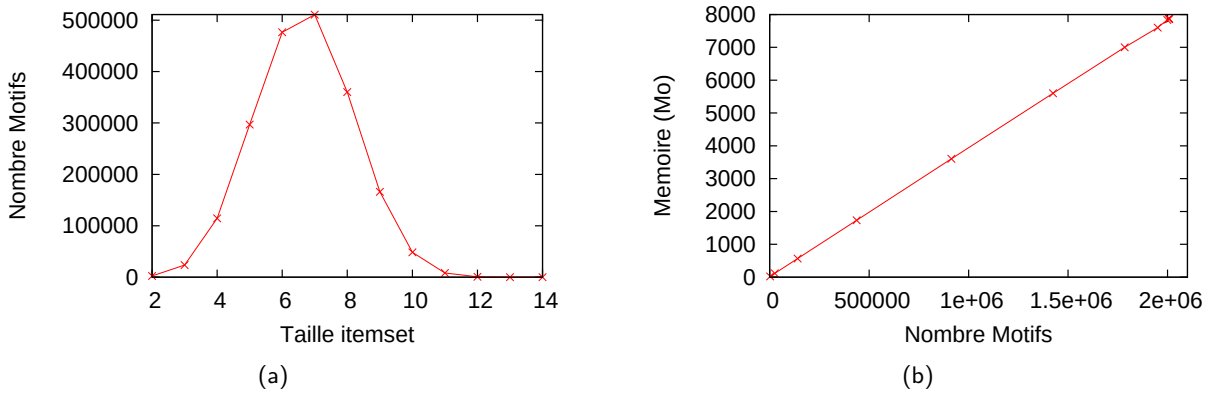



Fig 3.5 – Base C100I1000 avec minFreq=88% (a) Nombre de motifs extraits par longueur d'itemset (b) Mémoire utilisée en fonction du nombre de motifs

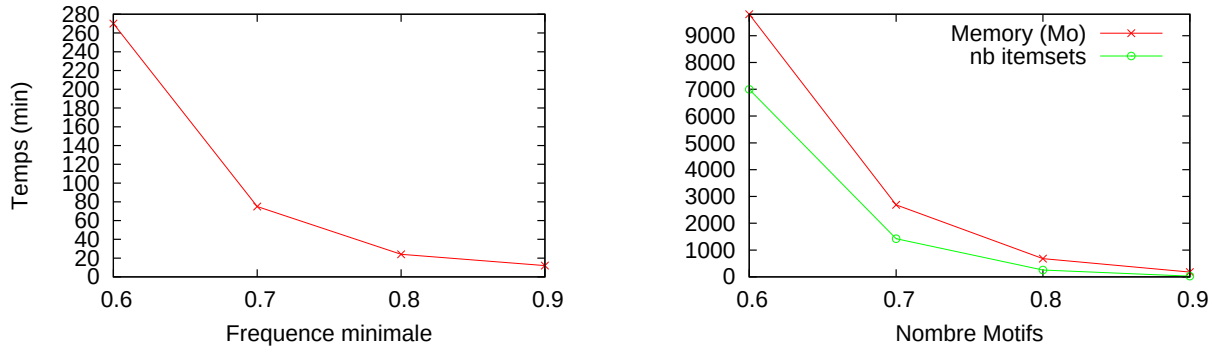
Comme le support minimal reste élevé, les résultats sur la base C100I1000 ne permettent pas d'affirmer que la méthode est robuste face à des faibles fréquences sur des bases denses. Cependant, l'approche permet d'obtenir une très grande quantité d'informations, même avec une fréquence élevée. Ce résultat démontre que cette méthode peut être utilisée dans un cas réel et est à replacer dans le contexte des bases génomiques. Sur ce type de bases, les experts recherchent les comportements fréquents ce qui s'avère généralement difficile à obtenir : la plupart des algorithmes ne terminent pas l'extraction ou bien n'extraient aucune information pertinente.
Les résultats des expérimentations sur la base C5000I100 sont résumés par les figures 3.6a et 3.6b. L'objectif ici est d'étudier le comportement de l'algorithme sur une base contenant plus d'objets que d'items. Tout d'abord, l'algorithme supporte bien la baisse du seuil de fréquence minimal, puisque nous parvenons ici à 60% pour environ 7000 itemsets graduels. La consommation mémoire est plus élevée que pour la base précédente : plus de 9 Go sont nécessaires à l'extraction de ces 7000 itemsets. En effet, même si le nombre d'itemsets est plus petit que pour la base C100I1000, le nombre de clients matérialisés dans les ensembles de conflit est bien plus conséquent.

Le temps nécessaire à l'extraction est également plus long. Cela démontre que le calcul de la fréquence est plus long lorsque le nombre d'objets est élevé. En effet, ce calcul est répété moins de fois que pour la base C100I1000, mais sur des matrices binaires plus grandes, ce qui ralentit le processus.

### Heuristique et complétude

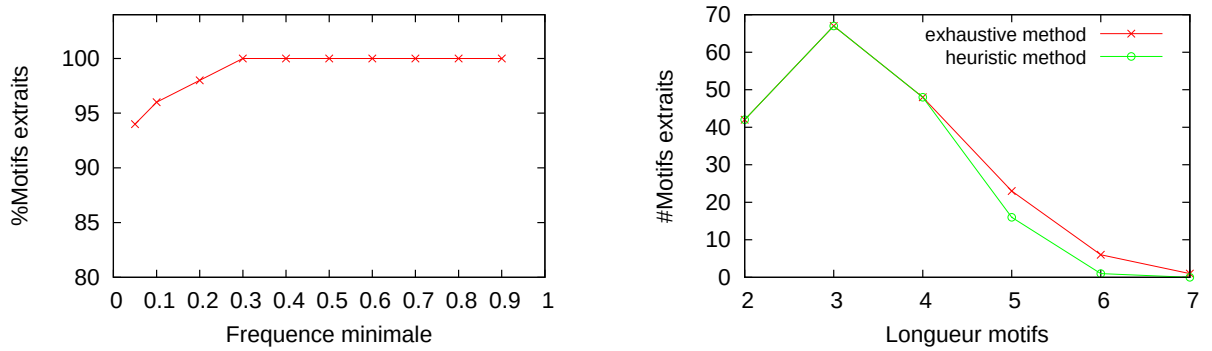
La méthode heuristique n'est pas complète, ce qui signifie qu'il y a une perte de motifs fréquents. Nous avons étudié cette perte, en comparant les résultats obtenus sur deux petites bases avec ceux obtenus par la méthode complète présentée à la section 3.4.

La figure 3.7a montre le nombre de motifs extraits comparé au nombre de motifs exacts sur la base C100I10. Jusqu'à un seuil minimal de 30%, l'approche heuristique extrait 100% des motifs, ce qui signifie que pour ce jeu de données, cette approche est exacte jusqu'à un support de 30%. Nous avons déterminé sur cette même base l'impact de la longueur des motifs sur le nombre de motifs extraits.




(a) (b)

Fig 3.6 – Base C5000I100 (a) Temps d'extraction en fonction de minFreq (b) Nombre de motifs extraits et mémoire utilisée




(a) (b)

Fig 3.7 – (a) Nombre de motifs extraits par rapport au nombre total de motifs (%) (b) Nombre de motifs extraits par niveau pour C100I10V100 avec minFreq=0.05

La figure 3.7b montre que l'heuristique écarte des motifs fréquents à partir d'une longueur de 5. La méthode proposée étant susceptible d'écarter des motifs fréquents à partir des 3-itemsets, ce résultat est encourageant. Notons qu'au pire des cas, l'algorithme a extrait 94% des motifs fréquents.

La longueur des motifs étant le facteur déterminant de l'efficacité de notre méthode, nous avons relancé ces mêmes tests sur une base contenant 100 objets et 100 attributs. La figure 3.8a (mise à l'échelle logarithmique sur l'axe du nombre de motifs) montre que l'heuristique écarte des itemsets fréquents à partir d'une fréquence minimale fixée à 80%. A ce seuil, les itemsets de longueur maximale contiennent six items graduels. Cependant, l'efficacité de l'heuristique reste acceptable, puisque plus de 80% des motifs fréquents sont extraits (figure 3.8a).

La figure 3.8b met en évidence l'avantage fondamental de notre heuristique : là où une méthode complète s'effondre, l'heuristique parvient à terminer l'extraction. Dans ce cas, la méthode complète s'effondre avec une fréquence minimale de 30% tandis que la méthode heuristique se termine, et double

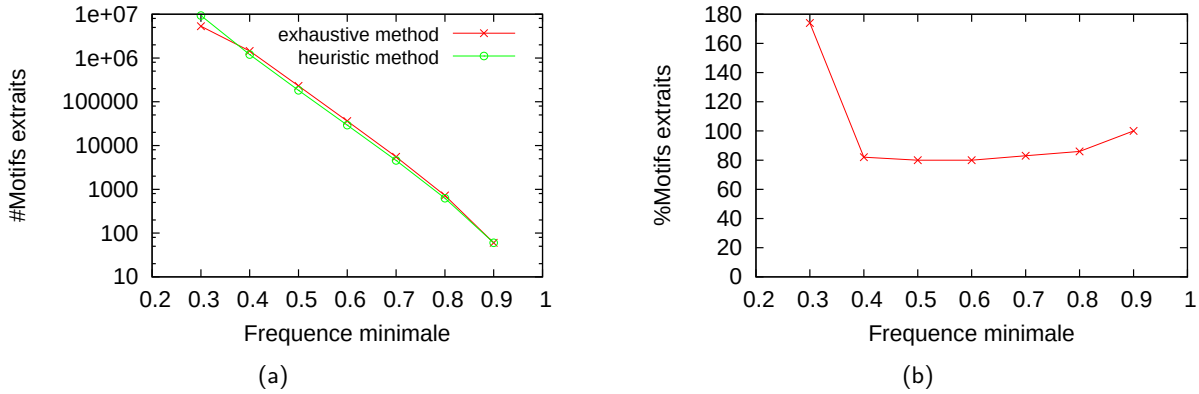



Fig 3.8 – (a) Nombre de motifs extraits par rapport au nombre total de motifs (%) (b) Nombre de motifs extraits par niveaux pour C100I10V100 avec minFreq=0.05

le nombre de motifs fréquents. Cela permet de présenter aux utilisateurs des itemsets fréquents de plus grande taille et qui n'auraient pas pu être extraits avec une autre méthode, malgré un léger biais sur leur fréquence.

### 3.3.5 Discussion

L'inconvénient majeur de la méthode exposée ci-dessus est qu'elle n'est pas complète. En effet, l'utilisation d'une heuristique implique que dans certains cas, la fréquence d'un itemset graduel peut être sous-évaluée. Ainsi, si l'utilisateur fixe un seuil minimal légèrement supérieur à la fréquence obtenue, et si cette fréquence n'est pas la fréquence réelle, cet itemset graduel ne sera pas extrait.

Ce phénomène s'explique par le fait que nous faisons des choix chaque fois que nous rencontrons plus d'un ensemble de conflits maximal. Évidemment, nous pouvons changer la politique de sélection de cet ensemble maximal : dans ce mémoire, nous supprimons l'objet le plus petit lexicographiquement, mais il serait possible de choisir de manière aléatoire, ou encore selon des règles fixées par l'expert. Cependant, quel que soit ce choix, la fréquence calculée sera toujours inférieure ou égale à la fréquence réelle. Ce phénomène ne s'applique pas au niveau local, mais pour les sur-itemsets.

Prenons pour exemple la base du tableau 3.15. Lors de la construction de l'ensemble représentatif associé à l'itemset $(A^{\geq}\ B^{\geq})$, notre algorithme supprimera les objets $\{o_x, o_y, o_z\}$. Cependant, il se trouve que pour l'itemset $(A^{\geq}\ B^{\geq}\ C^{\geq}\ D^{\geq}\ E^{\geq})$, $\{o_x, o_y, o_z\}$ est l'ensemble représentatif maximal. Notre algorithme choisit une solution différente et trouve $\{o_1, o_4\}$, ce qui résulte en un support plus faible.

Dans ce contexte, comment choisir l'objet à supprimer ? La problématique principale réside dans le fait que lors de la génération d'un itemset de niveau $n$, il est impossible de prédire la meilleure solution pour les itemsets de longueur $n+1$. Ainsi, l'extraction exhaustive des itemsets graduels est un problème difficile. Une solution consiste à conserver tous les ensembles représentatifs associés à un itemset graduel. Nous décrivons une méthode exhaustive répondant à cette problématique dans la section 3.4.

| | A | B | C | D | E |
|---|---|---|---|---|---|
| $o_1$ | 3 | 1 | 3 | 3 | 1 |
| $o_2$ | 4 | 2 | 1 | 4 | 2 |
| $o_3$ | 5 | 3 | 4 | 1 | 3 |
| $o_4$ | 6 | 4 | 3 | 5 | 4 |
| $o_5$ | 7 | 1 | 5 | 2 | 5 |
| $o_6$ | 8 | 1 | 6 | 6 | 1 |
| $o_x$ | 1 | 20 | 10 | 15 | 10 |
| $o_y$ | 2 | 30 | 20 | 40 | 20 |
| $o_z$ | 2.5 | 40 | 30 | 50 | 30 |

Tab 3.15 – Base de données problématique

## 3.4 Une approche complète pour l'extraction d'itemsets graduels

### 3.4.1 Structures de données adéquates

Une solution au problème du choix des objets soulevé dans la section 3.3 consiste à conserver tous les choix possibles. En d'autres termes, nous souhaitons conserver l'ensemble de toutes les listes constituant $G_s^{\mathcal{D}}$. Cependant, toutes ces possibilités doivent être conservées pour chaque itemset graduel, rendant l'espace mémoire occupé exponentiel. Il est donc primordial d'utiliser une structure satisfaisant les contraintes suivantes :

- Occupation mémoire compacte
- Jointure efficace : la structure ne devra être conservée que pour les itemsets de niveau $n$, puis effacée lors de la génération des niveaux $n+1$.

Les deux contraintes principales nous conduisent à adopter une représentation binaire. L'utilisation de structures binaires n'est pas nouvelle dans le contexte de la fouille de données. Il a d'ailleurs été démontré que les algorithmes les utilisant sont plus efficaces [AFGY02]. Le principe général est le suivant : plutôt que de conserver un tableau contenant les $n$ objets supportant un itemset, une structure de $\frac{|\mathcal{O}|}{8}$ octets est utilisée. Si un objet supporte un itemset, alors le bit correspondant à son index est mis à 1, et à 0 sinon. Lors de la jointure de deux itemsets, il suffit alors d'effectuer un ET binaire (noté dans ce mémoire $\wedge$), puis de compter le nombre de bits à 1 afin d'obtenir la fréquence du nouvel itemset.

Dans le contexte des itemset graduels, nous avons montré dans la section 3.3.2 qu'une intersection entre ensembles représentatifs locaux pouvait mener à une perte d'information. Afin d'utiliser la structure la plus compacte possible, nous ne représenterons qu'une seule fois les informations redondantes sur l'ensemble des solutions possibles. Par exemple, nous avions établi à la section 3.3.1 que quatre listes ordonnées d'itemsets représentent l'ensemble des solutions pour l'itemset graduel ($g_1^{\geq}$ $g_2^{\geq}$). En regardant de plus près les tableaux 3.8a, 3.8b, 3.8c et 3.8d, nous observons que les objets $p_1$, $p_2$ et $p_3$ apparaissent dans chacune de ces listes, et dans le même ordre. Cette partie est donc commune à l'ensemble de toutes les solutions. Puis, soit la liste ordonnée inclue $p_4$, soit elle inclut $p_5$. Nous retrouvons plus loin le même phénomène avec les objets $\{p_6, p_7\}$ et $p_8$.

L'ensemble des solutions $G_s^{\mathcal{D}}$ peut être représenté graphiquement par un diagramme de Hasse[3]. Un diagramme de Hasse est constitué de sommets et d'arc où :

- Chaque sommet représente un élément de l'ensemble des solutions
- Un arc relie deux sommets si $o_1 * o_2, * \in \{\geq, \leq\}$ est tracé
- Le diagramme garde une logique visuelle : si $o_1 * o_2$, alors $o_1$ est placé au-dessus de $o_2$
- On ne note ni la reflexivité, ni la transitivité

La figure 3.9 montre le diagramme de Hasse obtenu pour l'ensemble $G_s^{\mathcal{D}}$ associé à l'itemset graduel ($g_1^{\geq}$ $g_2^{\geq}$). Avec une telle représentation, chaque $\mathcal{L} \in G_s^{\mathcal{D}}$ représente un *chemin* de l'une des racines à l'une des feuilles. Dans la figure 3.9, il y a bien quatre chemins.

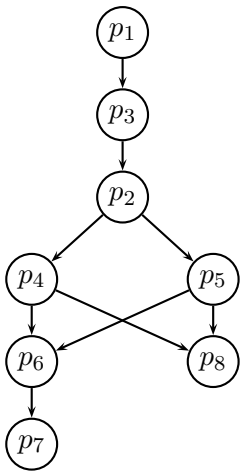



Fig 3.9 – Diagramme de Hasse représentant $G_s^{\mathcal{D}}$ pour $s =(g_1^{\geq}\ g_2^{\geq})$

Pour représenter de telles structures en mémoire, nous utilisons des matrices binaires, que nous formalisons de la manière suivante :

**Définition 14.** *(Matrice binaire des ordres) Soit $s$ un itemset graduel, $\mathcal{G}_s$ l'ensemble des listes ordonnées le représentant, et $\mathcal{O}_{\mathcal{G}_s}$ l'ensemble des objets de $\mathcal{G}_s$. $\mathcal{G}_s$ peut être représenté par une matrice binaire $M_{\mathcal{G}_s} = (m_{o_i,o_j})_{o_i \in \mathcal{O}_{\mathcal{G}_s}, o_j \in \mathcal{O}_{\mathcal{G}_s}}$, où $m_{o_i,o_j} \in \{0,1\}$.*

S'il existe une relation d'ordre entre $o_i$ et $o_j$, alors le bit correspondant à la ligne $o_i$ et à la colonne $o_j$ est mis à 1, et à 0 sinon. Par exemple, la matrice du tableau 3.16 montre la matrice construite pour représenter $\mathcal{G}_s$. Comme cette matrice contient huit objets, il sera nécessaire d'allouer $8^2 = 64$ bits, soit 8 octets. Nous ne réduisons pas la matrice à sa fermeture transitive. Par exemple, toute la ligne de l'objet $p_1$ est mise à 1.

3. De tels diagrammes ont été proposés par le mathématicien allemand Helmut Hasse, et permettent de représenter graphiquement un ordre fini

| $\vec{\Gamma}$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $p_7$ | $p_8$ |
|---|---|---|---|---|---|---|---|---|
| $p_1$ | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $p_2$ | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| $p_3$ | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| $p_4$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| $p_5$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| $p_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| $p_7$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_8$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Tab 3.16 – Matrice binaire $M_{\mathcal{G}_s}$ associée à $(g_1^{\geq}\ g_2^{\geq})$

### 3.4.2 Algorithmes

#### Génération des candidats

Le principal intérêt de la structure binaire représentée ci-dessus réside dans le fait qu'elle facilite grandement l'opération de jointure. En effet, l'ensemble de toutes les solutions étant mémorisé, il est maintenant possible d'effectuer une intersection entre deux matrices binaires. Cela nous garantit la préservation des ordres communs.

Nous procédons de la manière suivante :

1. **Génération des 1-itemsets graduels :** pour chaque item $i$ de la base $\mathcal{DB}$, l'item graduel $i^{\geq}$ est construit en ordonnant les $t_{\mathcal{O}}[i]$ selon la relation d'ordre $\geq$. A chaque nœud du premier niveau est associé la matrice binaire correspondante. Contrairement à l'heuristique, nous ne trions que sur une dimension. De plus, l'ordre étant enregistré dans la matrice, il ne sera plus nécessaire de lire la base de données. Par exemple, au premier niveau de l'arbre, les matrices représentant les figures 3.10a, 3.10b et 3.10c sont stockées.
2. **Génération des n-itemsets graduels :** la longueur des itemsets graduels est augmentée par passe. Nous utilisons le théorème 1 afin de construire les matrices binaires associées aux itemsets graduels résultant des jointures. L'utilisation de l'opérateur binaire ET nous garantit un processus efficace en terme de temps d'exécution : en effet, les opérations bit-à-bit figurent parmi les plus performantes en programmation.

**Théorème 1.** *Soit $s''$ un itemset graduel obtenu par la jointure des itemsets $s$ et $s'$. Alors*

$$M_{\mathcal{G}_{s''}} = M_{\mathcal{G}_s} \wedge M_{\mathcal{G}_{s'}}$$

A la fin du processus de jointure de niveau $k$, nous obtenons toutes les matrices binaires associées aux $k$-itemsets. Cependant, quelques informations représentées ne seront pas utiles. En effet, certains graphes peuvent contenir des *nœuds isolés*, c'est-à-dire des nœuds n'ayant ni père ni fils. Ces objets ne participent pas au calcul de la fréquence, et les conserver revient à une perte de mémoire et de temps, puisqu'ils seront considérés lors des jointures ultérieures. Ces opérations sont réalisées par l'algorithme 3.

Afin d'illustrer notre propos, nous construisons la matrice binaire associée à l'itemset graduel $(g_1^{\geq}\ g_2^{\geq}\ G^{\leq})$, qui s'obtient par jointure des itemsets $(g_1^{\geq}\ g_2^{\geq})$ et $(g_1^{\geq}\ G^{\leq})$. Lors de la génération des 2-itemsets,

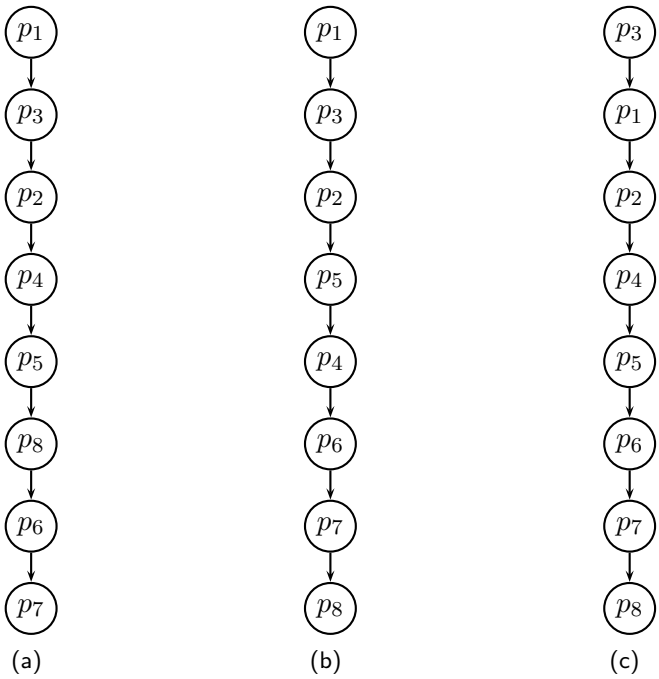



Fig 3.10 – Diagrammes de Hasse pour (a) $\mathcal{G}_{g_1^{\geq}}$, (b) $\mathcal{G}_{g_2^{\geq}}$, et (c) $\mathcal{G}_{G^{\geq}}$

**Algorithme 3** : Joindre

**Données** : Deux matrices $M_{\mathcal{G}_s}$ et $M_{\mathcal{G}_{s'}}$

**Résultat** : La matrice $M_{\mathcal{G}_{ss'}}$

1 $M_{ss'} = Initialiser(\mathcal{T}_{\mathcal{G}_s} \cap \mathcal{T}_{\mathcal{G}'_s})$
2 $M_{ss'} = M_s \ ET \ M_{s'}$
3 $M_{ss'} \leftarrow EffacerNoeudIsole(M_{ss'})$
4 **retourner** $M_{ss'}$

nous construisons la matrice binaire associée à $(g_1^{\geq} \ G^{\leq})$ reportée au tableau 3.11b. Le diagramme de Hasse correspondant est dessiné à la figure 3.11a. Le résultat de l'opération binaire $\wedge$ (algorithme 3, ligne 2) produit la matrice représentée au tableau 3.17b. Le diagramme de Hasse correspondant, dessiné à la figure 3.11b, montre bien que les nœuds $p_7$ et $p_8$ sont isolés : ils n'ont ni père, ni fils. Le tableau 3.18 montre la matrice stockée une fois les nœuds isolés supprimés.

## Calcul de la fréquence

Si elle simplifie l'opération de jointure, l'utilisation de matrices binaires complique en revanche le calcul de la fréquence. Selon la définition 9, la fréquence est le rapport du plus grand ensemble de solutions sur le nombre d'objets de la base. Il est donc nécessaire de dégager le plus long chemin de la matrice binaire. Cette problématique est non sans rappeler les problèmes de cheminement dans un graphe, où la plupart des algorithmes existant consistent à extraire le plus court chemin entre deux sommets [Dij71, Flo62].

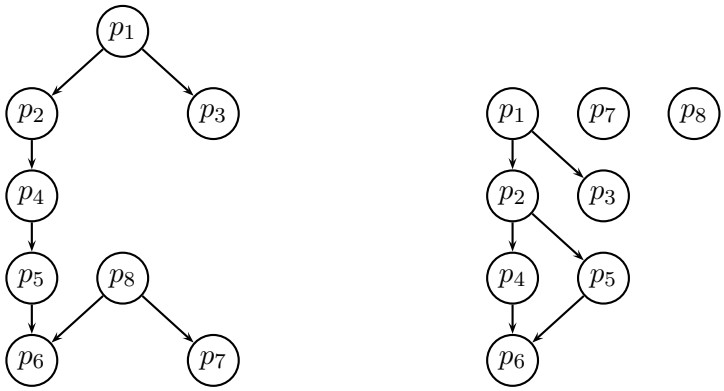
| $\vec{\Gamma}$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $p_7$ | $p_8$ |
|---|---|---|---|---|---|---|---|---|
| $p_1$ | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| $p_2$ | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| $p_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_4$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| $p_5$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $p_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_7$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_8$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

(a)

| $\vec{\Gamma}$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $p_7$ | $p_8$ |
|---|---|---|---|---|---|---|---|---|
| $p_1$ | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| $p_2$ | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| $p_3$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_4$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $p_5$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $p_6$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_7$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_8$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(b)

Tab 3.17 – Matrices binaires (a) $M_{\mathcal{G}_s}$ associée à $(g_1^{\geq}\ G^{\leq})$, (b) $M_{\mathcal{G}_s}$ associée à $(g_1^{\geq}\ g_2^{\geq}\ G^{\leq})$




(a) (b)

Fig 3.11 – Diagrammes de Hasse représentant (a) $G_s^{\mathcal{D}}$ pour $s =(g_1^{\geq}\ G^{\leq})$, (b) $G_s^{\mathcal{D}}$ pour $s =(g_1^{\geq}\ g_2^{\geq}\ G^{\leq})$

| $\vec{\Gamma}$ | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ |
|---|---|---|---|---|---|---|
| $p_1$ | 0 | 1 | 1 | 1 | 1 | 1 |
| $p_2$ | 0 | 0 | 0 | 1 | 1 | 1 |
| $p_3$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_4$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $p_5$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $p_6$ | 0 | 0 | 0 | 0 | 0 | 0 |

Tab 3.18 – Matrice binaire obtenue après suppression des nœuds isolés

Une solution naïve consiste à modifier les algorithmes de recherche de plus court chemin afin de les adapter à la problématique de recherche du plus long chemin. Cependant, la complexité de ces algorithmes (cubique pour la plupart) rend leur utilisation difficile en fouille de données. En effet, la fréquence est calculée après chaque jointure ce qui peut représenter un très grand nombre d'appels.

Nous nous sommes donc orientés vers un algorithme ne considérant qu'une seule fois chaque objet de la matrice binaire. Une telle méthode est légèrement plus consommatrice en mémoire, mais présente

le fort avantage d'être en $\mathcal{O}(n)$, et donc d'être faiblement consommatrice en terme de temps.

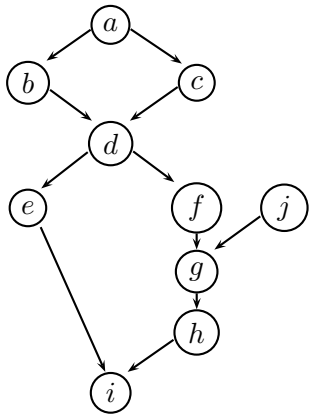



Fig 3.12 – Diagramme de Hasse exemple pour $\mathcal{G}_s$

| Liste | Longueur |
|---|---|
| $\{abdei\}$ | 5 |
| $\{abdfghi\}$ | 7 |
| $\{acdei\}$ | 5 |
| $\{acdfghi\}$ | 7 |
| $\{jghi\}$ | 4 |

Tab 3.19 – les listes d'objets composant $\mathcal{G}_s$

Nous illustrons notre méthode à l'aide de l'exemple de la figure 3.12. Celle-ci montre un diagramme de Hasse composé de cinq listes ordonnées d'objets de longueur variable. Ces listes ainsi que leur cardinalité sont énumérées dans le tableau 3.19. Au sein d'un $\mathcal{G}_s$, un objet peut appartenir à plusieurs listes ordonnées. C'est par exemple le cas de l'objet $g$ qui appartient à trois listes. Pour cette raison, un objet peut être affecté à différentes positions dans une liste. Ainsi, dans les seconde et quatrième listes, l'objet $g$ est en cinquième position, alors que dans la dernière il se trouve en seconde position. Pour le calcul de fréquence, nous souhaitons maximiser cette position (et donc conserver la cinquième).

Lorsqu'un nœud est parcouru pour la première fois, il est impossible de prédire si sa position actuelle est la position maximale. Nous conservons donc une sorte de "mémoire" des parcours contenant le résultat des parcours des autres nœuds. Lorsque plusieurs solutions se présentent, nous conservons la solution maximale.

La méthode décrite ci-dessus est réalisée par l'Algorithme 4. Cet algorithme récursif prend en entrée le nœud $o$ que nous souhaitons visiter, ainsi que la mémoire d'appels précédents mise à jour. Le résultat de son exécution est la mise à jour de la mémoire pour les nœuds fils de l'objet $o$. La stratégie suivante est adoptée : un nœud n'ayant pas de fils est considéré comme une feuille et l'index mémoire lui correspondant est mis à 1. Lorsqu'un tel nœud est rencontré, la récursion est arrêtée (ligne 3). Sinon, la récursion assure que lorsque qu'un objet $o$ est parcouru, tous les fils $o'$ de $o$ ont déjà été traités. A la fin, nous sélectionnons le fils ayant la mémoire la plus grande (ligne 11). La mémoire de l'objet courant $o$ sera égale à cette sélection plus un.

| Opération | a | b | c | d | e | f | g | h | i | j |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | *Avant appel Initialisation de la mémoire à -1* | | | | | | | | | |
| | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | 1 | $-1$ |
| **2** | *Appel 1 sur* **a** | | | | | | | | | |
| **3** | *Appel 2 sur* **b** *puis sur* **c** | | | | | | | | | |
| **4** | *Appel 3 sur* **d** | | | | | | | | | |
| **5** | *Appel 4 sur* **e** *puis sur* **f** | | | | | | | | | |
| **6** | *Appel 5 sur* **i** | | | | | | | | | |
| | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | $-1$ | 1 | $-1$ |
| **7** | *Retour appel 5* | | | | | | | | | |
| | $-1$ | $-1$ | $-1$ | $-1$ | 2 | $-1$ | $-1$ | $-1$ | 1 | $-1$ |
| **8** | *Retour appel 4 sur* **e** | | | | | | | | | |
| **9** | *Appel 6 sur* **g** | | | | | | | | | |
| **10** | *Appel 7 sur* **h** | | | | | | | | | |
| | $-1$ | 4 | $-1$ | 3 | 2 | $-1$ | $-1$ | 2 | 1 | $-1$ |
| **11** | *Retour appel 7* | | | | | | | | | |
| | $-1$ | 4 | $-1$ | 3 | 2 | $-1$ | 3 | 2 | 1 | $-1$ |
| **12** | *Retour appel 6* | | | | | | | | | |
| | $-1$ | 4 | $-1$ | 3 | 2 | 4 | 3 | 2 | 1 | $-1$ |
| **13** | *Retour appel 4 sur* **f** | | | | | | | | | |
| | $-1$ | 4 | $-1$ | 3 | 5 | 4 | 3 | 2 | 1 | $-1$ |
| **14** | *Retour appel 3* | | | | | | | | | |
| | $-1$ | 6 | $-1$ | 3 | 2 | 4 | 3 | 2 | 1 | $-1$ |
| **15** | *Retour appel 2 sur* **b** | | | | | | | | | |
| | $-1$ | 6 | 6 | 3 | 2 | 4 | 3 | 2 | 1 | $-1$ |
| **16** | *Retour appel 2 sur* **c** | | | | | | | | | |
| | 7 | 6 | 6 | 3 | 2 | 4 | 3 | 2 | 1 | $-1$ |
| **17** | *Retour appel 1* | | | | | | | | | |
| **18** | *Appel 8 sur* **j** | | | | | | | | | |
| | 7 | 6 | 6 | 5 | 2 | 4 | 3 | 2 | 1 | 4 |
| **19** | *Retour appel 8* | | | | | | | | | |

Tab 3.20 – Trace du calcul de la mémoire sur l'exemple 3.12

**Algorithme 4** : RecursiveCovering

**Données** : Un objet $o$
La mémoire provenant des appels antérieurs $Memory$
**Résultat** : $Memory$ remplie en fonction des objets parcourus

```
1  Sons ← GetSons(o)                    /* tous les o' de valeur 1 à la ligne o  */
2  si Sons = ∅ alors
3  |  Memory[o] = 1 ;
4  sinon
5  |  pour chaque i ∈ Sons faire
6  |  |  si Memory[i] = −1 alors
7  |  |  |  RecursiveCovering(i, Memory)
8  |  |  fin
9  |  fin
10 |  pour chaque i ∈ Sons faire
11 |  |  Memory[o] = max(Memory[o], Memory[i] + 1)
12 |  fin
13 fin
```

Le tableau 3.20 donne une trace de l'exécution de l'algorithme 4 sur l'exemple 3.12. Le diagramme de Hasse possède deux racines $a$ et $j$. Nous procédons en suivant l'ordre lexicographique. L'algorithme est appelé sur $a$, avec une mémoire initialisée à -1. Récursivement, les nœuds $b$, $d$, $e$ et $i$ sont parcourus. $i$ étant une feuille, la remontée de l'algorithme modifie la mémoire pour $b$, $d$, $e$ et $i$. Puis l'algorithme est rappelé sur $f$ (formalisé par l'opération 5 du tableau). Cela a pour effet de mettre à jour la mémoire pour $f$, $g$, $h$. Au moment de la mise à jour de $d$, l'algorithme choisit le fils ayant la position maximale pour s'aligner, dans ce cas $f$. Le même processus est répété pour $a$ qui se voit attribuer la position 7. L'appel sur la seconde racine $i$ est immédiat, puisque $g$ a déjà été parcouru. A la fin de cette méthode, l'algorithme retourne 7 pour cardinalité maximale.

### 3.4.3 Expérimentations

Nous avons mené nos expérimentations sur deux types de jeux de données : des jeux de données synthétiques, créés à partir du générateur décrit à la section 3.3.4 et des jeux de données issus de bases réelles. Ainsi, nous comparons dans un premier temps le comportement de la méthode complète avec celle de la méthode heuristique. Pour cela, nous avons réutilisé les jeux de données de la section 3.3.4.

#### Données synthétiques

Le tableau 3.21 montre le temps d'extraction ainsi que le nombre de motifs extraits pour la base C100I1000. La méthode complète génère le même nombre de motifs pour un seuil de fréquence minimal fixé à 90%, mais en 17 minutes, ce qui est moins que la méthode heuristique. Cela s'explique par le fait qu'il n'est pas nécessaire de parcourir la matrice binaire afin d'éliminer les objets en conflit. Cet écart de temps entre les deux approches se creuse pour un support de 88%, puisqu'il faut 90 minutes pour

extraire environ deux millions de motifs (contre 130 minutes avec la méthode heuristique).

| minFreq | #Itemsets | Temps (Min) |
|---|---|---|
| 90% | 381985 | 17 |
| 88% | 2019740 | 90 |

Tab 3.21 – Nombre d'itemsets et temps d'extraction pour la base C100I000

Les figures 3.13a et 3.13b montrent le comportement de l'algorithme dans le cas d'une base dense contenant peu d'items et beaucoup d'objets. On constate que le temps d'extraction avec la méthode complète est inférieur et permet d'extraire un plus grand nombre de motifs. En revanche, la méthode exhaustive utilise plus de mémoire car elle génère plus de motifs et donc plus de matrices binaires à stocker.

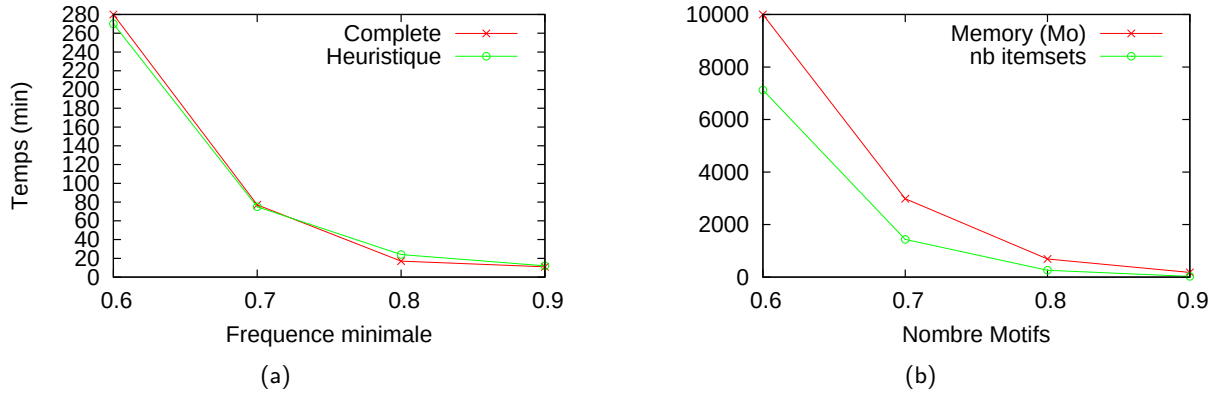



Fig 3.13 – Base C5000I100 (a) Temps d'extraction en fonction de minFreq (b) Nombre de motifs extraits et mémoire utilisée

Les expérimentations sur jeux de données synthétiques montrent donc que la méthode complète se comporte comme attendu : elle génère plus de motifs, en moins de temps, mais ne permet pas d'éviter le crash mémoire provoqué par l'explosion combinatoire et peut même dans certains cas l'atteindre plus vite que la méthode heuristique.

### Données réelles

Nous avons utilisé deux jeux de données réelles : un jeu est issu de questionnaires de patients atteints de la maladie d'Alzheimer, et l'autre issu d'analyse de puces dans le contexte de l'analyse de cancer du sein.
Notre but est ici de valider l'apport des règles graduelles, en répondant aux questions suivantes :

- *les règles sont-elles compréhensibles par les experts ?*
- *en retirent-ils une information utile ?*
- *quel est le comportement de l'algorithme sur des jeux de données réelles ?*

La maladie d'Alzheimer est une maladie neurodégénérative du tissu cérébral qui entraîne la perte progressive et irréversible des fonctions mentales et notamment de la mémoire [4]. Les chercheurs en psychologie du laboratoire EPSYLON (Université de Montpellier 3) étudient cette maladie, et les résultats présentés ici s'inscrivent dans le cadre du projet PEPS LAMAL [5]. L'équipe de Praxiling évalue les souvenirs. Pour ce faire, les chercheurs ont constitué des jeux de données à partir d'interviews structurées de patients et de tests évaluant les performances de la mémoire des patients (scores). Au cours des entretiens, les psychologues demandent aux patients de se remémorer un bon moment de leur vie, ainsi qu'un mauvais moment et un moment neutre. Ils analysent ensuite les souvenirs concernant les sons, les dispositions spatiales ou encore leurs sensations durant ces moments. Le jeu de données ainsi constitué et préparé contient 33 patients et 122 attributs.

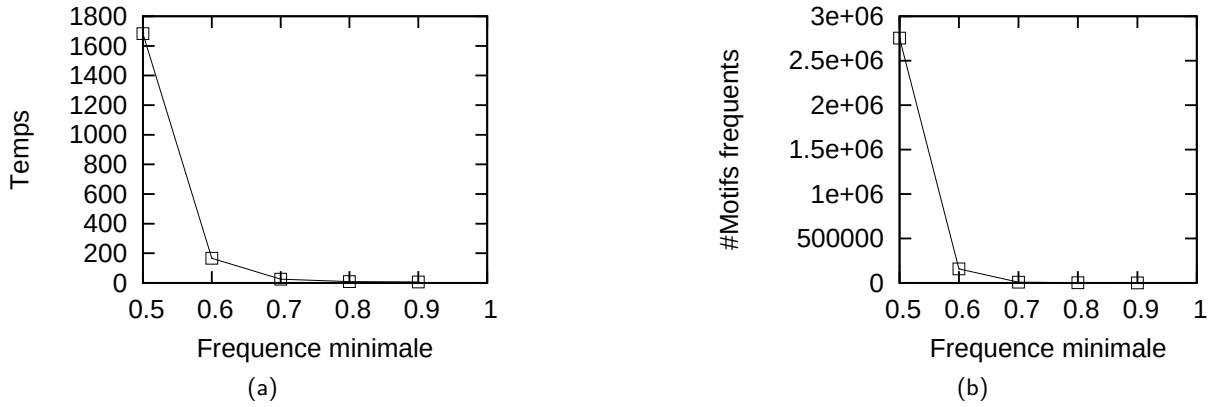



Fig 3.14 – Base Alzheimer (a) temps d'extraction en fonction de minFreq (b) nombre de motifs extraits

Les figures 3.14a et 3.14b montrent que le temps d'extraction pour un support minimal allant de 90% à 60% est très acceptable, puisqu'il varie de 10 secondes à environ deux minutes, pour extraire environ 300 motifs fréquents. En revanche, en dessous de 60%, le temps d'extraction et le nombre de motifs extraits croit exponentiellement : une vingtaine de minutes sont nécessaires afin d'extraire environ deux millions cinq cent mille motifs. Au delà des performances en terme de calcul, cela démontre que baisser le support au dessous de 60% n'apporte rien : non seulement il y a trop de règles pour qu'un être humain puisse les interpréter, mais en plus la base contient un très grand nombre de corrélations dont la fréquence est proche de 60%.

Voici des règles pointées par les experts comme apportant des informations déjà connues (donc validantes) :

- *Plus le souvenir de la disposition spatiale des personnes est bon et plus le souvenir du moment de la journée est bon, alors plus le souvenir de l'endroit est bon* (87.88%)
- *Plus il y a d'identifications durant le test RI48 et plus le souvenir des dispositions spatiales et du temps sont bons, alors plus le score au test MMS est élevé* (81.82%)
- *Plus un patient est âgé et moins il se souvient des sons d'un mauvais moment, alors moins son*

4. http ://fr.wikipedia.org/wiki/Maladie_d%27Alzheimer

5. Langage, Mémoire et Alzheimer : une approche des maladies neurodégénératives fondée sur la densité des idées. Fouille de données pour l'extraction de corrélations entre différents indices neuropsychologiques

*score au test MMS est élevé* (81.82%)

En revanche, le nombre de résultats est souvent trop élevé pour avoir une bonne interprétation. Cela s'explique par la redondance des items contenus dans les règles graduelles. Une solution de diminution du nombre de règles est donc proposée au chapitre 4.

Les jeux de données concernant le cancer du sein sont fournis par l'équipe Identité et Plasticité Tumorale de l'IRCM. Pour chaque tumeur, les chercheurs ont réalisé une analyse génomique en utilisant les puces à ADN. Celles-ci fournissent les données d'expression de quelque 5000 gènes. S'ajoutent les données cliniques, qui concernent le patient : son âge, le grade de la tumeur, le nombre de récidives, le nombre de ganglions infectés... Sur les 156 attributs, seulement 4 attributs numériques sont intéressants pour les experts. Le jeu de données constitué contient 108 tumeurs et 8000 attributs. L'expression de chaque clone est comprise entre -2 et 3. Le tableau 3.22 résume les différentes expérimentations que nous avons menées. La base étant trop dense, notre algorithme ne parvient pas à s'exécuter sur sa totalité. Nous avons donc, sur avis des experts, découpé cette base en fonction des chromosomes. De plus, une mini-base ne contenant que les expressions des gènes des chromosomes 8, 11 et 17 a été contruite à des fin de test : ce sont sur ces chromosomes que l'on retrouve les altérations les plus fréquentes.

| **Nbr gènes** | **MinSup** | **NbRegles** | **Longeur maxi** |
|---|---|---|---|
| 534 | 90 | 1024 | 2 |
| | 50 | 295821 | 3 |
| 1131 | 90 | 8163 | 4 |
| | 50 | 123126 | 4 |
| 1635 | 90 | 21706 | 4 |
| | 50 | 265580 | 4 |
| 2238 | 90 | 38139 | 4 |
| | 80 | 1536635 | 8 |
| Chr 8 11, 17 | 90 | 523 | 3 |
| | 50 | 328317 | 3 |

Tab 3.22 – Résumé des expérimentations sur la base cancer

Comme nous l'avons vu pour la base des psychologues, le nombre d'itemsets fréquents est beaucoup trop élevé pour une analyse humaine. Si le support est élevé, les experts ne retirent pas d'informations utiles, mais lorsqu'il est un peu plus faible, le nombre de motifs explose, ce qui ne permet pas la découverte de motifs intéressants. Ce phénomène n'est pas nouveau en fouille de données : il arrive régulièrement que les algorithmes extraient plus d'informations que celles présentées dans la base initiale. Il s'agit alors soit de post-traiter les résultats, en appliquant par exemple des méthodes de clustering, soit de raffiner l'extraction, en ajoutant par exemple de nouvelles contraintes. Ces aspects sont discutés dans le chapitre suivant.

### 3.4.4 Discussion

L'approche présentée ci-dessus est complète et plus efficace en terme de temps d'extraction que la méthode heuristique. Cependant, comme le montrent nos expérimentations, diverses pistes de recherche restent à explorer. Même si les consommations en ressources mémoire et temporelles sont tout à fait satisfaisantes compte tenu des spécificités des différentes bases, elles démontrent qu'il faut explorer deux axes : la réduction du nombre d'itemsets graduels afin de rendre la méthode plus accessible aux différents experts, et la réduction de la consommation des ressources pour les bases trop denses.

La réduction de la consommation des ressources peut être envisagée par la compression des itemsets fréquents (extraire des itemsets graduels clos), ou encore par la réduction du nombre de calculs de fréquence. Nous pensons que l'utilisation du principe d'inclusion/exclusion peut être utilisé afin de déduire la fréquence d'un itemset à partir d'un autre itemset de même longueur. Considérons par exemple les itemsets $s_1 = (i_1^{\geq}\ i_2^{\geq})$ et $s_2 = (i_1^{\geq}\ i_2^{\leq})$. Deux objets ayant la même valeur ne permettent pas de prédire si un objet participe à l'ensemble représentatif de $s_1$ ou à celui de $s_2$, alors qu'avec les relations d'ordre restreintes, il est possible de déterminer clairement à quel ensemble représentatif cet objet appartiendra. Ces propriétés doivent être étudiées afin de proposer des optimisations adaptées à notre méthode.

La réduction du nombre d'itemsets peut se faire au travers du durcissement de la contrainte de fréquence. En effet, nous avons considéré dans ce chapitre une vision optimiste de la fréquence, où deux objets ayant des valeurs égales pour un même attribut participent tous deux à la fréquence, ce qui donne lieu aux propriétés de la section 3.3.3. Cependant, dans certains contextes la sémantique de l'égalité n'est pas adaptée à la réalité. Dans ce cas, il est nécessaire de définir une nouvelle mesure de fréquence.

Une piste intéressante consiste à ne pas supprimer l'égalité, mais à la pondérer. Par exemple, considérons le cas où les valeurs varient fréquemment de 100 unités d'un objet à l'autre. Une variation d'une seule unité n'aura alors pas le même poids. Il s'agir alors de proposer des mesures estimant la "*force de variation*" d'un itemset graduel, et de filtrer les itemsets en fonction de cette mesure. Les mesures statistiques telles que la covariance ou l'entropie peuvent être envisagées. L'une des principales difficultés est que ce type de mesure n'est pas anti-monotone. Dans le chapitre suivant, nous proposons une solution pour la prise en compte de l'égalité durant l'extraction de motifs graduels.

# Chapitre 4

# Gradualité, typicité et atypicité

## 4.1 Introduction

Nous avons présenté dans le chapitre précédent deux algorithmes permettant d'extraire de manière automatique des itemsets graduels à partir de bases de données numériques. Les expérimentations ont montré que les deux méthodes étaient particulièrement adaptées sur de petites bases, mais qu'elles étaient très sensibles en terme de qualité de résultats au support minimal fixé par l'utilisateur. Un support trop faible résulte en une très grande quantité d'itemsets extraits, alors que paradoxalement, un seuil légèrement plus élevé conduit à un très faible nombre de résultats, bien souvent déjà connus des experts.

Dans ce chapitre, nous répondons aux problématiques suivantes : *comment extraire des règles graduelles potentiellement intéressantes pour l'expert ? Comment utiliser les itemsets graduels afin d'extraire des connaissances atypiques ?* Nous répondons au travers de deux propositions distinctes. Dans un premier temps, nous retravaillons la sémantique de la fréquence graduelle afin de proposer une mesure plus adaptée à la réalité de nos bases. Nous étudions différentes mesures, ainsi que la possibilité de les appliquer sur des bases réelles dans la section 4.2. Cette première partie se conclut par différentes expérimentations ainsi que sur la description d'un outil de navigation mis en place spécifiquement pour notre contexte (section 4.2.4).

Dans un second temps, nous exploitons la notion d'*élément local anormal*. Ce type d'élément, couramment désigné sous le terme "outlier", est remarquable car il ne suit pas l'évolution que suivent les autres éléments. Dans le contexte graduel, il s'agira pour nous de détecter les éléments qui suivent bien le sens de variation commun, mais dont la valeur est sensiblement plus élevée que les autres. Un bref état de l'art est dressé à la section 4.3.2, nous détaillons notre proposition à la section 4.3.3 et résumons différentes expérimentations à la section 4.3.4. Enfin, ce chapitre se termine par une discussion.

## 4.2 Sémantique du comptage dans le contexte graduel

Dans cette section, nous nous intéressons à la sémantique associée à la mesure de fréquence d'un itemset graduel. Dans le chapitre précédent, nous avions une vision optimiste de la fréquence, et c'est pour cette raison que les objets de valeurs égales participaient à l'incrémentation du support. Cependant, lorsque le jeu de données contient trop de valeurs égales, il est difficile d'extraire les sens de variation dominants. Dans ce cas, les fréquences d'itemsets de variation "*contradictoires*" (comme par exemple $(A^{\geq}B^{\geq})$ et $(A^{\geq}B^{\leq})$) seront très proches, voire égales. Nous recherchons donc à raffiner ces résultats. Pour ce faire, nous changeons la sémantique associée à la mesure de fréquence, ce qui entraîne la définition d'une nouvelle contrainte de fréquence.

### 4.2.1 Travaux existants

S'il existe de très nombreux travaux concernant l'adaptation de la mesure de confiance, comme par exemple la conviction et le lift [BMUT97], ou encore le "leverage" [PS91], les solutions alternatives à la contrainte de fréquence sont plus difficiles à mettre en œuvre. En effet, il s'agit ici de conserver de *bonnes propriétés*, comme par exemple l'anti-monotonie. Ainsi, dans [PHL01], les auteurs discernent trois types de contraintes :

- **Les contraintes anti-monotones :** il s'agit de contraintes vérifiant la propriété d'anti-monotonie ($\forall X \subseteq Y, C(Y) \Rightarrow C(X)$, par exemple la fréquence)
- **Les contraintes monotones :** il s'agit de contraintes vérifiant la propriété inverse à l'anti-monotonie, $\forall X \subseteq Y, C(X) \Rightarrow C(Y)$, comme par exemple la contrainte sum$(s.g_2) \geq 1$
- **Les contraintes succinctes :** pour déterminer si un itemset satisfait ou non une telle contrainte, il faut que chaque item composant l'itemset vérifie la contrainte. Par exemple, la contrainte $\min(s.prix) \geq v$ est succincte.

Les auteurs montrent ensuite qu'il existe des *contraintes convertibles anti-monotones (resp. monotones)* : si l'on considère un certain ordre sur les attributs, ces contraintes peuvent être converties afin de retrouver les propriétés d'anti-monotonie (resp. de monotonie). En outre, les auteurs démontrent qu'il n'est pas possible d'intégrer directement les contraintes convertibles dans un algorithme par niveau tel qu'Apriori.

[BL05] montre qu'au contraire ce type de contrainte peut-être utilisé à partir d'algorithmes par niveau. Les auteurs définissent ainsi les *contraintes anti-monotones relaxées* : si un itemset $s$ tel que $|s| > 2$ satisfait une telle contrainte, alors il existe au moins un sous-ensemble de $s$ de cardinalité $|s-1|$ qui satisfait la contrainte. Les auteurs introduisent ainsi une nouvelle classe de contraintes englobant les contraintes convertibles monotones, et définissent des algorithmes efficaces permettant d'exploiter ces propriétés. Un état de l'art très complet ainsi que de nouvelles approches basées sur les contraintes (via la transposition de contraintes notamment) peuvent être trouvées dans [Rio05]

Très récemment, des travaux ont été proposés afin de mesurer la force de variation d'une règle graduelle. Dans [MSSV08], une extension est proposée afin de mesurer l'amplitude de variation entre les couples d'objets. Plutôt que de considérer de manière binaire que la valeur entre deux objets augmente ou diminue, les auteurs quantifient cette variation en effectuant la différence de valeurs. Une mesure adaptée des dépendances floues proposée par les mêmes auteurs dans [BBS$^+$05] est alors utilisée à la place de la fréquence utilisée dans [BCS$^+$07].

[LLR09] proposent une méthode alternative combinant l'algorithme présenté au chapitre précédent et celui de [BCS$^+$07]. Il s'agit ici d'utiliser les règles graduelles dans le contexte de découverte de corrélations de rangs multiples. En effet, de par la définition de la fréquence, les règles graduelles obligent à conserver l'ordre des objets de la base d'un itemset à l'autre, ce qui les rend particulièrement adaptées à la recherche de rangs. Cependant, les fréquences telles que définies précédemment ne permettent pas de relier sémantiquement les règles au rang. Ainsi, plutôt que rechercher la plus grande liste d'ensembles, les auteurs recherchent le nombre de paires d'objets respectant un itemset. A partir des matrices binaires proposées au chapitre 3, il suffit de sommer le nombre de bits à 1. La méthode proposée se dégage alors de la recherche de la plus grande liste ordonnée d'objets, améliorant ainsi considérablement la complexité de notre méthode.

[KH10] recherchent également des corrélations de rang multiple en utilisant les règles graduelles. Les auteurs proposent une mesure basée sur les relations d'ordre floues nommées "rang gamma" et permettant de diminuer l'influence des objets dont la valeur est trop similaire. Cette mesure est utilisée à la place de la mesure de fréquence classique de l'algorithme Apriori. Les auteurs présentent également quelques résultats d'expérimentations sur une base œnologique et montrent que les règles graduelles

obtenues sont sémantiquement interprétables.

Les jeux de données construits à partir des expressions génomiques contiennent un domaine de valeur restreint. En effet, les expressions des gènes sont en premier lieu filtrées et normalisées, ce qui borne drastiquement leur valeur inférieure et supérieure. Lors de l'extraction d'itemsets graduels en utilisant les opérateurs de comparaison $\geq$ et $\leq$, nous obtenons trop de règles contradictoires, comme par exemple $Freq(A^{geq}B^{geq}) = Freq(A^{geq}B^{leq})$. Une analyse statistique de la base (présentée à la section 4.2.4) montre que la base contient des groupes de valeurs très proches, sans pour autant être strictement égales. De plus, les faibles différences de valeurs ne doivent pas être toujours prises en compte car elles peuvent provenir d'imperfections de mesures ou de prétraitement.

Les mesures proposées par [MSSV08] et [KH10] sont basées sur la théorie des sous-ensembles flous. Au-delà de nécessiter la définition de sous-ensembles flous par les experts, il est difficile d'intégrer ce type de mesure dans l'algorithme complet proposé précédemment. La mesure de fréquence proposée dans [LLR09] est une alternative très intéressante, mais elle ne satisfait pas les critères sémantiques définis par les experts. La principale différence réside dans la pondération choisie, basée sur le nombre de couples des objets de la base plutôt que sur la plus longue liste ordonnée. La mesure écarte les objets ne respectant pas la gradualité, mais ne prend pas en compte les valeurs plus ou moins proches.

Enfin, toutes ces alternatives sont présentées dans le contexte de la recherche de rang de corrélation multiple, ce qui induit des mesures de fréquences parfaitement adaptées à ce contexte. Cependant, nous souhaitons prendre en compte la sémantique de l'égalité des valeurs, ainsi que son impact sur la fréquence d'un itemset graduel. Même si nous parlons également de "*force de variation*", la sémantique associée ainsi que le but recherché ne sont pas les mêmes. Nous ne souhaitons pas déterminer des corrélations de rang, mais plutôt proposer de nouvelles mesures intégrant les spécificités des bases génomiques.

Dans la section suivante, nous étudions la prise en compte de l'égalité lors de l'évaluation du support. Nous présentons diverses mesures de fréquence répondant aux contraintes sémantiques relevées par les experts, et nous étudions leurs propriétés ainsi que leur possible intégration au sein des méthodes précédemment proposées.

### 4.2.2 Vers différentes mesures de fréquence

Afin d'illustrer nos propos, nous utiliserons la base de données du tableau 4.1. Cette base contient les caractéristiques principales rencontrées dans les cas réels : pour chaque itemset graduel considéré, la fréquence n'est pas systématiquement de 100% ; la base contient des valeurs égales, ainsi que des valeurs manquantes (représentées par le signe NA).

Dans la suite de cette section nous utiliserons les notations ainsi que le vocabulaire suivants :

- La *longueur d'un itemset graduel* est le nombre d'items le composant. On la note $|s|$
- La *concaténation* d'itemsets graduels est noté $s_1 \,.\, s_2$. Ainsi $|s_1 \,.\, s_2| = |s_1| + |s_2|$.
- La cardinalité du *domaine d'un item graduel* $i$ est noté $|dom(i)|$. Par exemple, $|dom(A)| = 5$
- Une *liste ordonnée d'objets* de la base par rapport à un itemset $s$ est noté $l_s = (o_1, ..., o_n)$. Par exemple, pour $s = C^{>}$, $l_s = (o_1, o_3, o_4)$ (avec les opérateurs $\{<, >\}$)

| Obj | A | B | C |
|---|---|---|---|
| $o_1$ | 0 | 1 | 11 |
| $o_2$ | 0 | 2 | 11 |
| $o_3$ | 1 | 4 | 12 |
| $o_4$ | 2 | 3 | 17 |
| $o_5$ | 0 | 3 | NA |
| $o_6$ | 3 | 4 | NA |
| $o_7$ | 3 | 6 | NA |
| $o_8$ | 4 | 3 | NA |

Tab 4.1 – Base d'exemple pour la recherche de support contenant 8 objets et 3 items

- L'ensemble de *toutes les listes ordonnées* de la base par rapport à un itemset $s$ est noté $\mathcal{L}_s$. Par exemple, $s = C^{>}$, $\mathcal{L}_s = \{(o_1, o_3, o_4), (o_2, o_3, o_4)\}$. (avec les opérateurs $\{<, >\}$)
- Nous sommes intéressés par les listes de $\mathcal{L}_s$ ayant la *cardinalité maximale*. De telles listes sont appelées listes maximales. Par abus de notation, on désignera cette cardinalité par $\mathcal{L}_s^{max} = \max_{l \in \mathcal{L}_s} |l|$. Par exemple, $\mathcal{L}_{c^{>}}^{max} = 3$
- Nous recherchons tous *les objets appartenant à une liste maximale*. Nous noterons cet ensemble $\mathcal{M} = \{o \in \mathcal{O} : \exists\, l, |l| = \mathcal{L}_s^{max}, o \in l\}$. Par exemple, pour $s = (A^{>}B^{>})$, $\mathcal{M} = \{o_2, o_3, o_4, o_5, o_6, o_7\}$
- La *fréquence* (ou support) d'un itemset graduel $s$ est notée $F(s)$
- L'*arrangement* de $k$ objets parmi $n$ est donné par la formule $A_n^k = \dfrac{n!}{(n-k)!}$

La notion de gradualité fait appel à l'ordonnancement des objets de la base en fonction d'un itemset graduel. Étant donné un ensemble $\mathcal{O}$, plusieurs sous-ensembles peuvent satisfaire l'itemset $s$. Nous cherchons à conserver une vision "optimiste" de la fréquence, tout en écartant les objets de valeurs égales. Ainsi, toutes les mesures présentées ci-dessous utilisent la fonction $\max$ afin de sélectionner le meilleur ensemble d'objets respectant un itemset, ce qui aboutit à la mesure de fréquence la plus élevée dans chaque contexte. Évidemment, il est possible d'adapter cette fonction à des contextes différents, par exemple en utilisant la fonction $\min$.

### Fréquence classique

La fréquence classique se base sur des notions de fréquence des règles d'association classiques. Il s'agit ici de faire un rapport entre l'ensemble des objets respectant un itemset donné et l'ensemble total des objets de la base. Cependant, le contexte graduel ajoute une dimension supplémentaire dont la prise en compte est primordiale : la dimension d'ordre. La mesure ci-dessous, présentée au chapitre précédent, est originalement donnée pour un algorithme utilisant les opérateurs $\{\leq, \geq\}$. On mesure le plus grand sous-ensemble d'objets ordonnés indépendamment des objets le composant.

**Définition 15.** *(Fréquence classique) Soit $s = (i_1^{*1} ... i_n^{*n})$ un itemset graduel et $\mathcal{L}_s$ l'ensemble des listes d'objets ordonnées respectant $s$. Alors la fréquence $F(s)$ est de $\dfrac{\mathcal{L}_s^{max}}{|\mathcal{O}|}$.*

Notons que dans ce contexte, il est possible de rencontrer des cycles dans le diagramme de Hasse

correspondant à la matrice binaire construite.

Avec l'utilisation des opérateurs $\{\leq, \geq\}$, nous obtenons :

$$\begin{aligned} F(A^{\geq}) &= 1 \\ F(B^{\geq}) &= 1 \\ F(C^{\geq}) &= 1 \\ F(A^{\geq}B^{\geq}) &= \frac{6}{8} = 0.75 \\ F(A^{\geq}C^{\geq}) &= \frac{4}{8} = 0.5 \\ F(B^{\geq}C^{\geq}) &= F(A^{\geq}B^{\geq}C^{\geq}) = \frac{3}{8} = 0.375 \end{aligned}$$

Il est possible de prendre en compte l'égalité des valeurs entre les objets afin que l'absence de variation n'augmente pas la fréquence, en utilisant les opérateurs $\{<, >\}$. Dans ce cas, les objets ayant une valeur égale par rapport à un item donné participeront à deux listes ordonnées bien distinctes. Par exemple, pour l'itemset $A^{\geq}$ nous obtenons $\mathcal{L}_{A^{\geq}} = \{o_1, o_2, o_5, o_3, o_4, o_6, o_7, o_8\}$ et pour l'itemset $A^{>}$ nous obtenons $\mathcal{L}_{A^{>}} = \{(o_1, o_3, o_4, o_6, o_7), (o_1, o_3, o_4, o_7, o_8), (o_2, o_3, o_4, o_6, o_7), (o_2, o_3, o_4, o_7, o_8), (o_5, o_3, o_4, o_6, o_7)$ $, (o_5, o_3, o_4, o_7, o_8)\}$, car $o_1 = o_2 = o_5$ et $o_6 = o_7$. Ces égalités conduisent donc à la construction de $3 \times 2 = 6$ listes ordonnées différentes. Les listes construites au premier niveau sont illustrées par les figures 4.1a, 4.1b et 4.1c. Ces constatations conduisent à la propriété suivante :

**Propriété 5.** *Soit $s = i_1^{*_1}...i_n^{*_n}$ un itemset graduel tel que $*_1...*_n \in \{<, >\}$. Alors*

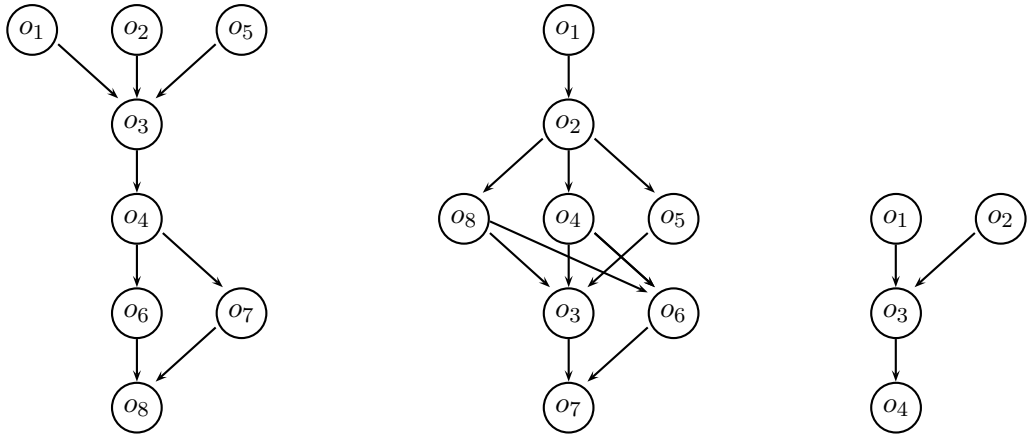
$$Freq(s) \leq \frac{\min\limits_{i \in s}(|dom(i)|)}{|\mathcal{O}|}$$

.

Nous obtenons alors les fréquences suivantes :

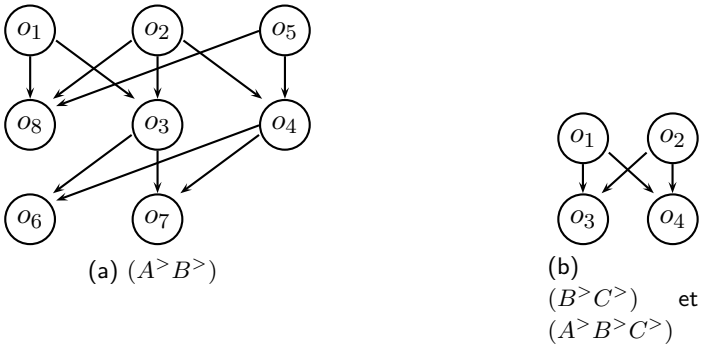
$$\begin{aligned} F(A^{>}) &= F(B^{>}) = \frac{5}{8} = 0.625 \ (\leq \frac{|dom(A)|}{|\mathcal{O}|}, \leq \frac{|dom(B)|}{|\mathcal{O}|}) \\ F(C^{>}) &= F(A^{>}B^{>}) = F(A^{>}C^{>}) = \frac{3}{8} = 0.375 \ (\leq \frac{|dom(C)|}{|\mathcal{O}|}) \\ F(B^{>}C^{>}) &= F(A^{>}B^{>}C^{>}) = \frac{2}{8} = 0.25 \ (\leq \frac{|dom(C)|}{|\mathcal{O}|}) \end{aligned}$$

La fréquence classique utilisant les opérateurs $\{<, >\}$ est anti-monotone. Cependant, cette mesure de fréquence est très contraignante car elle ne prend pas en compte la proportion d'objets participant à une fréquence. Par exemple, le diagramme de Hasse de la figure 4.1c possède un chemin de longueur maximale égale à celui de la figure 4.2a. Or il est évident dans le second graphe que presque tous les objets participent à la fréquence, alors que dans le premier il n'y en a que 50%. Cette mesure de fréquence n'est donc pas adaptée à l'utilisation des opérateurs $\{<, >\}$, car elle ne prend en compte que la longueur des diagrammes sans en considérer la largeur.




(a) $A^{>}$ (b) $B^{>}$ (c) $C^{>}$ et $(A^{>}C^{>})$

Fig 4.1 – Diagrammes de Hasse obtenus pour les items graduels de la base 4.1 en excluant l'égalité




(a) $(A^{>}B^{>})$ (b) $(B^{>}C^{>})$ et $(A^{>}B^{>}C^{>})$

Fig 4.2 – Diagrammes de Hasse obtenus pour les itemsets graduels de la base 4.1 en excluant l'égalité

**Support basé sur l'arrangement**

L'une des solutions envisagées consiste à considérer toutes les combinaisons ordonnées possibles pour un ensemble d'objets et le nombre de combinaisons effectivement réalisées. Pour un itemset graduel $s$, la taille de la plus longue liste ordonnée sera au mieux de $min_{(i \in s)}(|dom(i)|)$. Il s'agira alors de recenser les objets réalisant un ou plusieurs de ces chemins maximaux. Nous définissons la **fréquence combinatoire** de la manière suivante :

**Définition 16.** *(fréquence combinatoire) Soit $s = (i_1^{*1} ... i_n^{*n})$ un itemset graduel et $m = min_{i \in s}(|dom(i)|)$ la plus petite cardinalité de domaine parmi les items composant $s$. Il existe donc $A^m_{|\mathcal{O}|}$ chaînes maximales. La fréquence combinatoire est définie par $F(s) = \dfrac{|\mathcal{M}|}{A^m_{|\mathcal{O}|}}$*

En utilisant la définition 16 nous obtenons les fréquences suivantes :

$$\begin{aligned} F(A^{>}) &= \frac{8}{A_8^5} = 0.001 \\ F(B^{>}) &= \frac{8}{A_8^5} = 0.001 \\ F(C^{>}) &= \frac{4}{A_8^3} = 0.01 \\ F(A^{>}B^{>}) &= \frac{7}{A_8^5} = 0.001 \\ F(A^{>}C^{>}) &= \frac{4}{A_8^3} = 0.01 \\ F(B^{>}C^{>}) &= \frac{4}{A_8^3} = 0.01 \\ F(A^{>}B^{>}C^{>}) &= \frac{4}{A_8^3} = 0.01 \end{aligned}$$

Cependant, cette mesure n'est pas la plus adaptée pour l'extraction de règles graduelles, principalement pour deux raisons. Tout d'abord, la fréquence perd la propriété d'anti-monotonie : pour s'en convaincre, il suffit de remarquer que $F(A^{>}B^{>}) < F(A^{>}B^{>}C^{>})$. Ensuite, l'utilisation de la formule d'arrangement augmente fortement la complexité de notre algorithme. En effet, ce calcul est connu comme étant long, voire parfois irréalisable pour certaines valeurs. Enfin, nous notons que les valeurs de fréquence sont extrêmement faibles : le nombre de *solutions possibles* augmente très rapidement, alors que le nombre de *solutions réalisées* reste faible, ce qui induit une proportion bien souvent très faible.

Les arguments cités ci-dessus rendent la mesure de fréquence combinatoire inadaptée à l'extraction d'itemsets graduels, bien que la sémantique qu'elle véhicule reflète la logique de comptage attendue. Nous nous tournons donc vers des solutions de pondération afin de conserver un maximum d'objets respectant la fréquence tout en pénalisant ceux dont les valeurs sont égales.

### Fréquence locale pondérée

Idéalement, la mesure de fréquence doit correspondre à la sémantique suivante : "*la proportion d'objets qui participent à la gradualité d'un itemset*". Cette constatation nous amène non plus à mesurer uniquement la *longueur du chemin maximal obtenu*, mais également à considérer le *nombre d'objets réalisant l'un de ces chemins*. Nous définissons la **fréquence locale pondérée** de la manière suivante :

**Définition 17.** *(fréquence locale pondérée) Soit $s = (i_1^{*1}...i_n^{*n})$ un itemset graduel, $|\mathcal{M}|$ le nombre d'objets appartenant à une liste ordonnée maximale associé à l'itemset $s$. La fréquence locale pondérée est définie par :*

$$F(s) = \frac{|\mathcal{M}|}{|\mathcal{L}_s|} \times \frac{\mathcal{L}_s^{max}}{\min\limits_{i \in s}(|dom(i)|)}$$

Nous proposons donc de pondérer la fréquence par la proportion du nombre d'objets participant à un chemin maximal. Cette méthode permet de mettre en évidence la "force de gradualité" de l'itemset, car

elle prend en compte le *chemin maximal obtenu* et les objets réalisant l'un de ces chemins maximaux. En utilisant la définition 17, nous obtenons les fréquences suivantes :

$$\begin{aligned} F(A^{\geq}) &= \frac{8}{8} \times \frac{5}{5} = 1 \\ F(B^{\geq}) &= \frac{8}{8} \times \frac{5}{5} = 1 \\ F(C^{\geq}) &= \frac{4}{8} \times \frac{3}{3} = 0.5 \\ F(A^{\geq}B^{\geq}) &= \frac{7}{8} \times \frac{3}{5} = 0.52 \\ F(A^{\geq}C^{\geq}) &= \frac{4}{4} \times \frac{3}{3} = 1 \\ F(B^{\geq}C^{\geq}) &= \frac{4}{4} \times \frac{2}{3} = 0.66 \\ F(A^{\geq}B^{\geq}C^{\geq}) &= \frac{4}{4} \times \frac{2}{3} = 0.66 \end{aligned}$$

Cette mesure présente le fort avantage d'une complexité réduite comparée à celle de la fréquence combinatoire tout en prenant en compte le degré de gradualité réalisé. Cependant, cette fréquence n'est pas anti-monotone : $F(A^{>}B^{>}) < F(A^{>}B^{>}C^{>})$. Cela s'explique par le fait que nous considérons le nombre d'objets du graphe associé à l'itemset pour la pondération. Par exemple, nous divisons par 8 pour $F(A^{>}B^{>})$, alors que nous ne divisons que par 4 pour $F(A^{>}B^{>}C^{>})$. Cette dernière remarque nous amène à globaliser notre mesure de fréquence.

### Fréquence globale pondérée

Nous considérons que la fréquence locale est composée de deux parties distinctes : d'un côté la proportion des objets réalisant la gradualité (premier membre de la multiplication) et d'un autre côté le degré de gradualité réalisé. La seconde partie de la fréquence est locale car elle se réfère à l'itemset $s$. Il est difficile de se détacher de cette localité. En revanche, la première partie de la formule peut être "globalisée" en divisant non plus par le nombre d'objets des listes ordonnées associées à $s$, mais par le nombre total d'objets de la base. Nous obtenons alors la définition suivante :

**Définition 18.** *(fréquence globale pondérée) Soit $s = (i_1^{*1}...i_n^{*n})$ un itemset graduel et $|\mathcal{O}|$ le nombre d'objets de la base de données. La fréquence globale est définie par :*

$$F(s) = \frac{|\mathcal{M}|}{|\mathcal{O}|} \times \frac{\mathcal{L}_s^{max}}{min_{(i \in s)}(|dom(i)|)}$$

.

En utilisant la définition 18, nous obtenons les fréquences suivantes :

$$F(A^{>}) = \frac{8}{8} \times \frac{5}{5} = 1$$
$$F(B^{>}) = \frac{8}{8} \times \frac{5}{5} = 1$$
$$F(C^{>}) = \frac{4}{8} \times \frac{3}{3} = 0.5$$
$$F(A^{>}B^{>}) = \frac{7}{8} \times \frac{3}{5} = 0.52$$
$$F(A^{>}C^{>}) = \frac{4}{8} \times \frac{3}{3} = 0.5$$
$$F(B^{>}C^{>}) = \frac{4}{8} \times \frac{2}{3} = 0.3$$
$$F(A^{>}B^{>}C^{>}) = \frac{4}{8} \times \frac{2}{3} = 0.3$$

Cette nouvelle mesure de fréquence possède plusieurs propriétés, que nous démontrons dans la suite de cette section. Nous répondons aux questions suivantes : *la fréquence globale pondérée est-elle anti-monotone ? Est-il possible de remplacer la mesure de fréquence classique par la fréquence globale pondérée ?* Afin de répondre à ces différentes questions, nous utilisons le lemme technique 1, qui se base sur l'intersection des listes ordonnées d'objets :

**Lemme 1.** *Soit* $(s.s_i.s_j)$ *un itemset graduel obtenu à partir de* $(s.s_i)$ *et* $(s.s_j)$. *Si* $(o * o') \notin \mathcal{L}_{(s.s_i)}$, *alors* $(o * o') \notin \mathcal{L}_{(s.s_i.s_j)}$. *D'où* $\mathcal{L}^{max}_{(s.s_i)} \geq \mathcal{L}^{max}_{(s.s_i.s_j)}$ *et* $\mathcal{L}^{max}_{(s.s_j)} \geq \mathcal{L}^{max}_{(s.s_i.s_j)}$.

**Propriété 6.** *La mesure de fréquence globale pondérée n'est pas anti-monotone*

*Démonstration.* Nous utilisons les notations simplifiées suivantes : $n_s = |\mathcal{M}_s|$, $m_s = \mathcal{L}^{max}_s$, $M_s = min_{i \in s}(|dom(i)|)$

Nous raisonnons par l'absurde. Supposons que la mesure de fréquence est anti-monotone.
Soit $s = s_n.s_i \prec s' = s_n.s_i.s_j$. Alors $F(s) \geq F(s')$.
Par définition,

$$\frac{n_s}{|\mathcal{O}|} \times \frac{m_s}{M_s} \geq \frac{n_{s'}}{|\mathcal{O}|} \times \frac{m_{s'}}{M_{s'}} \tag{4.1}$$
$$\Leftrightarrow n_s \times \frac{m_s}{M_s} \geq n_{s'} \times \frac{m_{s'}}{M_{s'}} \tag{4.2}$$

Or, nous savons que $n_s \geq n_{s'}$, nous nous intéressons donc uniquement à l'inégalité $\frac{m_s}{M_s} \geq \frac{m_{s'}}{M_{s'}}$. Par définition,

$$M_{s'} = \begin{cases} M_{s.s_i} \text{ si } |dom(s_i)| < |dom(s_j)| \\ M_{s.s_j} \text{ sinon} \end{cases}$$

Si $M_{s'} = M_{s.s_i}$, alors nous obtenons

$$\frac{m_s}{M_{s.s_i}} \geq \frac{m_{s'}}{M_{s.s_i}} \tag{4.3}$$
$$\Leftrightarrow m_s \geq m_{s'} \tag{4.4}$$

Ce qui prouve que dans ce cas la fréquence est anti-monotone.

Si $M_{s'} = M_s$, alors

$$\frac{m_s}{M_{s.s_i}} \geq \frac{m_{s'}}{M_{s.s_j}} \tag{4.5}$$

$$\text{Or } M_{s.s_i} > M_{s.s_j} \Rightarrow \frac{1}{M_{s.s_i}} < \frac{1}{M_{s.s_j}} \tag{4.6}$$

L'équation 4.6 montre qu'il existe des cas où la fréquence n'est pas anti-monotone, par exemple $\frac{(m_s=3)}{(M_{s.s_i}=10)} < \frac{(m_{s'}=2)}{(M_{s.s_j}=5)}$. □

**Propriété 7.** *(monotonie) La mesure de fréquence globale pondérée n'est pas monotone.*

*Contre exemple.* Soit $\alpha = 0.5$ un seuil de fréquence minimale, alors $(A^{>}B^{>})$ est fréquent, mais pas $(A^{>}B^{>}C^{>})$, ce qui contredit la propriété de monotonie.

**Lemme 2.** *Soit $s.s_i.s_j$ un itemset graduel. Alors $F(s.s_i.s_j) > F(s.s_i) \Rightarrow F(s.s_j) > F(s.s_i.s_j)$*

*Démonstration.* Dans la démonstration de la propriété 6 nous montrons que nous avons $F(s.s_i.s_j) > F(s.s_i)$ si et seulement si $|dom(s_j)| < |dom(s_i)|$ (équation 4.6).
Nous raisonnons par l'absurde. Supposons $F(s.s_i.s_j) > F(s.s_j)$. Alors, par définition :

$$\frac{n_{s.s_i.s_j}}{|\mathcal{O}|} \times \frac{m_{s.s_i.s_j}}{M_{s.s_i.s_j}} \geq \frac{n_{s.s_j}}{|\mathcal{O}|} \times \frac{m_{s.s_j}}{M_{s.s_j}} \tag{4.7}$$

$$\Leftrightarrow \quad n_{s.s_i.s_j} \times \frac{m_{s.s_i.s_j}}{M_{s.s_i.s_j}} \geq n_{s.s_j} \times \frac{m_{s.s_j}}{M_{s.s_j}} \tag{4.8}$$

Or $M_{s.s_i.s_j} = M_{s.s_j}$ car $F(s.s_i.s_j) > F(s.s_i)$. Donc

$$n_{s.s_i.s_j} \times \frac{m_{s.s_i.s_j}}{M_{s.s_i.s_j}} \geq n_{s.s_j} \times \frac{m_{s.s_j}}{M_{s.s_j}} \tag{4.9}$$

$$\Leftrightarrow \quad n_{s.s_i.s_j} \times \frac{m_{s.s_i.s_j}}{M_{s.s_j}} \geq n_{s.s_j} \times \frac{m_{s.s_j}}{M_{s.s_j}} \tag{4.10}$$

$$\Leftrightarrow \quad n_{s.s_i.s_j} \times m_{s.s_i.s_j} \geq n_{s.s_j} \times m_{s.s_j} \tag{4.11}$$

$$\text{Or} \quad n_{s.s_i.s_j} \geq n_{s.s_i} \tag{4.12}$$

$$\Rightarrow \quad m_{s.s_i.s_j} \geq m_{s.s_j} \tag{4.13}$$

L'équation 4.13 est en contradiction avec le lemme 1. Donc $F(s.s_i) > F(s.s_i.s_j)$ □

### 4.2.3 Extraction d'itemset graduels avec fréquence globale pondérée

Habituellement, les algorithmes de type générer-élaguer considèrent les items dans un ordre lexicographique. Par exemple, le tableau 4.2 représente l'ordre de génération des itemsets graduels pour la base de données 4.1 en utilisant l'opérateur $\geq$. L'ordre de génération suit l'ordre des lignes du tableau.

Lorsque l'on substitue la fréquence globale pondérée à la fréquence classique, la méthode d'énumération par ordre lexicographique n'est pas complète, comme le montre l'exemple 12 :

| Niveau | Itemsets Candidats |
|---|---|
| 1 | $A^{\geq}$<br>$B^{\geq}$<br>$C^{\geq}$ |
| 2 | $(A^{\geq}B^{\geq})$<br>$(A^{\geq}C^{\geq})$<br>$(B^{\geq}C^{\geq})$ |
| 3 | $(A^{\geq}B^{\geq}C^{\geq})$ |

Tab 4.2 – Ordre de génération des itemsets graduels pour l'opérateur $\geq$

**Exemple 12.** *Supposons un seuil de fréquence minimal $\alpha$ fixé par l'utilisateur. La fréquence globale pondérée n'étant pas anti-monotone, il est possible de rencontrer des itemsets tels que $F(i_1^{>}i_2^{>}i_3^{>}) \geq \alpha$ et $F(i_1^{>}i_2^{>}) < \alpha$. En suivant l'ordre alpha-numérique, l'itemset graduel $(i_1^{>}i_2^{>})$ sera élagué, ne permettant pas la découverte de l'itemset graduel $(i_1^{>}i_2^{>}i_3^{>})$.*

Nous proposons donc de considérer l'ordre de génération en fonction de **l'ordre croissant des cardinalités de domaine**. Dans l'exemple du tableau 4.1, nous avons $|dom(A)| = 5$, $|dom(B)| = 5$ et $|dom(C)| = 3$. Ainsi, nous générerons les itemset graduels dans l'ordre donné par les lignes du tableau 4.3 :

| Niveau | Itemsets Candidats |
|---|---|
| 1 | $C^{>}$<br>$A^{>}$<br>$B^{>}$ |
| 2 | $(C^{>}A^{>})$<br>$(C^{>}B^{>})$<br>$(A^{>}B^{>})$ |
| 3 | $(C^{>}A^{>}B^{>})$ |

Tab 4.3 – Ordre de génération proposé

Notons que la contrainte de fréquence globale pondérée appartient à la classe des contraintes convertible présentées dans [PHL01]. Une contrainte est convertible anti-monotone si à partir d'un ordre donné sur les items, si un itemset $s$ satisfait cette contrainte, alors *tous* les préfixes de cet itemset (dans le même ordre) satisfont également cette contrainte. C'est ce que démontre le théorème 2. [PHL01] établit que l'on ne peut pas utiliser des contraintes convertibles avec les techniques d'extraction par niveau.

La fréquence globale pondérée appartient à la classe des *contraintes anti-monotones avec perte* définie par [BL05], puisque cette classe est une généralisation de la classe des contraintes convertibles. De manière informelle, si un itemset $s$ satisfait une contrainte anti-monotone avec perte, alors il existe au moins un itemset de longueur $|s| - 1$ qui satisfait cette contrainte. Dans [BL05], les auteurs proposent une méthode basée sur la réduction de données afin d'extraire les itemsets contraints à partir d'algorithmes par niveau : si une transaction $t$ ne contient pas au moins un item satisfaisant la contrainte de

fréquence classique et la contrainte anti-monotone sans perte, alors il peut être éliminé de la base de données. Ainsi, la méthode proposée réduit la taille de la base de données originale à chaque itération de l'algorithme.

Cette méthode ne peut pas être utilisée dans notre contexte : dans [BL05], les contraintes anti-monotones sans perte sont utilisées en conjonction avec la contrainte de fréquence classique, et la preuve de complétude se base entre autre sur l'utilisation de cette contrainte de fréquence classique. Or, notre fréquence globale pondérée n'est pas en conjonction avec la fréquence classique, mais plutôt la pénalise en fonction du nombre de valeurs égales. De plus, la méthode de [BL05] consiste à éliminer une transaction de la base, or le contexte graduel induit une comparaison entre transactions. Dans le cas où cette transaction n'est pas totalement isolée (c'est-à-dire qu'elle appartient à la liste des solutions $\mathcal{L}_s$), il n'est pas possible de prédire au niveau $k$ si cette transaction n'appartiendra pas à la liste solution maximale à un niveau ultérieur. Une transaction ne peut être éliminée que dans le cas unique où elle n'appartient pas à l'ensemble des solutions $\mathcal{L}_s$.

Ces arguments nous montrent que l'utilisation de la méthode de [BL05] n'est pas applicable telle quelle dans le contexte graduel. Cependant, la démonstration du théorème 2 indique qu'une telle contrainte peut être substituée à la contrainte de fréquence dans une approche par niveau.

**Théorème 2.** *(complétude) Soit $DB$ une base de donnée transactionnelle et $\alpha$ un seuil de fréquence minimale. Alors la méthode basée sur l'ordre croissant des cardinalités de domaine extrait tous les itemsets graduels tels que $F(s) \geq \alpha$.*

*Démonstration.* Soit l'itemset graduel $s.s_i.s_j$ tel que $F(s.s_i.s_j) > \alpha$. Par construction, $s.s_i.s_j$ sera généré

- Soit par concaténation de $s.s_i$ et $s.s_j$ (a)
- Soit par concaténation de $s.s_j$ et $s.s_i$ (b)

Pour construire $s.s_i.s_j$, trois cas se présentent :

Cas 1 : $F(s.s_i) \geq \alpha$ et $F(s.s_j) \geq \alpha$. Par construction, $s.s_i.s_j$ sera généré.

Cas 2 : $F(s.s_i) < \alpha$ et $F(s.s_i.s_j) \geq \alpha$.
Alors $F(s.s_j) \geq \alpha$ d'une part (lemme 2), et $|dom(s_j)| < |dom(s_i)|$ d'autre part (démonstration 4.2.2, équation 4.6).
Par construction $s.s_i.s_j$ ne sera pas généré par (a) mais par (b), puisque $|dom(s_j)| < |dom(s_i)|$. $s.s_i.s_j$ sera donc découvert.

Cas 3 : $F(s.s_j) < \alpha$ et $F(s.s_i.s_j) \geq \alpha$.
Nous utilisons le même raisonnement que pour le cas 2 : $F(s.s_i) \geq \alpha$ d'une part (lemme 2), et $|dom(s_i)| < |dom(s_j)|$ d'autre part (démonstration 4.2.2, équation 4.6).
Par construction $s.s_i.s_j$ ne sera pas générée par (b) mais par (a), puisque $|dom(s_i)| < |dom(s_j)|$. $s.s_i.s_j$ sera donc découvert.
Dans tous les cas, l'itemset graduel $s.s_i.s_j$ est généré. S'il est fréquent, il sera alors découvert. □

### 4.2.4 Expérimentations

Dans cette section, nous présentons les expérimentations menées à partir de la base de données génomique du cancer du sein. Notre but est de comparer l'impact de cette nouvelle mesure en terme de nombre d'itemsets graduels extraits et présentés à l'expert. Dans un second temps, nous présentons le prototype d'un outil de visualisation simple, mais adapté à un utilisateur non informaticien.

La base génomique contient 4408 items pour 108 patients. Un item correspond à l'expression d'un gène, et est identifié par le chromosome sur lequel se trouve ce gène, ainsi que par sa position. Une fois normalisée, l'expression d'un gène est comprise entre -2 et 3. Le tableau 4.4 montre les classifications des valeurs d'expression utilisées par les biologistes de l'IRCM de Montpellier. Les experts recherchent les gènes se sur-exprimant ou se sous-exprimant, et n'étudient pas les gènes qui s'expriment normalement. Ainsi, un premier prétraitement consiste à mettre à 0 toutes les valeurs comprises dans la borne $[-0.15, 0.138[$.

| Intervalle d'expression | | Type d'expression |
|---|---|---|
| $[-2, -0.15[$ | $\leftrightarrow$ | sous-exprimé |
| $[-0.15, 0.138[$ | $\leftrightarrow$ | normalement exprimé |
| $[0.138, 3]$ | $\leftrightarrow$ | sur-exprimé |

Tab 4.4 – Type d'expression en fonction des valeurs d'expression

Les experts appliquent également un filtrage de qualité : sur l'ensemble des patients, les items ayant moins de 50% de valeurs renseignées sont éliminés. A l'issue de ce traitement, nous obtenons une base contenant 4259 items à laquelle nous ajoutons 5 items cliniques.

Cependant, la base obtenue reste trop dense pour être fouillée dans des temps acceptables d'une part, et réactive à la baisse de support d'autre part. Ainsi, nous avons décidé sur les conseils de nos experts de découper cette base en plusieurs sous-bases en fonction des chromosomes. Le tableau 4.6 illustre les bases obtenues.

| Nom | Opérateurs |
|---|---|
| Fréquence classique (FC) | $\{\leq, \geq\}$ |
| Fréquence classique avec Egalité (FCE) | $\{<, >\}$ |
| Fréquence Globale Pondérée (FGP) | $\{<, >\}$ |

Tab 4.5 – Type de fréquences utilisées et opérateurs associés

Nous comparons les différentes fréquences énumérées au tableau 4.5. Les valeurs d'expressions sont représentées par des décimales à trois chiffres après la virgule. Dans ce contexte particulier, une égalité stricte n'a pas de sens. En effet, 0.002 et 0.004 sont des valeurs suffisement proches pour être considérées comme égales. Nous nous basons donc sur un seuil minimal de différence fixé par l'utilisateur (définition 19) lors de la construction de la matrice binaire.

**Définition 19.** *Soient $o$ et $o'$ deux objets, $i$ un item et $\Delta = abs(t_o[i] - t_{o'}[i])$. On considère que $t_o[i] = t_{o'}[i]$ si $\Delta < \alpha$, avec $\alpha$ fixé par l'utilisateur.*

| Intervalle d'attributs | Chromosomes | Nombre d'items |
|---|---|---|
| 0-518 | 1 | 518 |
| 519-1094 | 2, 3 | 576 |
| 1095-1580 | 4, 5 | 485 |
| 1581-2161 | 6, 7, 8 | 580 |
| 2162-2697 | 9, 10, 11 | 535 |
| 2698-3180 | 12, 13, 14 | 482 |
| 3181-3594 | 15, 16, 17 | 413 |
| 3595-3958 | 18, 19, 21 | 363 |
| 3959-4259 | 22, X, Y | 300 |

Tab 4.6 – Bases de données préparées pour les expérimentations

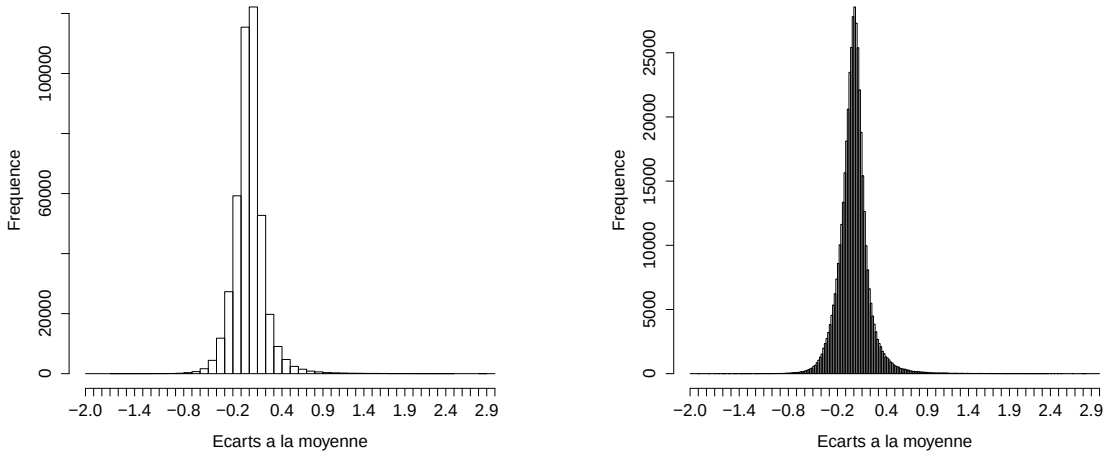
Un histogramme des écarts est utilisé afin de déterminer à partir de quelle variation nous considérons que les valeurs ne sont plus assez proches pour être considérées égales. La figures 4.3a montre l'histogramme des écarts par colonne. Ce calcul a été effectué en prenant la distance à la moyenne pour chaque colonne (bac) puis en additionnant les fréquences des écarts sur l'ensemble de la base. On note une écrasante majorité de faibles écarts (environ 0.1).

La figure 4.3b montre un diagramme avec une granularité plus fine. Cela montre que l'on est en présence de données prétraitées : centrées autour de zéro et avec une très forte concentration avec valeurs ayant un faible écart. Afin de comparer, l'histogramme a été réalisé en utilisant la moyenne globale (sur toute la base). La figure 4.3b montre des résultats similaires à l'utilisation de la moyenne par attribut. En nous basant sur ces constatations, nous avons fixé $\alpha = 0.1$, ce qui signifie que pour être considérée comme participant à la gradualité, la variation minimale entre deux objets doit être supérieure à 0.1.

Nous avons ensuite lancé l'algorithme d'extraction des règles utilisant les trois fréquences du tableau 4.5 pour les seuils minimaux suivants : 90%, 80%, 70%, 60%, 50% et 20%.

**Résultats**

Les figures 4.5a à 4.7c montrent les résultats obtenus sur les 9 sous-bases de données. Pour la grande majorité des bases, la fréquence classique utilisant les opérateurs $\{<, >\}$ s'avère inadaptée. Les figures 4.5b et 4.5c montrent qu'il n'y a pas de motifs fréquents pour un seuil minimal de 50%. Cela signifie que la cardinalité de domaine minimale est une contrainte très forte sur cette base, d'autant plus que soit le seuil est trop élevé et il y a très peu (voire pas du tout) d'itemsets extraits, soit le seuil est trop faible et le nombre de motifs explose, ce qui rend l'interprétation des résultats très difficile.




(a) (b)

Fig 4.3 – (a) Histogramme des écarts par colonne (b) Histogramme des écarts par rapport à la moyenne

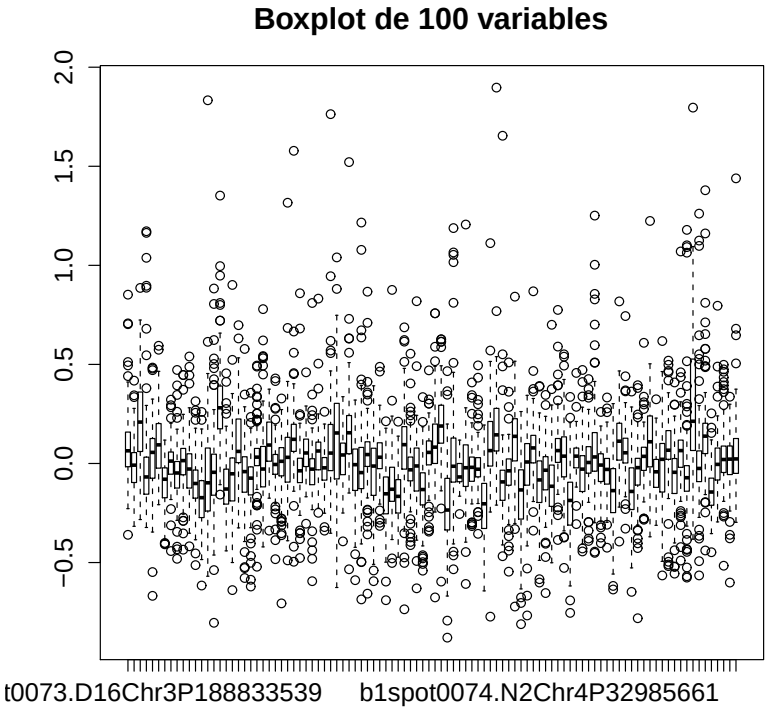



Fig 4.4 – Boxplot pour 100 éléments au hasard

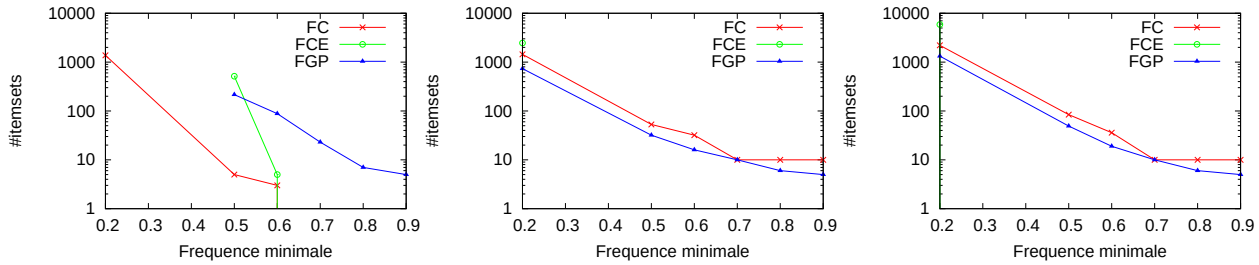



Fig 4.5 – Comparaisons du nombre de motifs obtenus en fonction de la base de fréquence pour (a) i518 (b) i1096 et (c) i1580

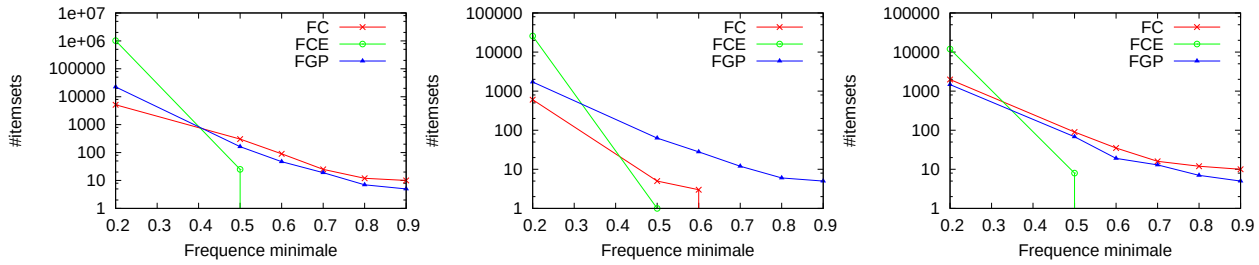



Fig 4.6 – Comparaisons du nombre de motifs obtenus en fonction de la base de fréquence pour (a) i2161 (b) i2697 et (c) i3180

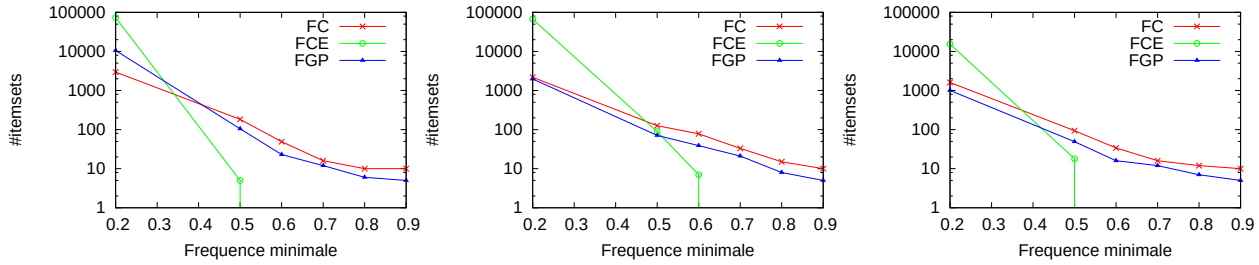



Fig 4.7 – Comparaisons du nombre de motifs obtenus en fonction de la base de fréquence pour (a) i3594 (b) i3958 et (c) i4259

Les fréquences FC et FGP montrent des résultats intéressants. Sur des seuils minimaux de fréquence allant de 90% à 50%, la FGP extrait moins de motifs que la fréquence classique avec opérateurs $\{\leq, \geq\}$. Cela signifie que l'égalité participe activement à former de plus long chemins. Une étude plus poussée des motifs obtenus avec une fréquence d'au moins 90% sur ces bases montre que ce sont les items provenant de la base clinique qui sont corrélés. En effet, les domaines de ces attributs sont plutôt faibles, à l'image par exemple du grade d'une tumeur, dont les valeurs sont 1, 2 ou 3.

En revanche, avec un seul minimal de fréquence à 20%, la FGP extrait plus d'itemsets que la FC. Ce phénomène s'explique par le fait suivant : avec la fréquence classique, il est toujours possible de fabriquer des plus petites chaînes, mais plus difficile de corréler ces plus petites chaînes sur plusieurs itemsets. La figure 4.8 le montre bien : le nombre d'itemsets extraits avec la FGP est plus élevé notamment car l'algorithme parvient à des niveaux plus profonds. Cela signifie qu'il y a une corrélation entre beaucoup d'items (jusqu'à 12 pour la base I2161) lorsque l'on considère le chemin maximal *réalisé* par rapport au chemin maximal *possible*. Ces itemsets sont d'autant plus renforcés que plusieurs objets de la base participent à ce phénomène.

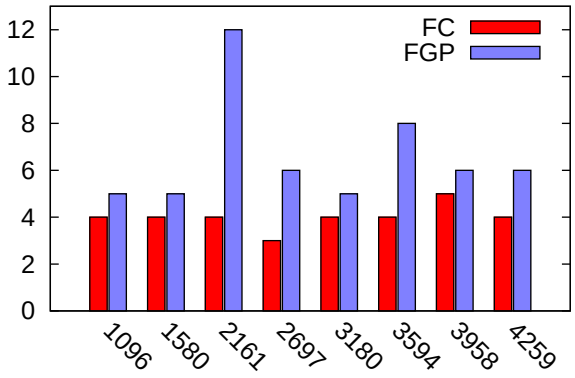



Fig 4.8 – Longueurs des itemsets extraits pour un seuil de fréquence minimal à 20%

Afin de présenter les résultats à un expert non informaticien, nous avons opté pour une interface graphique simple et pratique. Habituellement, la recherche de règles d'association graduelles est effectuée comme un post-traitement à partir de l'ensemble des itemsets extraits. Dans notre cas, nous avons souhaité ajouter une couche d'abstraction supplémentaire en effectuant un nouveau processus de fouille sur les résultats. Les règles d'association démultiplient les résultats à partir d'itemsets fréquents. Afin d'éviter ce phénomène, nous appliquons l'algorithme Apriori sur les résultats. Dans ce cas, chaque itemset est considéré comme une transaction. Notons que nous aurions pu utiliser des algorithmes de clustering tels que [JNK06, XSBT09], mais ici, notre but était de présenter les implications entre items.

La figure 4.9 montre une prise d'écran de l'interface de présentation des résultats proposée. Dans un souci de portabilité, celle-ci a été développée en php5 et en Ajax. Dans un premier temps, l'utilisateur choisit le fichier de résultats qu'il veut visionner, ainsi que les seuils de fréquence et de confiance minimales qu'il souhaite (haut de la figure 4.9). Les règles d'association extraites sont ensuite présentées et permettent à l'utilisateur d'avoir une vision rapide des itemsets graduels fréquemment corrélés. Dans notre exemple, 3 règles sont extraites et montrent que 5 items sont fréquemment corrélés. L'utilisateur peut ensuite consulter les itemsets du jeu de données sélectionné, en surlignant certains items graduels.

L'outil est fonctionnel : si l'utilisateur le souhaite, il peut ne visionner que les itemsets contenant un ou plusieurs items choisis, ou alors simplement visionner l'intégralité du jeu de données. Dans la figure 4.9, l'utilisateur a choisi de visionner les itemsets contenant les items Grade : :Richardson, OVS et b1spot0058-I5Chr11P127767333. Notons que cette interface est encore perfectible, et qu'il existe d'autres pistes de visualisation [AHL09, XSBT09]. D'autre part, un outil de visualisation permettant, dans la même philosophie, à l'utilisateur de sélectionner les règles qu'il souhaite voir a été proposé dans [MNSVS05].

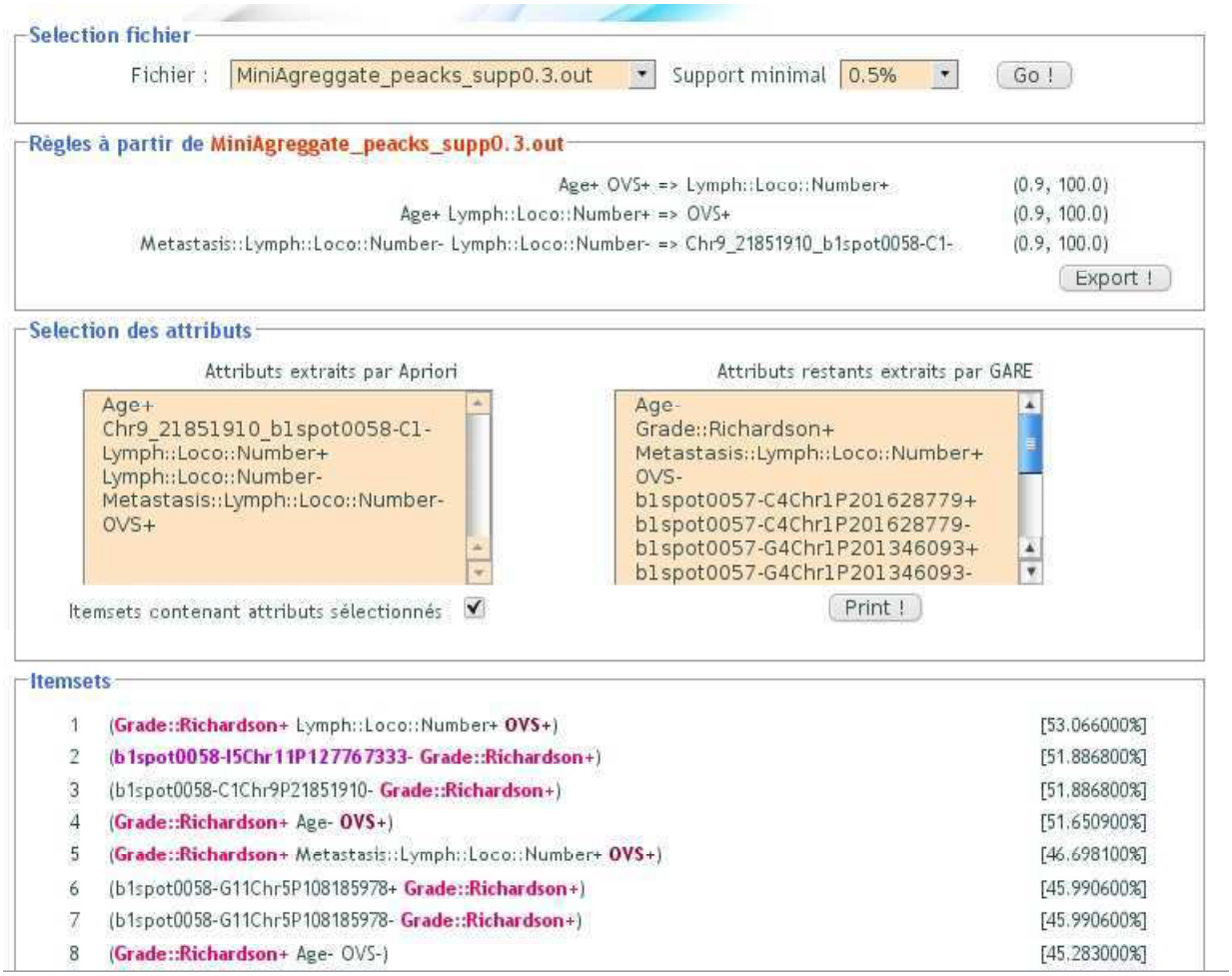



Fig 4.9 – Interface présentée à l'utilisateur

### 4.2.5 Conclusion

Nous avons présenté dans ce début de chapitre une nouvelle fréquence, appelée fréquence globale pondérée. Nous avons démontré que cette fréquence possède les propriétés de contrainte anti-monotone convertible, et surtout qu'il était possible de l'intégrer directement au processus de fouille, sans pour autant passer par des techniques de réduction de données.

Nos motivations concernant cette nouvelle mesure sont doubles. D'une part, nous souhaitons renfor-

cer la valeur sémantique des itemsets, notamment pour les itemsets de plus faible fréquence. D'autre part, l'égalité biaisait les fréquences de la plupart des itemsets contradictoires, leur attribuant une fréquence quasi-identique. La nouvelle sémantique proposée améliore considérablement ce problème, puisque nous notons moins d'itemsets contradictoires extraits, et que leur fréquence est sensiblement différente (au moins 10% d'écart).

Ce travail nous a permis de compléter les propositions présentées au chapitre 3, et également de mettre en œuvre un protocole d'expérimentation plus complet, au travers de la découpe de la base initiale, trop dense pour être fouillée directement. Les résultats sont actuellement en cours d'analyse.

Dans cette première partie, nous avons montré comment raffiner les résultats au travers de la réduction du nombre d'itemsets extraits. Pour cela, nous avons souhaité prendre en compte la "force de variation", en considérant que des "objets de valeurs égales" ne participaient pas à la variation. En revanche, nous n'avons pas pris en compte les objets dont la variation était plus forte que les autres. Nous proposons d'identifier ces objets en combinant l'heuristique du chapitre 3 à des techniques d'extraction de comportement inattendus.

## 4.3 Itemsets graduels et atypicité

### 4.3.1 Introduction

La détection de *comportement anormaux*, ou *outliers*, ou encore *anomalies* consiste à découvrir les objets ou motifs ne suivant pas un comportement attendu. Ce type d'élément est étudié par les statisticiens depuis de nombreuses années. Cependant, ce n'est que plus récemment que la communauté de fouille de donnée s'est intéressée à ce type d'objets. En effet, plutôt que découvrir des connaissances fréquentes, parfois trop nombreuses et déjà connues il s'avère parfois plus utile de rechercher des comportements inattendus. La recherche d'outliers est appliquée dans de nombreux domaines, tels que la détection de fraudes, d'intrusions ou encore de problèmes matériels.

Dans cette section, nous montrons comment extraire des objets ayant un comportement inattendu dans le contexte de la gradualité.

### 4.3.2 Travaux existants

Il existe de nombreux travaux permettant d'extraire des outliers à partir de méthodes de classification ou de clustering. Généralement, ces travaux utilisent la distance [KN98] ou encore différentes mesures statistiques [Haw80] afin de déterminer si un point est un outlier ou non. Dans ce chapitre, nous décrivons les travaux concernant la détection d'outliers à partir de motifs. Pour une lecture détaillée et très complète, le lecteur intéressé pourra se référer à [CBK07].

[Suz99, HLSL00] recherche des "*règles exceptionnelles*", qui contredisent des *règles d'association communes*. Par exemple, si la règle *"si un patient a le symptôme $A_1$ alors il a la maladie $B_1$"* est une règle commune, on considérera que la règle *"si un patient a les symptômes $A_1$ et $A_2$ alors il a la maladie $B_2$"* est exceptionnelle, car la prise en compte d'un nouveau facteur dans l'antécédent entraîne une conséquence différente. Pour extraire de telles contradictions, les auteurs s'appuient sur les mesures de fréquences et de confiance, en recherchant les règles $Y \cup Z \rightarrow X'$ ayant une confiance haute et une

faible fréquence telles que $Y \rightarrow X$ a un fort support et forte confiance et $Z \rightarrow X$ a une faible fréquence ou une faible confiance. Cependant, la méthode proposée repose sur la définition de cinq seuils laissés à la charge de l'utilisateur, ce qui la rend difficile à exploiter.

[BCMG04] propose d'extraire des anomalies comme "*si un patient a le symptôme $A_1$ alors il a la maladie $B_1$ cependant si un patient a le symptôme $A_1$ mais pas la maladie $B_1$ alors il a probablement la maladie $B_2$*". Les auteurs ne cherchent pas de règle contradictoire mais plutôt une alternative à la conséquence d'une règle fréquente. Afin d'extraire ce type de règles, les auteurs définissent une nouvelle mesure de confiance basée sur l'absence de certains itemsets.

[PGG+07] propose l'extraction de règles similaires, mais dans un contexte multidimensionnel : si la règle "*si un jeune consommateur a récemment reçu son permis de conduire, alors il va acheter une petite voiture*" est commune, "*Si un jeune consommateur dont le loisir préféré est le surf a récemment reçu son permis de conduire, alors il va acheter un van*" est exceptionnelle. De nouveau, les critères se basent sur la fréquence et le support, mais également sur la présence de caractère joker couramment utilisés dans le contexte multidimensionnel.

[SNV07] propose d'extraire les *itemsets rares*. Un itemset rare est un motif dont la fréquence est inférieure au seuil de fréquence minimal fixé. Afin de les extraires, les auteurs proposent l'algorithme Arima, composé de deux tâches : Apriori-Rare énumère tous les itemsets fréquents, et MRG-Exp, extrait uniquement les générateurs d'itemsets fréquents.

[PLT07b] proposent une aide à la navigation dans les cubes de données via la détection d'outliers. Les auteurs mettent en évidence à chaque niveau de hiérarchie $n$ les $k$ séquences les plus différentes des autres. Cette approche s'avère intéressante car elle est la seule à exploiter les différents niveaux de hiérarchie.

[LKLT08] propose d'extraire les transactions atypiques comparées à l'ensemble des transactions de la base. Pour cela, les probabilités d'apparition de courtes sous-séquences sont conservées de manière optimale dans un arbre probabiliste des suffixes. Une hypothèse de Markov permet de calculer la probabilité de chaque transaction en lisant celles des sous-séquences qu'elle contient dans l'arbre. Les transactions ayant le plus de sous-séquences de probabilité faible sont alors considérées comme atypiques.

Comme le montrent ces différents travaux, la recherche d'anomalies revêt plusieurs formes, selon lesquelles les algorithmes proposés diffèrent totalement. Nous avons regroupé ces différentes formes en trois catégories :

- Les **outliers globaux** sont des transactions de la base de données qui diffèrent des autres transactions. Par exemple, en considérant une base de données associant des descriptions à des protéines, on recherchera les protéines dont le comportement diffère des autres en se basant sur les descriptions. Cette approche considère que tous les items sont corrélés et cette hypothèse est utilisée durant le processus de découverte d'outliers.
- Les **outliers locaux** sont des objets ayant un comportement différent des autres en fonction d'un motif donné. Par exemple, si l'on extrait le motif "plus le gène $g_1$ s'exprime, alors plus le gène $g_2$ s'exprime" avec une fréquence de 99%, on déduira que 1% des objets de la base ne suivent pas

cette tendance.

- Les **règles inattendues** sont des règles qui contredisent une croyance (ou un motif extrait au préalable). Par exemple, nous pensons que "la ceinture de sécurité sauve des vies", ce qui est contredit par la règle "enfant et ceinture de sécurité implique mort".

Ainsi, nous différencions les inattendus concernant un objet ou une transaction de la base des inattendus concernant une règle. Dans le premier cas, il s'agit de présenter à l'utilisateur des objets de la base n'ayant pas le comportement attendu alors que dans le second, c'est une règle surprenante qui sera extraite.

Dans le contexte graduel, il est possible d'extraire toutes ces formes d'outlier. Dans ce chapitre, nous nous focalisons sur l'extraction d'outliers locaux. Dans notre contexte, nous ne recherchons pas un objet qui contredit un itemset, mais plutôt dont le comportement des autres diffère de par sa variation. Cet aspect vient alors compléter les mesures de fréquences présentées dans les sections précédentes, en investissant les objets dont la variation est plus élevée que les autres. De tels objets "accélèrent" la force de variation puisque c'est à partir de ceux-ci que les valeurs sont notablement plus élevées. Ils sont particulièrement intéressant pour les différents renseignements qu'ils fournissent :

- Si cet objet est en bout de liste ordonnée, cela peut-être soit un évènement correct mais isolé, soit une erreur de saisie dans la base de données
- On peut utiliser cet objet et l'itemset auquel il est associé pour déterminer quel item provoque cette accélération. Ainsi, l'expert pourra prendre en compte la plus grande influence d'un item par rapport aux autres.
- On peut également imaginer des applications telles que le clustering ou la classification : par exemple, si l'on extrait un gène dont la valeur augmente considérablement dans la liste, il peut être utile de se référer à des informations tel que le grade de la tumeur concernée afin de caractériser les valeurs d'expressions de cette tumeur.

Dans la suite de cette section, nous présentons une méthode permettant de présenter les objets en contradiction avec un itemset graduel donné. Nous nous basons sur l'heuristique présentée dans le chapitre précédent, principalement pour une raison pratique : cette méthode conserve les valeurs attribuées à un objet pour chaque niveau de l'arbre. Il sera alors plus facile d'ajouter une couche d'extraction des outliers, sans pour autant augmenter la complexité mémoire de notre méthode.

### 4.3.3 Extraction d'objets atypiques dans un cas graduel

Afin d'illustrer notre propos, nous utiliserons l'exemple du tableau 4.7, qui décrit l'âge et le bénéfice annuel de cinq entreprises. Sur cette base, l'itemset graduel $s =$"*plus l'entreprise est âgée, plus elle fait du bénéfice*" a une fréquence de 100%. On note que d'un objet à l'autre, la variation sur l'âge est constante : elle est de 10 mois. Cependant, la variation des bénéfices d'une entreprise à l'autre suit une loi de variation différente : elle est constante de l'entreprise $e_1$ à $e_3$ (10000€), mais quatre fois plus élevée pour $e_4$. Ainsi, nous pouvons affirmer que l'entreprise $e_4$ a une variation inattendue pour l'itemset graduel $s$, puisqu'elle "*casse*" la variation moyenne des bénéfices.

Plus formellement, nous définissons le problème d'extraction d'outliers locaux de la manière suivante : soit une base de données $DB$ et un seuil de fréquence minimal $minFreq$. Notre but est d'extraire d'une part tous les itemsets graduels qui supportent une variation constante, et d'autre part ceux qui

| Entreprise | Age (mois) | Bénéfices (€) |
|---|---|---|
| $e_1$ | 10 | 20,000 |
| $e_2$ | 20 | 30,000 |
| $e_3$ | 30 | 40,000 |
| $e_4$ | 40 | 80,000 |
| $e_5$ | 50 | 90,000 |

Tab 4.7 – Age et bénéfice de cinq entreprises

contiennent une variation "brutale". Plus particulièrement, nous souhaitons mettre en évidence les objets dont la variation n'est pas constante.

Nous définissons un outlier de la manière suivante :

**Définition 20.** *(outlier) Soit $s = (i_1^{*1}...i_n^{*n})$ un itemset graduel et $G^{\mathcal{D}}$ son ensemble représentatif associé. Soit $m$ la moyenne des valeurs de $dom(i_n)$, et $\varepsilon$ un seuil de variation minimal défini par l'utilisateur. Un objet $o$ est un outlier si $|t_o[i_n] - m| > \varepsilon$.*

En d'autres mots, un objet est un outlier par rapport à un itemset graduel $s$ si sa variation sur le dernier item de $s$ est plus grande qu'un seuil $\varepsilon$ défini par l'utilisateur. Dans notre contexte, il est possible d'utiliser plusieurs mesures de variations, quelques unes étant plus adaptées à certaines bases. Par exemple, une solution naïve consiste à considérer qu'un objet est un outlier s'il varie d'au moins deux fois la moyenne des variations. Par exemple, la moyenne des variations bénéfices du tableau 4.7 est de $17,500$ (en appliquant la formule 4.14).

$$\overline{T} = \frac{1}{n-1} \sum_{x=1}^{n} |i_x - i_{x-1}| \tag{4.14}$$

Ainsi, $e_4$ est considéré comme un outlier car il varie de $40,000$ ce qui est plus que $2 \times 17,500$.

Dans ce chapitre, nous proposons l'utilisation de l'inégalité de Chebyshev : sous des conditions valables pour la majorité des lois de probabilité, dans des jeux de données, presque toutes les valeurs sont proches de la moyenne. L'inégalité est donnée par l'équation 4.15, où $X$ est une variable aléatoire, $\mu$ une valeur attendue et $\sigma$ une variance finie.

$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \tag{4.15}$$

Nous notons $\overline{T}_{i_n}$ la moyenne associée à un itemset graduel $s = (i_1^{*1}...i_n^{*n})$ :

$$\overline{T}_{i_n} = \frac{1}{|G^{\mathcal{D}}|} \sum_{x=1}^{|G^{\mathcal{D}}|} t_x[i_n] \tag{4.16}$$

La variance est donnée par l'équation 4.17 :

$$\sigma = \frac{1}{|G^{\mathcal{D}}|} \sum_{x=1}^{|G^{\mathcal{D}}|} |t_x[i_n] - \overline{T}_{i_n}|^2 \tag{4.17}$$

Ainsi, les outliers sont les objets $o$ tels que

$$Pr(|t_o[i_n] - \overline{T}_{i_n}| \geq k\sigma) \leq \frac{1}{k^2} = \varepsilon \tag{4.18}$$

$$k = \frac{1}{\sqrt{\varepsilon}} \tag{4.19}$$

A partir du tableau 4.7, nous obtenons :

$$\overline{T}_{\text{Benefice}} = 52000 \qquad \sigma = 27857$$
$$\varepsilon = 0.1 \quad Cheb = ]7955, 96045[$$

Cela signifie que chaque objet dont la valeur appartient à l'intervalle $]7955, 96045[$ sera considéré comme un outlier. Appliquée tel quel sur un petit jeu de données tel que le notre n'est pas intéressant néanmoins, cette méthode est connue pour être pertinente sur de grandes bases de données. L'algorithme 5 décrit notre méthode.

**Algorithme 5** : ExtractOutliers

**Données** : Une base de données $DB$,
Un seuil de fréquence minimal $minFreq$,
Une déviation standard minimale $\varepsilon$

**Résultat** : Les itemsets graduels fréquents respectant $minFreq$ et contenant des outliers

```
L_1 ← ScanDB()
k ← 1
tant que |L_k| > 0 faire
    L_{k+1} ← Generate(L_k)
    pour chaque l ∈ L_2 faire
        si SupportCount(l) ≥ minFreq alors
            L_3 ← l
            out ← ComputeOutliers(ε)
            si |out| > 0 alors
                OutPut(l, out)
            fin
        fin
    fin
    k ← k + 1
fin
```

Les outliers sont calculés à la volée à chaque niveau du déroulement de l'algorithme. Nous ne conservons que les itemsets contenant des outliers, ainsi que les objets considérés comme outliers. L'inégalité de Chebyshev est implémentée par la fonction *ComputeOutliers*.

Avec cette méthode, les outliers extraits ne concernent que le dernier item de l'itemset. Comme nous utilisons un algorithme par niveau, les outliers de chaque niveau seront extraits indépendamment de ceux extraits au niveaux précédent. *Ces outliers peuvent-ils être pris en compte lors du calcul du*

*niveau suivant ?* Si un outlier appartient à la fois à l'ancien et au nouvel ensemble représentatif, alors nous pouvons penser que les deux items graduels concernés sont corrélés. Notons que l'inégalité de Chebyshev a été étendue au contexte multidimensionnel. Cependant, la complexité mémoire du calcul la rend difficilement applicable sur de grandes bases de données, c'est pourquoi nous ne l'avons pas utilisée dans ce contexte.

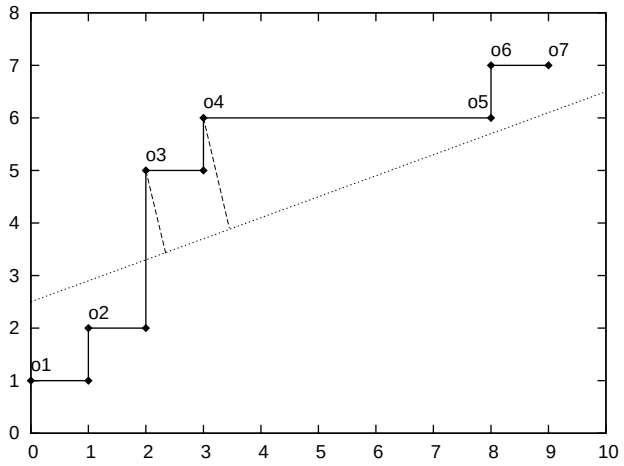



Fig 4.10 – Exemple d'un 2-itemset projeté

L'inégalité de Chebyshev permet la détection de valeurs extrêmes, c'est-à-dire des objets ayant une valeur éloignée de la moyenne. Cependant, dans notre contexte, la valeur de l'objet précédent à l'outlier doit être prise en compte. La figure 4.10 illustre ce processus. Ici, nous considérons un itemset graduel de taille 2, $s = i_1^{\geq} i_2^{\geq}$ et l'ensemble des objets $\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$ respectant $s$. Nous avons projeté les objets de $\mathcal{O}$ sur deux axes : l'axe $x$ pour les valeurs de $i_1$, et l'axe $y$ pour les valeurs de $i_2$. Supposons que l'inégalité de Chebyshev mette en évidence les objets $o_3$ et $o_4$, car leur distance à la moyenne est plus élevée que celle des autres objets. Cependant, nous recherchons les variations "brutales" et si la variation entre $o_2$ et $o_3$ n'est pas commune, ce n'est pas le cas de la variation entre $o_3$ et $o_4$. Nous ajoutons ainsi un critère supplémentaire à l'équation 4.15 afin de prendre en compte les outliers *directs*.

### 4.3.4 Expérimentations

Dans cette section, nous montrons les résultats de nos expérimentation sur un jeu de données réel concernant des joueurs de basketball [1].

La base de données contient $17$ items, listés dans le tableau 4.8.Nous avons lancé notre algorithme sur les $2000$ premiers joueurs de cette base.

Après une étude empirique, nous avons fixé le seuil de fréquence minimal à 0.2, soit 20%, et le pourcentage d'outliers à $0.05$, soit $5\%$. Notons qu'un pourcentage d'outlier plus grand que 10% n'apportera aucune information intéressante, car l'intervalle de Chebyshev sera trop près de la moyenne. Sur $2903$ itemsets graduels générés, notre algorithme extrait $307$ itemset graduels contenant des outliers.

1. `http://www.databasebasketball.com/`

| Id | Signification |
|---|---|
| leag | Ligue |
| gp | jeux joués |
| minutes | Minutes jouées |
| pts | Total de points |
| oreb | Rebonds offensifs |
| dreb | Rebonds defensifs |
| reb | Rebonds |
| asts | Aide totale |
| stl | Reprises |
| blk | Bloquages |
| turnover | Nombre total de retours |
| pf | Nombre total des foulées personnelles |
| fga | Filets de buts tentés |
| fgm | Filets de buts marqués |
| fta | Lancés libres tentés |
| ftm | Lancés libres réussis |
| tpa | Nombre de filets à 3 points tentés |
| tpm | Nombre de filets à 3 points marqués |

Tab 4.8 – Items de la base de données des joueurs de basketball

Le tableau 4.9 donne un aperçu du pourcentage de compression des résultats offert par notre méthode. De manière surprenante, le nombre d'itemsets extraits avec outliers n'évolue pas dans les mêmes proportions que le nombre d'itemsets extraits sans outliers. Cela confirme qu'il serait intéressant de définir une méthode plus élaborée, permettant de reporter en cascade les outliers d'un niveau $k$ au niveau $k+1$.

| Niveau | Sans Outliers | Avec Outliers | Pourcentage |
|---|---|---|---|
| 2 | 124 | 70 | 56 |
| 3 | 369 | 102 | 27 |
| 4 | 559 | 86 | 15 |
| 5 | 751 | 35 | 4 |
| 6 | 666 | 13 | 2 |
| 7 | 356 | 1 | 0.2 |
| 8 | 88 | 0 | 0 |
| Total | 2903 | 307 | |

Tab 4.9 – Comparaison du nombre d'itemsets extraits avec et sans outliers

Si l'analyse des résultats montre que la plupart des itemsets graduels extraits ont du sens, comme par exemple "*plus il y a de filets tentés, plus il y a de paniers*", elle permet aussi de voir les joueurs pour lesquels les valeurs donnent les écarts de variation les plus importants. Le tableau 4.10 montre quelques

itemsets graduels extraits avec leurs outliers correspondants.

| | **Itemset graduel** | **Freq** | **Outliers** |
|---|---|---|---|
| 1 | (turnover+ tpa+ tpm+) | 44 | DIENETR01<br>DIERKCO01<br>DIETRCO01<br>DIGREER01<br>DILLADU01<br>DILLAMI01<br>DILLCR01 |
| 2 | (dreb+ stl+ turnover+ tpa+ tpm+) | 26 | CALDWJO01<br>CALHOBI01 |
| 3 | (gp+ oreb- dreb- turnover-) | 26 | ABDELAL01<br>ABDULKA01<br>ABDULMA01<br>ABDULTA01<br>ABDURSH01<br>ABERNTO01 |
| 4 | (gp+ oreb- dreb- stl+ turnover-) | 26 | ABDELAL01<br>ABDULKA01<br>ABDULMA01<br>ABDULTA01<br>ABDURSH01<br>ABERNTO01<br>ABLEFO01 |
| 5 | (gp+ oreb- dreb- stl+ blk+ turnover-) | 26 | ABDELAL01<br>ABDULKA01<br>ABDULMA01<br>ABDULTA01<br>ABDURSH01<br>ABERNTO01 |
| 6 | (oreb+ dreb+ stl+ blk+ turnover+ tpa+ tpm+) | 25 | BUTLEGR01 |

Tab 4.10 – Quelques itemsets graduels extraits et leurs outliers

Les itemsets graduels 1 et 2 montrent que les outliers ne sont pas reportés sur les itemsets de plus grande longueur. : l'itemset 1 est inclus dans l'itemset 2, mais les outliers correspondant sont différents. Cela montre que les outliers de l'itemset 1 sont écartés par l'item dreb ou stl. En revanche, les itemsets graduels 3, 4 et 5 montrent un comportement opposé : la plupart des outliers sont reportés d'un motif à l'autre. Cela démontre que les joueurs en question montrent de plus importantes variations, et donc un comportement similaire sur les rebonds défensifs, les reprises et les blocages. L'itemset graduel 6 est

le plus long extrait sur cet échantillon de la base.

Les mêmes joueurs peuvent être considérés comme outliers pour plusieurs itemsets graduels différents. Nous nous sommes intéressés à la distribution des joueurs outliers par rapport aux itemsets. La figure 4.11 montre la distribution des outliers : l'axe $x$ représente le nombre de fois qu'un joueur est considéré comme outlier, et l'axe $y$ le nombre de joueurs concernés. Plus de la moitié des joueurs outliers (169 joueurs sur 307 joueurs) ne sont considérés qu'une seule fois comme outlier (c'est-à-dire qu'ils ne sont associés qu'a un seul itemset graduel), et un joueur est associé à 64 itemsets graduels différents. Cela signifie que ce joueur a une forte variation sur plusieurs items. L'analyse de ces items ainsi que de la position du joueur dans les chaînes les plus longues permettrait de connaître les raisons de son nombre d'apparitions.

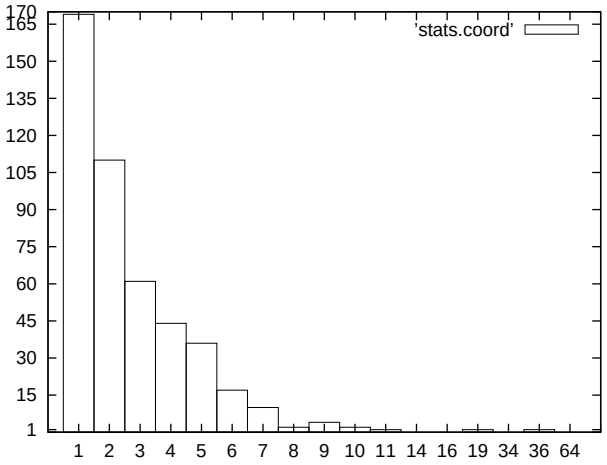



Fig 4.11 – Distribution des outliers

### 4.3.5 Conclusion

Dans ce chapitre, nous avons proposé deux solutions afin d'étudier les variations des objets au sein des itemsets graduels. La première proposition consiste à écarter les valeurs égales du comptage de la fréquence tout en considérant le nombre d'objets de la base participant activement à la gradualité. La seconde solution se concentre sur les objets ayant des variations plus fortes que les autres.

# Chapitre 5

# Gradualité et bases multidimensionnelles

## 5.1 Introduction

Le concept des entrepôts de données est apparu dans les années 1990. Ils permettent de stocker des masses de données historisées provenant de diverses sources dans une structure adaptée. Ces données sont ensuite utilisables sur demande des utilisateurs et chargées selon différents formats. Par exemple, le cube de données vise à présenter les données dans un format adapté aux décideurs, qui pourront naviguer à différents niveaux de granularité. La fouille de cube consiste à extraire de nouveaux types de connaissances telles que les règles d'associations multidimensionnelles, les motifs séquentiels multidimensionnels, ou encore plus récemment les blocs de données multidimensionnels. Cependant, il n'existe actuellement aucune méthode permettant d'extraire des éléments graduels à partir de telles structures. Pourtant, la plupart des données médicales sont multidimensionnelles par nature, par exemple parce qu'elles décrivent un résultat biologique ou médical en fonction de différents axes d'analyse comme l'âge, le sexe ou la ville d'habitation. Différents niveaux de hiérarchie sont souvent décrits (comme par exemple ville, département, région, ou encore les familles de maladies). De plus, les cubes de données présentent l'avantage de conserver des données numériques, sous un format agrégé.

Dans ce chapitre, nous nous intéressons à la découverte de règles d'association graduelles multidimensionnelles. Notre méthode s'appuie sur l'extraction de blocs de données. Nous résumons dans la section 5.2 les différentes notions associées aux cubes de données, ainsi que les différents travaux de fouille de données basés sur les cubes. Dans la section 5.3, nous présentons un algorithme efficace d'extraction de blocs de données. Ces blocs sont la base de notre algorithme d'extraction de règles d'association graduelles multidimensionnelles, présenté dans la section 5.4. Enfin, ce chapitre se termine par une discussion.

## 5.2 État de l'art

### 5.2.1 Cubes de données et fouille de données

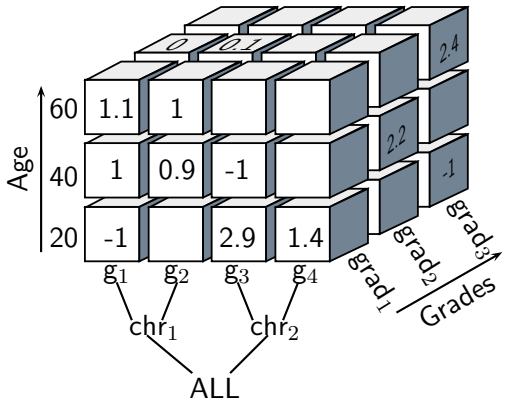



Fig 5.1 – Exemple de cube de données

Un cube de données est structuré par une ou plusieurs dimensions qui peuvent être numériques. Sur

ces dimensions, il est également possible de définir une hiérarchie, ce qui permet d'obtenir une vision plus ou moins précise des données. Un cube est composé de cellules, où chaque cellule correspond à la valeur prise à l'intersection des membres des dimensions. Les valeurs contenues dans ces cellules sont alors une *mesure* agrégée en regroupant des valeurs de données sources sur chaque dimension (on peut considérer la moyenne, le min, le max, etc...). La figure 5.1 illustre un tel cube. Il est structuré par trois dimensions : les gènes (flèche Gene), l'âge (flèche Age) et le grade de la tumeur. La dimension Gene contient les membres $grad_1$, $grad_2$ et $grad_3$ ; la dimension Age contient les membres $20$, $40$ et $60$. Une hiérarchie est établie sur les gènes, qui peuvent être regroupés par chromosomes. Enfin, le contenu des cellules représente la moyenne des expressions de gènes pour tous les patients de la base associés à ces valeurs de dimension. Ainsi, la cellule en bas à droite signifie que la moyenne d'expression du gène $g_4$ pour tous les patients ayant 20 ans et une tumeur de grade 1 est de 1.4.

La fouille de cube de données consiste à définir des méthodes capables d'extraire des connaissances à partir de données multidimensionnelles, agrégées, et potentiellement organisées à différents niveaux de hiérarchies. Différente de la fouille de données classique en raison des spécificités propres à cette organisation des données, elle nécessite la définition de nouvelles méthodes permettant à la fois d'extraire des connaissances intéressantes et pertinentes, mais aussi de faire face à de gros volumes de données en raison de la taille sans cesse croissante des cubes de données disponibles et des besoins grandissant d'applications en temps quasi réel des utilisateurs.

Dans ce contexte, de nombreux travaux ont été proposés ces dernières années, notamment pour extraire des règles d'association [KHC97, IKA02], des résumés flous [Lau02], ou encore des motifs séquentiels multidimensionnels [PHP+01, PCL+05]. Il est alors possible d'extraire des règles du type *"la plupart des gènes $g_1$ des tumeurs de grade 1 s'expriment faiblement"*, ou *"la plupart des gènes $g_1$ faiblement exprimés se retrouvent dans les tumeurs de grade 1* ou encore *"pour la plupart des tumeurs de grade 3, on trouve de fortes expressions dans le chromosome 1 pour les patients de 20 ans, puis des expressions de gènes $g_3$ faibles dans les tumeurs de grade 3"*.

Ces différentes règles, profitant de l'organisation des cubes de données, de la présence d'une ou plusieurs mesures et de leur multidimensionnalité et organisation multi-niveaux, permettent de renseigner le décideur sur les tendances présentes dans les cubes de données. Ces tendances sont difficiles à retrouver par une simple navigation non guidée par les opérateurs classiques OLAP. Notons que des travaux ont également été proposés pour retrouver des exceptions au sein de telles données [PGG+07, PLT07a].

Cependant, il n'existe pas à notre connaissance de méthode permettant d'extraire des règles graduelles de la forme *"Plus l'âge et le grade de la tumeur sont élevés, plus l'expression moyenne augmente"*. Or si l'ensemble des dimensions des cubes de données manipulés n'est pas toujours ordonné, il n'en reste pas moins que beaucoup peuvent l'être, notamment en observant les hiérarchies définies, comme par exemple les gènes selon leur chromosomes. Ce type de règles permet alors de retrouver des corrélations au sein de ces dimensions ordonnées.

Toutefois, les cubes de données étant volumineux et contenant des valeurs souvent très agrégées, il n'est pas possible de considérer chaque cellule individuellement pour vérifier si la corrélation est respectée. Afin de prendre en compte de manière plus souple les valeurs présentes dans les cellules du cube,

nous proposons donc de nous appuyer sur une représentation des cubes de données en blocs, comme proposé dans [CLL08, CMLL04, CLL07]. Ces blocs de données représentent des zones du cube quasi homogènes (au sens d'une confiance), décrivant par exemple que *quand la ville est N.Y. ou S.F. et que le produit est P1 ou P3 alors le niveau de ventes est 4*. Ces cubes de données, extraits par des algorithmes par niveau, présupposent qu'une représentation du cube a été définie. Une représentation correspond intuitivement à une façon d'agencer les valeurs des dimensions (en plaçant par exemple le produit $P3$ puis $P1$ puis $P4$ puis $P2$). Dans le cadre de dimensions ordonnées, cette représentation sera obtenue en ordonnant les dimensions.

### 5.2.2 Blocs de données

L'extraction de règles graduelles à partir de cubes se trouve au carrefour de plusieurs problématiques : l'extraction de connaissances au sein de cubes multidimensionnels (règles d'association et motifs séquentiels), la présentation des connaissances contenues dans le cube à l'utilisateur [CLM03], ou encore la prise en compte de la mesure [PLT07a] et des hiérarchies [CLL07].

Dans [CLM03], les auteurs mettent en évidence qu'il peut exister plusieurs représentations équivalentes pour des cubes définis sur plusieurs dimensions. Il suffit d'inverser l'ordre de présentation des membres d'une dimension pour obtenir une nouvelle représentation d'un même cube. Les auteurs montrent également qu'il existe des représentations plus pertinentes que d'autres selon des critères définis par l'utilisateur. Par exemple, une représentation pertinente peut être basée sur la mesure : on peut imaginer un ordre de présentation de la mesure dont on retrouverait les plus faibles valeurs à un *coin* du cube et les plus fortes au *coin opposé*. Les tableaux 5.1a et 5.1b montrent de telles représentations : les valeurs des cellules ne changent pas, mais des inversions sur les deux dimensions du cube réordonnent les cellules en plaçant les valeurs les plus faibles en bas à gauche et les plus fortes en haut à droite.

Les auteurs discernent les *représentations parfaites*, pour lesquelles aucune cellule ne contredit la contrainte fournie par l'utilisateur (l'ordre par exemple), des *représentations optimales*, qui contiennent des cellules contradictoires, mais qu'il est impossible de contourner. Le but est alors de calculer les meilleures représentations, en utilisant les opérateurs d'inversion OLAP. Cependant, le problème est NP-complet, ce qui rend la recherche de telles représentations difficile. De plus, même si la lecture du cube est facilitée pour l'utilisateur, il se peut que le fait de "casser" l'ordre entre certains membres d'une dimension rende l'interprétation difficile.

Dans [CLL07], les auteurs considèrent qu'il existe plusieurs représentations, mais ils en supposent une choisie (par l'utilisateur par exemple) comme support à la fouille. Il s'agit alors d'extraire des *blocs* permettant de dériver des règles de la forme "Si les produits sont $P_1, P2$ et $P_3$ et les mois d'achat sont juillet et janvier, alors le nombre moyen d'achats est 800". Une version améliorée de l'algorithme est présentée dans [CLL08] et permet de prendre en compte la hiérarchie des dimensions. Nous présentons ici brièvement la méthode de [CLL07], ainsi que les formalisation associées. Dans la suite de ce chapitre, nous conserverons ces notations.

Les cubes de données sont définis de manières diverses. Dans ce chapitre, nous considérons un ensemble de dimensions $D = \{d_1, d_2, ..., d_n\}$ où chaque dimension $d_i$ est définie sur un domaine fini de

| Patients avec gènes sur-exprimés | | | | | |
|---|---|---|---|---|---|
| gène 4 | 3 | 5 | 6 | 3 | 5 |
| gène 3 | 4 | 6 | 7 | 5 | 7 |
| gène 2 | 2 | 4 | 6 | 2 | 5 |
| gène 1 | 4 | 5 | 7 | 4 | 6 |
| | 20 ans | 30 ans | 40 ans | 50 ans | 60 ans |

(a)

| Patients avec gènes sur-exprimés | | | | | |
|---|---|---|---|---|---|
| gène 3 | 4 | 5 | 6 | 7 | 7 |
| gène 1 | 4 | 4 | 5 | 6 | 7 |
| gène 4 | 3 | 3 | 5 | 5 | 6 |
| gène 2 | 2 | 2 | 4 | 5 | 6 |
| | 20 ans | 50 ans | 30 ans | 60 ans | 40 ans |

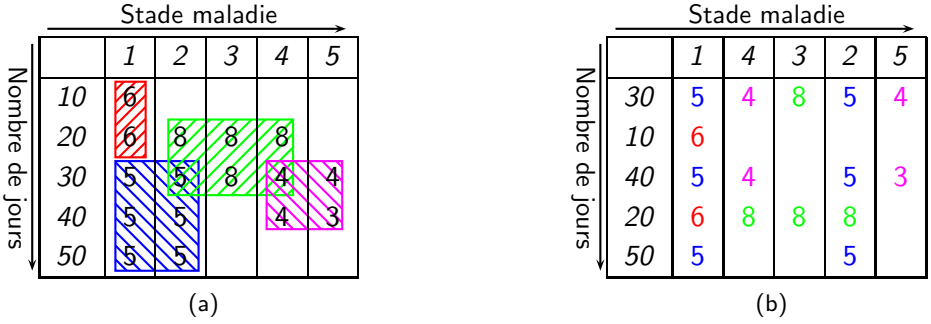
(b)

Tab 5.1 – Deux représentations différentes pour un cube de données

valeurs noté $dom_i$. Un cube de données est alors défini à partir de ces dimensions.

**Définition 21.** *(Cube) Un cube k-dimensionnel, ou simplement un cube $C$ est un n-uplet $\langle dom_1, ..., dom_k, dom_{mes}, m_C \rangle$ où :*

- $dom_1, ..., dom_k$ *sont des ensembles finis de symboles pour les membres associés avec les dimensions* $d_1, ..., d_k$ *respectivement*
- $dom_{mes}$ *un ensemble fini et totalement ordonné de valeurs de la mesure. Soit* $\perp \notin dom_{mes}$ *une constante (pour représenter les valeurs nulles). Alors* $dom_m = dom_{mes} \cup \perp$
- $m_C$ *est une application* $m_c : dom_1 \times ... \times dom_k \rightarrow dom_m$ *où* $m$ *est le domaine de la mesure.*




(a) (b)

Fig 5.2 – (a) Exemple de blocs de données (b) Représentation différente du même cube

Pour chaque $i = 1, ..., k$, un élément $v_i$ dans $dom_i$ est appelé une *valeur membre*. Une cellule $c$ d'un cube k-dimensionnel $C$ est un $(k+1)$-uplet $\langle v_1, ..., v_k, m \rangle$ tel que pour tout $i = 1, ..., k$, $v_i$ appartient à $dom_i$ et $m = m_C(v_1, ..., v_k)$. $m$ est appelé le *contenu* de $c$.

Par exemple, le tableau 5.2a illustre un cube à deux dimensions. Nous avons $C = \langle \{1, 2, 3, 4, 5\},$

$\{10, 20, 30, 40, 50\}, \{4, 5, 6, 8\}, m_C\rangle$. Nous avons deux dimensions $d_1$ = Stade de la maladie et $d_2$ = Nombre de jours, avec $dom_1 = \{1, 2, 3, 4, 5\}$ et $dom_2 = \{10, 20, 30, 40, 50\}$. $10$ est un membre de $dom_2$.

De manière générale, les cubes de données sont présentés à l'utilisateur sur une ou deux dimensions. Les membres des dimensions sont affichés par ordre alphabétique ou ordre d'insertion dans la base et les dimensions dont les membres peuvent être ordonnés (dimension temporelle par exemple) sont affichées de manière ordonnée si insérées dans cet ordre. Cependant, cette représentation est arbitraire. En effet, il est possible d'afficher les membres dans un ordre différent. Les mêmes données seront alors présentées à l'utilisateur différemment. Par exemple, nous aurions pu afficher la figure 5.2a sous la représentation de la figure 5.2b, où l'ordre des lignes a été modifié (ainsi que celui des colonnes). Il existe donc différentes représentations d'un même cube, qui sont formalisées de la manière suivante :

**Définition 22.** *(Représentation) Une représentation d'un cube k-dimensionnel $C$ est un ensemble $R = \{rep_1, ..., rep_k\}$ où pour chaque $i = 1, ..., k$, $rep_i$ est une application injective de $dom_i$ vers $\{1, ..., |dom_i|\}$*

Dans ce chapitre, nous considérons comme dans [CLL07] une représentation donnée. Cette représentation peut avoir été fixée par l'utilisateur ou provenir d'opérations antérieures. Nous ne traitons pas cet aspect dans ce chapitre. A partir de telles représentations, il est alors possible de retrouver les zones homogènes (au sens de la valeur de la mesure). Pour ce faire, [CLL07] proposent d'extraire des blocs de données, qui sont définis de la manière suivante :

**Définition 23.** *(Bloc) Un bloc $b$ est un ensemble de cellules définies sur un cube k-dimensionnel $C$ par $b = \delta_1 \times ... \times \delta_k$ où les $\delta_i$ sont des intervalles de valeurs consécutives de $dom_i$, pour $i = 1, ..., k$.*
*Remarque : il est possible que $\delta_i$ soit égal à tout $dom_i$ (valeur notée ici $ALL_i$).*

Tout comme les règles d'association, il est possible de définir des mesures de fréquence et de confiance pour les blocs :

**Définition 24.** *(Fréquence) Soit $count(b, m)$ le nombre de cellules ayant la valeur $m$ dans $b$, alors la fréquence d'un bloc $b$ de $C$ pour une valeur de mesure $m$ est*

$$Freq(b, m) = \frac{count(b, m)}{|C|}$$

**Définition 25.** *(Confiance) La confiance d'un bloc $b$ pour une mesure $m$ est*

$$Conf(b, m) = \frac{count(b, m)}{|b|}$$

Par exemple, il est possible d'extraire quatre blocs à partir du cube de la figure 5.2a :

- $b_1 = [1, 2] \times [30, 50]$, associé à la valeur 5, avec $Freq(b_1) = \frac{6}{25} = 0.24$ et $Conf(b_1) = \frac{6}{6} = 1$
- $b_2 = [2, 4] \times [20, 30]$, associé à la valeur 8, avec $Freq(b_2) = \frac{6}{25} = 0.24$ et $Conf(b_2) = \frac{4}{6} \simeq 0.66$
- $b_3 = [4, 5] \times [30, 40]$, associé à la valeur 4, avec $Freq(b_3) = \frac{4}{25} = 0.16$ et $Conf(b_3) = \frac{3}{4} = .075$
- $b_4 = [1, 1] \times [10, 20]$, associé à la valeur 6, avec $Freq(b_4) = \frac{2}{25} = 0.08$ et $Conf(b_4) = \frac{2}{2} = 1$

A partir de ces blocs, des règles "sémantiques" peuvent être proposées à l'utilisateur. Par exemple $b_1$ peut être formulé de la manière suivante : "*pour un nombre de jours compris entre 30 et 50 et pour chaque stade de maladie de 1 et 2, il y a 5 patients atteints*".

[CLL07] propose une méthode basée sur une génération par niveau afin d'extraire de tel blocs. Voici les principales étapes de l'algorithme, effectuées pour chaque mesure $m$ présente dans le cube :

1. Pour chaque dimension $d_i$, toutes les tranches contiguës sont générées. Informellement, une tranche est un hypercube ne contenant qu'un membre de $v_i \in d_i$ et $ALL_j$ sur toutes les autres dimensions. Par exemple, pour le cube de la figure 5.1, il y a trois tranches pour la dimension "âge" : $(ALL_{gene} \times 20 \times ALL_{grad})$, $(ALL_{gene} \times 40 \times ALL_{grad})$ et $(ALL_{gene} \times 60 \times ALL_{grad})$.
2. Pour chaque tranche, les intervalles des dimensions fixées à $ALL$ sont raffinées afin de minimiser la taille des blocs de données et d'en affiner la qualité. Puis, toutes les tranches telles que $Freq(\mathcal{T}(v_i), m) > \sigma$ sont conservées dans un ensemble $\mathcal{L}_1$.
3. Pour chaque dimension, les membres apparaissant dans chaque intervalle sont extraits et stockés sous la forme $(d_i, v)$.
4. Pour chaque dimension, les intervalles maximaux sont calculés en fonction des couples précédents.
5. L'algorithme entre ensuite dans une phase "à la apriori". Il s'agit d'augmenter la taille des blocs dimension par dimension à chaque passe, en combinant les blocs trouvés à la passe précédente. L'algorithme s'arrête lorsqu'il n'y a plus de blocs fréquents.

Afin d'illustrer cet algorithme, nous le déroulons pour la mesure $8$ et $\sigma = 5$ (les blocs fréquents doivent contenir au moins 5 cellules). Le tableau 5.2 résume ces différentes étapes. Tout d'abord, les tranches sont construites. Pour plus de lisibilité, nous avons regroupé les tranches par dimension. Notons que quelle que soit la mesure et le seuil de fréquence minimal considéré, ces tranches seront les mêmes. A l'issue de cette étape, 10 tranches sont construites. Elles ont toutes la même fréquence, soit 5 cellules. Ensuite, ces tranches sont raffinées pour calculer les intervalles de cellules contenant la valeur 8. Les tranches sur les membres $(d_1, 1)$, $(d_1, 5)$, $(d_2, 10)$, $(d_2, 40)$ et $(d_2, 50)$ sont élaguées, car elles ne contiennent pas de cellules ayant la valeur 8. En revanche, 5 tranches sont conservées, chacune ayant des intervalles différents. Au pas 3, les membres des dimensions fréquemment présentes sont isolés, puis combinés au pas 4. A ce stade, pour la valeur 8, il ne reste qu'un intervalle par dimension. Lors du pas 5, ces deux intervalles sont combinés et permettent d'extraire le bloc $b_2$. Cette méthode est répétée pour chaque mesure présente dans le cube.

### 5.2.3 Discussion

L'algorithme présenté dans [CLL07] n'est pas complet. D'une part, l'élagage basé sur la mesure de confiance en utilisant un algorithme par niveau peut écarter des candidats respectant les seuils de support et confiance minimale à l'étape de génération suivante. D'autre part, la maximisation des intervalles à l'étape 4 peut mener à ne pas considérer des blocs fréquents aux étapes ultérieures. Un exemple détaillé est donné dans [CLL08].

Cependant, l'utilisation des représentations et blocs de données s'avère intéressante, car ces techniques permettent un "remodelage" du cube afin de faire émerger les corrélations de valeurs. De plus,

| Mesure | Dimensions | |
|---|---|---|
| Pas 1 : construire les tranches pour chaque dimension | | |
| | $[1,1] \times ALL_{d_2}$ | $ALL_{d_1} \times [10,10]$ |
| | $[2,2] \times ALL_{d_2}$ | $ALL_{d_1} \times [20,20]$ |
| | $[3,3] \times ALL_{d_2}$ | $ALL_{d_1} \times [30,30]$ |
| | $[4,4] \times ALL_{d_2}$ | $ALL_{d_1} \times [40,40]$ |
| | $[5,5] \times ALL_{d_2}$ | $ALL_{d_1} \times [50,50]$ |
| Pas 2 : raffiner les tranches | | |
| 8 | $[2,2] \times [20,20]$ | $[2,4] \times [20,20]$ |
| 8 | $[3,3] \times [20,30]$ | $[3,3] \times [30,30]$ |
| 8 | $[4,4] \times [20,20]$ | |
| Pas 3 : sélectionner les membres fréquents pour chaque dimension | | |
| 8 | $(d_1, 2)$ | $(d_2, 20)$ |
| 8 | $(d_1, 3)$ | $(d_2, 30)$ |
| 8 | $(d_1, 4)$ | |
| Pas 4 : Calculer les intervalles maximaux par dimension | | |
| 8 | $[2,4]$ | $[20,30]$ |
| Pas 5 : Génération des blocs | | |
| 8 | $b_2 = [2,4] \times [20,30]$ | |
| 8 | $Freq(b_2) = 0.24$, $Conf(b_2) = 0.66$ | |

Tab 5.2 – Déroulement de l'algorithme pour $m = 8$ et $\sigma = 5$

les blocs proposent une perspective intéressante afin d'éliminer le bruit car une mesure de support et de confiance est associée à chaque bloc. Ainsi, les cellules vides font baisser ces mesures, ce qui permet de raffiner les parties du cube sélectionnées jusqu'à n'obtenir que les parties pertinentes. Dans ce chapitre, nous optons donc pour une définition des règles d'associations graduelles basées sur les blocs de données. Nous pensons que la gradualité est une valeur ajoutée forte, puisqu'elle met en évidence la corrélation de variations qu'il existe d'une part entre les membres d'une même dimension, et d'autre part entre les valeurs des blocs, augmentant ainsi la sémantique associée à un bloc. Enfin, il est possible de comparer les valeurs des différents blocs définis sur ces dimensions. Cela place la méthode graduelle comme un complément à la méthode de fouille de cube présentée ci-dessus.

Les données médicales sont multidimensionnelles par nature, et plus particulièrement dans le cas de données cliniques, où chaque attribut (taux de cholestérol, âge, médicamentation) peut être vu comme une dimension [PJ98, PL08]. Les premières expérimentations menées avec les implémentations proposées dans [CLL07] n'ont au préalable pas permis d'extraire de blocs de données, pour des raisons de performances. Dans ce chapitre, nous répondons aux questions suivantes : *est-il possible de proposer un algorithme d'extraction de bloc de données robuste sur des données médicales ? Comment utiliser les blocs de données afin d'extraire des règles graduelles multidimensionnelles ?* Et enfin, *est-il possible d'extraire de telles règles à la volée, c'est-à-dire au fur et à mesure de la découverte des blocs ?*

## 5.3 Extraction de blocs dans les bases de données multidimensionnelles

Dans cette section, nous présentons un nouvel algorithme d'extraction de blocs. Celui-ci présente l'avantage d'être complet, et nous permet d'extraire des blocs de données à partir de nos bases médicales. Nous proposons une extraction en deux étapes. Dans un premier temps, nous fabriquons des *classes* de blocs. Ces classes sont définies en fonction de la taille des intervalles et permettent de connaître facilement la fréquence associée à chaque bloc. De plus, certaines de ces classes sont incluses dans d'autres. Il est ainsi possible de générer par jointure tous les blocs multidimensionnels d'une classe à partir d'une autre classe. Cela nous permet de décider d'un *plan d'exécution*, c'est-à-dire de décider quelle classe sera utilisée afin de fabriquer tous les blocs de la classe suivante.

Afin d'illustrer notre propos, nous utilisons un exemple fourni par [CLL07], illustré par le tableau 5.3.

| | | | | | | |
|---|---|---|---|---|---|---|
| $P_1$ | 6 | 6 | 8 | 5 | 5 | 2 |
| $P_2$ | 6 | 8 | 5 | 5 | 6 | 75 |
| $P_3$ | 8 | 5 | 5 | 2 | 2 | 8 |
| $P_4$ | 8 | 8 | 8 | 2 | 2 | 2 |
| | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ |

Tab 5.3 – Base exemple à deux dimensions

Cette base est définie sur deux dimensions : $dom(d_1) = \{V_1, ..., V_6\}$ et $dom(d_2) = \{P_1, ..., P_4\}$.

### 5.3.1 Définition

A partir d'un cube de données, il est aisé de calculer pour chacune des dimensions tous les intervalles possibles. Notons $\mathcal{I}_d$ l'ensemble des intervalles possibles associé à la dimension $d$. Pour $d$ contenant $|dom(d)|$ membres, le nombre d'intervalles est de :

$$|\mathcal{I}_d| = \frac{|dom(d)|(|dom(d)| + 1)}{2}$$

Par exemple, nous pouvons construire $10$ intervalles avec les membres de la dimension $d_2$, référencés dans le tableau 5.4. De plus, il est possible de regrouper ces intervalles en fonction de leur taille. Par exemple, pour la dimension $d_2$, il y a $4$ intervalles de taille 1, 3 intervalles de taille 2, 2 intervalles de taille 3 et 1 intervalle de taille 4.

| **Taille** | **Intervalle** | **Taille** | **Intervalle** |
|---|---|---|---|
| 1 | $[P_1, P_1]$ | 2 | $[P_2, P_3]$ |
| 1 | $[P_2, P_2]$ | 2 | $[P_3, P_4]$ |
| 1 | $[P_3, P_3]$ | 3 | $[P_1, P_3]$ |
| 1 | $[P_4, P_4]$ | 3 | $[P_2, P_4]$ |
| 2 | $[P_1, P_2]$ | 4 | $[P_1, P_4]$ |

Tab 5.4 – L'ensemble des intervalles associé à $d_2$

A partir de ces informations, nous pouvons décrire de manière générale l'ensemble des blocs composant la base en combinant les différentes longueurs d'intervalles. Afin de mieux gérer l'ensemble de

ces blocs, nous définissons l'équivalence entre blocs, ce qui nous permet de ranger ces blocs par classe. Une classe est de la forme $< i_1 > \times < i_2 > \times ... \times < i_n >$, où chaque $i_j$ est une taille d'intervalle.

**Définition 26.** *(Équivalence entre bloc) Soient $b = \delta_1 \times ... \times \delta_k$ et $b' = \delta'_1 \times ... \times \delta'_k$ deux blocs. On dit que $b$ et $b'$ sont équivalents et on note $b \equiv b'$ si $\forall i \in \{1, ..., k\}, |\delta_i| = |\delta'_i|$. Comme toute relation d'équivalence, nous avons les propriétés suivantes :*

- *réflexivité ($b \equiv b$)*
- *symétrie ($b \equiv b \Leftrightarrow b' \equiv b$)*
- *transitivité ($b \equiv b' \wedge b' \equiv b'' \Rightarrow b \equiv b''$)*

La définition 26 permet de formaliser les classes d'équivalence :

**Définition 27.** *(Classe d'équivalence d'un bloc) Soit $\mathcal{B}$ l'ensemble des blocs d'un cube. La classe d'équivalence d'un bloc $b \in \mathcal{B}$, notée $Cl(b)$ est l'ensemble des images de $b$ par la relation $\equiv$ :*

$$Cl(b) = \{b' \in \mathcal{B} | b \equiv b'\}$$

.

Dans un cube de données contenant $k$ dimensions, le nombre de classes d'équivalence est de :

$$\prod_{i=0}^{k} |dom_i|$$

Par exemple, pour la base du tableau 5.3, nous pouvons construire $6 \times 4 = 24$ classes différentes, affichées dans le tableau 5.5. Notons que la dernière classe étant définie sur la taille maximale de tous les intervalles, elle représentera toujours la base complète.

| Classe | Exemple | Classe | Exemple |
|---|---|---|---|
| $Cl_1 =< 1 > \times < 1 >$ | $b_1 = [V_1, V_1] \times [P_1, P_1]$ | $Cl_{13} =< 4 > \times < 1 >$ | $b_{13} = [V_1, V_4] \times [P_1, P_1]$ |
| $Cl_2 =< 1 > \times < 2 >$ | $b_2 = [V_1, V_1] \times [P_1, P_2]$ | $Cl_{14} =< 4 > \times < 2 >$ | $b_{14} = [V_1, V_4] \times [P_1, P_2]$ |
| $Cl_3 =< 1 > \times < 3 >$ | $b_3 = [V_1, V_1] \times [P_1, P_3]$ | $Cl_{15} =< 4 > \times < 3 >$ | $b_{15} = [V_1, V_4] \times [P_1, P_3]$ |
| $Cl_4 =< 1 > \times < 4 >$ | $b_4 = [V_1, V_1] \times [P_1, P_4]$ | $Cl_{16} =< 4 > \times < 4 >$ | $b_{16} = [V_1, V_4] \times [P_1, P_4]$ |
| $Cl_5 =< 2 > \times < 1 >$ | $b_5 = [V_1, V_2] \times [P_1, P_1]$ | $Cl_{17} =< 5 > \times < 1 >$ | $b_{17} = [V_1, V_5] \times [P_1, P_1]$ |
| $Cl_6 =< 2 > \times < 2 >$ | $b_6 = [V_1, V_2] \times [P_1, P_2]$ | $Cl_{18} =< 5 > \times < 2 >$ | $b_{18} = [V_1, V_5] \times [P_1, P_2]$ |
| $Cl_7 =< 2 > \times < 3 >$ | $b_7 = [V_1, V_2] \times [P_1, P_3]$ | $Cl_{19} =< 5 > \times < 3 >$ | $b_{19} = [V_1, V_5] \times [P_1, P_3]$ |
| $Cl_8 =< 2 > \times < 4 >$ | $b_8 = [V_1, V_2] \times [P_1, P_4]$ | $Cl_{20} =< 5 > \times < 4 >$ | $b_{20} = [V_1, V_5] \times [P_1, P_4]$ |
| $Cl_9 =< 3 > \times < 1 >$ | $b_9 = [V_1, V_3] \times [P_1, P_1]$ | $Cl_{21} =< 6 > \times < 1 >$ | $b_{21} = [V_1, V_6] \times [P_1, P_1]$ |
| $Cl_{10} =< 3 > \times < 2 >$ | $b_{10} = [V_1, V_3] \times [P_1, P_2]$ | $Cl_{22} =< 6 > \times < 2 >$ | $b_{22} = [V_1, V_6] \times [P_1, P_2]$ |
| $Cl_{11} =< 3 > \times < 3 >$ | $b_{11} = [V_1, V_3] \times [P_1, P_3]$ | $Cl_{23} =< 6 > \times < 3 >$ | $b_{23} = [V_1, V_6] \times [P_1, P_3]$ |
| $Cl_{12} =< 3 > \times < 4 >$ | $b_{12} = [V_1, V_3] \times [P_1, P_4]$ | $Cl_{24} =< 6 > \times < 4 >$ | $b_{24} = [V_1, V_6] \times [P_1, P_4]$ |

Tab 5.5 – Classes d'équivalence de blocs construites à partir du cube exemple

A chacune de ces classes est associé un certain nombre de blocs, qu'il est possible de calculer à l'avance. Par exemple, la classe $Cl_{12} =< 3 > \times < 4 >$ contiendra les quatre blocs suivants :

- $b_{12} = [V_1, V_3] \times [P_1, P_4]$, support $12/24 = 0.5$
- $b_{122} = [V_2, V_4] \times [P_1, P_4]$, support $12/24 = 0.5$
- $b_{123} = [V_3, V_5] \times [P_1, P_4]$, support $12/24 = 0.5$
- $b_{124} = [V_4, V_6] \times [P_1, P_4]$, support $12/24 = 0.5$

La génération des blocs de données se fait en utilisant l'opérateur union. L'union de deux blocs est définie à partir de l'union d'intervalles suivante :

**Définition 28.** *(Union d'intervalles) Soient $\delta$ et $\delta'$ deux intervalles. L'union de $\delta$ et $\delta'$, notée $\delta \cup \delta'$ est définie par* $[\min(\delta, \delta'), \max(\delta, \delta')]$

**Définition 29.** *(Union de blocs) Soient $b = \delta_1 \times ... \times \delta_k$ et $b' = \delta'_1 \times ... \times \delta'_k$ deux blocs. L'union de $b$ et $b'$, notée $b \cup b'$ est définie par $\delta_1 \cup \delta'_1 \times ... \times \delta_k \cup \delta'_k$.*

Par exemple, si l'on unit les blocs $b_{12}, b_{122}, b_{123}$ et $b_{124}$, nous obtenons les blocs suivants :

- $b_{12} \cup b_{122} = [V_1, V_4] \times [P_1, P_4] = b_{16}$
- $b_{122} \cup b_{123} = [V_2, V_5] \times [P_1, P_4] = b_{161}$
- $b_{123} \cup b_{124} = [V_3, V_6] \times [P_1, P_4] = b_{162}$
- $b_{12} \cup b_{123} = [V_1, V_5] \times [P_1, P_4] = b_{20}$
- $b_{12} \cup b_{124} = [V_1, V_6] \times [P_1, P_4] = b_{24}$

La génération des blocs de données au travers de l'union permet également de définir l'inclusion entre blocs, puis de manière plus générale l'inclusion entre classes.

**Définition 30.** *(Inclusion d'intervalles) Soient $\delta$ et $\delta'$ deux intervalles. On dit que $\delta$ est inclus dans $\delta'$ et on note $\delta \subseteq \delta'$ si* $\min(\delta) \geq \min(\delta')$ *et* $\max(\delta) \leq \max(\delta')$.

**Définition 31.** *(Inclusion de blocs) Soient $b = \delta_1 \times ... \times \delta_k$ et $b' = \delta'_1 \times ... \times \delta'_k$ deux blocs. On dit que $b$ est inclus dans $b'$ et on note $b \subseteq b'$ si $\forall \delta_i \in b, \delta'_i \in b', \delta_i \subseteq \delta'_i$*

Par exemple, nous avons les inclusions de blocs suivantes :

- $b_{12} = [V_1, V_3] \times [P_1, P_4] \subseteq b_{16} = [V_1, V_4] \times [P_1, P_4]$
- $b_{123} = [V_3, V_5] \times [P_1, P_4] \subseteq b_{161} = [V_2, V_5] \times [P_1, P_4]$
- $b_{123} = [V_3, V_5] \times [P_1, P_4] \subseteq b_{162} = [V_3, V_5] \times [P_1, P_4]$

Nous définissons alors l'inclusion entre classes d'équivalences de la manière suivante :

**Définition 32.** *(Inclusion de classes d'équivalences) Soient $Cl$ et $Cl'$ deux classes d'équivalences. On dit que $Cl$ est inclus dans $Cl'$ et on note $Cl \prec Cl'$ si* $\{\forall b_i \in Cl, \exists b_j \in Cl' \mid b_i \subseteq b_j\}$

Afin d'illustrer notre propos, nous considérons l'exemple des classes d'équivalence $Cl_{10}$ (tableau 5.6) et $cl_{14}$ (tableau 5.7). Les inclusions de blocs suivantes montrent que $cl_{10} \prec Cl_{14}$ :

- $b_{100} \subseteq b_{140}$, $b_{101} \subseteq b_{141}$, $b_{102} \subseteq b_{142}$, $b_{103} \subseteq b_{142}$
- $b_{104} \subseteq b_{143}$, $b_{105} \subseteq b_{144}$, $b_{106} \subseteq b_{145}$, $b_{107} \subseteq b_{145}$
- $b_{108} \subseteq b_{146}$, $b_{109} \subseteq b_{147}$, $b_{110} \subseteq b_{148}$, $b_{111} \subseteq b_{148}$

| $Cl_{10} =< 3 > \times < 2 >$ | | |
|---|---|---|
| $b_{100} = [P_1, P_3] \times [V_1, V_2]$ | $b_{104} = [P_1, P_3] \times [V_2, V_3]$ | $b_{108} = [P_1, P_3] \times [V_3, V_4]$ |
| $b_{101} = [P_2, P_4] \times [V_1, V_2]$ | $b_{105} = [P_2, P_4] \times [V_2, V_3]$ | $b_{109} = [P_2, P_4] \times [V_3, V_4]$ |
| $b_{102} = [P_3, P_5] \times [V_1, V_2]$ | $b_{106} = [P_3, P_5] \times [V_2, V_3]$ | $b_{110} = [P_3, P_5] \times [V_3, V_4]$ |
| $b_{103} = [P_4, P_6] \times [V_1, V_2]$ | $b_{107} = [P_4, P_6] \times [V_2, V_3]$ | $b_{111} = [P_4, P_6] \times [V_3, V_4]$ |

Tab 5.6 – Tous les blocs appartenant à la classe d'équivalence $Cl_{10}$

| $Cl_{14} =< 4 > \times < 2 >$ | | |
|---|---|---|
| $b_{140} = [P_1, P_4] \times [V_1, V_2]$ | $b_{143} = [P_1, P_4] \times [V_2, V_3]$ | $b_{146} = [P_1, P_4] \times [V_3, V_4]$ |
| $b_{141} = [P_2, P_5] \times [V_1, V_2]$ | $b_{144} = [P_2, P_5] \times [V_2, V_3]$ | $b_{147} = [P_2, P_5] \times [V_3, V_4]$ |
| $b_{142} = [P_3, P_6] \times [V_1, V_2]$ | $b_{145} = [P_3, P_6] \times [V_2, V_3]$ | $b_{148} = [P_3, P_6] \times [V_3, V_4]$ |

Tab 5.7 – Tous les blocs appartenant à la classe d'équivalence $Cl_{14}$

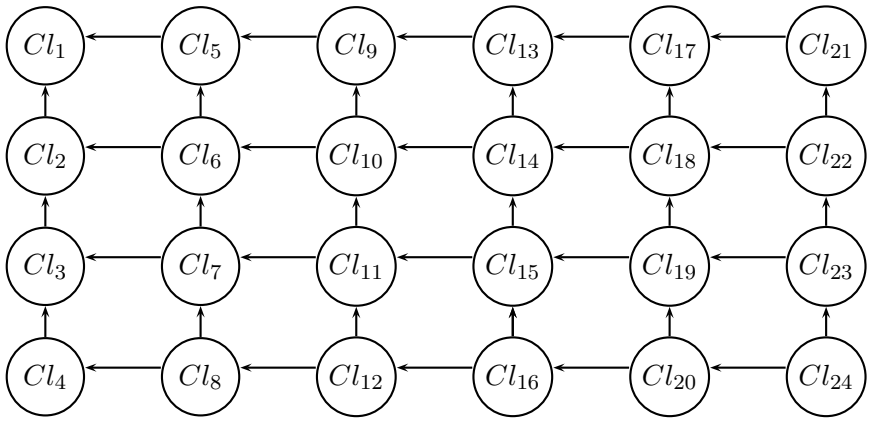



Fig 5.3 – Treillis d'inclusion des classes

Notons que l'ensemble de ces classes muni de l'inclusion définie ci-dessus forme un treillis. La figure 5.3 montre le treillis d'inclusion des classes pouvant être générées à partir de l'exemple du tableau 5.3. On lit que $Cl_{10}$ est inclus dans $Cl_{14}$ (les deux noeuds sont reliés). Cela signifie que tous les blocs associés à la classe $Cl_{1O}$ sont inclus dans au moins un bloc associé à la classe $Cl_{14}$

Dans notre contexte, un tel treillis possède des propriétés intéressantes. Par exemple, avant de matérialiser en mémoire l'ensemble des blocs appartenant à une classe d'équivalence, il est possible de connaître leur fréquence (en multipliant les entiers la définissant). Cela permet alors de ne générer et tester que les blocs qui pourront a priori être fréquents. De plus, la génération d'une classe $Cl'$ à partir d'une autre classe $Cl$ se fait par une simple jointure des blocs appartenant à $Cl$. Dans la sous-section suivante, nous décrivons les algorithmes correspondants.

### 5.3.2 Algorithmes

L'algorithme 6 décrit la méthode principale d'extraction des blocs. A partir du treillis de la figure 5.3, nous définissons l'ordre de génération des blocs de données, en suivant l'ordre d'inclusion des

classes (ligne 1). L'ensemble des classes dont la fréquence des blocs associée est de $\sigma$ permettront d'atteindre toutes les classes de fréquence supérieure. Ces classes "*racines*" sont récupérées à la ligne 2. Elles définissent les blocs de plus petite taille. Ensuite, les classes suivantes sont générées récursivement en suivant l'ordre donné par le treillis. Cette étape est réalisée dans les lignes 3-5.

**Algorithme 6** : ExtractBloc

**Données** : Un cube de données $C$,
le seuil de fréquence minimal $\sigma$
le seuil de confiance minimal $\gamma$

**Résultat** : Tous les blocs respectant les seuils de fréquence et de confiance

1 $\mathcal{C} \leftarrow$BuildLattice$(d_1, null, null, null)$
2 $R \leftarrow$ GetRoots()
3 **pour chaque** $Cl \in \mathcal{C}$ **faire**
4 | RecursiveBloc ($Cl$, $\gamma$) ;
5 **fin**

L'algorithme 7 réalise la construction du treillis des classes d'équivalence. Pour ce faire, l'algorithme s'exécute récursivement sur chaque intervalle de chaque dimension (lignes 9-14). Lorsqu'une classe d'équivalence est complètement construite et que sa fréquence respecte le seuil minimal, l'ensemble des classes directement incluses dans celle-ci sont calculées (lignes 3-6).

L'algorithme 8 permet de générer l'ensemble des blocs appartenant à une classe d'équivalence à partir d'une classe plus générale. Pour ce faire, certains blocs d'une classe sont unis deux à deux afin de générer l'ensemble des blocs de la classe marquée comme suivante par l'algorithme 7 (ligne 4-6). Cependant, l'algorithme n'effectue pas toutes les combinaisons de blocs d'une classe : en effet, seule une union sur les blocs ayant des intervalles consécutifs permettent d'atteindre la classe la plus proche dans le treillis (ligne 5). En reprenant l'exemple précédant, $b_{101} \cup b_{102} = b_{140} \subseteq Cl_{14}$, mais $b_{100} \cup b_{103} \nsubseteq Cl_{14}$.

Pour chaque nouveau bloc généré, la confiance peut être calculée en utilisant le principe d'inclusion-exclusion : soit $b_i$ et $b_j$ les deux blocs à joindre, $m$ la valeur du bloc traitée et $x^m_{b_i \cap b_j}$ le nombre de cellules communes aux deux blocs à joindre (obtenues par intersection) contenant $m$. Alors [1]

$$Confiance(b_i \cup b_j) = Confiance(b_i) + Confiance(b_j) - x^m_{b_i \cap b_j}$$

Une fois le treillis des inclusions construit, nous utilisons l'algorithme récursif 8 (en profondeur) générant toutes les classes supérieures et s'arrêtant avant l'obtention de la classe finale (tout le cube). Au fur et à mesure de la génération des classes de niveau $k$, les blocs de niveau $k-1$ sont effacés de la mémoire, car ils ne seront plus réutilisés.

1. La confiance est exprimée ici en nombre de cellules et non en pourcentage

**Algorithme 7** : BuildLattice

**Données** : la dimension courante $d$,
la dimension précédente $d_p$,
l'intervalle courant $i$,
la classe $Cl$ en cours de construction
le seuil de fréquence minimal $\sigma$

**Résultat** : Le treillis des classes d'équivalences

```
1  si d = D.last ∧ Freq(Cl) ≥ σ alors
2      C ← Cl
3      Cl.addSon(getClassEq(C, d, i − 1))
4      pour chaque d' ∈ {d − 1 ... d_1} faire
5          Cl.addSon(getClassEq(C, d', i))
6      fin
7      retourner C
8  fin
9  sinon
10     pour chaque i ∈ {0 ... I_d} faire
11         Cl = Cl× < i >
12         BuildLattice (d + 1, d, i, Cl)
13     fin
14 fin
```

**Algorithme 8** : RecursiveBloc

**Données** : Une classe d'équivalence $Cl$ et tous les blocs lui appartenant
le seuil de confiance minimal $\gamma$

**Résultat** : Tous les blocs fréquents et respectant les seuils de fréquence et confiance

```
1  Cl' ← ∅
2  pour chaque d ∈ 𝒟 faire
3      b' = null
4      pour chaque b ∈ Cl faire
5          si IsConsecutive(b, b', d) alors
6              Cl' ← b ∪ b'
7              pour chaque Mesure m faire
8                  conf = Conf(b) + Conf(b') − x^m_{b∩b'}
9                  si conf ≥ γ alors Output(b ∪ b')
10             fin
11         fin
12     fin
13     RecusiveBloc(Cl')
14     Cl' ← ∅
15 fin
```

### 5.3.3 Expérimentations

L'algorithme exposé ci-dessus a été testé en terme de performance en temps et en mémoire. Nous disposons d'un générateur de cube de données naïf, qui génère les mesures de manière aléatoire. Les cubes sont donc très denses : il n'y a pas de cellules vides. Ce type de cube ne reflète pas la réalité, mais permet néanmoins de tester le comportement de l'algorithme. Nous utilisons dans cette section trois jeux de données décrits par le tableau 5.8. Les expérimentations ont été menées sur un ordinateur Precision WorkStation R5400 Xeon(R) CPU E5450 3.00GHz doté de 16Go de RAM.

| **Nom** | $\|\mathcal{D}\|$ | **#membres / dim** | **#Cellules** |
|---|---|---|---|
| D5V10 | 5 | 10 | 100000 |
| D5V100 | 5 | 100 | 10000000 |
| D7V10 | 7 | 10 | 10000000 |
| D10V10 | 10 | 10 | 10000000000 |

Tab 5.8 – Bases de données test pour l'extraction de blocs

Pour toutes ces expérimentations, nous fixons un seuil de confiance minimal très bas (10%), ce qui augmente nos chances d'extraire des blocs. Le temps d'extraction dépend très fortement du nombre de dimensions ainsi que du nombre de membres par dimension. Les expérimentations montrent que le temps d'extraction peut être très long et ne permet pas d'extraction en temps réel. En revanche, on note que l'exploitation en terme de mémoire n'est pas très élevée et reste constante du moment que les valeurs du cube sont chargées en mémoire. Les figures 5.4a et 5.4b montre les temps d'extractions nécessaires pour une valeur de fréquence minimale variant de 90% à 40%. On note qu'avec 5 dimensions, notre algorithme extrait en moins de 600 secondes un peu plus de 100000 blocs de données. En revanche, il faut plus de 1400 minutes pour extraire environ 40000 blocs avec 7 dimensions. Pour toutes ces expérimentations, la charge en mémoire n'a pas excédé 700Mo.

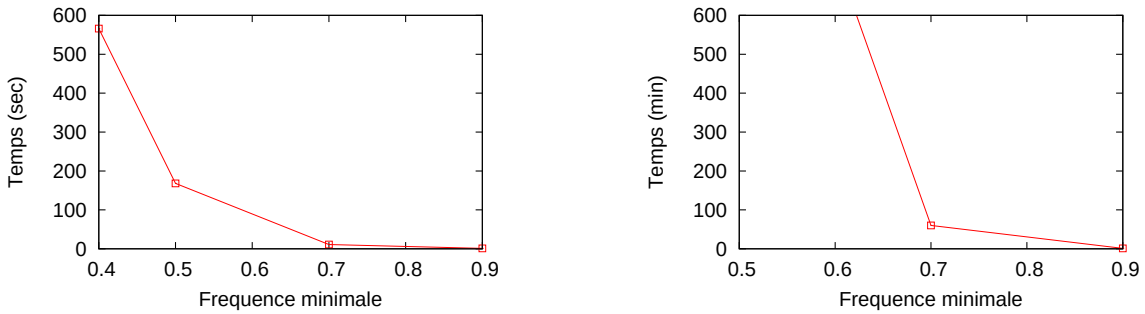



Fig 5.4 – Temps d'exécution pour (a) D5V10 (b) D7V10, (c)

Lors de l'extraction de blocs de données, le seuil de fréquence fixe par avance le nombre de blocs différents qu'il faudra explorer. C'est le seuil de confiance minimal qui permet ou non d'afficher un bloc en résultat. C'est ce que montre la figure 5.6 : le temps d'extraction varie très peu pour un même seuil de fréquence minimal et des seuils de confiance différents.

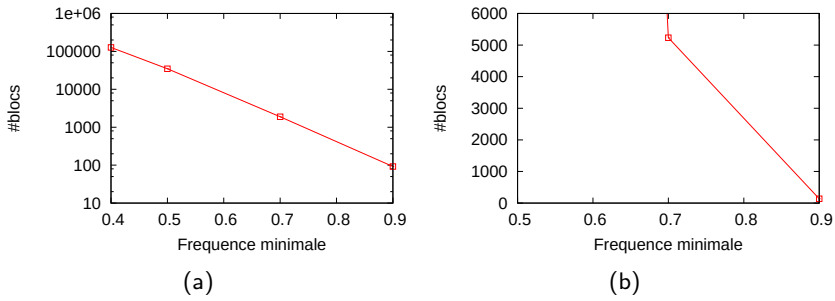



Fig 5.5 – Temps d'exécution pour (a) D5V10 (b) D7V10

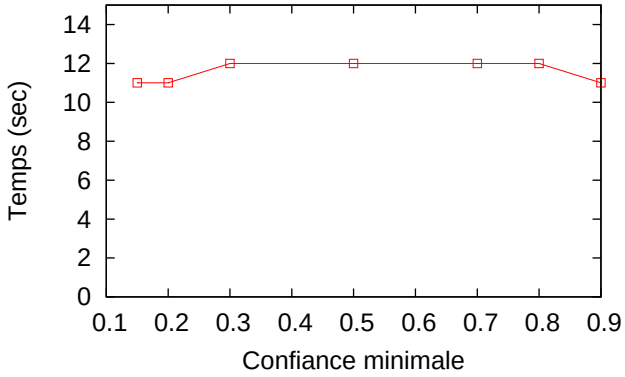



Fig 5.6 – Variation du seuil de confiance pour D5V10, avec minFreq = 40%

## Optimisations

La méthode présentée ci-dessus est perfectible au travers de diverses optimisations. Tout d'abord, les expérimentations menées ci-dessus montrent que les performances de l'algorithme dépendent fortement du nombre de membres par dimension ainsi que du seuil de confiance minimal. Or, notre algorithme procède par partitionnement du cube en sous-cubes constituant des blocs de données. La jointure entre ces sous-cubes se réalise facilement, d'autant plus lorsque les intervalles des membres d'une dimension sont contigus. Ces éléments montrent que les performances en terme de temps de notre méthode peuvent être grandement améliorées par des techniques de parallélisation, qui peuvent aisément être mises en œuvre.

Actuellement, notre méthode ne tient pas compte des cellules vides. Poutant, dans les jeux de données réels, ces cellules constituent la majorité du cube. Il conviendrait donc de les prendre en compte de manière plus efficace, notamment au travers d'un prétraitement. Par exemple, ces ensembles de cellules peuvent eux-même constituer des blocs de données, qu'il suffirait de prendre en compte lors de la génération des blocs. Le gain en terme de temps et mémoire serait alors non négligeable.

Dans la section suivante, nous présentons comment s'appuyer sur ces blocs pour extraire des motifs graduels multidimensionnels.

## 5.4 Extraction de règles graduelles multidimensionnelles

Dans cette section, nous expliquons comment extraire des itemsets graduels multidimensionnels à partir de blocs de données.

### 5.4.1 Les DG-sets

Nous souhaitons extraire des corrélations de variation de mesures (valeurs des blocs) associées à des dimensions dont les membres sont ordonnés. Par exemple, à partir du cube du tableau 5.2a, nous souhaiterions extraire "*Plus le nombre de jours de maladie diminue et plus le stade de la maladie augmente, alors plus la valeur des blocs augmente*".

De manière plus générique, on peut voir cette gradualité comme "*Plus (moins)* $d_1$*, ... , et plus (moins)* $d_n$*, alors plus la valeur des blocs augmente (diminue)*". Ce type de corrélations, que nous appellerons **DCG**, est clairement composée de corrélations de variations sur deux ensembles distincts :

- Les dimensions en elles-mêmes (première partie de la règle)
- La mesure (seconde partie de la règle)

**La première partie** concerne la gradualité sur les dimensions, ce qui revient à comparer les membres des dimensions. Ainsi, dans notre approche, nous considérons que le cube de données contient des dimensions ordonnées :

**Définition 33.** *(Dimension ordonnée) Une dimension* $d$ *est ordonnée si son domaine est muni d'une relation d'ordre total.*

Par exemple, les dimensions $d_1$ et $d_2$ du cube $C$ sont ordonnées, car elles peuvent être munie d'une relation d'ordre total : nous avons, pour $d_1$ : $10 \leq 20 \leq 30 \leq 40 \leq 50$, et pour $d_2$ : $1 \leq 2 \leq 3 \leq 4 \leq 5$. La définition 33 nous permet d'introduire la notion de gradualité sur les dimensions. Ainsi, nous avons les notions sémantiques "*le nombre de jours augmente*" ou "*le nombre de jours diminue*".

Dans ce chapitre, nous ne considérons que les dimensions ordonnées, ce qui signifie que les dimensions qui ne sont pas ordonnables seront ignorées par notre méthode.

**La seconde partie** concerne l'augmentation ou la diminution de la mesure, au travers de l'utilisation des blocs. De manière plus formelle, nous définissons une DCG de la manière suivante :

**Définition 34.** *(Dimension graduelle) Une 1-DG est de la forme* $[d^*, *_m]$*, où* $d^*$ *est une dimension graduelle telle que* $*_m \in \{\leq, \geq\}$ *est un opérateur se rapportant à la mesure (valeur des blocs).*

Notons l'utilisation des opérateurs de comparaison $\{\leq, \geq\}$, qui permettent de conserver les cubes ayant des valeurs égales. Cela nous permet de maximiser l'ensemble des blocs supportant une règle. Toutefois, les blocs de valeurs égales participeront à la fois au support de l'augmentation et de la diminution.

**Définition 35.** *(DG-set) Soit* $C = \langle dom_1, ..., dom_k, dom_m, m_C \rangle$ *un cube. Une DG-set est de la forme* $[\{d_l^{*l}, ..., d_i^{*i}\}, *_m]$*, où* $\{d_l^{*l}, ..., d_i^{*i}\}$ *est un ensemble de dimensions graduelles telles que* $\forall j = 1..i, d_j \in C$ *et* $*_m \in \{\leq, \geq\}$ *est un opérateur se rapportant à la mesure (valeur des blocs).*

Dans ce mémoire, nous considérons que les règles multidimensionnelles graduelles peuvent être générées à partir des DG-Set en post-traitement. Par abus de langage, une règle multidimensionnelle graduelle est en réalité un DG-Set. Dans notre contexte, comparer deux mesures revient à comparer deux blocs. Nous définissons donc l'ordre entre blocs de la manière suivante :

**Définition 36.** *(ordre entre blocs) Soit* $b = \delta_1 \times ... \times \delta_n$ *et* $b' = \delta'_1 \times ... \times \delta'_n$ *deux blocs définis sur les dimensions* $d_1, ..., d_n$ *ayant pour valeur associée* $m$ *et* $m'$ *respectivement. Soit* $r = [\{d_1^{*_1}, ..., d_n^{*_n}\}, *_m]$ *une DG-set. On dit que* $b$ *précède* $b'$ *en fonction de* $r$ *si :*

- $m *_m m'$
- $\forall j \in \{1, .., k\}$, $\min(\delta_j) *_j \min(\delta'_j) \wedge \max(\delta_j) *_j \max(\delta'_j)$

*On note* $b \lhd_r b'$.

Par exemple, lorsque l'on considère la règle multidimensionnelle graduelle $r_1 = [\{CSP^{\leq}\}, \geq]$ (*Plus le stade de la maladie diminue, plus la valeur des blocs augmente*), nous avons $b_1 \leq b_2$. De plus, $\min([1,2]) \leq \min([2,4]) \wedge \max([1,2]) \leq \max([2,4])$ $b_1$ précède donc $b_2$ ($b_1 \lhd_{r_1} b_2$). En revanche, si l'on considère $b_1 = [1]$ et $b_4 = [1,2]$, nous n'avons ni $b_1 \lhd_{r_1} b_4$, ni $b_4 \lhd_{r_1} b_1$, car $\min([1]) = \min([1,2]) \wedge \max([1]) \leq \max([1,2])$.

La généralisation à $n$ blocs ordonnés se fait alors de la manière suivante :

**Définition 37.** *(Liste de blocs ordonnés) Soit* $r = [\{d_1^{*_1}, ..., d_n^{*_n}\}, *_m]$ *une DG-set. Soit* $x$ *un entier appartenant à* $\mathbb{N}^*$. *Une liste de taille* $x$ *de blocs* $\mathcal{L} =< b_1, ..., b_x >$ *respecte* $r$ *si* $\forall i, j \in \{1, ..., x\}$ $b_i \lhd_r b_j$.

Par exemple, deux listes respectent la règle $[\{CSP^{\leq}\}, \geq]$ : $\mathcal{L}_1 =< b_3, b_2, b_4 >$ et $\mathcal{L}_2 =< b_1, b_2, b_4 >$. Afin de mesurer la représentativité d'une règle sur un cube, nous proposons d'utiliser une mesure de fréquence définie de la manière suivante :

**Définition 38.** *Soit* $C$ *un cube,* $\mathcal{B}$ *le nombre de blocs extraits sur ce cube et* $\mathcal{G}_r = \{\mathcal{L}_1 ... \mathcal{L}_z\}$ *l'ensemble de toutes les listes respectant* $r$. *Alors* $Freq(r) = \frac{\max_{1 \leq i \leq z}(|\mathcal{L}_i|)}{\mathcal{B}}$

### 5.4.2 Propriétés des DG-sets

Dans cette partie, nous montrons que nos définitions sont compatibles avec les propriétés classiques en fouille de données. Ainsi, nous retrouvons par exemple la propriété d'anti-monotonie. Pour ce faire, nous redéfinissons la notion d'inclusion de la manière suivante :

**Définition 39.** *(Inclusion) Soient* $r = [\{d_1^{*_1}, ..., d_n^{*_n}\}, *_m]$ *et* $r' = [\{d_1^{*_1\prime}, ..., d_o^{*_o}\}, *'_m]$ *deux DG-sets.* $r$ *est inclus dans* $r'$ *si*

- $*_m = *'_m$
- $\forall d$ $(d \in \{d_1^{*_1}, ..., d_n^{*_n}\} \Rightarrow d \in \{d_1^{*_1\prime}, ..., d_o^{*_o}\})$

*On note* $r \sqsubseteq r'$

Par exemple, $[\{d_1^{\leq}, d_2^{\geq}\}, \leq] \sqsubseteq [\{d_1^{\leq}, d_2^{\geq}, d_3^{\geq}\}, \leq]$. Par contre, $[\{d_1^{\leq}, d_2^{\geq}\}, \leq]$ n'est pas inclus dans $[\{d_1^{\leq}, d_2^{\geq}, d_3^{\geq}\}, \geq]$.

**Proposition 4.** *(Anti-monotonie DG-set) Soient* $r$ *et* $r'$ *deux DG-sets, nous avons :* $r \sqsubseteq r' \Rightarrow Freq(r) \geq Freq(r')$.

*Démonstration.* Soient deux DG-set $r_k$ et $r_{k+1}$ tels que $r_k \subseteq r_{k+1}$, avec $k$ et $k+1$ la longueur de ces DG-sets. Soit $ml$ la liste de taille maximale $\mathcal{G}_{r_k}$. Nous avons $\forall b, b' \in ml$ :

- si $\neg(b \lhd_{r_{k+1}} b')$ alors $b'$ ne fera pas partie de $\mathcal{G}_{r_k}$ et par conséquent $Freq(r_k) > Freq(r_{k+1})$
- si $(b \lhd_{r_{k+1}} b')$, alors $b'$ fera partie de $\mathcal{G}_{r_k}$ et par conséquent $Freq(r_k) = Freq(r_{k+1})$

Ainsi, nous avons $Freq(r_k) \geq Freq(r_{k+1})$. □

Comme dans les méthodes d'extraction de connaissance classiques, l'anti-monotonie des DG-sets permet de tronquer l'espace de recherche dès qu'un ensemble ne respecte pas le support minimal. Cependant, le nombre de combinaisons différentes de corrélations graduelles à considérer reste plus élevé que pour l'extraction d'itemsets. Par exemple, pour $n$ dimensions, il existe $2^{n+1}$ DG-sets à tester (contre $2^n$ dans le cas classique). Comme dans le chapitre 3, nous utilisons la notion de complémentarité due à la gradualité afin de réduire l'espace de recherche. De même que pour les motifs graduels présentés précédemment nous avons :

**Définition 40.** *(complémentaire) Soit* $r = [\{d_1^{*_1}, ..., d_n^{*_n}\}, *_m]$ *une DG-set. Sa DG-set complémentaire est* $c(r) = [\{d_1^{'*_1}, ..., d_n^{'*_n}\}, *'_m]$ *si* $\forall j \in [1, n] d_j = d'_j$ *et* $*_j = c_*(*'_j)$ *et* $*_m = c_*(*'_m)$*, où* $c_*(\geq) = \leq$ *et* $c_*(\leq) = \geq$.

Par exemple, $c([\{d_2^{\geq}\}, \leq]) = [\{d_2^{\leq}\}, \geq]$, mais $c([\{d_2^{\geq}\}, \leq]) \neq [\{d_2^{\geq}\}, \geq]$.

**Proposition 5.** *Soit* $r$ *un DG-set tel que* $c(r)$ *est le complémentaire de* $r$*. Alors l'ensemble des listes composant* $\mathcal{G}_r$ *est le même que celles composant* $\mathcal{G}_{c(r)}$.

**Corollaire 2.** $Freq(r) = Freq(c(r))$

Le corollaire 2 montre que le support de la moitié des DG-sets peut être déduit de manière automatique. Il ne sera donc pas nécessaire de les générer. D'autre part, il se trouve que ces DG-sets sont en réalité des informations redondantes. En effet, ce corollaire montre que "*Plus le nombre de jours de maladie diminue et plus le stade de la maladie augmente alors plus la valeur augmente*" est exactement la même chose que "*Moins le nombre de jours de maladie diminue et moins le stade de la maladie augmente alors plus la valeur diminue*".

### 5.4.3 Algorithme

La méthode adoptée afin d'extraire les règles graduelles multidimensionnelles repose sur les algorithmes décrits au chapitre 3. Cependant, dans les chapitres précédents, l'espace de recherche était composé des attributs de la base et la fréquence reposait sur le nombre d'objets de la base dont les variations de valeurs suivaient le sens de l'itemset graduel considéré. Dans le contexte multidimensionnel, et plus particulièrement celui des blocs, l'espace de recherche et les objets sont différents. Ici, l'espace de recherche est constitué des dimensions. La fréquence est basée sur le nombre de blocs.

Nous utilisons un algorithme par niveau qui augmente à chaque passe le nombre de dimensions graduelles. Ainsi, à l'image de l'algorithme Apriori de [AS94], nous construisons un arbre des préfixes. Dans cette structure, chaque nœud contient une dimension graduelle associée à un opérateur graduel sur la mesure (correspondant à la seconde partie de la règle). Le chemin d'un nœud à la racine représente une DG. Le corollaire 2 nous permettant de ne générer que la moitié des DG, nous illustrons dans ce chapitre l'extraction de connaissances graduelles sur l'augmentation de la mesure (les supports des diminutions

sont obtenus en inversant les opérateurs). De plus, nous rappelons que les dimensions considérées sont des dimensions ordonnées, les dimensions non ordonnées étant ignorées.

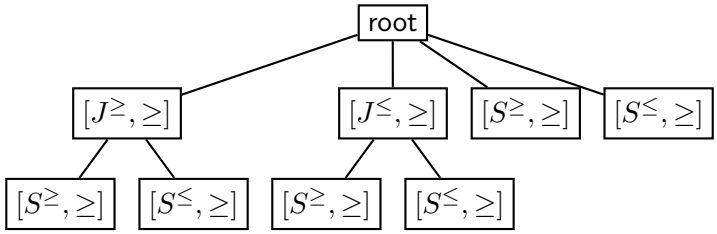



Fig 5.7 – Un exemple d'arbre préfixé pour l'extraction de DG-sets

La figure 5.7 montre l'arbre des préfixes généré pour extraire des DG-sets à partir du tableau 5.2a. L'arbre s'étend sur deux niveaux car il n'y a que deux dimensions. Nous remarquons que nous avons généré 4 noeuds au niveau 2, au lieu des 8 recouvrant la totalité des corrélations graduelles possibles. D'autre part, l'arbre généré est déséquilibré, ce qui permet d'éviter toute redondance.

Apriori est un algorithme *par niveau*, c'est-à-dire que que les noeuds de niveau $k$ sont générés par jointure sur les noeuds du niveau $k-1$. L'algorithme exploite ainsi la propriété d'anti-monotonie en alternant les étapes de génération et de comptage jusqu'à ne plus avoir de candidats fréquents.

L'extraction de gradualité nécessite de conserver pour chaque candidat généré l'ensemble des ordres possibles (l'ensemble $\mathcal{G}_r$), afin de considérer le meilleur lors de la génération du DG-set de taille $k+1$. Cependant, cet ensemble dépend fortement du nombre de blocs extraits. Ainsi, nous utilisons la structure binaire proposée précédemment. Cette matrice binaire est définie sur les blocs, et non plus sur les objets de la base :

$$\forall a, b \in \{1, ..., x\} \times \{1, ..., x\}, t_a, t_b \in \mathcal{T}_{\mathcal{G}_r} \begin{cases} m_{t_a,t_b} = 1 & si\ t_a \lhd_r t_b, \\ m_{t_a,t_b} = 0 & sinon \end{cases}$$

En reprenant l'exemple précédent, l'algorithme construit lors de la première passe les matrices représentées par les tableaux 5.9a et 5.9b :

| $\vec{\Gamma}$ | $b_1$ | $b_2$ | $b_3$ | $b_4$ |
|---|---|---|---|---|
| $b_1$ | 1 | 1 | 0 | 1 |
| $b_2$ | 0 | 1 | 0 | 1 |
| $b_3$ | 0 | 1 | 1 | 1 |
| $b_4$ | 0 | 0 | 0 | 1 |

(a)

| $\vec{\Gamma}$ | $b_1$ | $b_2$ | $b_3$ | $b_4$ |
|---|---|---|---|---|
| $b_1$ | 1 | 1 | 0 | 0 |
| $b_2$ | 0 | 1 | 0 | 0 |
| $b_3$ | 0 | 0 | 1 | 0 |
| $b_4$ | 0 | 1 | 0 | 1 |

(b)

Tab 5.9 – Matrice binaire pour (a) $[J^{\leq}, \geq]$ et (b) $[S^{\geq}, \geq]$

Ensuite, l'algorithme de calcul de fréquence glouton présenté au chapitre 3 est utilisé. A l'issue de ce processus, nous avons extrait en deux temps des règles graduelles multidimensionnelles basées sur les blocs.

## 5.5 Expérimentations

Dans cette section, nous décrivons les expérimentations menées. Nous avons implémenté les algorithmes présentés ci-dessus en C++. Ces algorithmes ont été testés en terme de temps, de mémoire et de nombre de motifs extraits. Pour ce faire, les blocs utilisés sont générés de manière aléatoire, en prenant en compte le nombre de dimensions ($|D|$), le nombre de membres par dimension (compris entre 0 et 10), le nombre de valeurs de blocs différentes ($|V|$) ainsi que le nombre de blocs ($|\mathcal{B}|$). Nous avons généré les trois fichiers présentés dans le tableau 5.10 :

| Name | $\|D\|$ | $\|V\|$ | $\|\mathcal{B}\|$ |
|---|---|---|---|
| D5V10B40 | 5 | 10 | 40 |
| D10V100B500 | 10 | 100 | 500 |
| D10V100B5000 | 10 | 100 | 5000 |

Tab 5.10 – Spécifications des jeux de test

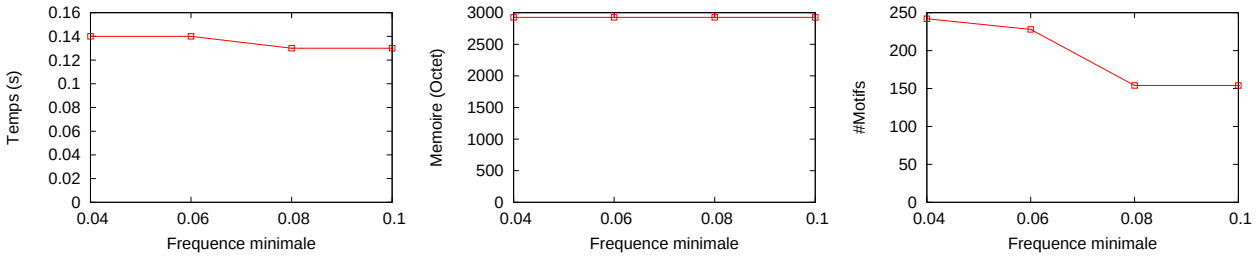
Les expérimentations ont été menées afin de mesurer les consommations en terme de temps, et mémoire en fonction du support minimal. De plus, nous avons conservé le nombre de DG-sets extraits. Les jeux de données étant générés de manière aléatoire, il est nécessaire de baisser le support afin de trouver des DG-sets. Les expérimentations ont été menées sur un serveur possédant un processeur Intel(R) Xeon(R) CPU E5450 @ 3.00GHz, et ayant 16Go de mémoire vive.

Les résultats obtenus sont satisfaisants en terme de temps d'exécution et de mémoire. Ainsi, pour un jeu contenant un faible nombre de dimensions et de blocs, il faut environ 1 seconde pour d'extraire environ 250 DG-sets. L'algorithme est particulièrement sensible au nombre de blocs, plus qu'au nombre de dimensions et de valeurs de dimensions. C'est ce que montrent les figures 5.8a, 5.9a et 5.10c. En effet, le nombre de blocs, fixé à 5000 dans le jeu de données D10V100B500, rend le temps d'exécution plus long : environ 17 minutes pour un support minimal fixé à 0.1 sont nécessaires à l'extraction de 80 DG-sets. En revanche, pour le même nombre de dimensions et le même support, mais seulement 500 blocs, il faut environ 30 secondes afin d'extraire 120 DG-sets.

Nous avons également exécuté notre algorithme sur le jeu de données reflétant la réalité *Chess Endgame Database for White King and Rook against Black King (KRK) – Black-to-move Positions Drawn or Lost in N Moves*[2]. Ce jeu calcule, pour une position donnée d'un roi blanc, d'une tour blanche et d'un roi noir, le nombre de coups à jouer de manière optimale afin de mener à la victoire de la partie blanche. Notons que pour certaines positions, il peut y avoir match nul (considéré comme valeur nulle dans notre cas). Le cube de données a alors été construit de la manière suivante :

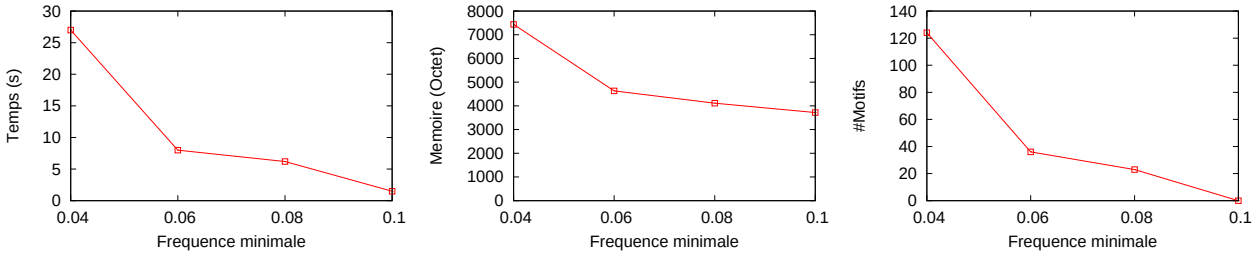
- la dimension 1 ($D1$) représente la distance euclidienne entre le roi blanc et le roi noir,
- la dimension 2 ($D2$) représente la distance euclidienne entre le roi blanc et la tour blanche,
- la dimension 3 ($D3$) représente la distance euclidienne entre le roi noir et la tour blanche.

2. http ://archive.ics.uci.edu/ml/datasets/Chess+(King-Rook+vs.+King)



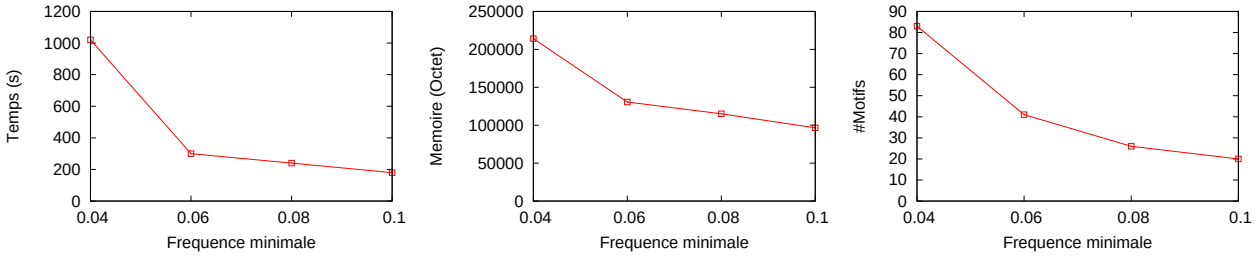

(a) (b) (c)

Fig 5.8 – Pour D5V10B40, en fonction du support (a) Temps d'exécution, (b) Mémoire utilisée, (c) Nombre de DG-sets extraits




(a) (b) (c)

Fig 5.9 – Pour D10V100B500, en fonction du support (a) Temps d'exécution, (b) Mémoire utilisée, (c) Nombre de DG-sets extraits




(a) (b) (c)

Fig 5.10 – Pour D10V100B5000, en fonction du support (a) Temps d'exécution, (b) Mémoire utilisée, (c) Nombre de DG-sets extraits

Le jeu contient 28056 instances réparties sur 648 cellules après calcul du cube en agrégeant sur les attributs non présents dans le schéma multidimensionnel (group by). La mesure d'une cellule contient le nombre de pas à jouer afin de mener à la victoire des blancs, et null si match nul. L'extraction de blocs a été effectuée avec un support de 5 cellules, et une confiance minimale de 80% (4 cellules sur 5). Nous avons obtenu 1816 blocs différents. Nous avons ensuite extrait les blocs multidimensionnels graduels en utilisant un faible support : 2%. Le tableau 5.11 résume les résultats obtenus.

| | | | |
|---|---|---|---|
| $D1^{\geq}D2^{\geq}, 18.1\%$ | $D1^{\leq}D2^{\geq}, 10.9\%$ | $D1^{\geq}D2^{\geq}D3^{\geq}, 4.6\%$ | $D1^{\leq}D2^{\geq}D3^{\geq}, 2.5\%$ |
| $D1^{\geq}D2^{\leq}, 10.2\%$ | $D1^{\leq}D2^{\leq}, 10.2\%$ | $D1^{\geq}D2^{\geq}D3^{\leq}, 4\%$ | $D1^{\leq}D2^{\geq}D3^{\leq}, 3.5\%$ |
| $D1^{\geq}D3^{\geq}, 12.7\%$ | $D1^{\leq}D3^{\geq}, 12.1\%$ | $D1^{\geq}D2^{\leq}D3^{\geq}, 2.6\%$ | $D1^{\leq}D2^{\leq}D3^{\geq}, 3.6\%$ |
| $D1^{\geq}D3^{\leq}, 14.1\%$ | $D1^{\leq}D2^{\leq}, 18.7\%$ | $D1^{\geq}D2^{\leq}D3^{\leq}, 2.7\%$ | $D1^{\leq}D2^{\leq}D3^{\leq}, 5\%$ |
| $D2^{\geq}D3^{\geq}, 12.6\%$ | $D2^{\leq}D3^{\geq}, 9.3\%$ | | |
| $D2^{\geq}D3^{\leq}, 17.1\%$ | $D2^{\leq}D3^{\leq}, 14.5\%$ | | |

Tab 5.11 – Blocs graduels obtenus sur le jeu de données Chess

De ces expérimentations, nous déduisons les règles suivantes : plus la distance entre le roi blanc et le roi noir est élevée et plus la distance entre le roi noir et la tour blanche est élevée alors moins le nombre de coups à jouer pour gagner diminue. En revanche, si l'on ajoute que la distance entre la tour blanche et le roi blanc augmente (respectivement diminue) alors le support du nombre de coups à jouer passe à 5% (respectivement 3.5%). De manière générale, nous déduisons de ces résultats que les manières de gagner optimales se jouent entre deux distances : soit la distance entre le roi blanc et la tour blanche est faible, soit la distance entre le roi noir et la tour blanche est élevée. En revanche, nous montrons que ces trois distances ne sont pas liées par une co-variation graduelle puisqu'aucun résultat n'est produit par notre algorithme.

## 5.6 Discussion

Dans ce chapitre, nous proposons une approche originale permettant d'extraire, à partir de cubes de données, des règles graduelles multidimensionnelles de la forme *Plus le nombre de jours de maladie augmente et plus le stade de la maladie augmente, plus le nombre de patients est grand*. Ces règles sont extraites en considérant des dimensions ordonnées et des blocs de données extraits selon la valeur de la dimension. Cette approche permet de dégager les tendances qui sont présentes dans les cubes de données. Nous nous appuyons pour ce faire sur une méthode de découverte à partir d'algorithmes par niveau en faisant croître le nombre de dimensions présentes dans les règles graduelles générées, et en considérant une représentation binaire pour représenter les ordres entre blocs décrivant la valeur de mesure en fonction des valeurs des dimensions. Nos expérimentations prouvent que l'algorithme est efficace en terme d'utilisation mémoire, et qu'il est fortement dépendant du nombre de blocs. Le nombre de dimensions en revanche influe peu sur le temps d'exécution.

La sémantique des règles extraites peut-être améliorée, notamment en intégrant la notion de hiérarchie lors de l'extraction. En effet, l'une des particularités des cubes de données est d'associer à chaque

dimension une hiérarchie permettant ainsi à l'utilisateur d'affiner le niveau de granularité des informations lors de sa navigation. Cependant, l'intégration de ces structures de données au processus de fouille peut s'avérer compliqué notamment lors de l'agrégation au niveau hiérarchique supérieur, comme le montrent [IKA02, DHL+04] ou encore [PLT08]. Dans notre contexte, la mesure choisie lors de l'agrégation conditionne la dernière partie de la règle graduelle multidimensionnelle, puisqu'elle influera directement sur les valeurs associées aux blocs extraits. Toutefois, l'intégration de règles graduelles généralisées permettrait d'améliorer le support et donc la pertinence des résultats présentés à l'utilisateur.

Dans [CLL07], les auteurs utilisent la confiance comme contrainte d'élagage des blocs. Bien que non-antimonotone, cela permet d'explorer des blocs qui n'auraient jamais pu être atteints en utilisant uniquement la contrainte de fréquence. Cela demande une étude théorique préalable qui permettrait de borner les erreurs lors des générations suivantes.

Enfin, notre méthode ne prend pas en compte le recouvrement des blocs. En cas de recouvrement total, nous prenons en compte le bloc de taille maximale afin de présenter la règle la plus représentative à l'utilisateur. Cependant, ces recouvrements peuvent apporter potentiellement des informations intéressantes, puisque plus ciblées. D'autre part, les "chevauchements", sans recouvrement total, offrent une sémantique différente, que nous n'avons pas exploitée au travers de la méthode proposée.

En conclusion, l'extraction de motifs graduels à partir de bases multidimensionnelles est prometteuse, notamment dans le contexte actuel de construction de nombreux entrepôts de données médicaux. Il s'agira alors de valider expérimentalement les approches décrites dans ce chapitre sur de gros volumes de données. Au sein de ces entrepôts de données, la prise en compte de l'aspect historisé des données sera importante.

Dans le chapitre suivant, nous abordons l'impact de cette prise en compte pour découvrir des motifs graduels intégrant la notion de temporalité. Nous étudions pour ce faire les liens entre motifs graduels et bases de données séquentielles. Nous nous intéressons également aux liens entre les motifs graduels, Pareto et les skylines.

# Chapitre 6

# Gradualité, skylines et motifs séquentiels

Dans ce chapitre, nous mettons en relation notre travail avec des problématiques quelques peu différentes, mais régulièrement abordées en fouille de données. Les skylines (ou opérateurs de Pareto) définissent, à partir d'une requête utilisateur, l'ensemble des solutions offrant un meilleur compromis. Il s'agit ici de définir des méthodes capables de répondre à de telles requêtes dans un temps acceptable. Dans une première partie, nous décrivons les travaux relatifs à cette problématique, puis nous démontrons que les itemsets graduels permettent de répondre à cette problématique. Nous décrivons alors les différentes possibilités d'utilisation des itemsets graduels dans le contexte de la recherche des skylines.

Dans la seconde partie de ce chapitre, nous abordons les motifs séquentiels multidimensionnels graduels. Nous expliquons deux visions différentes de la gradualité à partir de base de données séquentielles, et montrons que la problématique d'extraction de motifs séquentiels graduels est complexe.

## 6.1 Itemsets graduels et skylines

Dans cette partie, nous abordons le concept des skylines. Après un bref état de l'art, nous expliquons en quoi les skylines et les motifs graduels sont liés. Nous montrons comment l'algorithme complet présenté au chapitre 3 permet de résoudre de manière efficace les différentes problématiques liées aux skylines.

### 6.1.1 Travaux existants

Les skylines visent à déterminer les objets d'une base de données offrant les meilleures solutions selon divers critères. L'un des exemples le plus courament cité et sûrement le plus intuitif est celui du choix d'un hôtel : considérons un touriste qui recherche un hôtel près de la plage au meilleur prix. Après consultation de divers hôtels sur internet, notre touriste sélectionne les hôtels de la base 6.1.

| Hotel | Distance (m) | Prix (€) |
|---|---|---|
| $h_1$ | 100 | 70 |
| $h_2$ | 300 | 60 |
| $h_3$ | 500 | 40 |
| $h_4$ | 100 | 80 |
| $h_5$ | 400 | 70 |

Tab 6.1 – Exemple d'hôtels avec leur prix et distance à la plage

Selon les critères voulus, les hôtels $h_4$ et $h_5$ ne sont pas les meilleurs : $h_4$ est plus cher que $h_1$ alors qu'ils sont à même distance de la plage, et $h_5$ est plus loin que $h_1$ alors qu'il est au même prix. En revanche, $h_1$, $h_2$ et $h_3$ offrent de bons compromis. Ces points de "compromis", représentés par la ligne rouge sur la figure 6.1, constituent *l'ensemble skyline* des dimensions "prix" et "distance". De tels ensembles sont activement utilisés durant les processus de décision faisant intervenir plusieurs critères, et permettent de mieux comprendre les données.

Notons que les skylines sont l'équivalent de l'*optimum de Pareto* (ou *front de Pareto*) très utilisé par exemple en théorie de l'économie. Il s'agit dans ce contexte de trouver l'ensemble des solutions optimisant une problématique, comme par exemple le bon équilibre entre production et consommation,

répartition des richesses et bien être etc... Pour cette raison, les skylines sont parfois désignés dans la littérature sous le terme "*opérateurs de Pareto*". De plus, si l'utilisation des skylines est relativement récente dans la communauté fouille de données, il existe de nombreuses problématiques proches et étudiées depuis de très nombreuses années en statistique : les enveloppes convexes, les top-k ou encore les plus proches voisins.

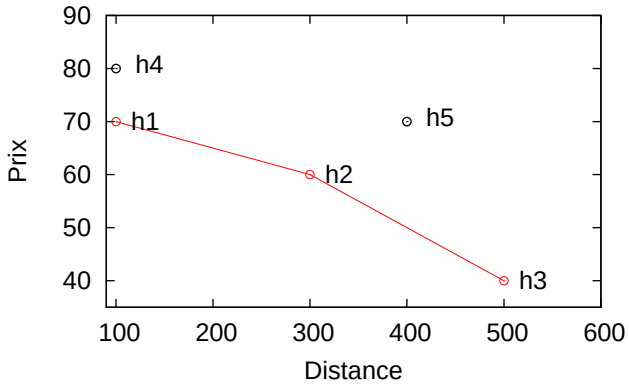



Fig 6.1 – Projection des hôtels sur le prix et la distance

De manière informelle, la problématique d'extraction de l'ensemble skyline consiste à trouver tous les points $p$ parmi l'ensemble des objets de la base et sur $\mathcal{D}$ dimensions n'étant *dominés* par aucun autre point sur ces dimensions. Dans la littérature, les dimensions désignent les attributs de la base de données et les points les objets. Dans un souci de cohérence de vocabulaire dans l'ensemble de ce manuscrit, nous utiliserons le terme "*item*" plutôt que le terme "*dimension*", et le terme "*objet*" plutôt que le terme "*point*". Dans ce contexte, un objet $o$ *domine* un objet $o'$ selon un ensemble d'items $\mathcal{I} = \{i_1, .., i_n\}$ si $\forall i \in \mathcal{I}, o[i] \leq o'[i]$. Ainsi, un "*sous-ensemble skyline*" désignant dans la littérature les points dominants sur un sous-ensemble de dimensions sera appelé ici "*itemset graduel*"

A notre connaissance, la première utilisation des skylines en base de données est introduite par [BKS01] avec des requêtes SQL ayant la forme suivante :

SELECT ... FROM ... WHERE ...
GROUP BY ... HAVING ...
SKYLINE OF [DISTINCT] $i_1$ [MIN | MAX | DIFF], ..., $i_n$ [MIN | MAX | DIFF]
ORDER BY ...

où MIN, MAX et DIFF spécifient si la valeur doit être la plus petite, la plus grande ou simplement différente. Pour répondre à ce type de requête, les auteurs introduisent plusieurs algorithmes :

- L'algorithme Blocks-Nested-Loops (BNL) consiste à comparer tous les objets de la base deux à deux sur les attributs de la requête SQL
- L'algorithme Divide & Conquer (D & C) consiste à diviser la base de données selon les objets en plusieurs partitions, de trouver les skylines sur chacune des partitions, puis de fusionner les objets skylines de chaque partition

- L'algorithme basé sur les arbres B-Tree[1] consiste à parcourir deux à deux les B-Tree afin de faire émerger les skylines. Cela revient à maintenir des listes ordonnées et à en faire émerger les objets incomparables ayant les index les plus petits.

Les auteurs proposent une étude expérimentale afin de comparer le comportement de ces différents algorithmes. Les bases de données utilisées varient de 5 à 10 items et de 10 à 100 Mo. L'algorithme BNL est plus efficace que D & C lorsqu'il y a peu d'items (de l'ordre de 5 items), mais s'effondre au-delà.

[CGGL03] propose une optimisation de l'algorithme BNL. Les auteurs démontrent qu'ordonner au préalable les données de la base selon une fonction monotone améliore de manière considérable les performances de BNL. En effet, tester si un objet $o$ domine un autre objet $o'$ dans ce contexte revient à comparer les scores de $o$ et $o'$. Les expérimentations, effectuées sur des bases de 100Mo et 7 items montrent une amélioration considérable des temps de réponses.

[PTFS03] propose l'algorithme Branch-and-Bound Skyline (BBS) basé sur le principe des plus proches voisins. Pour ce faire, les auteurs utilisent un R-Tree, structure populaire en informatique et permettant de retrouver rapidement les objets les plus proches (en terme de distance) d'un objet $o$. BBS parcoure le R-Tree à l'aide d'un tas de manière optimale : il évalue et ajoute un nœud (un objet) dans l'arbre en commençant par les objets les plus proches de l'origine. A chaque itération, l'algorithme étend un nœud, l'évalue et l'élague s'il n'est pas skyline. L'algorithme s'arrête lorsqu'il n'y a plus d'objets dans le tas. Les expérimentations montrent de meilleures performances que BNL, mais sur des bases à 5 items. Une étude expérimentale plus poussée ainsi qu'une discussion sur les divers types de skylines peuvent être trouvées dans [PTFS05].

[GSG05] introduisent l'algorithme Linear-Elimination-Sort for Skyline (LESS), dont l'avantage majeur est de résoudre une requête skyline en $\mathcal{O}(dn)$. Pour ce faire, les auteurs combinent les algorithmes BNL, SFS et FLET, un algorithme de calcul d'enveloppe convexe [BCL90]. Les principales améliorations se portent sur l'évaluation des skylines et le tri externe des données. Les quelques expérimentations décrites démontrent que LESS est largement plus efficace que SFS pour 7 items.

[XZT08] propose la notion de "*relation de domination flexible*" : les auteurs constatent que la définition des skylines ne permet pas à l'utilisateur un contrôle poussé des résultats. Par exemple, lorsqu'il recherche les hôtels proposant les meilleurs compromis, l'utilisateur peut souhaiter privilégier l'une des caractéristiques, comme par exemple le prix de l'hôtel. Pour résoudre ce problème, les auteurs définissent les $\varepsilon$-Skylines, dont le paramètre $\varepsilon$ permet de conserver l'ensemble classique des objets skylines, de le diminuer ou de l'augmenter. De plus, les auteurs associent à chaque item une fonction de préférence (un poids) qui permet de privilégier certains objets. Les auteurs montrent comment adapter l'algorithme SFS à cette problématique, et définissent l'algorithme IFR, basé sur les R-Tree. Les auteurs expérimentent sur une sous-partie de la base NBA[2], qui contient les statistiques des joueurs sur 16 377 enregistrements. Au final, le nombre d'items considéré est de 7, et les temps de réponse sont acceptables.

1. arbre équilibré stockant les données sous une forme triée et permettant une exécution des opérations d'insertion et de suppression en temps amorti logarithmique

2. www.databasebasketball.com

Les travaux présentés ci-dessus ont pour idée initiale d'inclure les skylines au sein de requêtes SQL. Les différentes propositions concernent surtout la manière de calculer les skylines en fonction de cette requête. Les algorithmes peuvent être classés selon trois catégories : les approches basées sur les boucles imbriquées, les approches basées sur des stratégies de division, et les approches basées sur les techniques d'indexation. Les auteurs considèrent généralement de plus différentes variantes de la requête classique, pour lesquelles il est possible d'adapter facilement leurs propositions :

- Les **requêtes Top-**$k$ consistent à extraire les $k$ objets d'un ensemble de données minimisant une fonction de préférence donnée par l'utilisateur
- Le $K$-**Skyband** [GSYZ09] retourne les objets qui sont dominés par au plus $K-1$ objets. Cette notion est une généralisation des skylines, puisqu'un skyline est un 1-Skyband.
- Les **requêtes skylines contraintes** consistent à ne considérer que les objets dont le domaine est inclus dans une contrainte fixée par l'utilisateur. Par exemple, $Prix < 80$ peut être une contrainte.

### Le skycube

Le temps de réponse pour chaque requête skyline peut être long, et contient des itérations redondantes : relecture de la base de données, recherche des points dominants sur des ensembles d'items déjà explorés. Partant de ce constat, [YLL+05] propose de calculer le SKYCUBE, un cube de données contenant les résultats de toutes les requêtes skylines possibles. Cependant, il existe $2^{|\mathcal{I}|}-1$ sous-ensembles skylines (dans notre contexte itemsets graduels). De plus, chaque itemset graduel contient au moins un objet skyline : cela rend obligatoire le calcul de chaque itemset, sans pouvoir utiliser de mesures possédant des propriétés anti-monotones afin d'élaguer le SKYCUBE. Afin d'illustrer le SKYCUBE par un exemple, nous utilisons la base de données du tableau 6.2.

| | **A** | **B** | **C** | **D** |
|---|---|---|---|---|
| $o_1$ | 1 | 4 | 5 | 7 |
| $o_2$ | 1 | 3 | 6 | 7 |
| $o_3$ | 2 | 3 | 5 | 8 |
| $o_4$ | 3 | 5 | 5 | 1 |
| $o_5$ | 2 | 2 | 3 | 1 |

Tab 6.2 – Base de données exemple

Il s'agit alors de calculer les skylines de tous les itemsets présents dans le treillis de la figure 6.2.

Pour ce faire, les auteurs proposent deux algorithmes :

- L'algorithme BUS est un algorithme par niveau qui calcule les skylines des plus petits itemsets aux plus grands itemsets, en utilisant des filtres adaptés
- L'algorithme TDS est de type D& C et utilise deux stratégies de partage : l'une pour la fusion et l'autre pour les parents communs.

Les études expérimentales montrent que TDS surpasse BUS en terme de performance. Les auteurs ont également étendu cette approche dans [PYL+06], où ils proposent la notion de "*groupe skyline*" : ce sont des sous-ensembles d'objets ayant la même valeur sur un itemset, et appartenant également

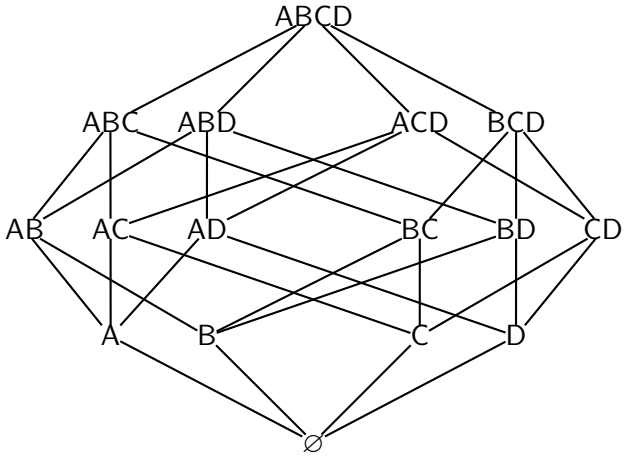



Fig 6.2 – Treillis des itemsets de la base 6.2

au skyline de l'itemset. Par exemple, $o_1$ et $o_2$ partagent la même valeur pour A. $(\{o_1, o_2\}, A)$ est donc un groupe skyline. De plus, $o_1$ et $o_2$ partagent la même valeur sur D. $(\{o_1, o_2\}, (A, D))$ est donc un groupe skyline maximal. Ainsi, plutôt que de construire le treillis des itemsets, les auteurs construisent le treillis des groupes de skylines. La figure 6.3 illustre un tel treillis. Les groupes skylines donnent une explication sémantique à la notion de skylines, puisqu'il est possible de déduire quels *sous-ensembles décisifs* conduisent à la création d'un nouveau groupe. Cependant, les groupes skylines induisent un coût de calcul supplémentaire. Dans [PWcXW07], les auteurs proposent les algorithmes Stellar et SkyEy, qui calculent tous les groupes skylines sans énumérer tous les sous-ensembles skylines.

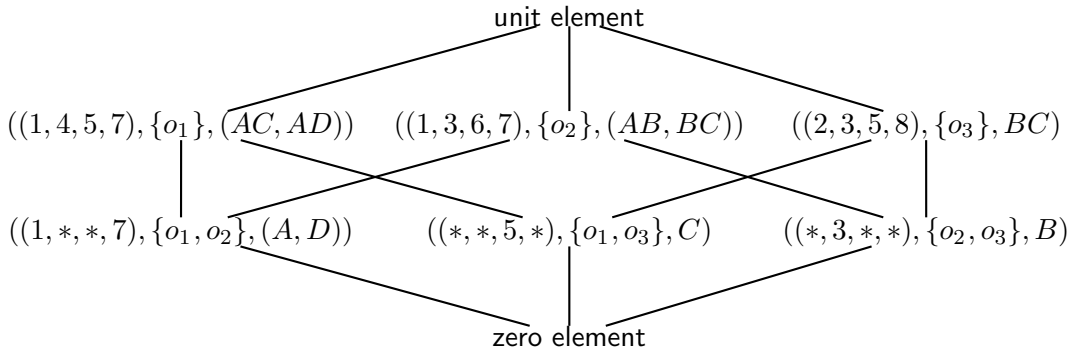



Fig 6.3 – Treillis des groupes skylines de la base 6.2

[KLRK10] propose d'utiliser des ensembles binaires afin de calculer l'intégralité du skycube. Les auteurs proposent deux méthodes : soit les objets de la base sont testés pour tous les itemsets (algorithme Point-based), et retenus s'ils en font partie, soit les combinaisons de valeurs sont testées afin de trouver les objets skylines associés (algorithme Value-based). Les expérimentations démontrent que l'algorithme Value-based est plus efficace que l'algorithme Point-based. De plus, [KLRK10] est la première approche démontrant des performances plus efficaces que [PYL$^+$06] pour les bases dont la cardinalité du domaine des attributs est inférieure à 100. Au delà, l'algorithme TDS est plus performant.

| | A | B | C | D |
|---|---|---|---|---|
| $o_1$ | 4 | 3 | 2 | 2 |
| $o_2$ | 5 | 1 | 1 | 2 |
| $o_3$ | 1 | 4 | 4 | 1 |
| $o_4$ | 3 | 5 | 5 | 1 |
| $o_5$ | 2 | 3 | 3 | 1 |

Tab 6.3 – Base exemple

[RPK10] est à notre connaissance l'approche la plus récente pour calculer le SKYCUBE. Les auteurs se basent sur des règles de dérivations afin de calculer les skylines clos. Les auteurs proposent deux algorithmes par niveau, *orion* et *orion-tail* (ce dernier utilise une technique d'élagage supplémentaire) ainsi que l'algorithme en profondeur *orion-clos* qui calcule les skylines clos. Les expérimentations montrent que les performances de ces trois nouveaux algorithmes surpassent les performances de Stellar et SkyEy. Cependant, il n'existe aucune comparaison entre [KLRK10] et [RPK10].

La recherche de skyline dans le contexte des bases de données est relativement récente, et actuellement très active dans la communauté fouille de données, comment en témoignent les publications [RPK10] et [KLRK10]. Nous avons vu dans le panorama cité ci-dessus que ce n'est que très récemment que des algorithmes très efficaces ont été proposés. La problématique de recherche de skylines est liée à la recherche de motifs graduels : dans les deux cas, l'ordre des valeurs des domaines des attributs régule les résultats et les algorithmes. Cependant, les skylines se concentrent sur les objets qui ne sont dominés par aucun autre, alors que les motifs graduels se concentrent sur les plus grandes chaînes d'ordre. Dans la section suivante, nous montrons comment adapter l'algorithme complet d'extraction d'itemsets à l'extraction des skylines et des variantes de requêtes skylines.

### 6.1.2 Calculer les skylines à partir des itemsets graduels

Dans cette section, nous utilisons l'exemple du tableau 6.3.

Le tableau 6.4 montre tous les skylines obtenus à partir de cette base :

| Itemset | Skyline | Itemset | Skyline |
|---|---|---|---|
| A | $\{o_3\}$ | BD | $\{o_2, o_5\}$ |
| B | $\{o_2\}$ | CD | $\{o_2, o_5\}$ |
| C | $\{o_2\}$ | ABC | $\{o_1, o_2, o_3, o_5\}$ |
| D | $\{o_3, o_4, o_5\}$ | ABD | $\{o_2, o_3, o_5\}$ |
| AB | $\{o_2, o_3, o_5\}$ | ACD | $\{o_1, o_2, o_3, o_5\}$ |
| AC | $\{o_1, o_2, o_3, o_5\}$ | BCD | $\{o_2, o_5\}$ |
| AD | $\{o_3\}$ | ABCD | $\{o_1, o_2, o_3, o_5\}$ |
| BC | $\{o_2\}$ | | |

Tab 6.4 – Skylines calculés à partir de l'exemple 6.3

Plus formellement, la domination d'un objet sur un autre est définie de la manière suivante :

**Définition 41.** *(domination) Soit $s = (i_1^{\geq}...i_n^{\geq})$ un itemset graduel. $o$ domine $o'$ si $\forall i \in s, i[o] \leq i[o']$ et $\exists j \in s, j[o] < j[o']$. On note $o \prec_s o'$.*

**Définition 42.** *(skyline) Soit $s = (i_1^{\geq}...i_n^{\geq})$ un itemset graduel. $o$ appartient au skyline de $s$ ($o \in SKY(s)$) s'il n'est dominé par aucun autre objet sur $s$.*

Dans le contexte de l'extraction d'itemsets graduels, nous avons le théorème 3 :

**Théorème 3.** *Soit $s = (i_1^{\geq}...i_n^{\geq})$ un itemset graduel. Alors*

$$SKY(s) = \{o \in \mathcal{O}, M^s_{\bullet o} = \overrightarrow{0}\} \cup \{o \in \mathcal{O}, \forall o' \in \mathcal{O}, M^s_{oo'} \wedge M^s_{o'o} = 1\}$$

*où $M^s$ est la matrice binaire associée à l'itemset $s$, $M^s_{\bullet o}$ est la colonne correspondant à l'indice $o$ de la matrice, et $\overrightarrow{0}$ est le vecteur nul de longueur $|\mathcal{O}|$.*

*Démonstration.* Soit $M^s$ la matrice associée à $s = s_k.s_l$ telle que $M^s = M^{s_k} \wedge M^{s_l}$. Soit $o$ tel que $M^s_{\bullet o} = \overrightarrow{0}$, alors

- Soit $M^{s_k}_{\bullet o} = \overrightarrow{0}$ alors il n'existe aucun $o'$ tel que $i[o'] \leq i[o], \forall i \in s_k$. Donc comme $s_k \subset s$, il n'existe aucun $o'$ tel que $i[o'] \leq i[o], \forall i \in s$
- Soit $M^{s_l}_{\bullet o} = \overrightarrow{0}$ alors il n'existe aucun $o'$ tel que $i[o'] \leq i[o], \forall i \in s_l$. Donc comme $s_l \subset s$, il n'existe aucun $o'$ tel que $i[o'] \leq i[o], \forall i \in s$
- Soit $M^{s_k}_{\bullet o} \neq \overrightarrow{0}$ et $M^{s_l}_{\bullet o} \neq \overrightarrow{0}$, donc $\nexists o', M^{s_k}_{o'o} = M^{s_l}_{o'o} = 1$. Donc on peut avoir $M^{s_k}_{o'o} = 1$, mais dans ce cas $M^{s_l}_{o'o} = 0$ ou inversement. On en déduit que sur $s$, $o'$ ne peut être tel que $i[o'] \leq i[o], \forall i \in s$.

Supposons maintenant que $M^{s=s_k.s_l}_{\bullet o} \neq \overrightarrow{0}$, et que si $M^s_{o'o} = 1$, alors $M^s_{oo'} = 1$. Cela signifie que $i[o'] = i[o]$, $\forall i \in s_k$ et $j[o'] = j[o]$, $\forall j \in s_l$. Ces deux objets sont donc confondus et ne sont pas dominés l'un par l'autre car la seconde condition de domination (stricte) n'est pas vérifiée. Nous ne gardons alors qu'un seul membre de cette classe d'objets. Pour tous les objets $o'$ restants, $M^s_{o'o} = 0$. La colonne $M^s_{\bullet o}$ devient donc nulle, et aucun autre objet ne domine $o$.
Nous avons démontré que $\{o \in \mathcal{O}, M^s_{\bullet o} = \overrightarrow{0}\} \cup \{o \in \mathcal{O}, \forall o' \in \mathcal{O}, M^s_{oo'} \wedge M^s_{o'o} = 1\} \subseteq SKY(s)$.
Soit un objet $o$ tel que $o \in SKY(s)$. Alors il n'existe pas d'objet $o' \in \mathcal{O}$ qui domine $o$. Cela signifie $\nexists o'$ tel que $o' \leq o$ sur toutes les dimensions, et $o' < o$ sur au moins une dimension. Alors $o \leq o', \forall o' \in \mathcal{O}$. Donc :

- Soit il n'y a pas d'objet confondu avec $o$, et $o < o'$ sur toutes les dimensions. On a $M^s_{\bullet o} = \overrightarrow{0}$
- Soit $\exists o'$ tel que $o = o'$. Par construction, $M^s_{oo'} \wedge M^s_{o'o} = 1$

Nous avons démontré que $SKY(s) \subseteq \{o \in \mathcal{O}, M^s_{\bullet o} = \overrightarrow{0}\} \cup \{o \in \mathcal{O}, \forall o' \in \mathcal{O}, M^s_{oo'} \wedge M^s_{o'o} = 1\}$. □

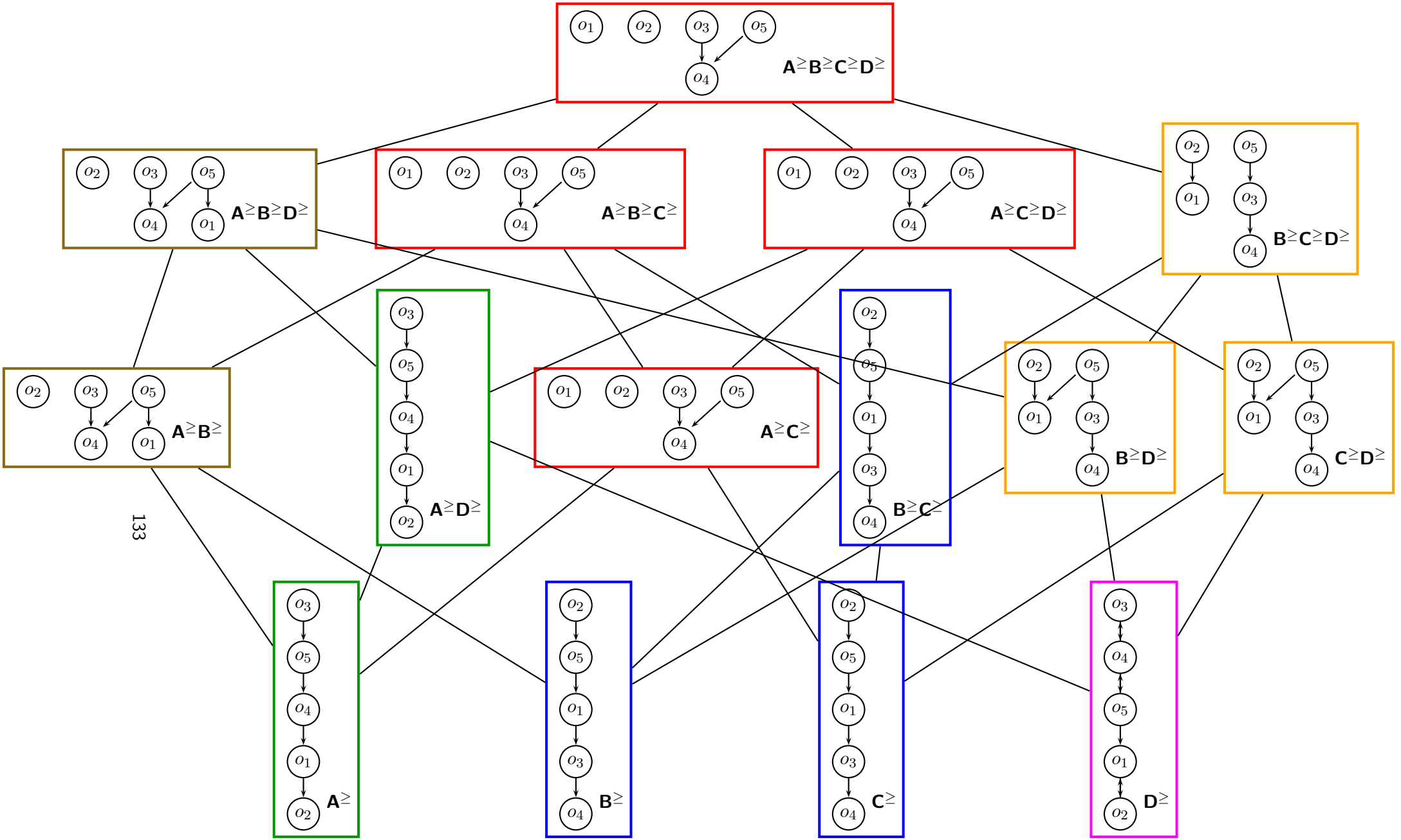





Fig 6.4 – Diagrammes de Hasse générés pour l'exemple 6.2

En d'autres termes, le théorème 3 montre que les "*racines*" des diagrammes de Hasse correspondant aux matrices binaires sont les skylines. Par exemple, considérons la figure 6.4, qui représente tous les diagrammes de Hasse générés pour la base du tableau 6.2. Par souci de lisibilité, nous les avons ordonnés selon le treillis des itemsets. Les racines du diagramme de Hasse associé à l'itemset graduel $(B^{\geq}D^{\geq})$ sont $o_2, o_5$, et correspondent bien aux sous-espaces skylines recensés dans le tableau 6.4. On note également que les objets $o_3, o_4, o_5$ sont sélectionnés comme racine de $D^{\geq}$, car ils sont égaux et aucun autre objet ne les domine. Ils vérifient la partie $\{\forall o' \in \mathcal{O}, M^s_{oo'} \wedge M^s_{o'o} = 1\}$ du théorème.

Le théorème 3 montre qu'il est possible d'utiliser l'algorithme d'extraction complet présenté au chapitre 3 afin de répondre aux différentes requêtes skylines.

**Requêtes skylines classiques**

Supposons une requête skyline de la forme :

SELECT ... FROM ... WHERE ...
GROUP BY ... HAVING ...
SKYLINE OF [DISTINCT] $i_1$ [MIN | MAX], ..., $i_n$ [MIN | MAX]
ORDER BY ...

Alors il suffit de construire les matrices binaires associées aux items $i_1 ... i_n$, en utilisant l'opérateur de comparaison $\geq$ pour MIN et $\leq$ pour MAX (les racines représentent les plus petites valeurs pour $\geq$). Nous effectuons ensuite une intersection (ET binaire) entre toutes ces matrices, et calculons les racines de la matrice finalement obtenue. Ces racines sont les skylines demandés. Ces étapes sont résumées par l'algorithme 9.

**Algorithme 9** : SkyQuery

**Données** : Une base de données $BD$,
Requête skyline $SKY(\mathcal{I})$
**Résultat** : Ensemble des objets de $SKY(\mathcal{I})$

```
1 M ← 1 /* Initialisation de tous les bits de la matrice résultat à 1        */
2 pour chaque i ∈ I faire
3 |   M' ← ConstructMat(i, *)
  |   /* Construire la matrice binaire en fonction de l'ordre associé à l'item
  |      */
4 |   M ← METM'
5 fin
```

**Résultat** : GetRoots$M$

L'avantage de cette approche est l'utilisation de la structure binaire, qui permet de faire émerger rapidement les objets skylines de $SKY(\mathcal{I})$. Contrairement aux approches de [BKS01, PTFS03, GSG05], nous n'avons pas besoin de comparer les objets à chaque itération afin de déterminer les objets skylines courants, ni de fusionner des objets aux retours des itérations D& C. En ce sens, notre méthode sera

plus rapide. Dans ce contexte, l'utilisation d'une fonction d'ordre telle que proposée par [CGGL03] est réalisée par la ligne 3 de notre algorithme. En revanche, nous souffrons de la structure binaire pour des bases de données contenant beaucoup d'objets. L'efficacité mémoire sera donc moindre.

L'algorithme 9 est une généralisation permettant de répondre à la problématique des top-$k$ skylines.

### Le K-SKYBAND

Le K-SKYBAND est une généralisation des skylines. Il consiste à trouver les objets qui sont dominés par au plus $K-1$ objets. Dans ce contexte, nous aurons besoin de calculer les niveaux associés à chaque objet de la matrice finale afin de trouver ceux possédant au plus $K-1$ pères.

**Corollaire 3.** $K-SKYBAND(s) = \{\forall o \in M^s, |M^s_{\bullet o}| \leq K-1\}$

*Démonstration.* Le corollaire 3 est une généralisation du théorème 3 □

Dans ce cas, la découverte des objets skylines associés est encore plus directe, puisque nous n'avons pas à vérifier les objets égaux dans la matrice. Contrairement à [GSYZ09], nous n'avons pas besoin de stratégie de partitionnement des données. Comme pour les requêtes skylines, nous pensons qu'une méthode basée sur les règles graduelles sera plus performante dans le cas d'un jeu de données ayant un nombre d'objets susceptible de tenir en mémoire.

### Le SKYCUBE

Le skycube consiste à calculer l'ensemble des sous-skylines, ce qui correspond dans notre cas à tous les itemsets graduels de la base. Cependant, nous ne pouvons pas utiliser la mesure de fréquence afin de réduire l'espace de recherche. Il existe donc un risque d'explosion mémoire liée au phénomène de l'explosion combinatoire.

L'approche [RPK10] ne conserve pas tous les ordres en mémoire. Les règles de dérivation permettent d'inférer une majeure partie des objets skylines pour chaque sous-ensemble. En ce sens, cette approche aura de meilleures performances qu'une approche basée sur les itemsets graduels. Notre approche devra alors être améliorée, afin de pouvoir se comparer à celle-ci.

Nous pensons qu'une approche basée sur l'extraction des itemsets graduels clos pourrait nous rapprocher des performances présentées dans [RPK10] et [KLRK10]. Afin d'illustrer notre propos, nous reprenons le treillis de la figure 6.4. L'algorithme classique énumère les itemsets par niveau, du niveau 1 à 4. Les cadres de différentes couleurs sur la figure montrent les "skylines clos", c'est-à-dire les sous-ensembles skylines partageant les mêmes objets skylines. A l'exception de $(B^{\geq}C^{\geq}D^{\geq})$, tous les diagrammes de Hasse ayant la même couleur sont identiques. Cela signifie que les groupes représentés par des couleurs peuvent être représentés par un même nœud.

Cette constatation rappelle la définition des itemsets clos proposée dans [ABLP10]. Un itemset graduel $s$ est clos s'il n'existe pas d'itemset graduel $s'$ tel que $s' \subset s$ et $Freq(s) = Freq(s')$. Cette notion de clos est basée sur la fréquence, et non sur les skylines. Les premières expérimentations menées à partir de cette définition montrent une nette amélioration des performances de notre algorithme. En cela, nous pensons que les itemset graduels clos peuvent être très efficaces dans le contexte de découverte des

skylines.

La principale différence entre les objets skylines et les itemsets graduels repose sur le fait que dans un cas la mesure de fréquence est primordiale alors que dans l'autre seules les racines comptent. Cependant, des objets qui ne sont pas racines à un niveau $n$ peuvent le devenir à un niveau $n+1$. *Est-il possible de détecter à l'avance quels objets vont remonter dans l'arbre ?* Pour résoudre cette question, nous envisageons deux pistes :

- L'utilisation du **principe d'inclusion-exclusion** peut permettre de détecter à l'avance si un objet appartenant à $s_i$ et/ou à $s_j$ se trouvera dans le diagramme associé à l'itemset graduel $s_i.s_j$. Il s'agira alors de définir comment déterminer l'index de cet objet, ou du moins de borner son index potentiel.
- L'utilisation de **règles de dérivations** peut permettre d'inférer directement sur les matrices de niveau $n$ à partir des matrices de niveau $n-1$. Il s'agira alors de définir et démontrer plusieurs propriétés provenant de l'ordre des objets.

Enfin, les diverses propositions énumérées dans les travaux existants montrent qu'il existe d'autres structures que les matrices binaires permettant de conserver efficacement les ordres. On peut par exemple citer les B+-Tree ou encore les R-Tree. Il s'agit alors d'étudier la mise en œuvre du remplacement de notre structure binaire, ainsi que les éventuels gains ou pertes de mémoire et leur impact sur la complexité de nos propositions.

### 6.1.3 Les skylines et la gradualité, qui résout quoi ?

Nous avons démontré que la recherche de skylines et la recherche d'itemsets graduels étaient liés, et que nos algorithmes d'extraction d'itemsets graduels permettaient de résoudre la problématique d'extraction d'objets skylines. *Les approches proposées pour l'extraction de skylines sont-elles adaptables à l'extraction de motifs graduel ?* Cela paraît difficile. En effet, les approches sus-citées élaguent les ensembles non solutions. Ainsi, il n'est pas possible pour ces approches de déterminer le ou les plus longues sous-chaines d'objets ordonnés.

Les stratégies du type D& C recherchent les solutions optimales des sous-espaces de recherche, puis les fusionnent. Dans le cas de la gradualité, il est difficile de fusionner les ordres découverts dans ces sous-espaces. C'est pourquoi ce type de stratégie n'est pas envisageable dans le contexte graduel.

Les stratégies du type BNL sont semblables à celle que nous proposons. Cependant, elles ne sont conçues que pour répondre à des requêtes skylines, et non pour générer le SKYCUBE. En cela, elles ne sont pas adaptées à notre contexte.

Les approches *orion*, *orion-clos* et *orion-tail* n'énumèrent pas les ordres possibles à chaque niveau, et c'est d'ailleurs ce qui les rend si efficaces. Cependant, comment inférer à partir du niveau $n$ les ordres du niveau $n+1$ sans avoir au préalable construit une liste de ces ordres ? Il n'est donc pas possible d'adapter une telle approche. [KLRK10] utilise également une structure binaire, qui permet de conserver les valeurs des objets pour chaque item. Cette structure est pensée pour vérifier à quel skyline appartient chaque objet. Une telle stratégie ne fonctionne pas pour les itemsets graduels : on peut en effet déterminer si un objet appartient à l'ensemble solution d'un itemset graduel, mais il faut considérer *l'ensemble des objets solutions* pour déterminer la fréquence associée à cet itemset.

Tous ces arguments ainsi que les propositions présentées dans la section 6.1.2 nous amènent à penser que la recherche de motifs graduels inclut la découverte de skylines. Partant de ce constat, on peut proposer de nouveaux types de skylines :

- **Les skycubes avec dominations inversées :** toutes les propositions sur les skycubes ne considèrent qu'un sens d'ordonnancement. Les objets dominant d'autres objets sont ceux ayant la valeur minimale. Cependant, on peut considérer les objets minimisés sur une dimension, et maximisés sur d'autre. Il ne s'agit pas ici d'ordonner au préalable dans un ordre ascendant ou descendant, mais de considérer que $(A^{\geq}B^{\geq}C^{\geq})$ doit appartenir au skycube au même titre que $(A^{\geq}B^{\leq}C^{\geq})$ ou encore $(A^{\geq}B^{\leq}C^{\leq})$. La principale difficulté réside dans le fait que l'ensemble des requêtes skyline est doublé (cela aboutit à la génération de deux fois plus d'ensembles skylines) ;
- **Les requêtes skylines avec mesure de fréquence associée :** on peut se demander la valeur sémantique que véhicule un ensemble skyline qui contient presque l'intégralité des objets de la base. Par exemple, considérons les skylines associés à l'itemset graduel $(A^{\geq}B^{\geq}C^{\geq}D^{\geq})$ sur la figure 6.4. Dans ce cas, quatre objets sur cinq sont des skylines, soit 80% de la base de données. Si l'on considère la sémantique associée à ces objets, il sera préférable pour l'utilisateur de rechercher les objets skylines qui dominent *au moins* $k$ objets de la base.

Nous avons étudié dans cette section les liens entre itemsets graduels et ensembles skylines. Ces types de motifs s'appliquent sur des bases non séquentielles. Dans la section suivante, nous nous intéressons aux bases séquentielles, et analysons les différents types de motifs graduels qu'il est possible d'extraire.

## 6.2 Les motifs séquentiels graduels

Nous introduisons d'abord brièvement les travaux relatifs à l'extraction de motifs séquentiels, puis nous décrivons deux types de gradualité qui peuvent être utilisés conjointement pour de tels motifs.

Les motifs séquentiels sont utiles dès lors qu'il existe une notion d'ordre dans la base, comme par exemple le temps. Il s'agit alors non plus d'extraire des itemsets, mais des séquences d'itemsets ordonnés. Par exemple, "*le gène $g_1$ et le gène $g_2$ s'expriment et plus tard le gène $g_3$ se sous-exprime et le gène $g_4$ s'exprime*" est un motif séquentiel. Notons que l'on ne sait pas combien de temps s'est écoulé entre les réactions des gènes $g_1$ et $g_2$ d'une part, et celles des gènes $g_3$ et $g_4$ d'autre part.

Depuis leur introduction par [AS95], les motifs séquentiels ont été très étudiés par la communauté fouille de données. En effet, l'introduction de l'ordre entre les itemsets ouvre l'espace de recherche, et complexifie le processus de découverte. Ainsi, le succès de l'extraction réside dans la manière de parcourir efficacement l'espace de recherche. Les mêmes paradigmes que pour les itemsets graduels ont émergé : *générer-élaguer* [AS95, MCP98], et *pattern-growth* [PHMa+04]. Dans le premier cas, il s'agit de parcourir l'espace de recherche niveau par niveau, en augmentant la taille des séquences à chaque passe, alors que dans le second, il s'agit de parcourir l'espace de recherche en profondeur d'abord. Différentes variantes très efficaces basées sur la notion de classes d'équivalence ont également vu le jour [Zak01, RCP08].

Ces différents algorithmes fonctionnent sur des bases binaires : un gène s'est ou ne s'est pas exprimé. Avec l'augmentation des capacités de stockage, les bases de données numériques sont devenues une réalité. Les algorithmes classiques d'extraction ont donc été adaptés afin de gérer les données quantitatives.

Parmi les solutions proposées, on trouve les approches basées sur les intervalles [KLNS07], ou encore les approches utilisant la logique floue [FLT07].

Plus récemment, [FMLT08a, FMLT08b] proposent d'extraire des motifs séquentiels flous graduels. Ces motifs prennent en compte le temps écoulé entre chaque itemset (plus ou moins de temps s'est écoulé entre telle et telle série d'événements) ainsi que la gradualité sur les itemsets. Par exemple, les auteurs proposent des motifs comme "*une augmentation lente du nombre de connexions à la page KBM précède une longue période d'augmentation de connexions à KOML, après une courte période. Puis vient une augmentation lente du nombre de connexions à DJAVA*.".

Dans ce chapitre, nous définissons deux types de motifs séquentiels graduels : les motifs inter, et les motifs intra. Nous illustrons nos définitions à l'aide de la base de données de la table 6.5. Elle décrit les valeurs prises par trois différents items sur différentes dates.

| Objet | Date | A | B | C |
|---|---|---|---|---|
| $o_1$ | $d_1$ | 200 | 15 | 0 |
| | $d_2$ | 250 | 15 | 1 |
| | $d_3$ | 100 | 10 | 1 |
| | $d_4$ | 150 | 5 | 0 |
| $o_2$ | $d_1$ | 250 | 30 | 2 |
| | $d_2$ | 200 | 25 | 1 |
| | $d_3$ | 300 | 20 | 1 |
| | $d_4$ | 300 | 15 | 0 |
| $o_3$ | $d_1$ | 300 | 60 | 4 |
| | $d_2$ | 350 | 30 | 5 |
| $o_4$ | $d_1$ | 300 | 150 | 8 |
| | $d_2$ | 300 | 25 | 3 |

Tab 6.5 – Base de données séquentielle exemple

La gradualité peut être placée dans les motifs séquentiels à deux niveaux :

1. en considérant la *variation temporelle au sein d'un même objet*, on parlera de **motifs séquentiels intra**

2. en considérant la *variation de valeurs d'un objet à un autre objet*, on parlera de **motifs séquentiels inter**

### 6.2.1 Les motifs séquentiels graduels intra

Les motifs séquentiels graduels intra concernent les variations temporelles d'un même objet : étant donné un objet et un item, nous pouvons suivre la variation de la valeur d'une date à une autre. Cela permet de mesurer l'évolution au cours du temps. Par exemple, considérons l'item B pour l'objet $o_3$ : sa valeur décroît au cours du temps, puisqu'elle commence à 60 pour $d_1$ et finit à 30 pour $d_2$. Cela correspond à l'itemset graduel $(B^{\leq})$.

Dans ce cas, l'ordre classique entre les itemsets dans ce motif séquentiel ne change pas et n'intègre pas de notion graduelle. Au contraire, la notion graduelle est incluse directement au sein des objets, ce qui nous permet de ne pas noter de tels motifs sous la forme de listes d'itemsets graduels, mais sous la forme d'ensemble d'items graduels. Par exemple, $(A^{\geq}B^{\leq})$ signifie que sur au moins deux dates $d_i$ et $d_{i+1}$ la valeur de A a augmenté, et qu'à ces mêmes dates la valeur de l'objet pour B a diminué. Nous définissons ce type de motifs de la manière suivante :

**Définition 43.** *(Motifs séquentiels graduel intra) Soit $I^* = \{i_1^*, ..., i_m^*\}$ un itemset graduel. Soit $o$ un objet de la base. On dit que $o$ supporte $I^*$ s'il existe $\{d_k, k \geq 2\}$ inclus dans l'ensemble des dates associées à $o$, tel que les dates $\{d_k\}$ respecte $I^*$, sous la contrainte chronologique suivante : $\forall k, l, d_k \leq d_l \Rightarrow d_k(i_j) * d_l(i_j)$ pour tout $j = 1, ..., m$.*
*Sous cette contrainte, $I^*$ est appelé motif séquentiel graduel intra, en raison de la gradualité sur le temps des itemsets à l'intérieur de chaque objet.*

| Objet | Motif inter | Listes de dates |
|---|---|---|
| $o_1$ | $(A^{\geq}B^{\geq}C^{\geq})$ | $(d_1d_2)$ |
| | $(A^{\geq}B^{\leq}C^{\geq})$ | $(d_1d_2)$ |
| | $(A^{\leq}B^{\leq}C^{\geq})$ | $(d_1d_3)(d_1d_4)(d_2d_3)$ |
| | $(A^{\leq}B^{\leq}C^{\leq})$ | $(d_2d_3d_4)$ |
| | $(A^{\geq}B^{\leq}C^{\leq})$ | $(d_3d_4)$ |
| $o_2$ | $(A^{\geq}B^{\leq}C^{\leq})$ | $(d_1d_3d_4)(d_2d_3d_4)$ |
| | $(A^{\leq}B^{\leq}C^{\leq})$ | $(d_1d_2)$ |
| $o_3$ | $(A^{\geq}B^{\leq}C^{\geq})$ | $(d_1d_2)$ |
| $o_4$ | $(A^{\geq}B^{\leq}C^{\leq})$ | $(d_1d_2)$ |
| | $(A^{\leq}B^{\leq}C^{\leq})$ | $(d_1d_2)$ |

Tab 6.6 – Liste de dates ordonnées par client et par motif graduel intra

La fréquence d'un motif séquentiel graduel intra sur l'ensemble de la base de données est définie de la manière suivante :

**Définition 44.** *(Fréquence d'un motif séquentiel intra) Le cardinal de l'ensemble des objets qui supportent $I^*$ est la fréquence de $I^*$ notée $Freq(I^*)$ ; on cherche tous les $I^*$ inclus dans l'ensemble des items graduels tels que $Freq(I^*) \geq minFreq$, avec $minFreq$ un seuil de fréquence minimal fixé par l'utilisateur.*

Par exemple, le tableau 6.6 montre les motifs graduels intra que l'on peut extraire de la base objets par objets. A chaque motif graduel, nous avons associé dans la dernière colonne les listes des dates les validant. Ainsi, à partir de la base 6.5, nous pouvons extraire les motifs graduels intra associés aux fréquences suivantes :

- $S_1 = (A^{\geq}B^{\geq}C^{\geq})$, $Freq(S_1) = \frac{1}{4} = 0.25$
- $S_2 = (A^{\leq}B^{\leq}C^{\leq})$, $Freq(S_2) = \frac{3}{4} = 0.75$
- $S_3 = (A^{\geq}B^{\leq}C^{\leq})$, $Freq(S_3) = \frac{3}{4} = 0.75$
- $S_4 = (A^{\geq}B^{\leq}C^{\geq})$, $Freq(S_4) = \frac{2}{4} = 0.5$

Notons qu'il est également possible de donner des poids aux objets supportant les motifs graduels intra, en pondérant le nombre de transactions réalisant la variation par le nombre de transactions de l'objet. Par exemple, pour $S_3$ nous obtenons $Freq(S_3) = (\frac{2}{4} + \frac{3}{4} + 0 + 1) \times \frac{1}{4} = 0.56$, car :

- deux transactions sur quatre supportent $S_3$ pour $o_1$,
- trois transactions sur quatre supportent $S_3$ pour $o_2$,
- aucune transaction ne supporte $S_3$ pour $o_3$
- deux transactions sur deux supportent $S_3$ pour $o_4$

Avec ce comptage, nous obtenons les fréquences suivantes :

- $Freq(S_1) = (\frac{2}{4} + 0 + 0 + 0) \times \frac{1}{4} = 0.125$
- $Freq(S_2) = (\frac{3}{4} + \frac{2}{4} + 0 + 1) \times \frac{1}{4} = 0.56$
- $Freq(S_3) = (\frac{2}{4} + \frac{3}{4} + 0 + 1) \times \frac{1}{4} = 0.56$
- $Freq(S_4) = (\frac{2}{4} + 0 + 1 + 0) \times \frac{1}{4} = 0.375$

Pour extraire de tels motifs, un algorithme naïf consiste à lire les transactions objet par objet, et déterminer la liste des motifs graduels intra que cet objet contient. On peut imaginer placer ces motifs dans une structure d'arbre préfixé au fur et à mesure de leur découverte : si un motif est déjà connu, on augmente son support, sinon on crée un nouveau chemin dans l'arbre. Un tel algorithme s'avère peu efficace en temps, puisqu'il va générer tous les motifs supportés par au moins un objet de la base, et ne pourra pas utiliser des techniques d'élagage basées sur la fréquence.

### 6.2.2 Les motifs séquentiels graduels inter

Dans ce cas, nous ne considérons pas la gradualité d'une date à une autre, mais d'un objet à l'autre. Cela signifie que la valeur d'un item est comparée entre deux objets, comme nous le considérons dans les chapitres précédents. Ce type de motifs séquentiels graduels permet de mettre en évidence une variation croissante sur toute la base. Par exemple, le motif séquentiel $(A^{\geq})(C^{\geq})$ signifie "*une augmentation de A est suivie par une augmentation de C*".

Nous posons comme hypothèse préalable que les dates associées à chaque objet sont les mêmes.

**Définition 45.** *(itemset graduel inter) Soit $I^* = \{i_1^*, ..., i_m^*\}$ un itemset graduel. On dit qu'un ensemble d'objets $\mathcal{O}' \subset \mathcal{O}$ supporte $I^*$ s'il existe une date $d$ telle que $\mathcal{O}'$ respecte $I^*$ à la date $d$. La fréquence de $I^*$ est alors la cardinalité du plus grand ensemble $\mathcal{O}'$ qui le supporte.*

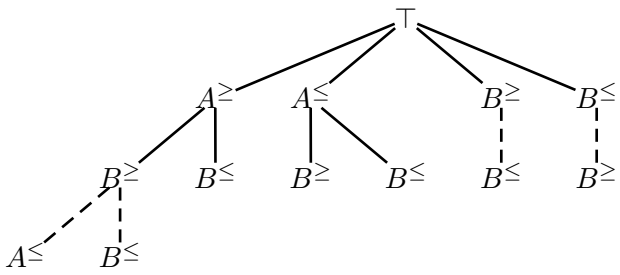



Fig 6.5 – Une partie de l'arbre des préfixes construit à partir de la base de la table 6.5

**Définition 46.** *(motif séquentiel graduel inter) Soit $S^* = \langle I_1^*, ..., I_k^* \rangle$ une liste ordonnée d'itemsets graduels de longueur $l$. $S^*$ est appelé motif séquentiel graduel inter.*

**Définition 47.** *(Fréquence d'un motif séquentiel graduel inter) On dit que $\mathcal{O}'$ supporte $S^*$ s'il existe $l$ dates successives $d_1, ..., d_l$ telles que $\mathcal{O}'$ respecte $I_1^*$ à la date $d_1$, ..., $\mathcal{O}'$ respecte $I_l^*$ à la date $d_l$. La fréquence de $S^*$ est alors le cardinal du plus grand ensemble $\mathcal{O}'$ qui le supporte.*

Par exemple, $S = \langle (A^{\geq})(C^{\leq}) \rangle$ est un motif séquentiel graduel inter. Pour $o_1$ et $o_2$, nous observons que A augmente de $d_1$ à $d_2$ et que C diminue de $d_2$ à $d_3$. La découverte de tels motifs est une tâche bien plus complexe que les problématiques exposées précédemment dans ce manuscrit. En effet, il s'agit ici de considérer d'une part l'ordre des itemsets, et d'autre part l'ordre temporel. Voici quelques motifs séquentiels graduel inter que l'on peut extraire à partir de la base 6.5 :

- $S_1 = \langle (A^{\geq}B^{\leq}) \rangle$, $Freq(S_1) = 1$, car $A[o_1, d_1] \leq A[o_2, d_1] \leq A[o_3, d_1] \leq A[o_4, d_1]$ et $B[o_1, d_1] \geq B[o_2, d_1] \geq B[o_3, d_1] \geq B[o_4, d_1]$
- $S_1 = \langle (A^{\geq}B^{\geq})(C^{\leq}) \rangle$, $Freq(S_1) = 0.5$, car $A[o_1, d_1] \leq A[o_2, d_1]$, $B[o_1, d_1] \leq B[o_2, d_1]$ et $C[o_1, d_3] \geq C[o_2, d_3]$

L'ordre temporel fait perdre la propriété de complétude. Pour s'en convaincre, il suffit de constater que $Freq(\langle (A^{\geq}B^{\geq})(C^{\leq}) \rangle) \neq Freq(\langle (A^{\leq}B^{\leq})(C^{\geq}) \rangle)$. De ce fait, l'espace de recherche est démultiplié. Par exemple, la figure 6.5 montre une petite partie de l'arbre PST qui peut être construit en utilisant les items de la table 6.5. Dans un arbre de préfixes, chaque nœud représente un item, et il existe deux types d'arcs : ceux qui correspondent à un lien dans un itemset (dessiné en traits pleins) et ceux correspondant au début d'un nouvel itemset (en pointillés). Le chemin de la racine à une feuille représente un motif séquentiel graduel.

### Motifs séquentiels graduels et motifs d'évolution

Ces propositions peuvent être comparées aux *motifs d'évolution* [FMLT08a, FMLT08b], puisque leur définition est assez similaire à celle des motifs séquentiels graduels au cours du temps. Cependant, la solution proposée par Fiot & Al. diffère de la nôtre : les auteurs transforment un degré d'appartenance en une *base de données de tendance* en soustrayant les valeurs d'une date à une autre. L'extraction est ensuite directement appliquée sur cette nouvelle base. Dans [FMLT08a, FMLT08b], les auteurs sont plus intéressés par mesurer la "force" de la gradualité. En effet, ils mesurent si l'évolution (au cours de

la période de la première date à la dernière) est “lente”, “rapide”... Cela les mène à une solution basée sur des méthodes différentes, comme les fréquences floues et les contraintes de temps.

## 6.3 Discussion

Dans ce chapitre, nous avons expliqué pourquoi l'ensemble des solutions retournées par les itemsets graduels associés aux opérateurs $\{\leq, \geq\}$ inclue l'ensemble skyline associé au même itemset. Ce travail ouvre la voie à de nombreuses perspectives comme celles citées dans la section 6.1.3.

Nous avons également présenté les différentes manières de combiner la gradualité et les motifs séquentiels. L'ordre entre les itemsets introduit par les motifs séquentiels rend très difficile l'extraction de motifs séquentiels graduels, qu'ils soient intra ou inter. La problématique d'extraction de motifs séquentiels graduels, et plus particulièrement la recherche de motifs inter, reste ouverte.

# Chapitre 7

# Bilan, perspectives et conclusion

En 1956, le premier disque dur fabriqué par IBM pesait plus d'une tonne pour une capacité de 5Mo. La course à la miniaturisation tout en élevant les capacités de ce matériel permet d'atteindre actuellement 2To. En 50 ans, la capacité des disques durs a été multipliée par un facteur de 1 000 000 tandis que le prix d'un mega octet a été divisé par 1,3 million [1]. Ces capacités de stockage sont désormais accessibles à tous et se sont immiscées dans absolument tous les domaines. Le monde médical n'a pas échappé à cette évolution : par exemple, la base de données PaQuid [2] recense depuis 1987 une cohorte de 3777 sujets afin de mieux comprendre la maladie d'Alzheimer et les syndromes qui lui sont apparentés. De même, des données issues de tests psychologiques peuvent être numérisées et stockées.

En 1977, Frédérick Sanger propose une méthode enzymatique dans le but de séquencer les acides nucléiques : l'ancêtre de la puce à ADN est né. De nos jours, une puce permet de séquencer simultanément plusieurs milliers de séquences d'ADN. Les puces à ADN constituent l'une des plus grandes avancées dans le domaine de la génomique et leurs applications sont multiples : le diagnostic clinique, l'identification et l'expertise médico-légale, sans oublier la compréhension et le traitement de diverses maladies. L'investissement massif dans cette nouvelle technologie ainsi que la démultiplication des expériences ont engendré une très grande masse d'informations. Si les outils de stockage permettent désormais d'emmagasiner toutes ces données, les exploiter à l'aide de techniques d'analyses pertinentes reste un problème ouvert.

Ces nouvelles données possèdent des caractéristiques bien particulières décrites tout au long de ce mémoire, qui constituent un véritable défi pour la fouille de données. Dans ce mémoire, nous nous sommes intéressés à l'extraction de motifs graduels de la forme "plus le gène $g_1$ se sur-exprime et le gène $g_2$ se sous-exprime, alors plus le gène $g_3$ se sur-exprime". Notre travail se focalise sur l'aspect théorique d'une part, à travers la définition de plusieurs formes de motifs graduels ainsi que l'élaboration de divers algorithmes efficaces, et sur l'aspect expérimental d'autre part. Nous résumons dans la section 7.1 les diverses contributions de ce mémoire, avant de dresser un bilan de leur mise en œuvre. Dans la section 7.2, nous décrivons les perspectives soulevées par ce travail.

1. Sources : http ://www.clubic.com/article-39236-1-50-ans-disque-dur-petit-historique.html
2. http ://cm2r.enamax.net/onra/index.php ?option=com_content&task=view&id=65&Itemid=66

## 7.1 Synthèse du travail effectué

Le panorama des travaux existants dressé au chapitre 2 montre qu'il existe diverses approches de fouille de données appliquées au domaine médical. Dans ce chapitre, nous nous sommes focalisés sur les algorithmes d'extraction de motifs, puisque nous n'avons pas traité de techniques de classification ou de clustering dans le reste du mémoire. Les approches présentées sont majoritairement pensées pour une application aux bases génomiques : expressions de gènes, séquences d'ADN et séquences de protéines. Nous avons mis en évidence les limitations principales de ces approches : aucune ne prend en compte de manière satisfaisante l'aspect quantitatif des expressions de gènes. De plus, ces approches ne sont pour la plupart pas génériques. Nous nous sommes donc concentrés sur la question suivante : *comment extraire des informations potentiellement utiles et prenant en compte l'aspect quantitatif de la plupart des bases médicales ?* Pour y répondre, nous avons développé les techniques décrites dans cette section.

### 7.1.1 Extraction de règles graduelles relationnelles

Le chapitre 3 porte sur la définition formelle ainsi que l'extraction des itemsets graduels. Les travaux fédérateurs en fouille de données de ce type d'itemsets sont relativement récents [Hül02, BCS+07], et se situent dans le contexte de la logique floue. Les mesures de représentativité de ce type d'itemsets sont basées soit sur des régressions linéaires, soit sur le dénombrement des couples d'objets dont la valeur évolue sous-ensembles flous par sous-ensembles flous.

De notre point de vue, si la logique floue offre un cadre sémantique particulièrement appréciable pour les utilisateurs, nous pensons que la gradualité peut être définie en dehors du contexte flou. Nous avons posé les définitions pouvant servir de base aux travaux sur les motifs graduels, en considérant que la notion de gradualité devient effective si pour un item donné, les valeurs varient dans le même sens d'une transaction à une autre. Notre objectif était double : d'un côté permettre la corrélation d'items qui varient dans des sens différents et d'autre part conserver une mesure de fréquence reflétant la proportion des objets de la base variant dans un même sens.

L'étape suivante a été de proposer des méthodes efficaces. En effet, si de nombreux algorithmes, basés sur les corrélations binaires, fonctionnent bien pour l'extraction d'items classique, la nécessité de prendre en compte les corrélations de variations les rend inefficaces dans le cas d'items graduels. Dans un premier temps, nous avons pensé à une heuristique qui maintient une liste du plus grand ensemble d'objets variants. Le résultat des diverses expérimentations montre qu'elle permet d'extraire des itemsets plus longs que l'approche complète. En cela, notre heuristique permet d'extraire des itemsets potentiellement intéressants pour les experts, puisqu'ils mettent en évidence les co-variations de nombreux items. De plus, sa définition permet l'intégration de la recherche d'autre types de connaissances, tels que les outliers.

Cependant, cette méthode est basée sur l'ordre lexicographique des items et peut être amenée à faire des choix au niveau $k$ qui auront pour conséquence l'élimination d'itemsets fréquents au niveau $k+n$. Nous nous sommes donc intéressés à l'élaboration d'une méthode complète, basée sur une structure de données adéquate. Les performances en terme de temps et d'exploitation mémoire sont meilleures que celles de l'heuristique. Toutefois, comme elle découvre tous les itemsets graduels là où l'heuristique en

écarte, cette dernière méthode s'effondre avant l'heuristique sur nos plus grandes bases.

Afin de tester ces deux méthodes de manière efficace sur des jeux de données synthétiques denses, nous avons élaboré un générateur simple, mais qui permet de faire varier les items dans un seul sens pour la majorité des objets. Cela nous permet d'extraire des itemsets n'ayant pas ou peu d'itemsets contradictoires. Dans la littérature, nous trouvons peu d'informations sur les bases de données des expérimentations concernant les approches [Hül02, BCS+07]. A titre d'indication, dans [KH10], la base de données UCI concernant la qualité des vin[3] est utilisée. Cette base contient 4898 transactions décrivant 11 items. Nos expérimentations montent jusqu'à 5000 transactions pour 100 items, soit dix fois plus d'items. En ce sens, nous pensons que notre algorithme d'extraction d'itemsets graduels figure parmi les plus performants.

Une perspective intéressante consiste à comparer ces différentes approches. Cela nécessitera la mise en place d'un protocole adapté ne pénalisant pas le "côté flou" des approches antérieures.

### 7.1.2 Étude de la sémantique de la fréquence

Les diverses expérimentations menées tant avec l'approche heuristique qu'avec l'approche complète sur jeux de données réelles (et plus particulièrement sur les bases d'expression de gènes) montrent qu'il est nécessaire d'étudier la sémantique reflétée par la mesure de fréquence. En effet, dans le contexte graduel, les opérateurs de comparaison choisis ont un impact important sur la qualité et le nombre d'itemsets extraits. De plus, la sémantique de la fréquence graduelle n'est pas facile à appréhender au premier abord.

Partant de ces constats, nous avons proposé diverses mesures de fréquences alternatives. L'un des plus grands enjeux était la prise en compte des valeurs égales. Cependant, nous avons constaté que l'utilisation d'opérateurs de comparaison stricts rendait notre approche très sensible au seuil de fréquence minimal : soit nous n'obtenions pas ou très peu d'itemsets graduels, soit nous en obtenions beaucoup trop et avec une trop faible fréquence pour être intéressants.

La fréquence graduelle précédemment proposée ne prenait pas en compte le nombre total d'objets participant à la gradualité. En effet, il est possible que la variation maximale soit réalisée par plusieurs chemins, donc par différents objets indépendants. Nous avons donc proposé une mesure permettant de pondérer la fréquence graduelle classique par le nombre d'objets réalisant la gradualité. Nous avons étudié les propriétés de cette mesure de fréquence, et montré comment l'intégrer à l'algorithme précédemment proposé.

De plus, nous avons proposé une méthode permettant d'identifier les objets ayant une variation anormalement forte. Cette proposition s'inscrit dans la philosophie de l'étude de la sémantique de la variation, puisque les itemsets contenant de tels objets sont restitués. Nous pensons que dans l'avenir, la mise en évidence de comportements atypiques dans un contexte médical permettra la découverte de connaissances intéressantes.

3. http ://archive.ics.uci.edu/ml/datasets/Wine+Quality

Enfin, nous avons implémenté un outil de visualisation adapté à un non-informaticien. Actuellement, nos résultats sont en cours d'analyse et leur validation constitue une perspective à part entière.

### 7.1.3 Extraction de règles graduelles multidimensionnelles

Nous nous sommes intéressés à l'extraction de la gradualité dans un contexte non relationnel. Notre choix s'est porté sur les cubes de données, dont la structure semble parfaitement adaptée au milieu médical. En effet, les cubes de données offrent la possibilité d'agréger les données et de naviguer à différent niveaux de granularité. De plus, des hiérarchies peuvent être définies sur les membres des dimensions. Une telle vision des données s'avère adaptée lorsque l'on sait que les connaissances médicales sont fréquemment organisées sous forme de hiérarchies ou de thésaurus. On pourra citer par exemple Gene Ontology [4] qui décrit les gènes selon leurs fonctions moléculaires, leurs rôles dans les processus biologiques ou encore leurs localisations dans les composants cellulaires. Dans le cadre de l'Alzheimer, on pourra utiliser MESH [5], qui fournit un vocabulaire contrôlé très complet des termes médicaux.

Dans ce contexte, notre objectif était de permettre l'extraction de connaissances graduelles à partir de bases multidimensionnelles. Pour ce faire, nous nous sommes appuyés sur les blocs de données multidimensionnels [CLL08], qui donnent une représentation résumée des cubes de données. Dans un premier temps nous avons proposé un nouvel algorithme d'extraction de blocs utilisant des classes d'équivalences basées sur la cardinalité des intervalles. Les précédentes propositions étaient basées sur des heuristiques, et ne permettaient pas d'extraire l'ensemble complet des blocs de données respectant les seuils de mesure et de fréquence minimales. En revanche, notre méthode est complète. Notre objectif était de pouvoir paramétrer efficacement l'extraction de blocs, voire d'être capable de ne considérer qu'une seule cellule dans le seuil de fréquence minimal.

Les blocs de données se sont avérés être un cadre idéal à la définition des itemsets graduels multidimensionnels. Ainsi, nous avons mis en place une variante de l'algorithme s'appliquant aux bases relationnelles. Nos expérimentations sur jeux de données réelles montre l'efficacité de cette approche.

L'une des perspectives directes pour ce travail est la génération d'un cube de données à partir de la base génomique. Nous expérimenterons ainsi notre méthode sur un jeu de données réaliste. D'autre part, nous projetons de "fusionner" ces deux techniques afin d'extraire des itemsets graduels multidimensionnel au fur et à mesure de la découverte des blocs.

### 7.1.4 Gradualité, skylines et motifs séquentiels

Notre dernière contribution porte sur la mise en relation des itemsets graduels avec les skylines d'une part et les motifs séquentiels d'autre part. L'objectif ici était de déplacer la gradualité vers d'autres types de connaissances.

Les skylines mettent en évidence les objets de la base de données qui ne sont *dominés* par aucun autre objet. La notion de domination est définie par un opérateur $\min$, $\max$ ou diff sur le domaine des

4. http ://www.geneontology.org/
5. http ://www.ncbi.nlm.nih.gov/mesh

items. Dans ce contexte, un ensemble skyline en fonction d'un itemset montre les meilleurs compromis. La recherche sur les skylines a été un domaine très actif ces dernières années, car les applications sont reconnues dans les processus de prise de décision.

Nous avons démontré que les itemsets graduels permettent d'extraire directement les skylines. Les expérimentations présentes dans la litérature nous poussent à penser que l'adaptation de l'extraction automatique des itemsets graduels à la recherche de skylines permettrait d'obtenir de bons résultats. En effet, les bases considérées dépassent rarement 10 items. Notre algorithme est efficace sur des bases contenant un plus grand nombre d'items. De plus, la définition des règles graduelles que nous avons fournie permet de répondre à toutes les requêtes dérivées des skylines : les top-$k$ skylines, le $K$-skyband ou encore les requêtes skylines contraintes. Nous allons plus loin et montrons que les itemsets graduels peuvent être utilisés afin de trouver les skylines ayant des relations de dominance inverse sur les différents items. De plus, on peut imaginer conserver la fréquence afin de proposer des skylines dominant au moins $n$ objets, ce qui appuierait dans de nombreux cas leur sémantique.

Il est important de noter que nous pensons que la méthode décrite sera toujours moins efficace que [RPK10] dans le cas de jeux de données fortement corrélés. En revanche, nos expérimentations pourraient prouver que dans le cas de jeux anti-corrélés, notre approche se comporte plus efficacement.

La seconde partie de cette contribution concerne les motifs séquentiels. Ces motifs sont des séquences ordonnées d'itemsets, et posent un défi à la fouille dans le sens où l'espace de recherche est potentiellement infini. La mise en œuvre de la gradualité séquentielle est donc une tâche difficile, puisqu'il s'agit non seulement de conserver l'ordre sur les valeurs, mais également l'ordre sur les items. Nous avons défini deux types de gradualité séquentielle : l'une portant sur la gradualité temporelle et l'autre portant sur la gradualité de valeurs.

L'une des perspectives de ces contributions est leur mise en œuvre sur des bases de données synthétiques dans un premier temps afin de mesurer leurs performances, puis sur des jeux de données réels dans un second temps.

## 7.2 Perspectives

Les perspectives ouvertes par ces différentes contributions sont nombreuses. Nous développons dans cette section les principales.

### 7.2.1 Gradualité et statistique

Nos expérimentations sur jeu de données réelles montrent la nécessité d'un prétraitement judicieux des données. Pour cela, nous pensons nous orienter vers diverses méthodes statistiques.

En premier lieu, il serait intéressant d'étudier les corrélations de variables par exemple en utilisant la corrélation de Pearson [ESBB98, Sal10]. Deux méthodes se présentent alors : soit nous utilisons cette corrélation lors d'un prétraitement des données, ce qui permet la mise en évidence de relations, soit elle est utilisée comme contrainte lors du processus de fouille.

Une autre piste consiste à se tourner vers les méthodes de réduction de données telles que les approches de sélection de variables (couramment désignées sous le terme de "*sélection de gènes discriminants*" en biologie) ou les méthodes d'analyse multivariée utilisant des projections telles que l'Analyse en Composante Principale (ACP). En effet, l'ACP met en évidence des combinaisons de différentes variables (dans notre contexte les items) résumant au mieux l'ensemble des variables. Un tel prétraitement résulterait alors en une base plus petite, donc plus facile à fouiller.

### 7.2.2 Condensation et filtrage des itemsets graduels

Dans ce manuscrit, nous avons étudié deux méthodes visant à réduire le nombre d'itemsets extraits. La première consiste à changer la définition du support tandis que la seconde consiste à sélectionner les itemsets contenant des variations atypiques.

L'une des perspectives à court terme est la définition de nouvelles mesures permettant le filtrage des itemsets graduels. Il existe en effet de nombreuses contraintes réduisant le nombre de motifs extraits. On citera par exemple les contraintes sur les domaines des valeurs ou encore sur la longueur des motifs extraits. Nous pensons que l'intégration de ces paramètres donnerait une valeur ajoutée pour les utilisateurs, notamment dans le cadre d'une validation biologique.

Dans le chapitre 4, nous proposons d'extraire des outliers locaux par rapport à un itemset graduel. Il existe différent types d'exceptions et d'inattendus. Il serait donc intéressant d'étudier les outliers au niveau global (transaction outlier), ou mieux, les motifs graduels inattendus. En effet, compte tenu de l'information graduelle, il est possible de définir divers types d'inattendus : itemsets dont le sens de variation d'un item contredit une croyance ou des itemsets de taille plus courte (par exemple $(A^{\geq}B^{\leq}C^{\geq})$ peut être surprenant sachant $(A^{\geq}B^{\geq})$), ou encore itemsets graduels dont la fréquence obtenue est différente de la fréquence attendue. Ces dernières années, de nombreux travaux adoptant ces philosophies ont vu le jour, cependant il n'en existe aucun utilisant les règles graduelles.

Depuis la publication de la méthode complète d'extraction des itemsets graduels dans [DLT09b], de nombreux travaux basés sur ces définitions ont été proposés. Parmi eux, nous notons la recherche d'itemsets graduels clos, dans le cadre de la thèse de Sarah Ayouni, en co-tutelle entre la Faculté des Sciences de Tunis et le Lirmm. Nous pensons que les clos constituent une avancée majeure dans le cadre de l'extraction de motifs graduels à partir de grandes bases. Cependant, l'aspect graduel peut rendre la définition d'algorithmes efficaces complexe. Dans ce contexte, on peut imaginer une piste portant sur la recherche de sous-ordres communs, ou encore le maintien de listes des ordres propres à chaque item.

Dans le contexte graduel, les applications théoriques peuvent être multiples : au delà de l'extraction de motifs clos, on peut imaginer différent types d'heuristiques accélérant le processus de recherche, ou encore des critères d'élagages basés sur de telles listes. L'un des verrous majeurs sera alors la tenue en mémoire de telles listes.

Dans la lignée de ces idées, nous pensons que certains items "fédèrent" ou contraignent de manière plus évidente les ordres des autres itemsets. Par exemple, considérons le diagramme 4.1c du chapitre 4 : dans ce cas, les listes ordonnées de $C^{>}$ régulent celles de $(A^{>}C^{>})$. Détecter ces comportements améliorerait grandement l'efficacité de nos méthodes. Cela est d'autant plus vrai si l'on utilise les

opérateurs de comparaison $\{\leq, \geq\}$ et que la base contient un grand nombre d'égalités, puisqu'il sera possible de réordonner ces valeurs en fonction des valeurs non-égales lors de la conjonction. L'une des pistes de recherches à privilégier afin d'atteindre cet objectif concerne les règles de dérivations.

### 7.2.3 Restitution des motifs graduels

Dans le chapitre 4, nous avons posé les bases d'un outil de visualisation de nos résultats. L'interprétation des résultats est une étape à part entière du processus d'Extraction des Connaissances. Pour ce faire, elle s'appuie sur différents outils, tels que des outils d'évaluation (on parlera de post-traitement) et surtout de visualisation. Cependant, si les travaux de la communauté s'intéressent très fortement aux étapes précédentes (sélection, transformation et fouille de données), il reste des efforts considérables à faire pour la visualisation des résultats. Il existe des plateformes très connues permettant une visualisation graphique des résultats de certains algorithmes comme par exemple WEKA [6].

Toutefois, ces travaux ne proposent pas de placer l'utilisateur *au cœur de la navigation*. Pourtant, une bonne interprétation passe à notre sens par là. C'est à l'utilisateur de choisir quels motifs il veut visualiser, de quelle manière il veut les dérouler, les compresser ou encore les mettre en relation avec d'autres motifs.

Cette perspective est une perspective à long terme, qui se trouve à l'intersection de la fouille de données et de l'interaction homme-machine. On peut imaginer par exemple la mise en œuvre d'un nouveau processus de fouille dynamique sur les résultats de la fouille qui permettrait à l'utilisateur de compresser, décompresser ou remanier en temps réel les résultats obtenus.

### 7.2.4 Gradualité et intégration des connaissances

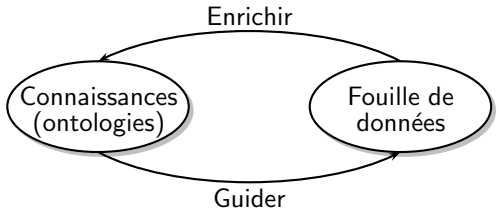



Fig 7.1 – Relation entre la fouille et les connaissances structurées

Comme nous l'avons souligné à la section 7.1.3, il existe de nombreuses structures d'organisation des connaissances approuvées par la communauté médicale, telles que les ontologies, les thésaurus, les vocabulaires contrôlés ou encore les pathways. Ces différentes structures nous renseignent sur la manière dont les concepts médicaux sont placés, ou encore la manière dont les composantes biologiques interagissent. Par exemple, un pathway décrit l'ensemble des réactions biochimiques qui conduisent un substrat de base à un produit final [7].

6. http ://www.cs.waikato.ac.nz/ml/weka/
7. source : http ://fr.wikipedia.org/wiki/Voie_m%C3%A9tabolique

Ces dernières informations peuvent s'avérer extrêmement utiles et leur rôle peut être double. D'une part, les résultats de la fouille peuvent permettre d'enrichir les connaissances existantes [DJBF+08], en soulignant des relations nouvelles. D'autre part, les connaissances et interactions connues peuvent guider le processus de fouille, soit durant la génération des connaissances [MDNST05], soit durant l'élagage. La figure 7.1 résume ce principe.

### 7.2.5 Mise en relation niveau micro et niveau macro

Durant ce travail, nous avons essayé de mettre en évidence les corrélations existantes entre les données de niveau micro et les données de niveau macro. Nous considérons dans ce manuscrit que les données concernant les plus petites unités de l'être vivant, comme les séquences de protéines, les séquences de gènes ou encore les expressions de gènes sont des données micro. Les données macro désignent les données à plus grande échelle, comme les données environnementales, les données sur l'état civil du patient ou encore les données de son suivi médical.

Il n'existe à notre connaissance pas de méthode de fouille de données permettant de lier concrètement ces différents niveaux de données. Dans ce mémoire, nous avons ajouté les données macro numériques aux données micro avant le processus de fouille. Par exemple, pour la base sur le cancer, nous avons conservé des attributs tels que l'âge, le grade de la tumeur, le nombre de récidives etc... Les résultats obtenus montrent que ces différents attributs sont fréquemment corrélés à des variations de gènes.

Cependant, on note que certains de ces attributs, notamment le grade d'une tumeur, conditionnent complètement l'extraction. Cela s'explique par la cardinalité du domaine de cet attribut, qui est de 3. Ainsi, on peut se demander s'il n'y aurait pas une meilleure façon d'intégrer ce type d'attributs. Par exemple, nous pouvons diviser la base en trois sous-bases, une pour chaque valeur de tumeurs, puis regarder les itemsets graduels communs à ces bases, ou alors spécifiques à chacune d'entre elles. Dans ce contexte, on parlera d'*itemsets discriminants* [Sal10].

Une autre solution consiste à se tourner vers les méthodes de classification. Nous avons utilisé les règles graduelles comme règles de classification dans [CDL+09] ([Dâr10] propose également d'utiliser les itemsets graduels de cette manière). La méthode n'a pas encore été testée, c'est une perspective à court terme.

## 7.3 Vers la création d'un outil de fouille complet pour les bases médicales

L'objectif principal de ce travail était la proposition d'algorithmes de fouille de données efficaces et permettant l'extraction de motifs utiles aux différents acteurs du domaine médical. L'ensemble des contributions décrites dans ce mémoire, ainsi que la démonstration de leur mise en œuvre sur des bases réelles montrent que nous avons atteint le premier objectif.

La plupart des algorithmes de fouille se heurtent au problème de la quantité de données extraites. Nos méthodes n'échappent pas à ce phénomène : sur certaines bases, la quantité de motifs graduels extraits est telle qu'il est quasiment impossible pour un expert de les interpréter.

Durant ces trois années de thèse, nous avons travaillé avec différents experts de l'INSERM et du

laboratoire LAMECO. Des échanges réguliers ont permis de mettre en place des protocoles d'expérimentations adaptés. La *validation biologique*, c'est-à-dire la démonstration de la pertinence des méthodes dans un contexte biologique comme dans le contexte des sciences humaines, est un travail long, qui nécessite un recul de plus de trois ans. Les experts ont souligné quelques motifs intéressants, car ils apportent une information déjà connue et donc valide. En ce sens, les motifs graduels décrivent d'une manière cohérente les données. Toutefois, les méthodes proposées ont besoin de la plupart des perspectives citées ci-dessus afin d'être pleinement exploitables.

Ce travail constitue également une avancée dans le sens où nous avons été capables d'appliquer des algorithmes de fouille de données à des bases constituant un défi pour la communauté. Ces méthodes présentent le fort avantage de donner des résultats exacts, ce qui n'introduit aucun biais avec des faux positifs. Enfin, nous avons pensé nos méthodes dans un cadre générique, ce qui signifie qu'elles sont également valides sur tous types de bases (bases de capteur, bases de log web etc). Pour conclure, nous pensons que ces travaux ouvrent la voie à de nombreux autres qui permettront, sur la base des définitions proposées, de continuer à améliorer la qualité des motifs extraits, jusqu'à contribuer, peut-être, à une découverte intéressante pour les experts du monde de la santé.

# Références


[ABLP10] Sarah Ayouni, Seddik Ben Yahia, Anne Laurent et Pascal Poncelet : Mining closed gradual patterns. *In International Conference of Artificial Intelligence and Soft Computing (ICAISC). LNAI.*, 2010.

[AFGY02] Jay Ayres, Jason Flannick, Johannes Gehrke et Tomi Yiu : Sequential pattern mining using a bitmap representation. *In KDD '02 : Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 429–435. ACM, 2002.

[AHL09] Guillaume Artignan, Mountaz Hascoet et Mathieu Lafourcade : Multiscale visual analysis of lexical networks. *Information Visualisation, International Conference on*, 0:685–690, 2009.

[AL99] Yonatan Aumann et Yehuda Lindell : A statistical theory for quantitative association rules. *In Journal of Intelligent Information Systems*, pages 261–270, 1999.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen et A. Inkeri Verkamo : *Fast discovery of association rules*, pages 307–328. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[APS07] Marie Agier, Jean-Marc Petit et Einoshin Suzuki : Unifying framework for rule semantics : Application to gene expression data. *Fundamental Informaticae*, 78(4):543–559, 2007.

[AS94] Rakesh Agrawal et Ramakrishnan Srikant : Fast Algorithms for Mining Association Rules. *In 20th International Conference on Very Large Data Bases, (VLDB'94)*, pages 487–499, 1994.

[AS95] Rakesh Agrawal et Ramakrishnan Srikant : Mining Sequential Patterns. *In the 11th IEEE International Conference on Data Engineering*, pages 3–14, 1995.

[BBS+05] Fernando Berzal, Ignacio Blanco, Daniel Sanchez, José María Serrano et Maria-Amparo Vila : A definition for fuzzy approximate dependencies. *Fuzzy Sets and Systems*, 149(1): 105–129, 2005.

[BCL90] Jon L. Bentley, Kenneth L. Clarkson et David B. Levine : Fast linear expected-time alogorithms for computing maxima and convex hulls. *In SODA '90 : Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 179–187, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.

[BCMG04] Fernando Berzal, Juan-Carlos Cubero, Nicolas Marin et Matias Gamez : Anomalous association rules. *In In IEEE Workshop on Alternative Techniques for Data Mining and Knowledge Discovery (ICDM'04)*, novembre 2004.

[BCS+07] Fernando Berzal, Juan-Carlos Cubero, Daniel Sanchez, Maria-Amparo Vila et José María Serrano : An alternative approach to discover gradual dependencies. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 15(5):559–570, octobre 2007.

[BE94] T. Bailey et C. Elkan : Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *In Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36, 1994.

[BFP07] Pascale Bergeret, Frédéric Flouvat et Jean-Marc Petit : Vers la génération de jeux de données synthétiques réalistes pour les motifs fréquents. *In Bases de Données Avancées (BDA'07)*, octobre 2007.

[BKS01] Stephan Börzsönyi, Donald Kossmann et Konrad Stocker : The skyline operator. *In Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 421–430, Washington, DC, USA, 2001. IEEE Computer Society.

[BL05] Francesco Bonchi et Claudio Lucchese : Pushing tougher constraints in frequent pattern mining. *In In Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference (PAKDD'05)*, pages 114–124, 2005.

[BLP97] Patrick Bosc, Ludovic Lietard et Olivier Pivert : Gradualité, imprécision et dépendances fonctionnelles. *In Bases de Données Avancées (BDA'97)*, 1997.

[BLP98] Patrick Bosc, Ludovic Lietard et Olivier Pivert : Extended functional dependencies as a basis for linguistic summaries. *In Principles of Data Mining and Knowledge Discovery, Second European Symposium (PKDD'98)*, pages 255–263, 1998.

[BMD90] Bernadette Bouchon-Meunier et Sylvie Despres : Acquisition numérique / symbolique de connaissances graduelles. *In Actes des 3emes journées nationales du PRC-GDR Intelligence Artificielle*, pages 127–138, Paris, La Défense, mars 1990.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman et Shalom Tsur : Dynamic itemset counting and implication rules for market basket data. *In Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, mai 1997.

[BPU99] Patrick Bosc, Olivier Pivert et Laurent Ughetto : On data summaries based on gradual rules. *In Proceedings of the 6th International Conference on Computational Intelligence, Theory and Applications*, pages 512–521. Springer-Verlag, 1999.

[BRBR05] Jérémy Besson, Céline Robardet, Jean-François Boulicaut et Sophie Rome : Constraint-based concept mining and its application to microarray data analysis. *Intelligent Data Analysis*, 9(1):59–82, 2005.

[Cal00] Andrea Califano : Splash : structural pattern localization analysis by sequential histograms. *Bioinformatics*, 16(4):341–357, April 2000.

[CBK07] Varun Chandola, Arindam Banerjee et Vipin Kumar : Outlier detection - a survey. Rapport technique, University of Minnesota, Department of Computer Science and Engineering, Minneapolis, août 2007.

[CDL+09] Yeow Wei Choong, Lisa Di Jorio, Anne Laurent, Dominique Laurent et Maguelonne Teisseire : Cbgp : Classication based on gradual patterns. *In In Proceedings of the International Conference on Soft Computing and Pattern Recognition*, 2009.

[CGGL03] Jan Chomicki, Parke Godfrey, Jarek Gryz et Donming Liang : Skyline with presorting. *In International Conference on Data Engineering (ICDE'03)*, volume 0, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[CL09] Jake Y. Chen et Stefano Lonardi : *Biological Data Mining*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, septembre 2009.

[CLL07] Yeow Wei Choong, Anne Laurent et Dominique Laurent : Summarizing data cubes using blocks. *In* P. Poncelet M. Teisseire F. Masseglia, éditeur : *Data Mining Patterns : New Methods and Applications*, page 36. IDEA Group Inc., 2007.

[CLL08] Yeow Wei Choong, Anne Laurent et Dominique Laurent : Mining multiple-level fuzzy blocks from multidimensional data. *Fuzzy Sets and Systems*, 159(12), 2008.

[CLM03] Yeow Wei Choong, Dominique Laurent et Patrick Marcel : Computing appropriate representations for multidimensional data. *Data Knowl. Eng.*, 45(2):181–203, 2003.

[CMB02] Matthieu Capelle, Cyrille Masson et Jean-François Boulicaut : *Mining Frequent Sequential Patterns under a Similarity Constraint*, volume 241, chapitre 10, pages 645–651. Lecture Notes in Computer Science, 2002.

[CMLL04] Yeow Wei Choong, Pierre Maussion, Anne Laurent et Dominique Laurent : Summarizing multidimensional databases using fuzzy rules. *In Proc. of the 10th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'04)*, pages 99–106, 2004.

[Dâr10] Georgiana-Lavinia Dârlea : *Un Système de Classification Supervisée à Base de Règles Implicatives*. Thèse de doctorat, Université de Savoie, Fev 2010.

[DCL93] Rose Dieng, Olivier Corby et Stéphane Lapalut : Acquisition of gradual knowledge. *In Proceedings of the 7th European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, pages 407–426, London, UK, 1993. Springer-Verlag.

[DD05] Didier Devaurs et Fabien De Marchi : Génération de bases de transactions synthétiques : vers la prise en compte des bordures. *In* Véronique Benzaken, éditeur : *21emes journées bases de données avancées (BDA'05)*, pages 119–133, octobre 2005.

[DGS02] Isabela Drummond, Lluís Godo et Sandra Sandri : Restoring consistency in systems of fuzzy gradual rules using similarity relations. *In* G.L. Ramalho G. Bittencourt, éditeur : *Lecture Notes in Artificial Intelligence*, volume 2507, pages 386–396. Springer-Verlag, Springer-Verlag, 2002.

[DHL+04] Guozhu Dong, Jiawei Han, Joyce Lam, Jian Pei, Ke Wang et Wei Zou : W. : Mining constrained gradients in large databases. *IEEE Transactions on Knowledge Discovery and Data Engineering*, 16, 2004.

[Dij71] E. W. Dijkstra : A short introduction to the art of programming. Rapport technique, Technische Hogeschool Eindhoven, Eindhoven, The Netherlands, Août 1971.

[DJBF+08] Lisa Di-Jorio, Sandra Bringay, Céline Fiot, Anne Laurent et Maguelonne Teisseire : Sequential patterns for maintaining ontologies over time. *In OTM '08 : Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems*, pages 1385–1403, Berlin, Heidelberg, 2008. Springer-Verlag.

[DP92] Didier Dubois et Henri Prade : Gradual inference rules in approximate reasoning. *Inf. Sci.*, 61(1-2):103–122, 1992.

[DP08] Didier Dubois et Henri Prade : Gradual elements in a fuzzy set. *Soft Comput.*, 12(2):165–175, 2008.

[DPG95] Didier Dubois, Henri Prade et Michel Grabisch : Gradual rules and the approximation of control laws. *Theoretical aspects of fuzzy control*, pages 147–181, 1995.

[DR04] Boyan N. Dimitrov et Vladimir Rykov : On reliability of hierarchical systems with gradual failures. *Journal of Mathematical Sciences*, 123(1):3802–3815, 2004.

[EP02] Eleazar Eskin et Pavel A. Pevzner : Finding composite regulatory patterns in dna sequences. *In ISMB*, pages 354–363, 2002.

[EPF05] T.P Exarchos, C. Papaloukas et D.I. Fotiadis : A novel methodology for myocardial ischaemia diagnosis using association rules. *European Medical and Biological Engineering Conference (EMBEC)*, 2005.

[ESBB98] M. B. Eisen, P. T. Spellman, P. O. Brown et D. Botstein : Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, December 1998.

[Flo62] Robert W. Floyd : Algorithm 97 : Shortest path. *Commun. ACM*, 5(6):345, 1962.

[FLT07] Céline Fiot, Anne Laurent et Maguelonne Teisseire : From crispness to fuzziness : Three algorithms for soft sequential pattern mining. *In IEEE Transactions on Fuzzy Systems*, volume 15, pages 1263–1277, 2007.

[FMLT08a] Céline Fiot, Florent Masseglia, Anne Laurent et Maguelonne Teisseire : Gradual trends in fuzzy sequential patterns. *In 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems*, 2008.

[FMLT08b] Céline Fiot, Florent Masseglia, Anne Laurent et Maguelonne Teisseire : Ted and eva : Expressing temporal tendencies among quantitative variables using fuzzy sequential patterns. *In 17th IEEE International Conference on Fuzzy Systems (FuzzIEEE'08)*, juin 2008.

[GDP03] Sylvie Galichet, Didier Dubois et Henri Prade : Fuzzy interpolation and level 2 gradual rules. *In Proc. of the 3rd European Society for Fuzzy Logic and Technology Conf. EUSFLAT'03*, pages pp. 506–511, Zittau Allemagne, 09 2003.

[GDP04] Sylvie Galichet, Didier Dubois et Henri Prade : Imprecise specification of ill-known functions using gradual rules. *International Journal of Approximate Reasoning*, 35:205–222, 2004.

[GSG05] Parke Godfrey, Ryan Shipley et Jarek Gryz : Maximal vector computation in large data sets. *In VLDB '05 : Proceedings of the 31st international conference on Very large data bases*, pages 229–240. VLDB Endowment, 2005.

[GSYZ09] Zhenqiang Gong, Guangzhong Sun, Jing Yuan et Yanjing Zhong : Efficient top-$k$ query algorithms using $k$-skyband partition. *In Infoscale*, pages 288–305, 2009.

[Haw80] Douglas Hawkins : *Identification of Outliers*. Chapman and Hall, London, UK, 1980.

[HJ06] Lawrence S Hon et Ajay N Jain : A deterministic motif finding algorithm with application to the human genome. *Bioinformatics*, 22(9):1047–1054, 2006.

[HLSL00] Farhad Hussain, Huan Liu, Einoshin Suzuki et Hongjun Lu : Exception rule mining with a relative interestingness measure. *In In Proceedings of Paci Asia Conference on Knowledge Discovery in DataBases (PAKDD'00)*, pages 86–97, 2000.

[HLW03] Tzung-Pei Hong, Kuei-Ying Lin et Shyue-Liang Wang : Fuzzy data mining for interesting generalized association rules. *Fuzzy Sets Syst.*, 138(2):255–269, 2003.

[HPY00] Jiawei Han, Jian Pei et Yiwen Yin : Mining frequent patterns without candidate generation. *In* Weidong Chen, Jeffrey Naughton et Philip A. Bernstein, éditeurs : *2000 ACM SIGMOD Intl. Conference on Management of Data*, pages 1–12. ACM Press, 2000.

[Hül02] Eyke Hüllermeier : Association rules for expressing gradual dependencies. *In PKDD '02 : Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 200–211, London, UK, 2002. Springer-Verlag.

[Hül07] Eyke Hüllermeier : *Why Fuzzy Set Theory is Useful in Data Mining*, chapitre 1, pages 1–16. Successes and New Directions in Data Mining. Idea Group Publishing, novembre 2007.

[IKA02] Tomasz Imielinski, Leonid Khachiyan et Amin Abdulghani : Cubegrades : Generalizing association rules. *Data Mining and Knowledge Discovery*, 6:219–258, 2002.

[iMAS+01] Jun ichi Miyazaki, Satonari Akutsu, Norihiro Satow, Chinami Hirao et Yao Yao : The gradual expression of troponin t isoforms in chicken wing muscles. *Journal of Muscle Research and Cell Motility*, 22(8):693–701, décembre 2001.

[JDGC07] Hazäel Jones, Didier Dubois, Serge Guillaume et Brigitte Charnomordic : A practical inference method with several implicative gradual rules and a fuzzy input : one and two dimensions. *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, pages 1–6, July 2007.

[JNK06] Nicolas Jay, Amedeo Napoli et François Kohler : Cancer patient flows discovery in drg databases. *In MIE 2006*, Maastricht/Pays-bas, 08 2006.

[KH10] Hyung-Won Koh et Eyke Hüllermeier : Mining gradual dependencies based on fuzzy rank correlation. *In In Proceedings of the 5th International of Soft Methods in Probabilities and Statistics (SMPS'10)*, 2010.

[KHC97] Micheline Kamber, Jiawei Han et Jenny Y. Chiang : Metarule-guided mining of multi-dimensional association rules. *In In Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining (KDD'97)*, pages 207–210, 1997.

[KLNS07] Chulyun Kim, Jong-Hwa Lim, Raymond T. Ng et Kyuseok Shim : Squire : Sequential pattern mining with quantities. *Journal of Systems and Software*, 80(10):1726–1745, 2007.

[KLRK10] Gayathri Tambaram Kailasam, Jin-Seung Lee, Jae-Won Rhee et Jaewoo Kang : Efficient skycube computation using point and domain-based filtering. *Information Sciences*, 180(7): 1090–1103, 2010.

[KN98] Edwin M. Knorr et Raymond T. Ng : Algorithms for mining distance-based outliers in large datasets. *In Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98)*, pages 392–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[Lau02] Anne Laurent : *Bases de données multidimensionnelles floues et leur utilisation pour la fouille de données*. Thèse de doctorat, université Paris 6, septembre 2002.

[LKLT08] Cécile Low-Kam, Anne Laurent et Maguelonne Teisseire : Mining for sequential outliers using a variable length markov model. *In 7th International Conference on Machine Learning and Applications (ICMLA'08)*, pages 571–576, Washington, DC, USA, décembre 2008. IEEE Computer Society.

[LLR09] Anne Laurent, Marie-Jeanne Lesot et Maria Rifqi : Graank : Exploiting rank correlations for extracting gradual itemsets. *In FQAS '09 : Proceedings of the 8th International Conference on Flexible Query Answering Systems*, pages 382–393, Berlin, Heidelberg, 2009. Springer-Verlag.

[MCP98] Florent Masseglia, Fabienne Cathala et Pascal Poncelet : The psp approach for mining sequential patterns. *In PKDD '98 : Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 176–184, London, UK, 1998. Springer-Verlag.

[MDNST05] Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli et Malika Smaïl-Tabbone : Méthode sémantique pour la classification et l'interrogation de sources de données biologiques. *In EGC 2005 - Atelier Modélisation des Connaissances*, Paris, 01 2005.

[MNSVS05] Sandy Maumus, Amedeo Napoli, Laszlo Szathmary et Sophie Visvikis-Siest : Fouille de données biomédicales complexes : extraction de règles et de profils génétiques dans le cadre de l'étude du syndrome métabolique. *In Journées Ouvertes Biologie Informatique Mathématiques - JOBIM 2005*, Lyon France, 2005.

[MPT04] Florent Masseglia, Pascal Poncelet et Maguelonne Teisseire : Pre-processing time constraints for efficiently mining generalized sequential patterns. *In 11th International Symposium on Temporal Representation and Reasoning (TIME '04)*, pages 87–95, 2004.

[MSSV08] Carlos Molina, José María Serrano, Daniel Sánchez et Maria-Amparo Vila : Mining gradual dependencies with variation strength. *Mathware & soft computing*, 15:75–93, 2008.

[PB05] R.G. Pensa et Jean-François Boulicaut : From local pattern mining to relevant bi-cluster characterization. *In* A. Fazel Famili, Joost N. Kok, José María Peña, Arno Siebes et A. J. Feelders, éditeurs : *IDA*, volume 3646 de *Lecture Notes in Computer Science*, pages 293–304. Springer, 2005.

[PCL+05] Marc Plantevit, Yeow Wei Choong, Anne Laurent, Dominique Laurent et Maguelonne Teisseire : M2SP : Mining Sequential Patterns Among Several Dimensions. *In 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*, Lecture Notes in Computer Science, pages 205–216. Springer, octobre 2005.

[PCT+03] Feng Pan, Gao Cong, Anthony K. H. Tung, Jiong Yang et Mohammed J. Zaki : Carpenter : finding closed patterns in long biological datasets. *In KDD '03 : Proceedings of the ninth*

*ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–642, New York, NY, USA, 2003. ACM.

[PGG+07] Marc Plantevit, Sabine Goutier, Françoise Guisnel, Anne Laurent et Maguelonne Teisseire : Mining unexpected multidimensional rules. *In ACM tenth international workshop on Data warehousing and OLAP (DOLAP'O7)*, pages 89–96, 2007.

[PHL01] Jian Pei, Jiawei Han et Laks V.S. Lakshmanan : Mining frequent itemsets with convertible constraints. *Data Engineering, International Conference on on Data Engineering (ICDE'01)*, pages 433–442, 2001.

[PHMa+04] Jian Pei, Jiawei Han, Behzad Mortazavi-asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal et Mei chun Hsu : Mining sequential patterns by pattern-growth : The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16:2004, 2004.

[PHP+01] Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen et Umeshwar Dayal : Multi-dimensional sequential pattern mining. *In CIKM '01 : Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88, New York, NY, USA, 2001. ACM.

[PJ98] Torben Bach Pedersen et Christian S. Jensen : Research issues in clinical data warehousing. *In SSDBM '98 : Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pages 43–52, Washington, DC, USA, 1998. IEEE Computer Society.

[PL08] Sellappan Palaniappan et Chua Sook Ling : Clinical decision support using olap with data mining. *International Journal of Computer Science and Network Security*, septembre 2008.

[PLT07a] Marc Plantevit, Anne Laurent et Maguelonne Teisseire : Extraction d'outliers dans des cube de données : une aide à la navigation. *In* Revue des Nouvelles Technologies de l'Information, éditeur : *Entrepots de Données et Analyse en ligne (EDA'07)*, pages 113–130, Poitiers, 2007.

[PLT07b] Marc Plantevit, Anne Laurent et Maguelonne Teisseire : Extraction d'outliers dans des cubes de données : une aide à la navigation. *In 3èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2007), Poitiers*, volume B-3 de *RNTI*, pages 113–130, Toulouse, Juin 2007. Cépaduès.

[PLT08] Marc Plantevit, Anne Laurent et Maguelonne Teisseire : Up and Down : Mining Multidimensional Sequential Patterns Using Hierarchies. *In* Johann Eder Tho Manh Nguyen Il-Yeol Song, éditeur : *Data Warehousing and Knowledge Discovery, 10th International Conference, DaWaK 2008*, Lecture Notes in Computer Science, pages 156–165. Springer Berlin / Heidelberg, septembre 2008.

[PMP01] G. Pavesi, G. Mauri et G. Pesole : An algorithm for finding signals of unknown length in dna sequences. *Bioinformatics*, 17 Suppl 1, 2001.

[PS91] Gregory Piatetsky-Shapiro : Discovery, analysis and presentation of strong rules. *In* Gregory Piatetsky-Shapiro et William J. Frawley, éditeurs : *Knowledge Discovery in Databases*, pages 229–248. AAAI Press, 1991.

[PTFS03] Dimitris Papadias, Yufei Tao, Greg Fu et Bernhard Seeger : An optimal and progressive algorithm for skyline queries. *In In SIGMOD*, pages 467–478, 2003.

[PTFS05] Dimitris Papadias, Yufei Tao, Greg Fu et Bernhard Seeger : Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1):41–82, 2005.

[PWcXW07] Jian Pei, Ada Wai-chee, Fu Xuemin et Lin Haixun Wang : Computing compressed multidimensional skyline cubes efficiently. *In International Conference on Data Engineering (ICDE'07)*, 2007.

[PYL+06] Jian Pei, Yidong Yuan, Xuemin Lin, Wen Jin, Martin Ester, Qing Liu, Wei Wang, Yufei Tao, Jeffrey Xu Yu et Qing Zhang : Towards multidimensional subspace skyline analysis. *ACM Trans. Database Syst.*, 31(4):1335–1381, 2006.

[RBCB03] Francois Rioult, Jean-François Boulicaut, Bruno Cremilleux et Jérémy Besson : Using transposition for pattern discovery from microarray data. *In The 8th ACM SIGMOD workshop on research issues in Data Mining and Knowledge Discovery (DMKD 03)*, pages 73–79, 2003.

[RCP08] Chedy Raïssi, Toon Calders et Pascal Poncelet : Mining conjunctive sequential patterns. *Data Min. Knowl. Discov.*, 17(1):77–93, 2008.

[RF98] Isidore Rigoutsos et Aris Floratos : Combinatorial pattern discovery in biological sequences : The teiresias algorithm. *Bioinformatics*, 14(1):55–67, 1998.

[Rio05] Francois Rioult : *Extraction de connaissances dans les bases de données comportant des valeurs manquantes ou un grand nombre d'attributs*. Thèse de doctorat, Université de Caen Basse-Normandie, novembre 2005.

[RMZ03] Ganesh Ramesh, William A. Maniatty et Mohammed J. Zaki : Feasible itemset distributions in data mining : theory and application. *In PODS '03 : Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 284–295, New York, NY, USA, 2003. ACM.

[RPK10] Chedy Raïssi, Jian Pei et Thomas Kister : Computing closed sycubes. *In In Proceedings of the 36th International Conference on Very Large Data Bases (VLDB'10)*, Singapore, 2010.

[SA96] Ramakrishnan Srikant et Rakesh Agrawal : Mining Quantitative Association Rules in Large Relational Tables. *In Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 1996.

[SAC90] Hamilton O. Smith, Thomas M. Annau et Srinivasan Chandrasegaran : Finding sequence motifs in groups of functionally related proteins. *In National Academy of Sciences of the United States of America*, volume 87, pages 826–830, janvier 1990.

[Sag98] Marie-France Sagot : Spelling approximate repeated or common motifs using a suffix tree. *Lecture Notes in Computer Science*, 1380:374–390, 1998.

[Sal10] Paola Salle : *Les motifs séquentiels pour les données issues des puces ADN*. Thèse de doctorat, I2S, Montpellier, juillet 2010.

[SNV07] Laszlo Szathmary, Amedeo Napoli et Petko Valtchev : Towards rare itemset mining. *In 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '07)*, volume 1, pages 305–312, Patras Grèce, 2007.

[Suz99] Einoshin Suzuki : Scheduled discovery of exception rules. *In Proceedings of the Second International Conference on Discovery Science (DS '99)*, pages 184–195, London, UK, 1999. Springer-Verlag.

[SVNG09] Laszlo Szathmary, Petko Valtchev, Amedeo Napoli et Robert Godin : Efficient vertical mining of frequent closures and generators. *In IDA '09 : Proceedings of the 8th International Symposium on Intelligent Data Analysis*, pages 393–404, Berlin, Heidelberg, 2009. Springer-Verlag.

[Wil82] R. Wille : Restructuring lattice theory : An approach based on hierarchies of concepts. *In Ordered Setsand in I. Rivals (Ed.)*, volume 23, 1982.

[WSB92] B. J. Wielinga, A. Th. Schreiber et J. A. Breuker : Kads : a modelling approach to knowledge engineering. *Knowl. Acquis.*, 4(1):5–53, 1992.

[XSBT09] Wei Xing, Paola Salle, Sandra Bringay et Maguelonne Teisseire : Demon-visualisation : un outil pour la visualisation des motifs séquentiels extraits à partir de données biologiques. *In Extraction et Gestion des Connaissances (EGC'09)*, janvier 2009.

[XZT08] Tian Xia, Donghui Zhang et Yufei Tao : On skylining with flexible dominance relation. *In ICDE '08 : Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 1397–1399, Washington, DC, USA, 2008. IEEE Computer Society.

[YLL+05] Yuan Yidong, Xuemin Lin, Qing Liu, Wei Wang, Jeffrey Xu Yu et Qing Zhang : Efficient computation of the skyline cube. *In IN VLDB*, pages 241–252, 2005.

[Zak01] Mohammed J. Zaki : Spade : an efficient algorithm for mining frequent sequences. *In Machine Learning Journal, special issue on Unsupervised Learning*, pages 31–60, 2001.

[ZH05] Mohammed J. Zaki et Ching-Jui Hsiao : Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):462–478, 2005.

# Articles publiés dans le cadre de cette thèse

### Revues internationales avec comité de lecture

- [DLT09a] L. Di Jorio, A. Laurent, M. Teisseire, "Gradual Rules : A Heuristic Based Method and Application to Outlier Extraction", *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*. 2009.

### Revues nationales avec comité de lecture

- [DBBLMT10] L. Di Jorio, S. Bringay, D. Brouillet, A. Laurent, S. Martin, M. Teisseire, "Fouille de données issues d'études psychologiques liées au vieillissement : extraction de règles graduelles", *Technique et science informatique (TSI), numéro spécial L'informatique à l'interface de l'activité humaine et sociale*. A paraître.
- [Dij10] L. Di Jorio, "Extraction de motifs graduels à partir de cubes de données", *Technique et science informatique (TSI), numéro Entrepôts de données et analyse en ligne*. A paraître.

### Conférences internationales avec comité de lecture

- [CDLLT09] Y. W. Choong, L. Di Jorio, A. Laurent, D. Laurent, M. Teisseire,"CBGP : Classication Based on Gradual Patterns", *In Proceedings of the International Conference on Soft Computing and Pattern Recognition*, 2009
- [DLT09b] L. Di Jorio, A. Laurent, M. Teisseire, "Mining Frequent Gradual Itemsets From Large Databases", *The 8th International Symposium on Intelligent Data Analysis, (IDA'09)*, Lyon
- [DLT08] L. Di Jorio, A. Laurent, M. Teisseire "Fast Extraction of Gradual Association Rules : A Heuristic Based Method", *IEEE/ACM International Conference on Soft Computing as Transdisciplinary Science and Technology, (CSTST 2008)*, Cergy-Pontoise, Paris

### Conférences nationales avec comité de lecture

- [DLT09c] L. Di Jorio, A. Laurent, M. Teisseire, "Extraction efficace de règles graduelles", *9èmes journées d'Extraction et Gestion des Connaissances, (EGC'09)*,199-204, Strasbourg
- [DCLT09] L. Di Jorio, Y.W. Choong, A. Laurent, M. Teisseire, "Règles graduelles et cubes de données : quand les blocs s'empilent !", *5èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2009)*, Montpellier

# Annexes

# Annexe A

# Biologie et bases génomiques

## A.1 La cellule

L'être humain est constitué de **cellules**, l'unité la plus petite du corps humain. Il existe des cellules de différents types et ayant différentes fonctions. En temps normal, ces cellules se divisent et grandissent afin de produire d'avantage de cellules en fonction des besoins du corps. Le mécanisme de reproduction et de transmission du patrimoine génétique assuré, la cellule meurt.

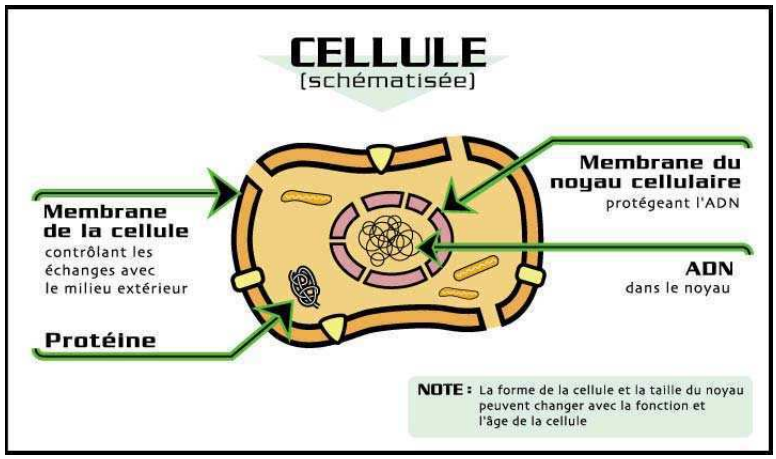



Fig A.1 – Schéma d'une cellule

La **cellule** (en latin *cellula* signifie *petite chambre*) est l'unité structurale, fonctionnelle et reproductrice constituant tout ou partie d'un être vivant. Chaque cellule est une entité vivante qui, dans le cas d'organismes multicellulaires, fonctionne de manière autonome, mais coordonnée avec les autres. Les cellules de même type sont réunies en tissus, eux-mêmes réunis en organes.

En bref, comme le montre la figure A.1, la cellule est constituée d'une membrane, et possède en son centre un noyau. C'est ce dernier qui nous intéresse, car il contient les instructions nécessaires à la réalisation des réactions chimiques (développement et fonctionnement de la cellule). Le support matériel de ces instructions sont les **chromosomes**.

## A.2 Le génome

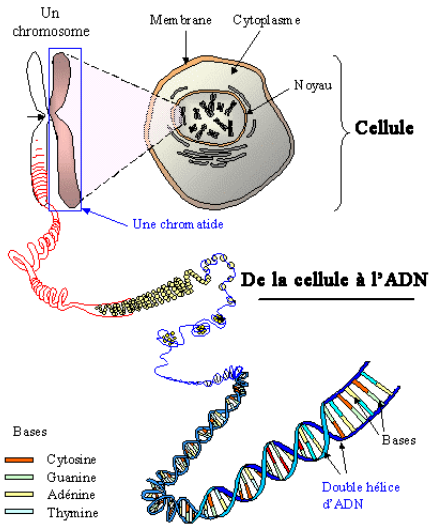



Fig A.2 – Du chromosome à l'ADN

Chez l'homme, le noyau contient 23 paires de chromosomes. Chaque chromosome est composé d'un exemplaire d'origine maternelle et d'un exemplaire d'origine paternelle expliquant la transmission des caractères génétiques de génération en génération. Chaque chromosome contient deux minuscules filaments : c'est la **double hélice d'ADN** (représenté à la figure A.3). L'ADN est donc le support de l'information génétique transmise lors du processus de reproduction cellulaire. L'ensemble du matériel génétique d'une espèce est appelé **génome**.

Le rôle de l'ADN ne s'arrête pas à porter l'information génétique. De manière générale, l'ADN sert à la **synthèse des protéines**. Les protéines sont indispensables à la vie, puisqu'elle remplissent par la suite diverses fonctions (transport, communication etc...). La synthèse des protéines s'effectue suivant deux grandes étapes : la transcription de l'ADN en ARN messager (ARNm) puis la traduction de l'ARNm en protéine. Ce processus est aussi appelé **expression de gène**, car un gène (portion d'ADN dont la position sur le chromosome est connue) va coder une ou plusieurs protéines. Nous expliquerons plus en détail la transcription à la section A.5.

Le matériel contenu dans la cellule est à la base du fonctionnement de l'organisme. Ainsi, la compréhension de tous ces mécanismes, qui s'avèrent souvent très complexes, permettra à terme de comprendre l'ensemble des disfonctionnements. Par exemple, une cellule devient cancéreuse suite à l'accumulation d'un nombre suffisant de défauts à l'intérieur du génome d'une cellule. Cela signifie que les gènes ont subi une modification, et donc que le matériel génétique supporté par les chromosomes a changé, ce qui va également modifier l'expression des gènes. Une manière de comprendre le cancer consiste à étudier ces changements. Pour cela, il est nécessaire de comprendre à quel niveau l'ADN est touché. Voyons tout d'abord ce qu'est exactement l'ADN.

## A.3 Acide désoxyribo-nucléique

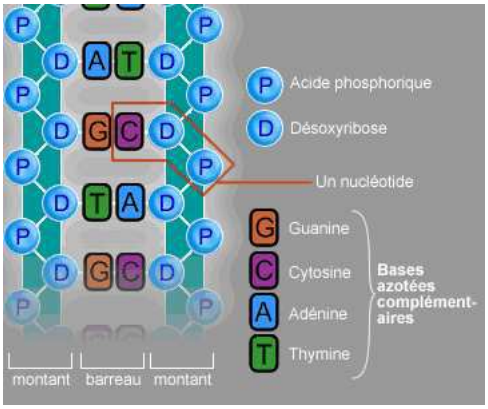



Fig A.3 – Schéma d'un brin d'ADN

L'ADN est composé de séquences de nucléotides on parle de polymère de nucléotides ou encore de polynucléotide. Chaque nucléotide est constitué de trois éléments liés entre eux :

- un groupe phosphate lié à :
- un sucre, le désoxyribose, lui-même lié à :
- une base azotée.

Il existe 4 bases azotées différentes : l'adénine (notée A), la thymine (notée T), la cytosine (notée C) et la guanine (notée G). La liaison entre les nucléotides se fait grâce à la répétition des sucre-phosphate. L'ADN est en fait composé de deux séquences ou brins, se faisant face, et formant une double hélice. Grâce à l'alternance des 4 bases azotées A,C,T,G, toutes ces séquences dans l'ADN constituent un message codé, portant les informations génétiques. Le lien entre l'information génétique, et les caractères de l'organisme (le phénotype), est gouverné par le **code génétique**.

Le code génétique est un système de correspondance entre les séquences de nucléotides de l'ADN et les séquences en acides aminés des protéines. En effet, l'enchaînement des quatre nucléotides A,C,T,G, doit coder l'enchaînement des 20 acides aminés dans les protéines. Le codage d'un acide aminé nécessite donc au minimum une suite de 3 bases.

## A.4 Les protéines

Une protéine, aussi appelée protide, est une macromolécule composée par une ou plusieurs chaîne(s) (ou séquence(s)) d'acides aminés liés entre eux par des liaisons peptidiques. En général, on parle de protéine lorsque la chaîne contient plus de 100 acides aminés. Dans le cas contraire, on parle de peptides et de polypeptides. L'enchaînement des acides aminés est codé par le génome et constitue la structure primaire.

Les acides aminés sont produits lors du processus de traduction, expliqué à la section A.5. Ceux-

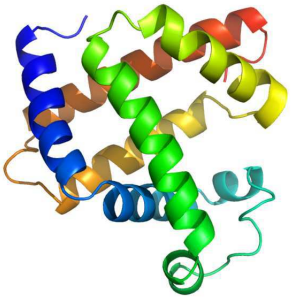


Fig A.4 – Schéma de la myoglobine

ci vont s'enrouler de manière différente, ce qui va produire différentes protéines. Cette propriété est importante à comprendre, puisqu'elle confère un aspect structurel aux protéines, ce qui se traduira par l'ajout d'une dimension lors d'une analyse automatique. La figure A.4 représente la myoglobine, et met en avant l'aspect structurel des protéines.

## A.5 La synthèse des protéines

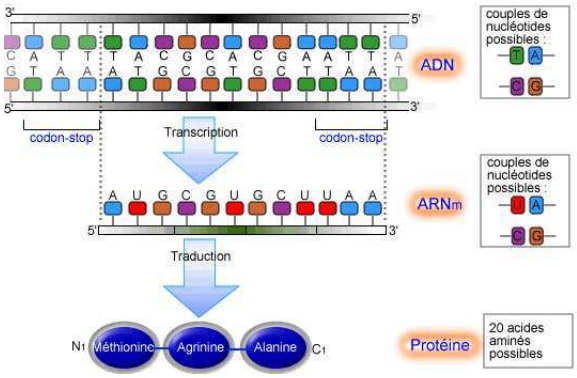



Fig A.5 – Schéma de la synthése des protéines

La synthèse des protéines s'effectue à partir des gènes. Comme illustré à la figure A.5, elle se déroule en deux étapes au moins : la transcription de l'ADN en ARN messager et la traduction de l'ARN messager en une protéine.

La transcription consiste à faire une "copie de travail" de l'ADN : l'ARN messager, noté $\text{ARN}_m$. Le début des gènes le long de la molécule d'ADN est marqué par des séquences spéciales applées **séquences consensus**. il ne s'agit en effet pas de séquences exactes mais de séquences approchées d'une séquence moyenne mais différant seulement par quelques paires de bases. Les deux utilisées pour repérer le début d'un gène sont les boîtes `CAAT` et `TATA`, du nom des nucléotides formant le coeur de la séquence moyenne et situées dans une zone précédant le gène appelée **promoteur** car elle initie la transcription du gène. Pour résumer, la transcription se passe de la manière suivante : une protéine dédiée à la transcription (généralement apellée **facteur de transcription**) des gènes va se fixer en un endroit précis de l'ADN, situé dans le promoteur. Ce complexe va parcourir la molécule d'ADN pour la lire. Il va tout d'abord

dérouler la molécule d'ADN, puis séparer les deux brins, puis assembler les bases azotées en se servant du brin complémentaire pour aboutir à la molécule d'ARN.

**Exemple 13.**
*le brin d'ADN non transcrit : A T A G C G T T C A G A A C T G A T A C G T A A*
*le brin d'ADN transcrit :* *T A T C G C A A G T C T T G A C T A T G C A T T*
*le brin d'ARN :* *A U A G C G U U C A G A A C U G A U A C G U A A*

Puis le processus de traduction de l'ARN$_m$ en acide aminé va avoir lieu. On peut diviser l'ARN$_m$ en plusieurs groupes de trois nucléotides appelés codon, et chaque codons correspond à un acide aminé. De manière analogue aux bruns d'ADN, l'ARN$_m$ possède un codon-initiateur, qui permet de commencer la traduction, et un codon-stop qui marque l'arrêt de la traduction.

**Exemple 14.**
*Le brin d'ARN messager est A U A G C G U U C A G A A C U G A U A C G U A A*
*Les codons sont* *AUA GCG UUC AGA ACU GAU ACG UAA*
*L'ARN$_t$ est* *UAU CGC AAG UCU UGA CUA UGC AUU*
*Les acides aminés* *Île Ala Phe Arg Thr Asp Thr X*

Les exemples 13 et 14 montrent la traduction d'une portion d'ADN en une séquence d'acides aminés qui vont composer une protéine [1].

## A.6 Puces à ADN : étude des gènes

La puce à ADN est un ensemble de molécules d'ADN fixées sur une surface qui peut être du verre, du silicium ou bien encore du plastique. Cette biotechnologie récente permet de quantifier le niveau d'expression des gènes (transcrits) dans une cellule d'un tissu donné (foie, intestin...), à un moment donné (embryon, adulte...) et dans un état donné (malade, saine...). Le processus CGH peut être découpé en trois grandes étapes :

1. **L'hybridation :** La puce est une plaque de petite taille sur laquelle sont fixés des brins monocaténaires (un seul brin au lieu des deux habituels) d'ADN, chacun correspondant au brin complémentaire d'un ARN messager (ARNm). On peut fixer sur cette plaque plusieurs dizaines de milliers de fragments d'ADN (et donc étudier l'expression d'autant de gènes). La première étape consiste donc à extraire les ARNm (provenant des gènes exprimés) de la cellule à analyser et à fixer des fluorochromes dessus (couleur rouge). Puis l'échantillon des ARNm est versé sur la plaque : chaque brin d'ARNm va s'hybrider au brin monocaténaire d'ADN qui lui est complémentaire pour former un double brin. La plaque est ensuite nettoyée pour éliminer les brins d'ARNm ne s'étant pas hybridés. Cette étape est représentée en haut de la figure A.6 : coloration de l'ADN des deux cellules puis hybridation.

2. **Analyse des résultats :** La puce est ensuite scannée au laser et une image de la puce est créée : chaque fois qu'il y a eu hybridation, le fluorochrome fixé sur l'ARNm a émis dans la longueur

1. Ces exemples sont tirés du site http ://fr.wikipedia.org/wiki/Synth%C3%A8se_des_prot%C3%A9ines.

d'onde du laser et cela est visible par un point de couleur. Des logiciels interprètent la luminosité de chaque point de la plaque contenant un ADN différent et en déduisent une mesure numérique de l'expression de chaque gène. Dans notre cas, les gènes dont il y a perte d'expression sont représentés en vert, un gain est représenté en rouge, et une expression stable apparaîtra en jaune. La mesure numérique de l'expression de chaque gène est représentée par le ratio rouge / vert, seuillé par les valeurs 0.5 et 2 (arbitrairement choisi par les biologistes). Le logarithme de base 2 est utilisé afin de normaliser les données en vue de test statistiques.

3. **filtrage résultats :** Il s'agit de normaliser chaque résultat, filtrer les réplicats et éditer un rapport de qualité, permettant de déterminer si l'échantillon est fiable ou non.

Ainsi, cette technique permet d'extraire les différences d'expression de gènes entre deux types de cellules, comme par exemple une cellule cancéreuse et une cellule normale. L'étape suivante consiste à analyser ces données. C'est là qu'intervient le processus de fouille de données.

## A.7 Vers une modélisation globale du système : les pathways

Le pathway est un outil de modélisation de systèmes complexes (avec des boucles de rétroaction), permettant d'étudier leur dynamique. Il peut être défini comme étant un ensemble d'entités en relation pouvant interagir les unes avec les autres et avec lui-même. Les pathways modélisent en général trois réseaux bien distincts :

- **Le réseau de régulation des gènes :** il est rare qu'un gène s'exprime seul. En réalité, certains gènes s'expriment en même temps car ils participent à la synthèse de plusieurs protéines liées. On parlera alors de **co-expression de gènes**. Le réseau de régulation des gènes tend donc à modéliser l'ensemble des co-expressions, activations et inhibitions des gènes. La difficulté réside dans le fait que ces réseaux sont totalement dynamiques, et que la découverte des co-expressions de gène est un problème ouvert.
- **Les cascades signalétiques :** elles visent à modéliser l'ensemble des réactions de l'organisme. Par exemple, lorsque l'organisme est attaqué par un virus, il se protège et déclenche ainsi tout un ensemble de réactions. On retrouve également dans cette modélisation les intéractions protéines-protéines.
- **Les voies métaboliques :** elles modélisent l'ensemble des réactions chimiques qui servent à produire un composant.

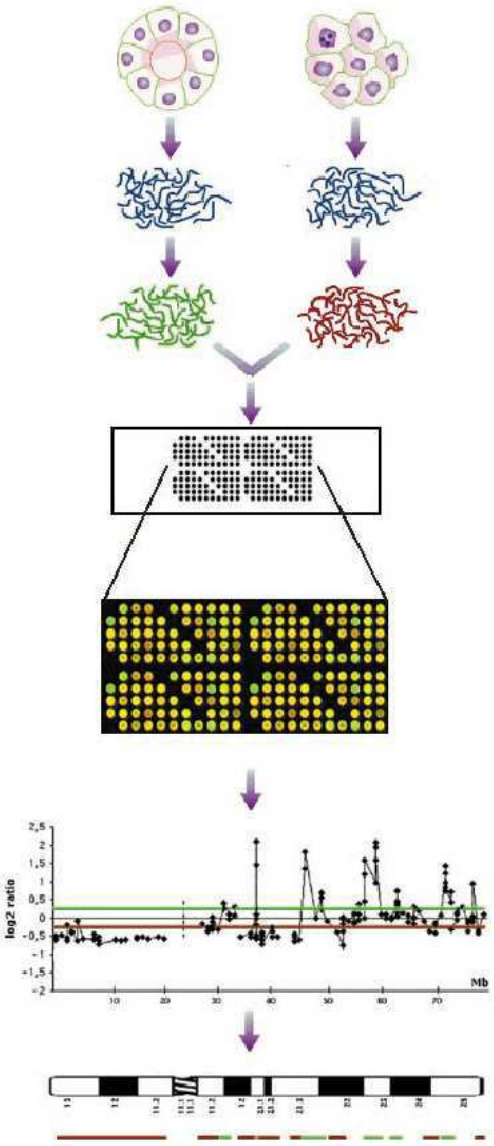



Fig A.6 – Processus CGH

**Recherche de motifs graduels et application aux données médicales**

Avec le développement des nouvelles technologies d'analyse (comme par exemple les puces à ADN) et de gestion de l'information (augmentation des capacités de stockage), le domaine de la santé a particulièrement évolué ces dernières années. En effet, des techniques de plus en plus avancées et efficaces sont mises à disposition des chercheurs, et permettent une étude approfondie des paramètres génomiques intervenant dans des problèmes de santé divers (cancer, maladie d'Alzheimer ...) ainsi que la mise en relation avec les paramètres cliniques. Parallèlement, l'évolution des capacités de stockage permet désormais d'accumuler la masse d'information générée par les diverses expériences menées. Ainsi, les avancées en terme de médecine et de prévention passent par l'analyse complète et pertinente de cette quantité de données. Le travail de cette thèse s'inscrit dans ce contexte médical. Nous nous sommes particulièrement intéressé à l'extraction automatique de motifs graduels, qui mettent en évidence des corrélations de variation entre attributs de la forme "plus un patient est âgé, moins ses souvenirs sont précis". Nous décrivons divers types de motifs graduels tels que les itemsets graduels, les itemset multidimensionnels graduels ou encore les motifs séquentiels graduels, ainsi que les sémantiques associées à ces motifs. Chacune de nos approches est testée sur un jeu de données synthétique et/ou réel.

**Gradual patterns extraction and application to health data**

With the raise of new biological technologies, as for example DNA chips, and IT technologies (e.g. storage capacities), health care domain has evolved through the last years. Indeed, new high technologies allow for the analysis of thousands of genomic parameters related to various deseases (as cancer, Alzheimer), and how to link them to clinical parameters. In parallel, storage evolutions enable nowadays researchers to gather a huge amount of data generated by biological experiments. This Ph.D thesis is strongly related to medical data mining. We tackle the problem of extracting gradual patterns of the form "the older a patient, the less his memories are accurate". To handle different types of information, we propose to extract gradualness for an extensive range of patterns : gradual itemsets, gradual multidimensionnal itemsets, gradual sequencial patterns. Every contribution is experimented on a synthetic or real datasets.

---

**Mots-clés** : Extraction de connaissances, fouille de données, règles d'association, gradualité, motifs graduels, itemsets graduels, motifs séquentiels, bases médicales.

**Keywords** : knowledge extraction, data mining, association rules, gradualness, gradualness patterns, gradual itemsets, sequential patterns, health databases.

---

**Discipline** : Informatique

**Laboratoire** : Laboratoire d'Informatique de Robotique et de Micro-électronique de Montpellier
Université Montpellier II - CNRS (UMR 5506) – CC 477
161 rue Ada - 34095 Montpellier cedex 5 - France