# Règles graduelles et cubes de données : quand les blocs s'empilent !

Lisa Di Jorio*, Yeow Wei Choong**
Anne Laurent * Maguelonne Teisseire***

*LIRMM – Université de Montpellier 2 – CNRS
161 rue Ada, 34392 Montpellier – FRANCE
{name}@lirmm.fr,
http://www.lirmm.fr/tatoo
**HELP University College – Kuala Lumpur – MALAYSIA
choongyw@help.edu.my
***Cemagref - UMR Tetis
maguelonne.teisseire@teledetection.fr

**Résumé.** Le couplage des méthodes de fouille de données et d'entrepôts de données permet d'extraire des informations pertinentes à partir de cubes de données. Dans ce contexte, de nombreuses approches ont été proposées, permettant par exemple d'extraire des règles d'association ou des motifs séquentiels. Cependant, il n'existe pas de méthodes permettant d'extraire des règles graduelles. Dans cet article nous nous intéressons donc à la découverte de telles règles corrélant des variations sur un ensemble de dimensions ordonnées avec des variations sur la mesure du cube. Nous découvrons par exemple des règles du type "*Plus la ville est de taille importante et la catégorie socio-professionnelle de catégorie supérieure, plus le nombre de produits vendus est grand*". Afin de découvrir ces règles de manière efficace et en prenant en compte les grandes tendances issues des cubes de données, nous nous appuyons sur des travaux précédents permettant d'extraire des blocs de données homogènes.

## 1 Introduction

La fouille de cube de données, initiée par Chen et Pei (2006) qui a introduit l'OLAP Mining, 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 grandissants d'applications en quasi temps réel des utilisateurs.

Dans ce contexte, des travaux ont été proposés ces dernières années, notamment pour extraire des règles d'association, des résumés flous, ou encore des motifs séquentiels multidimensionnels. Il est alors possible d'extraire des règles du type *la plupart des ventes de canoës à Boston sont faibles*, ou *la plupart des ventes de l'est sont effectuées à Boston* ou encore *pour la plupart des catégories socio-professionnelles, on trouve des achats de canoes dans une ville de l'Est puis des achats de combinaisons et de housses à San Francisco*. 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 Plantevit et al. (2007a,b).

Cependant, il n'existe pas à notre connaissance de méthode permettant d'extraire des règles graduelles de la forme *Plus la ville est de taille importante et la catégorie socio-professionnelle haute, plus le nombre d'achats est élevé*. 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 villes selon leur taille, selon leur degrés de respect de l'environnement, selon la note attribuée pour la qualité de vie, ou encore les catégories socio-professionnelles, la taille du magasin où s'est effectué l'achat, etc. Ce type de règles permet alors de retrouver des corrélations au sein de ces dimensions ordonnées. Nous proposons donc dans cet article les définitions nécessaires à la formalisation de notre approche, ainsi que les algorithmes associés. 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 Choong et al. (2008, 2004, 2007). 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. Nous montrons dans cet article qu'ordonner par ordre croissant ou décroissant ne modifie en aucun cas les règles graduelles extraites.

Dans la suite de cet article, nous décrivons dans la section 2 le contexte de notre étude et les travaux associés. Dans la section 3 nous introduisons les définitions liées à notre contribution, nous détaillons notre proposition dans la section 4, tandis que la section 5 présente les résultats des expérimentations menées. Enfin, la section 6 conclut et expose les principales pistes de recherche ouvertes par notre travail.

## 2 Travaux antérieurs

L'extraction de règles graduelles à partir de cubes se trouve au carrefour de plusieurs problématiques : l'extraction de connaissances au sein de cubes mmultidimensionnels(règles d'association et motifs séquentiels), la présentation des connaissances contenues dans le cube à l'utilisateur Choong et al. (2003), ou encore la prise en compte de la mesure Plantevit et al. (2007b) et des hiérarchies Choong et al. (2007). D'autre part, la gradualité en elle même est une notion assez récente dans la fouille de données. Ainsi, nous décrivons dans cette section les travaux dédiés à la présentation des cubes de données, puis les différents travaux menés dans le cadre de la gradualité.

Dans Choong et al. (2003), 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 de la dimension *produits* 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 permettraient de trouver un ordre de présentation des membres des dimensions amenant de fait un ordre de présentation de la mesure dont on retrouverait alors les plus faibles valeurs à un *coin* du cube et les plus fortes au *coin opposé*. 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 OLAP d'inversion. Cependant, le problème est NP-complet, et la

recherche de telles représentations peut s'avérer 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 Choong et al. (2007), les auteurs considèrent qu'il existe plusieurs représentations, mais ils ne travaillent que sur une seule. 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 Choong et al. (2008) et permet de prendre en compte la hiérarchie des dimensions.

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, les blocs proposent une perspective intéressante afin d'éliminer le bruit, de par la définition des blocs, car une mesure de support et de confiance est associée à chaque bloc, ainsi les cellules vides font baisser ces mesures. Cependant, les blocs n'offrent pas la possibilité d'expliquer facilement ces corrélations. C'est pourquoi, dans cet article, nous proposons d'extraire une information supplémentaire : la corrélation de variation qu'il existe d'une part entre les membres d'une même dimension, et d'autre part entre les valeurs des blocs.

La notion de gradualité, longuement étudiée dans le cadre de la logique floue, a été récemment étudiée dans le cadre de la fouille de données : Hüllermeier (2002); Berzal et al. (2007); Di-Jorio et al. (2008, 2009). En effet, plutôt que de considérer les règles graduelles comme prédéfinies, la fouille de données permet de découvrir de telles règles. Cette démarche devient nécessaire face à la quantité d'information relative aux systèmes.

Dans le cas de Hüllermeier (2002), il s'agit d'utiliser des méthodes telles que la régression linéaire et les coefficients de variation. L'approche peut extraire des règles d'associations de tendances contenant un grand nombre d'items. Cependant, il n'est pas possible de mélanger les sens de variation. Ainsi, les règles seront de la forme "*l'age augmente et le salaire augmente*", mais jamais de la forme "*l'age augmente et le salaire diminue*". Berzal et al. (2007) propose d'adapter l'algorithme Apriori à l'extraction d'itemsets graduels. L'approche permet de mélanger les sens de variation, mais sa complexité s'avère élevée, ce qui peut rendre l'extraction impossible dès lors que la base est trop grande. Enfin, nous avons proposé une heuristique (Di-Jorio et al. (2008)) puis une approche exhaustive (Di-Jorio et al. (2009)). La nouveauté réside dans le fait que nous n'utilisons plus la logique floue mais travaillons directement sur les valeurs d'une part, et la structure utilisée dans notre approche passe à l'échelle. Cependant, ces règles graduelles ne peuvent être directement appliquées sur des cubes de données. Pourtant, les cubes possédant des degrès d'information différents (membres de dimensions, hérarchies, aggregation), il peut s'avérer intéressant de les utiliser afin d'extraire de nouveaux types de corrélations.

Dans cet article, nous proposons d'extraire des règles graduelles à partir de cubes de données. Celles-ci permettent de corréler les variations sur les dimensions aux variations sur les valeurs des blocs. Pour ce faire, nous avons défini les spécificité que doivent avoir les dimensions, ainsi qu'une règle graduelle. Afin de comprendre la formalisation de la section 3, nous rappelons ici les formalisations des cubes, représentations et blocs proposées par Choong et al. (2007).

Les cubes de données sont définis de manière diverse. Dans cet article, 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 valeurs noté $dom_i$. Un cube de données est alors défini à partir de ces dimensions.

**Définition 1.** *(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 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.*

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 1 illustre un cube à deux dimensions. Nous avons $C = \langle\{10, 20, 30, 40, 50\}$, $\{1, 2, 3, 4, 5\}, \{4, 5, 6, 8\}, m_C\rangle$. Nous avons deux dimensions $d_1$ = Taille (représentant la taille de la ville) et $d_2$ = CSP (la catégorie professionnelle), avec $dom_1 = \{10, 20, 30, 40, 50\}$ et $dom_2 = \{1, 2, 3, 4, 5\}$. 1 est un membre de $dom_2$.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 10 | 6 | | | | |
| 20 | 6 | 8 | 8 | 8 | |
| 30 | 5 | 5 | 8 | 4 | 4 |
| 40 | 5 | 5 | | 4 | 4 |
| 50 | 5 | 5 | | 4 | 4 |

FIG. 1: Exemple de cube à deux dimensions, avec blocs

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, et les dimensions dont les membres peuvent être ordonnés (dimension temporelle par exemple), sont affichées de manière ordonnée. Cependant, cette représentation est arbitraire. En effet, il est possible d'afficher les membres dans un ordre différent. Les mêmes données seront alors présentées différemment. Par exemple, nous aurions pu afficher la figure 1 sous la représentation de la figure 2, où les lignes ainsi que les colonnes 2 et 4 ont été interverties. Il existe donc différentes représentations d'un même cube.

**Définition 2.** *(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 cet article, nous considérons comme dans Choong et al. (2007) une représentation donnée. Cette représentation peut avoir été fixée par l'utilisateur ou provenir d'opérations antérieures. Nous ne traitons pas cet aspect dans cet article. 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, Choong et al. (2007) proposent d'extraire des blocs de données, qui sont définis de la manière suivante :

**Définition 3.** *(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 $ALL_i$).

Prenons l'exemple de la figure 1, qui représente un cube à 2 dimensions. La dimension représentée sur les colonnes est la taille de la ville, et la dimension représentée sur les lignes est la catégorie socio-professionnelle. De ce cube, quatre blocs peuvent être extraits :

– $b_1 = [30, 50] \times [1, 2]$, associé à la valeur 5
– $b_2 = [20, 30] \times [2, 4]$, associé à la valeur 8
– $b_3 = [30, 50] \times [4, 5]$, associé à la valeur 4
– $b_4 = [10, 20] \times [1]$, associé à la valeur 6

| | CSP | | | | |
|---|---|---|---|---|---|
| | *1* | *4* | *3* | *2* | *5* |
| *10* | | | | 6 | |
| *30* | 4 | 5 | 8 | 5 | 4 |
| *40* | 4 | 5 | | 5 | 4 |
| *20* | 8 | 8 | 8 | 6 | |
| *50* | 4 | 5 | | 5 | 4 |

FIG. 2: Exemple de cube à deux dimensions, sans blocs

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 une ville ayant une taille de 30 à 50 et pour une catégorie socio-professionnelle de 1 et 2, la quantité de produits achetée est 5*". Cependant, les blocs permettent uniquement d'extraire des membres de dimension dont les valeurs sont contiguës. Les blocs mettent donc en évidence des membres corrélés, mais ne montrent pas de corrélations graduelles sur l'ensemble du cube. Par exemple, on extraira "Lorsque la taille de la ville est 30,40,50 et la catégorie socio-professionnelle est 1 ou 2, alors le nombre de ventes est de 5". Nous manquons alors la corrélation qu'il existe entre les valeurs de la dimensions : "plus la taille de la ville augmente" et "plus la catégorie socio-professionnelle augmente". De plus, dans ce cas de figure, il est alors possible de comparer les valeurs des différents blocs défini sur ces dimensions. C'est pourquoi nous proposons d'utiliser l'ensemble des blocs extraits afin d'extraire des corrélations graduelles à partir d'un cube de données. Notre méthode peut donc être vue comme un post-traitement à l'extraction de blocs.

# 3 Gradualité et blocs : une nouvelle formalisation

## 3.1 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 1, nous souhaiterions extraire "*Plus la distance à l'est diminue et plus la catégorie socio-professionnelle 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 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 4.** *(Dimension ordonnée) Une dimension* $d_i$ *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 elle peut 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 4 nous permet d'introduire la notion de gradualité sur les dimensions. Ainsi, nous pourrons dire "*plus la ville est de grande taille*" ou "*plus la ville est de petite taille*".

Dans cet article, nous ne considérons que les dimensions ordonnées, c'est-à-dire 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 5.** *(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 règles supportant une règles. Toutefois, les blocks de valeurs égales participeront à la fois au support de l'augmentation et de la diminution.

**Définition 6.** *(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 cet article, nous considérons que les règles multidimensionnelles graduelles peuvent être générées à partir des DG-Set en post-traitement. Par abus de language, 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 7.** *(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 respectivement* $m$ *et* $m'$*. Soit* $r = [\{d_1^{*_1}, ..., d_n^{*_n}\}, *_m]$ *une DG-set.* $b$ *précède* $b'$ *en fonction de* $r$ *si :*

- $m *_m m'$
- $\forall j \in [1, n], \min(\delta_j) *_j \min(\delta'_j) \wedge \max(\delta_j) *_j \max(\delta'_j)$

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

Par exemple, lorsque l'on considère la règle multidimensionnelle graduelle $r_1 = [\{CSP^{\leq}\}, \geq]$ (*Plus la catégorie socio-professionnelle diminue, plus la valeur des blocks 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 8.** *(Liste de blocs ordonnés) Soit* $r = [\{d_1^{*_1}, ..., d_n^{*_n}\}, *_m]$ *une DG-set. Une liste de blocs* $\mathcal{L} =<_L b_1, ..., b_x >_L$ *respecte* $r$ *si* $\forall b_i, b_j \in \mathcal{L}$ $b_i \lhd_r b_j$*.*

Par exemple, plusieurs listes respectent la règle $[\{CSP^{\leq}\}, \geq]$ : $\mathcal{L}_1 =<_L b_3, b_2, b_4 >_L$ et $\mathcal{L}_2 =<_L b_1, b_2, b_4 >_L$. 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 9.** *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* $Freq(r) = \frac{\max_{1 \leq i \leq z}(|\mathcal{L}_i|)}{\mathcal{B}}$

## 3.2 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'antimonotonie. Pour ce faire, nous redéfinissons la notion d'inclusion de la manière suivante :

**Définition 10.** *(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 1.** *(Antimonotonie 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 longeur 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'antimonotonie 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). C'est pourquoi nous proposons l'utilisation de la complémentarité afin de réduire l'espace de recherche :

**Définition 11.** *(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, $c([\{CSP^{\geq}\}, \leq]) = [\{CSP^{\leq}\}, \geq]$, mais $c([\{CSP^{\geq}\}, \leq]) \neq [\{CSP^{\geq}\}, \geq]$.

**Proposition 2.** *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 1.** $Freq(r) = Freq(c(r))$

Le corollaire 1 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 la taille de la ville diminue et plus la catégorie socio professionnelle augmente alors plus la valeur augmente*" est exactement la même chose que "*Moins la taille de la ville diminue et moins la catégorie socio-professionnelle augmente alors plus la valeur diminue*".

# 4 Mise en œuvre

Dans cette section, nous détaillons la méthode proposée afin d'extraire des règles graduelles à partir de blocs. Ainsi, dans un premier temps, nous expliquons comment est traversé l'espace de recherche. Ensuite, nous montrons comment utiliser une structure binaire correspondant à nos besoins et enfin, nous détaillons la manière de calculer le support en prennant en compte les divers choix possibles pour une même règle.

## 4.1 Recherche sur les dimensions

Nous proposons un algorithme par niveau qui augmente à chaque passe le nombre de dimensions graduelles. Ainsi, à l'image de l'algorithme Apriori de Agrawal et Srikant (1994), nous construisons un arbre des préfixes. Dans cette structure, chaque noeud 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 1 nous permettant de ne générer que la moitié des DG, nous illustrons dans cet article l'extraction de connaissances graduelles sur l'augmentation de la mesure (les supports des diminutions sont obtenues en inversant les opérateurs). De plus, nous rappelons que les dimensions considérées sont des dimensions ordonnées, les dimensions non ordonnées étant ignorées.

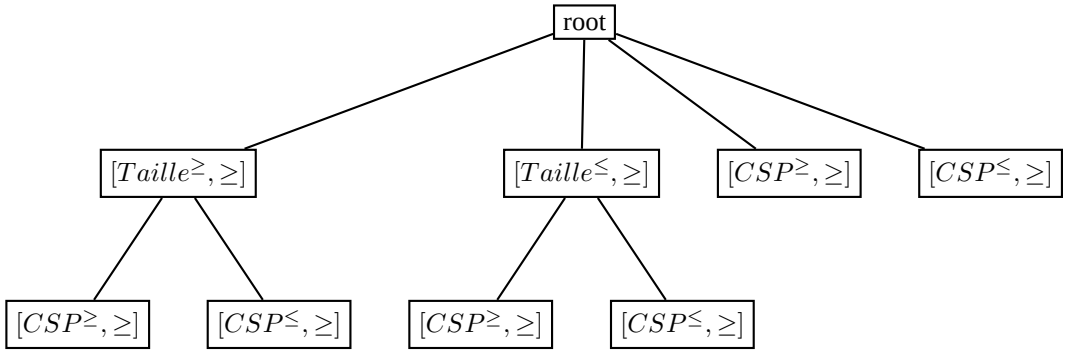



FIG. 3: Un exemple d'arbre préfixé pour l'extraction de DG-sets

La figure 4.1 montre l'arbre des préfixes généré pour extraire des DG-sets à partir du tableau 1. 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 lieux 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 1, nommé ExtraireDG-set, est la routine générale permettant l'extraction de l'ensemble des DG-set d'un cube de données.

**Algorithme 1** : ExtraireDG-set

**Entrées** : Un ensemble de blocs $\mathcal{B}$
Un seuil de fréquence minimal $\sigma$
**Sorties** : L'ensemble des DG-set

```
1  L_1 ← GenererNiveau1()
2  k ← 2
3  tant que L_{k-1} ≠ ∅ faire
4  |  L_k ← GenererNiveau(L_{k-1})  pour chaque l ∈ L_k faire
5  |  |  si CalculFreq(l) < σ alors
6  |  |  |  L_k ← {L_k \ l}
7  |  |  fin
8  |  fin
9  |  k ← k + 1
10 fin
```

Dans cet algorithme, trois étapes sont importantes : la génération du premier niveau, la génération des niveaux $k$ pour $k > 1$ et enfin le comptage de fréquence. La recherche de corrélations graduelles est très proche de l'extraction d'ordre, comme dans Pei et al. (2006). Une solution naïve consiste à ordonner sur une dimension, puis sur la suivante en fonction de la première et ainsi de suite. Cependant, l'ordre dans lequel est considérée chacune de ces dimensions est déterminant sur le résultat. Il suffit d'inverser par exemple la première et la seconde dimension pour obtenir des résultats différents. Dans notre contexte, il est important que l'extraction soit totalement indépendante de l'ordre dans lequel sont considérées les dimensions.

Une solution consiste alors à conserver toutes les solutions possibles pour un DG-set de taille $k$ donné, afin de considérer la meilleure lors de la génération du DG-set de taille $k + 1$. Cela revient à conserver, pour chaque noeud de l'arbre des préfixe, l'ensemble $\mathcal{G}_r$. Cependant, cet ensemble dépend fortement du nombre de blocs extrait. Ainsi, nous proposons dans la suite de cet article une structure adaptée à la conservation de cet ensemble.

## 4.2 Conservation binaire des ordres

L'utilisation de structure binaire n'est pas nouvelle en fouille de données. Ainsi, dans Ayres et al. (2002), des vecteurs binaires permettent de conserver les ensembles d'objets respectant un motif séquentiel. Les structures binaires peuvent également être adaptées à la problématique d'extraction de gradualité. En effet, il s'agit de conserver les relations communes à deux DG-sets.

Plutôt que de conserver une présence absence de bloc (comme dans un cas d'extraction classique), nous conservons la présence absence de relation entre blocs. Ainsi, pour un ensemble de blocs $\mathcal{B}$ extraits à partir d'un cube k-dimensionnel, il peut exister au pire $|\mathcal{B}| \times |\mathcal{B}|$ relations à conserver. C'est pourquoi, comme dans Di-Jorio et al. (2009) nous utilisons des matrices binaires, qui permettent de représenter huit absences ou présences de relation par octet.

**Définition 12.** *(Matrice des ordres) Soit $\mathcal{B}_{\mathcal{G}_r}$ l'ensemble des blocs contenus dans $\mathcal{G}_r$, et $d = |\mathcal{B}_{\mathcal{G}_r}|$. $\mathcal{G}_r$ peut être représenté par une matrice binaire $M_{\mathcal{G}_r} = m_{t_a,t_b,\forall t_a,t_b \in \mathcal{B}_{\mathcal{G}_r} \forall a,b \in [1,d]\times[1,d]}$, où $m \in \{0,1\}$.*

Nous adoptons la représentation suivante ; pour une règle $r$, nous construisons une matrice $M$ telle que :

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

Par exemple, en reprenant les quatre blocs extraits du cube défini sur les dimensions taille de la ville et catégorie socio-professionnelle (tableau 1), nous construisons la matrice binaire affichée au tableau 1a. La matrice se lit de la manière suivante : le bloc de la ligne considérée précède (1) ou non (0) la colonne considérée. Par exemple, pour l'élément de la première ligne seconde colonne, on lit que $b_2$ précède $b_1$.

Il est important de noter qu'avec notre approche, lorsqu'un bloc est "recouvert" par un autre bloc, il n'existe pas de relation de gradualité entre eux. Par exemple, si nous considérons $b_i = [1,4] \times [1,3]$ et $b_j = [2,3] \times [1,2]$, alors $b_i$ ne précède pas $b_j$, car l'intervalle défini sur la première dimension de $b_j$ est strictement inclus dans celui de $b_i$, ainsi que l'intervalle défini sur la seconde dimension.

La construction des 1-DG (premier niveau de l'arbre des préfixes) est particulière, puisqu'elle doit calculer les relations qui respectent la gradualité à la fois sur la dimension et sur la valeur des blocs. Cependant, les blocs sont définis comme des intervalles de membres de dimension. Il faut alors d'une part ordonner ces intervalles, et d'autre part ordonner sur la valeur des blocs. Afin d'effectuer cette étape de manière efficace, nous proposons l'algorithme 2, dont le principe est le suivant : construire une matrice binaire pour l'ordre sur les valeurs des blocs (ligne 2), et une matrice pour l'ordre en intervalle tel que décrit par la définition 7. L'intersection de ces deux matrices donne alors l'ordre pour le 1-DG.

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

(a)

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

(b)

TAB. 1: Matrice binaire pour (a) $[Ville^{\leq}, \geq]$ et (b) $[CSP^{\geq}, \geq]$

**Algorithme 2** : GenererDimension1

**Entrées** : Les intervalles $\mathcal{R}$ pour une dimension $d$
La matrice $M$ des ordres des valeurs entre blocs
Un opérateur $*$

**Sorties** : La matrice associée à $d^*, \geq$

1 $T_d \leftarrow OrdonnerIntervalles(d)$
2 $M_{int} \leftarrow GenererMatriceIntervalles(d)$
3 **retourner** $M_{int} \wedge M$

La jointure entre deux DG-sets $r$ et $r'$ doit conserver les relations entre blocs présente à la fois dans $r$ et dans $r'$. En utilisant les matrices binaires, cela revient à effectuer un ET binaire (noté $\wedge$) :

**Théorème 1.** *Soit $r''$ une DG-set générée à partr des deux DG-set $r$ et $r'$. Nous avons la relation suivante : $M_{\mathcal{G}_{r''}} = M_{\mathcal{G}_r} \wedge M_{\mathcal{G}_{r'}}$*

Ainsi, pour le DG-set $[\{Ville^{\leq}, CSP^{\geq}\}, \geq]$, nous obtenons la matrice du tableau 2a. Nous remarquons que le bloc $b_3$ est isolé : il n'est en relation avec aucun autre bloc. La gradualité ne pouvant être mesurée qu'à partir de relation, les blocs isolés sont supprimés, et nous conservons la matrice du tableau 2b.

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

(a)

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

(b)

TAB. 2: Matrice binaire pour (a) $[\{Ville^{\leq}, CSP^{\geq}\}, \geq]$ et (b) $[\{Ville^{\leq}, CSP^{\geq}\}, \geq]$

## 4.3 Un algorithme glouton pour le calcul de la fréquence

Le calcul de la fréquence est effectué une fois par nœud et permet de vérifier si le DG-set associé respecte le seuil minimal fourni par l'utilisateur. Dans le cadre d'un algorithme par niveau, cette opération est effectuée un nombre de fois exponentiel, c'est pourquoi il est primordial qu'elle soit exécutée rapidement. Les matrices binaires n'énumèrent qu'une seule fois les blocs. C'est pourquoi le diagramme de Hasse généré possède des cycles et plusieurs chemins par bloc (un bloc appartient à plusieurs listes ordonnées pour un même DG-set). L'algorithme utilisé est donc un algorithme glouton récursif, qui ne parcourt qu'une seule fois chaque élément d'une matrice binaire (algorithme 3).

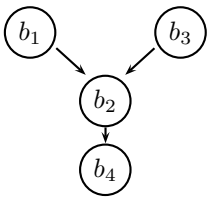
**Algorithme 3** : CalculFrequence

**Entrées** : Un bloc $b$
La mémoire du pas précédent $Memory$
**Sorties** : $Memory$ remplie

```
1  Fils ← GetFils(b)                    /* tous les b' mis à 1 dans la ligne b  */
2  si Sons = ∅ alors
3  |  Memory[node] = 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[node] = max(Memory[node], Memory[i] + 1)
12 |  fin
13 fin
```

Par exemple, la matrice du tableau 1a peut être représentée sous la forme d'un diagramme de Hasse, comme le montre la figure 3a. Dans cette représentation, pour $b \lhd_r b'$, le nœud $b$ est placé au dessus du nœud $b'$ et une flèche est tracée du nœud $b$ au nœud $b'$.

La mémoire pour l'algorithme 3 est initialisée à -1 pour les nœuds $b_1$ à $b_4$ (comme le montre le tableau 3b), puis est lancé sur les racines, $b_1$ et $b_3$. La passe sur $b_1$ appelle récursivement l'algorithme sur $b_2$ puis sur $b_4$. A la fin de cette passe, la mémoire est dans l'état montré au tableau 3c. L'algorithme est ensuite appelé su $b_3$, qui ne se lance pas récursivement, puisque le fils $b_2$ a déjà été renseigné. $b_3$ prend alors la valeur 3, et la fréquence de $[V^{\leq}, \geq]$ est de 3.



(a)

| $b_1$ | $b_2$ | $b_3$ | $b_4$ |
|---|---|---|---|
| -1 | -1 | -1 | -1 |

(b)

| $b_1$ | $b_2$ | $b_3$ | $b_4$ |
|---|---|---|---|
| 3 | 2 | -1 | 1 |

(c)

TAB. 3: (a) Diagramme de Hasse pour $[V^{\leq}, \geq]$ (b) Première passe de l'algorithme 3, (c) Seconde passe de l'algorithme 3

# 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 différents ($|V|$) ainsi que le nombre de blocs ($|\mathcal{B}|$). Nous avons généré trois fichiers :

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

TAB. 4: Spécifications des jeux de tests

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 posédant 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 afin d'extraire environ 250 DG-sets. L'algorithme est particulièrement sensible aux nombre de blocs, plus qu'au nombre de dimensions et de valeurs de dimensions. C'est ce que montrent les figures 4a, 5a et 6c. 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 nécessaire 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.

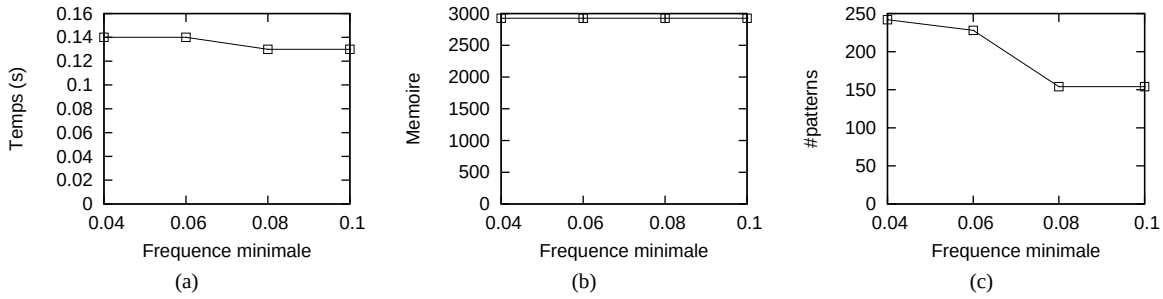



FIG. 4: Pour D5V10B40, en fonction du support (a) Temps d'exécution, (b) Mémoire utilisée, (c) Nombre de DG-sets extraits

# 6 Conclusion

Dans cet article, nous proposons une approche originale permettant d'extraire, à partir de cubes de données, des règles graduelles de la forme *Plus la ville est de taille importante et la CSP de catégorie supérieure, plus le nombre de produits vendus 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

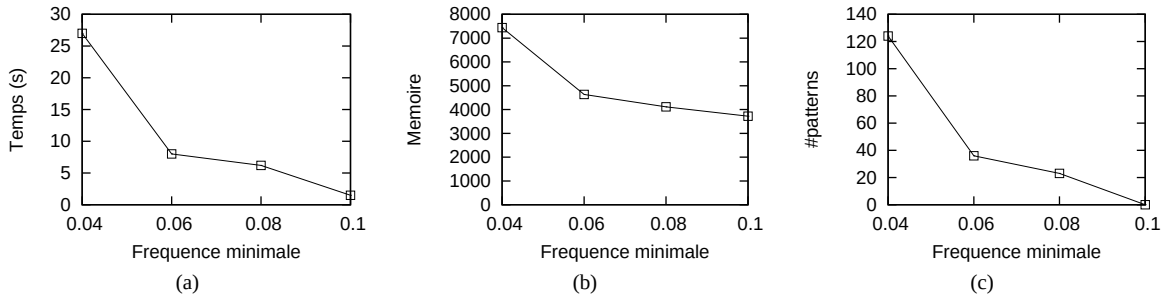



FIG. 5: Pour D10V100B500, en fonction du support (a) Temps d'exécution, (b) Mémoire utilisée, (c) Nombre de DG-sets extraits

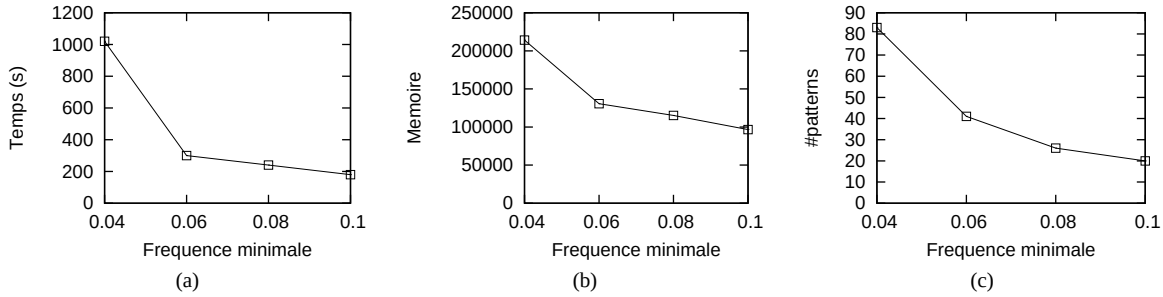



FIG. 6: Pour D10V100B5000, en fonction du support (a) Temps d'exécution, (b) Mémoire utilisée, (c) Nombre de DG-sets extraits

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'exécutions.

Dans nos futurs travaux, nous intégrerons la notion de hiérarchies de dimensions afin de retrouver les meilleures règles graduelles possibles (pertinence, support), et étudierons des méthodes étendues de calcul du support avec notamment la prise en compte de la confiance associée aux blocs et le relâchement de la notion d'ordre entre les blocs (en particulier pour prendre en compte les recouvrements).

# Références

Agrawal, R. et R. Srikant (1994). Fast Algorithms for Mining Association Rules. In *20th International Conference on Very Large Data Bases, (VLDB'94)*, pp. 487–499.

Ayres, J., J. Flannick, J. Gehrke, et T. Yiu (2002). Sequential pattern mining using a bitmap representation. In *KDD '02 : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, pp. 429–435. ACM.

Berzal, F., J.-C. Cubero, D. Sanchez, M.-A. Vila, et J. M. Serrano (2007). An alternative approach to discover gradual dependencies. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS) 15*(5), 559–570.

Chen, Y. et J. Pei (2006). Regression cubes with lossless compression and aggregation. *IEEE Trans. on Knowl. and Data Eng. 18*(12), 1585–1599.

Choong, Y., A. Laurent, et D. Laurent (2008). Mining multiple-level fuzzy blocks from multidimensional data. *Fuzzy Sets and Systems 159*(12).

Choong, Y., P. Maussion, A. Laurent, et D. Laurent (2004). 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)*, pp. 99–106.

Choong, Y. W., A. Laurent, et D. Laurent (2007). Summarizing data cubes using blocks. In P. Poncelet M. Teisseire F. Masseglia (Ed.), *Data Mining Patterns : New Methods and Applications*, pp. 36. IDEA Group Inc.

Choong, Y. W., D. Laurent, et P. Marcel (2003). Computing appropriate representations for multidimensional data. *Data Knowl. Eng. 45*(2), 181–203.

Di-Jorio, L., A. Laurent, et M. Teisseire (2008). Fast extraction of gradual association rules : A heuristic based method. In *International Conference on Soft Computing as Transdisciplinary Science and Technology (CSTST'08)*, Volume 1.

Di-Jorio, L., A. Laurent, et M. Teisseire (2009). Extraction efficace de règles graduelles. In *9èmes journées d'Extraction et Gestion des Connaissances*, pp. 199–204.

Hüllermeier, E. (2002). Association rules for expressing gradual dependencies. In *PKDD '02 : Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, London, UK, pp. 200–211. Springer-Verlag.

Pei, J., J. Liu, et K. Wang (2006). Discovering frequent closed partial orders from strings. *IEEE Trans. on Knowl. and Data Eng. 18*(11), 1467–1481.

Plantevit, M., S. Goutier, F. Guisnel, A. Laurent, et M. Teisseire (2007a). Mining unexpected multidimensional rules. In *DOLAP*, pp. 89–96.

Plantevit, M., A. Laurent, et M. Teisseire (2007b). Extraction d'outliers dans des cube de données : une aide à la navigation. In Revue des Nouvelles Technologies de l'Information (Ed.), *EDA'07 : Entrepots de Données et Analyse en ligne*, Poitiers, pp. 113–130.

## Summary

Coupling data mining and data warehousing allows for discovering relevant information from data cubes. In this Framework, several methods have been proposed, aiming for instance at discovering association rules or sequential patterns. However, no method has been proposed to discover gradual rules from such multidimensional databases. In this paper, we thus propose to discover correlations between a set of ordered dimensions with the measure evolution. Such rules are like "the higher the city size and the higher the Professional category, the higher the number of products sold". In order to efficiently build these rules and to take the major tendencies into account, we rely on the approach developped to discover blocks of homogeneous measure value.