Approche binaire pour la génération des fortes règles d'association

Thabet Slimani, Boutheina Ben Yaghlane, Khaled Mellouli

LARODEC, IHEC Carthage, Carthage Présidence 2016, Tunisia
thabet.slimani@issatm.rnu.tn
boutheina.yaghlane@ihec.rnu.tn
khaled.mellouli@ihec.rnu.tn

Résumé. Dans ce papier, nous proposons une nouvelle méthode d'extraction des règles d'association dans des bases de données relationnelles basée sur la technologie des *arbres de Peano* (Ptree). La structure de données utilisée pour représenter la base de données est un ensemble de Ptrees de base représentant chacun un vecteur binaire et tous ces Ptrees sont stockés dans des fichiers binaires. Nous montrons que la structure Ptree combinée avec la technique de réduction appelée *élagage par support minimum* produisent des règles d'association fortes et réduisent considérablement le temps de construction de l'association. En effet, notre approche présente l'avantage de ne pas effectuer des parcours coûteux de la base de données. Cette approche a été testée à travers un prototype que nous avons implémenté. Les résultats expérimentaux montrent que les règles d'association fortes sont générées dans un temps minimum comparativement à d'autres travaux.

Keywords : Fouille de données, Règle d'association, Itemsets fréquents, Arbres de Peano (Ptree), Règles ANDing.

1 Introduction

L'extraction de connaissances à partir des données ECD (KDD en anglais pour "Knowledge Discovery in Data") constitue un thème important du domaine de la fouille de données. Elle consiste en la recherche des connaissances implicites, cachées dans un ensemble volumineux d'informations stockées dans des bases de données. L'extraction des règles d'association est l'un des principaux problèmes traités de l'ECD. Ce thème a été initialement abordé par (Agrawal et al 1993) pour l'analyse des bases de données de transactions de ventes où chaque transaction est constituée d'une liste d'articles achetés (*items*), le but étant de découvrir les groupes d'articles (*itemsets*) achetés le plus fréquemment ensembles. Une *règle d'association* est une relation d'implication X→Y entre deux itemsets disjoints X et Y. Dans l'exemple du panier de la ménagère, cette règle indique que les transactions qui contiennent les items de l'ensemble X ont tendance à contenir les items de l'ensemble Y. Il existe deux mesures de qualié associées à chaque règle, *confiance* et *support*. Une règle apparaît dans l'ensemble de transactions D avec une confiance c si c% des transactions dans D qui contiennent X contiennent aussi Y. Une règle d'association X→Y a comme support s si s% des transactions dans D contiennent X∪Y. Une *règle d'association forte* est une règle dont le support

dépasse un seuil du support minimum (*MinSup*) et dont la confiance dépasse un seuil de confiance minimale (*MinConf*). Les seuils du support minimum et de la confiance minimale doivent être fixés d'avance par l'expert.

L'extraction des règles d'association à partir de larges bases de données est une tâche cruciale, la principale application est "l'analyse du panier de la ménagère", mais aujourd'hui, cette technique a fait l'objet de plusieurs applications réelles dans différents domaines nécessitant la recherche des groupements potentiels de produits ou de services, par exemple secteur bancaire, secteur de télécommunications, secteur médical pour la recherche de complications dûes à des associations de médicaments, ou encore traitement d'images (données spatiales), analyse des accès internet, ... Néanmoins, cette tâche est coûteuse en temps de calcul car elle consiste à faire plusieurs parcours de la base de données relationnelle pour l'identification des *itemsets fréquents*[1]. Les algorithmes de découverte de connaissances sont devenus inefficaces devant l'abondance de données d'où la nécessité d'utiliser des algorithmes rapides.

C'est dans ce cadre que se situe notre travail qui a pour objectif de proposer une nouvelle méthode permettant d'extraire des règles d'association fortes à partir des bases de données volumineuses. En effet, étant donnée une base de données relationnelle représentée sous forme d'un ensemble de tuples appelés aussi ensemble d'attributs[2] de nature diverse (des attributs de domaine binaire ou non binaire), nous proposons d'abord de convertir les données non binaires de la base relationnelle originale en une table binaire (Bitmap). Cette transformation est une caractéristique de la méthode *rough set* décrite dans (Munkata 1998). Ensuite, nous utilisons une structure de données, appelée *arbre de Peano* (Ptree), pour stocker, d'une façon compacte, toutes les informations utiles. Les Ptrees sont utilisés, dans ce cadre, pour étendre les opérations *ANding* (Ding et al. 2002b) des Ptrees sur les attributs de la base de données. En utilisant Ptrees, nous n'avons pas à faire des parcours coûteux de la base de données pour calculer les supports des itemsets puisque la base de données, comme nous l'avons déjà mentionné, est chargée dans un fichier binaire, et par conséquent l'algorithme B-ARM (en anglais "Binary Association Rule Mining"), que nous proposons, permettra d'extraire les itemsets fréquents sur divers types de bases de données en un temps minimum comparativement à d'autres travaux.

L'organisation de ce papier est la suivante : la Section 2 présente un état de l'art des travaux effectués sur les règles d'association. La Section 3 est une présentation sommaire de la technique Ptree, ensuite la section 4 est consacrée pour les caractéristiques des attributs d'une base de données. Dans la Section 5, nous présentons le principe et le modèle de la génération des règles d'association. La conversion des attributs aux Ptrees est abordée dans la Section 6. Notre algorithme de génération des règles d'association est présenté dans la Section 7 et les résultats expériementaux de notre prototype implémenté dans la Section 8. Enfin, dans la Section 9, nous concluons par quelques voies de recherche.

[1]Un *itemset* est dit "*fréquent*" si son support est supérieur à un seuil correspondant au support minimum (MinSup) spécifié préalablement par l'expert.

[2]Dans le cas des applications du panier de la ménagère, ces tuples sont représentés par un ensemble de transactions ou ensemble d'items.

2 Etat de l'art

Les travaux effectués sur les règles d'association dans le domaine de fouille de données sont très nombreux. L'algorithme de base Apriori (Agrawal et Srikant 1994) est l'algorithme le plus populaire, il a été développé pour la découverte des règles d'associations dans des bases de données larges. Une extension de l'algorithme Apriori, appelée algorithme DHP (Direct Hashing and Pruning), est proposée dans (Park et al 1995) utilisant la technique de hachage. Un algorithme plus récent appelé FDM (Fast Distributed Mining of association rules) a été proposé par (Cheung et al 1996), il est caractérisé par la génération d'un plus petit ensemble d'itemsets candidats et par la réduction du nombre de messages à passer. Pincer Search (Lin et Kedem 1998) étend l'algorithme Apriori pour générer les itemsets fréquents. Depth-project (Agarwal et al 2000) parcourt le treillis des itemsets en profondeur et utilise un réordonnancement dynamique afin de réduire l'espace de recherche. Un travail assez récent réalisé par (Delic et al 2002) procède à l'amélioration de la qualité des règles d'association par des techniques rough sets. Enfin, plus proches de notre travail, d'une part, l'algorithme FP-growth qui représente la base de transactions sous forme d'un arbre compressé appelé *FP-tree* (Han et al 2000) et d'autre part, l'algorithme MFItemsets (Maximal Frequent Itemsets) qui représente la base des transactions sous forme d'un arbre binaire et renvoie un ensemble de produits booléens correspondants aux itemsets fréquents maximaux associés aux transactions données (Salleb et Maazouzi 2002).

3 La technique Ptree

La structure de données *arbre de Peano* (Ptree), appelée aussi "*Peano Count Tree*" est une représentation compacte et efficace utilisée pour le stockage de la base de données sous forme de bits binaires (0 et 1). Cette structure a été initialement introduite pour la représentation de données spatiales telles que les données des applications RSI (*Remotely Sensed Imagery*) (Perrizo et al 2001a, Ding et al 2002a, Ding et al 2002b). Un Ptree est un arbre basé sur des *quadrants*. Le principe de Ptree consiste à diviser récursivement la totalité des données spatiales en des quadrants et à compter les bits 1 pour chaque quadrant, en formant de ce fait un arbre de calcul de quadrants. Dans la figure 1, 55 est le nombre de bits 1 dans l'image entière, Ce niveau racine est étiqueté de niveau 0. Les nombres 16, 8, 15 et 16 du niveau suivant (niveau 1) sont les nombres de bits 1 pour les quatre principaux quadrants dans l'ordre Z de la trame (supérieur-gauche, supérieur-droite, bas-gauche, bas-droite). Puisque les premiers et les derniers quadrants de niveau 1 se composent entièrement de 1-bits (appelés *quadrants pure-1*), les sous-arbres ne sont pas nécéssaires et ces branchements se terminent. De même, des quadrants composés entièrement de 0-bits s'appellent des *quadrants pure-0* et causent également l'arrêt des banchements. Ce processus se répète récursivement en utilisant l'ordre Z des quatre sous-quadrants à chaque nouveau niveau. Eventuellement, chaque niveau se termine puisqu'au niveau "feuille" tous les quadrants sont purs.

Les Ptrees sont un peu semblables, dans leur construction, à d'autres structures de données dans la littérature, par exemple Quadtrees (Samet 1984) et HHcodes[3]. Lors de

[3]http ://www.statkart.no/nlhdb/iveher/hhtext.html

l'utilisation de la structure Ptree, tout calcul de l'information peut être effectué d'une manière accélérée. L'analyse de performance effectuée dans (Ding et al 2002a) montre que Ptree produit un coût de calcul faible (en temps microprocesseur) et un espace de stockage réduit comparé avec les données originales.

Un PM-tree (*Peano mask tree*) est une variante de Ptree, particulièrement utile pour l'optimisation de l'opération *ANDing* entre deux Ptrees. PM-tree consiste à utiliser une logique de 3-valeurs, dans laquelle 1 est employé pour représenter un quadrant pure-1, 0 est employé pour représenter un quadrant de pure-0 et m est utilisé pour représenter un quadrant mixte. Un exemple de PM-tree est donné dans la figure 1.

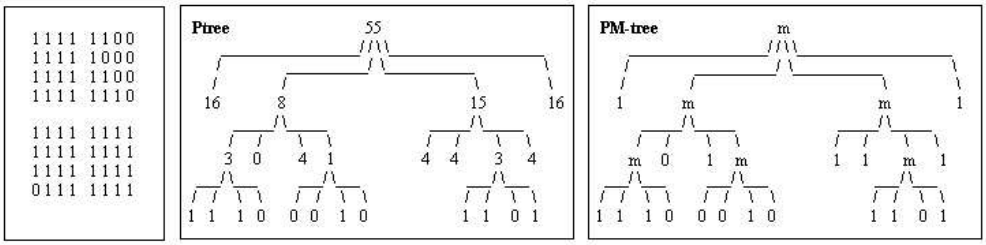

FIG. 1 – Image 8*8 et ses Ptree et PM-tree (Perrizo et al 2001b).

4 Caractéristique de la méthode

Une base de données est représentée par une table binaire ou table Bitmap dont les colonnes sont des attributs et chaque attribut possède un ensemble limité de valeurs (items) connues par domaine d'attribut de la base de données. La base de données peut avoir deux types de domaines : des attributs de domaine binaire DB et des attributs de domaine non binaire NDB.

4.1 Les attributs de domaine binaire

Un attribut de domaine binaire est représenté par un vecteur $\overrightarrow{v} \subset \{v1, v2\}$ de taille k tels que les valeurs $v1$ et $v2$ sont prises dans l'ensemble {0,1} et k est le nombre des k-uplet de valeurs prises dans {0,1}. Un 1-uplet représente un tuple de la base de données ou une transaction en terme du panier de la ménagère.

Une base de données A^n de dimension n est constituée de n vecteurs binaires, chacun de taille 2^n et chaque vecteur est constitué à son tour de 4 vecteurs binaires de taille $2^n/4$ (pour des raisons de simplicité, nous avons décomposé chaque vecteur binaire en 4 quadrants). Les lignes de A^n repésentent alors, toutes les combinaisons possibles de n valeurs binaires 0 et 1. Dans l'exemple donné dans TAB.1, la présence d'un article *ordinateur* dans une transaction ou son absence représente son domaine {acheté, non acheté} et la transformation binaire rend la valeur de l'attribut a_1=1 si l'ordinateur est acheté ou a_1=0 si l'ordinateur n'est pas acheté.

Tid	$Ordinateur$		Tid	a_1
1	$Acheté$		1	1
2	$Non\ acheté$	$\Longrightarrow$	2	0
3	$Acheté$		3	1
4	$Non\ acheté$		4	0
...	...		...	...

TAB. 1 – Transformation des données brutes en une représentation Bitmap pour des attributs de domaine binaire

4.2 Les attributs de domaine non binaire

Un attribut de domaine non binaire A^j est constitué de j items et la base de données est représentée de $\sum_{i=1}^{n} j * i$ vecteurs binaires tel que n est le nombre d'attributs du domaine non binaire (NDB). A titre d'exemple, pour une meilleure représentation du profit d'un client, on associe à l'attribut *revenu* le domaine constitué de $j = 3$ items {haut, moyen, faible} défini comme suit : a_1="revenu haut", a_2="revenu moyen" et a_3="revenu faible" et représenté par la table binaire suivante :

Tid	$Revenu$		Tid	a_1	a_2	a_3
1	$Haut$		1	1	0	0
2	$Moyen$	$\Longrightarrow$	2	0	1	0
3	$Faible$		3	0	0	1
4	$Haut$		4	1	0	0
...	...		...	...	...	...

TAB. 2 – Transformation des données brutes en une représentation Bitmap pour des attributs de domaine non binaire.

5 Génération des règles d'association

5.1 Principe

Etant donnée une base de données composée de n attributs associés à m tuples, l'idée principale de notre algorithme s'appuie sur l'algorithme Apriori (Agrawal et Srikant 1994) qui consiste à balayer la base de données pendant des passages multiples dans le but de rechercher des itemsets fréquents. Après chaque passage, le nombre des items dans les sous-ensembles des itemsets larges s'incrémente de 1 jusqu'à ce que les ensembles possibles dans la base de données sont construits, ensuite, l'algorithme utilise les larges itemsets pour générer les règles d'association. L'algorithme Apriori s'arrête lorsqu'aucun itemset large ne peut être crée. Ainsi, pour un nombre élevé d'items n et un nombre de tuples m très grand, on peut associer 2^n combinaisons possibles qui augmente le temps de calcul puisque le calcul s'effectue directement sur la base de données, d'où la nécessité d'architectures plus efficace qui permettent des accès rapides à de grands jeux de données.

Pour cela, nous avons adopté la technique Ptree dont l'avantage est de ne pas effectuer des parcours coûteux sur la base de données car les données originales sont converties en données spatiales (ou binaires), puis stockées dans un fichier binaire. En utilisant l'organisation REL (représentation de la base de données sous forme d'une table Bitmap), l'idée de Ptree consiste, d'une part, à diviser récursivement les données spatiales dont les colonnes de la table Bitmap forment chacune une bande de bits et d'éviter, d'autre part, de comparer à chaque fois les items 1 par 1 et de procéder, au contraire, à la comparaison des blocs de tuples, ce qui diminue le temps de calcul.

5.2 Modèle

La génération des règles d'association est modélisée sur la base du modèle *rough set* (Munkata 1998), dont l'univers Ω des attributs de la table Bitmap est divisé en deux ensembles disjoints, un ensemble constituant les *attributs de condition* Ω_c et un deuxième ensemble constituant les *attributs de décision* Ω_d.

Soit la TAB. 3 où les attributs de l'ensemble des données sont représentés par {X}, {Y} et {Z}. L'attribut X a deux valeurs {A et B}= {Compte débité, Compte nul}, l'attribut Y a trois valeurs {C, D et E}={revenu faible, revenu haut, revenu moyen} et l'attribut de décision Z a deux valeurs {F, G}={accorder prêt, ne pas accorder prêt}. Il y en a 7 items pour la table Bitmap résultante {A, B, C, D, E, F et G}.

Tidclient	*Compte*	*Revenu*	*Accorder prêt*		*Tid*	*A*	*B*	*C*	*D*	*E*	*F*	*G*
1	*Débité*	*Moyen*	*Oui*		1	1	0	0	0	1	1	0
2	*Débité*	*Faible*	*Non*		2	1	0	1	0	0	0	1
3	*Débité*	*Moyen*	*Oui*	$\Longrightarrow$	3	1	0	0	0	1	1	0
4	*Débité*	*Haut*	*Oui*		4	1	0	0	1	0	1	0
5	*Nul*	*Haut*	*Non*		5	0	1	0	1	0	0	1

TAB. 3 – Table de données originales et son équivalente Bitmap

Dans la première étape, toutes les règles possibles sont construites à partir de tous les attributs de la table Bitmap. Par exemple, pour X→Y ; 6 règles différentes peuvent être formées. Elles sont obtenues par le produit cartésien des items des deux attributs X et Y {A→C, A→D, A→E, B→D, B→E}. Le choix d'une règle d'association est basé sur les quantités numériques, support et confiance, définies par l'utilisateur, qui vont servir à valider l'interêt d'une telle règle, ainsi toutes les règles qui ne satisfont pas les contraintes du support minimum et de la confiance minimale sont éliminées.

6 Des attributs aux Ptrees

Dans notre approche, chaque item est représenté par un Ptree. Un ensemble de tuples, selon la représentation Ptree, doit être une puissance de 2. Le nombre des tuples dans une base de données est transformé à la plus proche puissance de 2, sachant que les tuples complétés sont des itemsets incluant uniquement la valeur 0. La table des tuples est représentée d'une manière analogique à la représentation ensembliste sous formes d'itemsets. Soit I un itemset, la transformation binaire consiste à mettre la valeur 1 à

l'item I_j s'il appartient à $\tilde{I}$, et 0 sinon. Dans l'approche Ptree, chaque colonne de la table Bitmap est représentée par un vecteur de bits dont la taille est divisible par 4, appellé *Ptree de base*.

Considérons la base de données de la table 4. Soient les attributs de condition de l'espace Ω_c formés par les attributs {A, B, C, D, E, F} désignant l'ensemble des produits {cartouche imprimante, lecteur vidéo, voiture, ordinateur, caméra, imprimante} et l'attribut de décision de l'espace Ω_d formé par {G} désignant l'attribut {logiciel graphique}. Soit $MinSup$=10% et $MinConf$=30% représentant le support minimum et la confiance minimale. TAB.4 présente cette base de données convertie sous forme d'une table Bitmap.

T_{id}	A	B	C	D	E	F	G		T_{id}	A	B	C	D	E	F	G
T_{id1}	oui	oui	non	non	non	non	non		T_{id1}	1	1	0	0	0	0	0
T_{id2}	oui	oui	oui	oui	oui	oui	non		T_{id2}	1	1	1	1	1	1	0
T_{id3}	non	oui	non	oui	non	non	oui		T_{id3}	0	1	0	1	0	0	1
T_{id4}	non	oui	non	non	oui	non	oui	⇒	T_{id4}	0	1	0	0	1	0	1
T_{id5}	non	non	non	oui	non	oui	oui		T_{id5}	0	0	0	1	0	1	1
T_{id6}	non	non	non	oui	oui	non	oui		T_{id6}	0	0	0	1	1	0	1
T_{id7}	non	oui	non	non	oui	non	non		T_{id7}	0	1	0	0	1	0	0
T_{id8}	non	oui	non	oui	oui	oui	non		T_{id8}	0	1	0	1	1	1	0

TAB. 4 – Exemple d'une base de données et sa table Bitmap correspondante

6.1 Construction des Ptrees

Pour des raisons de simplicité, nous supposons que le *fan-out* (les noeuds sortants) est quatre. Pour chaque vecteur de bits, on associe un Ptree de base. Il y en a six Ptrees de base pour l'univers Ω_c de la table 4, et puisque chaque Ptree présente un nombre de bits divisible par 4, donc, il est constitué par quatre sous-quadrants dont le quadrant d'origine est la totalité des bits formant un item de la table Bitmap. Il y a toujours n Ptrees de base pour une table Bitmap formée de n items. Comme nous l'avons déjà expliqué, l'idée de construction d'un Ptree consiste à diviser périodiquement le vecteur binaire (item) en des sous-quadrants et d'enregistrer le nombre des bits 1 pour chaque sous-quadrant en formant, de ce fait, un arbre de calcul des quadrants.

6.2 Stockage de Ptrees

D'une manière similaire à la génération de la séquence de Peano à partir des trames de données spatiales, on doit créer l'arbre Ptree d'une manière ascendente. La génération de Ptree dépend du nombre de *fan-out* dans les noueds internes de l'arbre Ptree et dans le noeud racine. Pour représenter Ptree avec différents *fan-out*s, on introduit la notation P-tree-(*r-i*) ; où r=*fan-out* du noeud racine et i=*fan-out* de tous les noeuds internes du niveau 1. On utilise dans notre travail la représentation P-tree-(4-4-n) càd on divise le nombre des tuples composant la base de données par 4 blocs (les blocs de transactions doivent avoir au minumum 4 tuples). Par exemple, si le nombre de tuples est inférieur à 16, on complète par des 0 pour obtenir le format Ptree sur 16 tuples. En règle générale, si le nombre de transactions est inférieur à 2^{n+1} et supérieur à 2^n alors le Ptree de base est stocké avec un nombre égal à 2^{n+1}.

6.3 Les opérations d'intersection (*ANDing*) des Ptrees

ANDing est une opération très importante et fréquemment utilisée pour les Ptrees. Il y a plusieurs voies d'exécuter l'opération *ANDing* de Ptree. Nous pouvons l'exécuter niveau par niveau à partir du niveau racine. TAB 5 donne les règles de l'opération *ANDing* de Ptrees. L'opérande 1 et l'opérande 2 sont deux Ptrees avec la racine X1 et X2 respectivement. En utilisant PM-trees, X1 et X2 pourraient être n'importe quelle valeur parmi 1, 0 et m. Par exemple, à l'opération *ANDing*, un Ptree des bits pures-1 avec n'importe quel Ptree aura comme conséquence la deuxième opérande et un Ptree des bits pures-0 avec n'importe quel Ptree aura comme conséquence le Ptree pure-0.

Opérande1	Opérande2	Résultat
1	X2	Sous-Arbre avec comme racine X2
0	X2	0
X1	1	Sous-Arbre avec comme racine X1
X1	0	0
m	m	4 si le résultat de 4 sous-quadrants = 0 ; m sinon

TAB. 5 – Les règles d'intersection des Ptrees.

7 Algorithme B-ARM (Binary Association Rule Mining)

Dans Apriori et la plupart des algorithmes ARM, la base de données doit être balayée entièrement pour calculer le support de chaque *itemset* candidat. Lorsque l'ensemble de transactions est large, le coût sera extrêmement élevé. Alors qu'à partir de l'opération *ANDing* des Ptrees, le support de chaque *itemset* candidat peut être obtenu directement, càd le compte du support est juste le compte de la racine de chaque Ptree de base, et par conséquent, il n'y a aucun besoin de balayer la base de données. En se basant sur cette idée, on a développé un nouveau algorithme, appelé *B-ARM*, pour la génération des règles d'association en utilisant des Ptrees. Comme pour l'algorithme Apriori, l'algorithme B-ARM sert à la découverte des *itemset*s fréquents.

Afin de générer des itemsets fréquents, l'algorithme B-ARM procède comme suit : Il commence par le stockage des Ptrees de base dans deux vecteurs, un vecteur pour le stockage des quadrants (les blocs des bits de chaque noeud interne) et un autre pour le stockage des noeuds racines, ensuite il s'agit de dégager les attributs fréquents dont le support est supérieur ou égal au support minimum spécifié par l'utilisateur. L'étape suivante consiste à itérer récursivement l'intersection par des opérations ANDing des Ptrees de base de la partie condition avec les Ptrees de base des attributs de décision, afin de dégager l'ensemble des nouveaux itemsets admissibles de la partie condition. Les règles sont générées après vérification de la confiance de la partie condition et la partie conclusion. L'algorithme s'arrête lorsqu'aucun itemset fréquent n'existe dans l'ensemble des attributs candidats C_k.

Algorithm *B-ARM*
Discrétisation de données ;
Stockage_Ptrees ;
Pour chaque attribut i $\subset \Omega_c$
    $C_1 = F_1$;
FinPour
$\mathbf{C_k} = C_1$;
Tant que ($\mathbf{C_k} \neq \emptyset$) **faire**
    Pour chaque attribut i $\subset \mathbf{C_k}$
        Pour chaque attribut j $\subset \Omega_d$
            F_{ij} = AND_Ptreebase(i,j) ;
            Stockage_Ptrees ;
        FinPour
        $F_k = F_k \cup F_i$; // *itemsets candidats*
    FinPour
    $\mathbf{C_k} = F_k\{c \in C_k |\ c.count >= MinSup\}$;
FinTantque

Dans la procédure de stockage de Ptree, on stocke les noeuds racines de Ptree dans un vecteur et les noeuds binaires (formant les sous-quadrants) dans un autre vecteur afin de faciliter l'accès aux données des sous-quadrants de chaque *bande* (colonne de la table Bitmap), en vue de les comparer pour vérifier les associations entre ces *bande*s. Soit N_t = nombre de tuples ; n est initialisé à 3 ; I désigne le nombre total des attributs.

Procédure *Stockage_Ptrees*
Pour (bandej=1 ; j<I ; j++)
    racine[j] :=rootcount(1 ; bandej)//vecteur qui stocke les racines des Ptrees.
    Si($2^n \leq N_t$ et $N_t < 2^{n+1}$) **alors**
        Pour (i :=N_t ; i$\leq 2^{n+1}$; i++)
            bandj[i+1] :=0 ;
        Finpour
    Finsi
    k :=0 ;
    Pour (i :=1 ; i$\leq 2^n; 2^n/4$) // calcul des racines ou sous-quadrants
        k :=k+1 ;
        rootsBandj[k] :=rootcount (1 ; bandj) ;// vecteur racines internes
    FinPour
    Pour (i :=1 ; i$\leq 2^n; 2^n/4$)
        Si (rootsBandj[i]<>2^n **ou** rootsBandj[i]<>0) **alors**
            bitsBandj[i] :=rootsBandj[i] ;//vecteur des bits
        Finsi
    Finpour
FinPour

La liste des règles fortes générées dans l'exemple décrit précédemment est résumée dans la table 6. Les règles sont classifiées par attribut de décision G ou F.

	Att décision = G	Att décision = F
Règles fortes à deux attributs	B→G D→G E→G F→G	A→F B→F C→F D→F E→F
Règles fortes à 3 attributs	B,D→G	A, B→F A,C→F A,D→F A,E→F B,C→F B,D→F C,D→F
Règles fortes à 4 attributs		A,B,C→F A,B,D→F A,B,E→F A,C,D→F B,C,D→F B,C,E→F B,D,E→F C,D,E→F
Règles fortes à 5 attributs		A,B,C,D→F A,B,C,E→F A,C,D,E→F B,C,D,E→F
Règles fortes à 6 attributs		A, B,C,D,E→F

TAB. 6 – Schéma illustratif des règles fortes retenues.

8 Implémentation et exprémentation

Nous avons développé un prototype de génération des règles fortes. La structure de données utilisée pour représenter la base de données (BD) est un ensemble de Ptrees de base représentant chacun un vecteur binaire de la table Bitmap et tous ces Ptrees sont stockés dans des fichiers binaires. Cette structure de données permet de charger complètement la BD volumineuse dans un fichier binaire. L'objectif de cette opération est d'éviter le balayage direct de la BD.

La transformation d'une BD en une représentation Bitmap et l'utilisation du schéma de production des règles d'association nécessite une comparaison objective de cette méthode avec d'autres travaux. La comparaison de notre travail est facilitée par l'utilisation d'un *benchmark data*[4]. On a choisi une BD (*car evaluation database*) à travers laquelle on a comparé notre travail. Cette BD comprend 1728 tuples et 25 valeurs d'attributs (items) dans la table Bitmap.

Nous avons comparé notre travail par rapport à deux approches Apr+ (Apriori+) et Rs+ (RS-Rules+) (Delic et al 2002). La procédure Apr+ est une extension de la méthode "*faster association rule*" combinée avec la procédure "*rough set*". Cette procédure utilise un attribut de décision fixe comme le principe de notre travail, mais, en plus de cela, notre travail ajoute la notion de la technique Ptree pour accélérer le temps de génération des règles d'association admissibles uniquement et utilise à chaque étape la partie condition de ces mêmes règles pour générer d'autres règles. Les règles dérivées sont produites sur la base de la réduction successive des règles inutiles.

[4]Le "benchmark data" se trouve dans "*UCI Repository of Machine Learning Database and Domain Theories*"(URL : ftp.ics.uci.edu/pub/machine learning-databases/Adult, /car, /mushroom).

Dans notre travail, on ne produit pas des règles redondantes, alors que dans Rs+ et Apr+, des règles redondantes sont produites et enlevées par la suite. Les premièrs résultats, non définitifs, donnés dans TAB.7 montre que notre travail produit un temps de génération des règles d'association égale à 0,083 min pour des attributs ayant comme valeurs d'items égale à 1 et un temps égale à 0,067 min pour des attributs ayant comme valeurs d'items égale à 0 qui est un temps réduit par rapport à Rs+ et Apr+. Notons que notre approche produit les mêmes règles générées par les méthodes Rs+ et Apr+.

Database	Car Evaluation					
Support minimum	0,10					
Confiance minimale	0,75					
Attribut de décision	1			0		
Méthode	Rs+	Apr+	B-ARM	Rs+	Apr+	B-ARM
Temps CPU (min)	1,15	1,12	0,083	1,10	3,15	0,067

TAB. 7 – Tableau comparatif des algorithmes de génération des règles d'association

9 Conclusion

La méthode de génération des règles d'association de notre algorithme B-ARM est basée sur l'élagage par confiance minimale et support minimum. La découverte des similarités entre des attributs/items a été basée sur des règles de comparaison des Ptrees. Notre travail est avantageux car, d'une part, il évite le balayage direct de la base de données qui est une opération coûteuse en mémoire et en temps de calcul qui dépasse largement la capacité des ordinateurs, malgré leurs évolutions rapide et, d'autre part, il offre un gain de comparaison des attributs/items car la comparaison s'effectue par bloc des tuples de la base de données.

Comme extension de ce travail, nous comptons utiliser la même méthode pour générer des règles d'association dans un temps limité tout en ajoutant aux règles d'association la contrainte du temps (domaine nouveau pour l'identification des séquences).

Références

Agarwal R.C., Aggarwal C.C. & Prasad V.V.V. (2000). Depth First Generation of Long Patterns. In *Proc. of the 6th Int. Conf. on Knowledge Discovery and Data Mining*, pp. 108-118.

Agrawal R., Imielinski T. & Swami A.N. (1993). Mining Association Rules between Sets of Items in Large Databases. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 207-213.

Agrawal R. & Srikant R. (1994). Fast Algorithms for Mining Association Rules. In *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, pp. 487-499.

Cheung C., Han J., Ng V.T., Fu A.W. & Fu Y. (1996). A Fast Distributed Algorithm for Mining Association Rules. In *Proc. of 1996 Int'l Conf. on Parallel and Distributed Information Systems (PDIS'96)*, Miami Beach, Florida, USA.

Delic D., Lenz L. & Neiling N. (2002). Improving the Quality of Association Rule Mining by means of Rough Sets. Free university of Berlin, Institute of Applied Computer Science, Garystr. 21, D-14195, Berlin, Germany.

Ding Q., Ding Q. & Perrizo W. (2002a). Association Rule Mining on Remotely Sensed Images using P-trees. In *Proceedings of the PAKDD*, Taipei, Taiwan, pp. 66-79.

Ding Q., Khan M., Roy A. & Perrizo W. (2002b). The Ptree Algebra. In *Proc. of ACM Symposium on Applied Computing (SAC'02)*, Madrid, Spain, pp. 413-417.

Han J., Pei J. & Yin Y. (2000). Mining Frequent Patterns without Candidate Generation. In *Proc. 2000 ACM SIGMOD Intl. Conference on Management of Data*, Chen W., Naughton J. & Bernstein P.A. (Eds), ACM Press, pp. 1-12.

Lin D. & Kedem Z. M. (1998). Pincer Search : A New Algorithm for Discovering the Maximum Frequent Set. In *Proc. Int. Conf. on Extending Database Technology.*

Munkata T. (1998). Rough Sets. In *Fundamentals of the New Artificial Intelligence*, New York : Springer-Verlag, pp. 140-182.

Park J.S., Chen M.S. & Yu P.S. (1995). An Effective Hash-based Algorithm for Mining Association Rules. In *Proc. 1995 ACM SIGMOD International Conference on Management of Data*, pp. 175-186.

Perrizo W., Ding Q., Ding Q. & Roy A. (2001a). On Mining Satellite and other Remotely Sensed Images. In *Proc. of Workshop on Research Issues on Data Mining and Knowledge Discovery*, pp. 33-44.

Perrizo W., Ding Q., Ding Q. & Roy A. (2001b). Deriving High Confidence Rules from Spatial Data using Peano Count Trees. In *Advances in Web-Age Information Management : Second International Conference* WAIM 2001, Wang X.S., Yu G. and Lu H. (Eds), Springer-Verlag, LNCS 2118, pp. 91-102.

Salleb A. & Maazouzi Z. (2002). Approche Booléenne pour l'extraction des itemsets fréquents maximaux. In *Conf. d'Apprentissage (CAp'02)*, Orléans, pp. 111-122.

Samet H. (1984). The Quadtree and Related Hierarchical Data Structures. In *Computing Surveys*, 16(2), pp. 187-260.

Summary

In this paper, we propose a new method for the association rule mining in relationnel data bases by the use of Peano tree (Ptree) technology. The data of the association rules model are organized in format REL which is a converted organization of the not-spatial data to spatial data (binary). A data base is seen as a Bitmap table whose each column represents an attribute and each row represents a tuple of data base. We show, in this work, that the Ptree structure, combined with the technique of reduction called pruning by minimum support, can produce strong rules by reducing considerably the time of association construction. Our prototype and experiments have shown that it is possible to produce a strong rules in a minimum time comparing to other works.