Extraction de Motifs sous Contraintes Quantifiées

Medhi Khiari, Arnaud Lallouet, Jérémie Vautard

Extraction de Motifs sous Contraintes Quantifiées

Mehdi Khiari[1] **Arnaud Lallouet**[1] **Jérémie Vautard**[2]

[1] GREYC (CNRS - UMR 6072) – Université de Caen Basse-Normandie
Boulevard du Maréchal Juin, 14000 Caen
[2] Chercheur indépendant
[1]{prenom.nom}@unicaen.fr [2]jeremie.vautard@gmail.com

Résumé

Au cours des dernières années, des approches d'extraction de motifs en fouille de données utilisant la PPC on été proposées. Ces approches ont montré leur utilité pour modéliser de manière flexible une large panoplie de contraintes, notamment les contraintes portant sur plusieurs motifs. Néanmoins, ces approches se basent sur les CSPs où toutes les variables sont quantifiées existentiellement. Or certaines requêtes n-aires (requêtes portant sur plusieurs motifs) requièrent la quantification universelle pour être modélisées de manière concise et élégante, comme par exemple la requêtes peak (un motif est considéré comme pic si tous ses voisins ont une valeur, par rapport à une mesure, inférieure à un seuil donné). Nous proposons dans cet article un cadre générique permettant la modélisation et la résolution de problèmes d'extraction de motifs sous contraintes quantifiées.

1 Introduction

L'Extraction de Connaissances dans les Bases de Données (ECBD) a pour objectif la découverte d'informations utiles et pertinentes répondant aux intérêts de l'utilisateur. L'extraction de motifs sous contraintes est un cadre proposant des approches et des méthodes génériques pour la découverte de motifs locaux [3]. Mais, ces méthodes ne prennent pas en considération le fait que l'intérêt d'un motif dépend souvent d'autres motifs et que les motifs les plus recherchés par l'utilisateur sont fréquemment noyés parmi une information volumineuse et redondante. C'est pourquoi la transformation des collections de motifs locaux en modèles globaux tels que les classifieurs ou le clustering [16, 10] est une voie active de recherche et la découverte de motifs sous contraintes portant sur des combinaisons de motifs locaux est un problème majeur. Dans la suite, ces contraintes sont appelées *contraintes n-aires* et les motifs concernés, *motifs n-aires*.

Il y a encore quelques années, peu de travaux concernant l'extraction de motifs n-aires ont été menés et les méthodes développées sont toutes ad hoc [23, 18]. Ceci est expliqué par la difficulté de la tâche due à l'ampleur de l'espace de recherche lorsqu'il s'agit d'extraire des motifs sous contraintes n-aires. Ce manque de généricité est un frein à la découverte de motifs pertinents et intéressants car chaque contrainte n-aire entraîne la conception et le développement d'une méthode ad hoc.

Au cours des quatre dernières années, des approches génériques d'extraction de motifs sous contraintes n-aires sont proposées [14, 11, 19]. Ces approches utilisent différentes méthodes de résolution. [14] utilise uniquement un solveur de CSP en proposant une modélisation sous forme de CSP du problème d'extraction de motifs sous contraintes n-aires. [11] propose un solveur dédié à l'extraction de motifs dont les méthodes de filtrage et de propagation sont inspirés d'algorithmes issus de la communauté PPC notamment AC-5 [12]. Enfin, [19] propose une méthode d'extraction utilisant un solveur SAT. Ces différentes approches sont associées à des langages de contraintes permettant à l'utilisateur de spécifier ses requêtes de manière déclarative. Ce dernier a ainsi plus de facilité à cibler des motifs de haut niveau plus faciles à interpréter.

Certaines requêtes n-aires (i.e., requêtes comportant des contraintes n-aires) requièrent en plus la quantification universelle pour être modélisées de manière concise et élégante, comme par exemple la requêtes *peak* [7] (un motif est considéré comme pic si tous ses voisins ont une valeur, par rapport à une mesure, inférieure à un seuil donné). On ne trouve dans la littérature que la description de la requête sans méthode de résolution associée. Ce type de requêtes n'est pas

Trans.	Items					
t_1	A	B	C		c_1	
t_2	A	B		D	c_1	
t_3	A	B		D	c_1	
t_4	A	B			c_1	
t_5		B		D		c_2
t_6	A	B	C	D		c_2
t_7	A					c_2

TABLE 1 – Jeu de données exemple

supporté par les méthodes génériques précédemment citées.

Nous proposons dans cet article un cadre générique permettant la modélisation et la résolution de problèmes d'extraction de motifs sous contraintes quantifiées.

L'article est organisé comme suit : la section 2 introduit le cadre général de l'extraction de motifs sous contraintes et la nécessité de développer des méthodes permettant l'extraction de motifs de haut niveau. Nous y présentons aussi deux requêtes n-aires d'extraction de motifs sous contraintes quantifiées. La section 3 trace un état de l'art des approches d'extraction de motifs utilisant la PPC et en particulier l'approche *Pure-CP*. La section 4 présente les cadre des QCSP et des QCSP$^+$. Les sections 5 et 6 proposent respectivement la nouvelle approche que nous proposons et l'étude expérimentale associée.

2 Extraction de motifs sous contraintes

2.1 Contexte et définitions

Soit $\mathcal{I}$ un ensemble de littéraux distincts appelés *items*, un motif ensembliste [1] d'items correspond à un sous-ensemble non vide de $\mathcal{I}$. Ces motifs sont regroupés dans le langage $\mathcal{L}_\mathcal{I} = 2^\mathcal{I} \backslash \emptyset$. Un contexte transactionnel est alors défini comme un multi-ensemble de motifs de $\mathcal{L}_\mathcal{I}$. Chacun de ces motifs, appelé transaction, constitue une entrée de la base de données.

Ainsi, le tableau 1 présente un contexte transactionnel $\boldsymbol{r}$ où 7 transactions étiquetées $t_1, \ldots, t_7$ sont décrites par 6 items $A, \ldots, D, c_1, c_2$.

L'extraction de motifs a pour but la découverte d'informations à partir de tous les motifs ou d'un sous ensemble de $\mathcal{L}_\mathcal{I}$. L'extraction sous contraintes cherche la collection de tous les motifs de $\mathcal{L}_\mathcal{I}$ présents dans $\boldsymbol{r}$ et satisfaisant un prédicat appelé *contrainte*. Ces motifs sont appelés *motifs locaux* ; ce sont des régularités observées dans certaines parties des données. La localité de ces motifs provient du fait que, vérifier s'ils satisfont une contrainte donnée, peut s'effectuer indépendamment des autres motifs.

Les notions de support et de fréquence d'un motif sont définies comme suit :

Définition 2.1 (Support, fréquence)
Une transaction $t \in \mathcal{T}$ supporte le motif X (ou X est présent dans t) ssi $X \subseteq t$ (ou $\forall i \in X, R(i,t) = 1$). Le ***support*** $\texttt{support}(X)$ *d'un motif X d'attributs est l'ensemble des transactions qui le supportent.*

Sa mesure de fréquence $\texttt{freq}(X)$ *est le cardinal du support.*

Il y a de nombreuses contraintes permettant d'évaluer la pertinence et la qualité des motifs locaux. Un exemple bien connu est celui de la contrainte de fréquence qui permet de rechercher les motifs X ayant une fréquence $\texttt{freq}(X)$ supérieure à un seuil minimal fixé $min_{\texttt{freq}} > 0$. De nombreux travaux [20] remplacent la fréquence par d'autres mesures permettant d'évaluer l'intérêt des motifs locaux recherchés. C'est le cas de la mesure d'aire : soit X un motif, $\texttt{area}(X)$ est le produit de la fréquence de X par sa taille, i.e., $\texttt{area}(X) = \texttt{freq}(X) \times \texttt{size}(X)$ où $\texttt{size}(X)$ désigne la longueur (i.e., le nombre d'items) de X.

En pratique, l'utilisateur est souvent intéressé par la découverte de motifs de plus haut niveau, comme par exemple les règles de classification les plus simples afin d'éviter le sur-apprentissage [26], ou encore des paires de règles d'exception [23] capables de révéler des caractéristiques globales des données. De tels motifs reposent sur des propriétés impliquant plusieurs motifs locaux et sont formalisés par la notion de *contrainte n-aire*.

Définition 2.2 (Contrainte n-aire) *Une contrainte n-aire est une contrainte portant sur n motifs.*

Définition 2.3 (Requête n-aire) *Une requête n-aire est une conjonction de contraintes contenant au moins une contrainte n-aire.*

2.2 Extraction de motifs de haut niveau

Un problème majeur dans les processus de l'ECBD est le nombre conséquent de motifs produits rendant leur utilisation difficile. Ainsi, les motifs les plus significatifs sont souvent noyés au milieu d'informations triviales ou redondantes. Ce problème est souvent rencontré par les extracteurs de motifs locaux dont le manque d'expressivité est frein à la découverte de motifs plus pertinents.

Ainsi plusieurs travaux se sont intéressés à réduire le nombre de motifs extraits pour obtenir des motifs plus pertinents. Plusieurs pistes se sont dégagées.

1. Dans cet article, nous nous intéressons au cas des motifs ensemblistes

[17, 5, 6, 25] proposent des méthodes permettant de réduire la redondance dans l'ensemble de motifs extraits, mais, même si ces approches comparent explicitement les motifs locaux entre eux, elles sont principalement fondées sur la réduction de la redondance entre motifs ou poursuivent des objectifs spécifiques tels que la classification.

[18, 22] présentent des approchent d'extraction de motifs dédiées à des classes particulières de contraintes n-aires. D'autres travaux se sont intéressés à des requêtes n-aires particulières pour lesquelles il proposent des méthodes de résolution ad hoc.

Enfin, très récemment, des approches génériques d'extraction de motifs sous contraintes n-aires sont proposées : l'idée clé de ses approches est de se baser sur un langage de contraintes à partir duquel il est possible de modéliser une très large panoplie de requêtes n-aires [14, 11, 19]. Néanmoins, aucune de ces approches ne permet de modéliser des requêtes contenant des contraintes quantifiées.

2.3 Motifs pics (peak)

Les motifs *pic* ou sommet (ou encore *peak*) [7] sont un exemple de requête n-aire se modélisant à l'aide de contraintes quantifiées.

Un motif pic est un motif qui affiche une valeur, par rapport à une mesure m donnée, assez grande par rapport à celles affichées par tous ses voisins. En d'autres termes, un motif pic a un comportement exceptionnel par rapport à ses voisins (le voisinage d'un motif est calculé par rapport à une distance d donnée).

Soit $m : \mathcal{L}_\mathcal{I} \rightarrow \mathbb{R}$ une mesure, δ un entier et ρ un réel, et soit d une distance, on a :

$$peak(X) \equiv \begin{cases} vrai & \text{si } \forall Y \in \mathcal{L}_\mathcal{I} \ tq \ d(X,Y) < \delta, \\ & \quad m(X) \geq \rho \times m(Y) \\ faux & \text{sinon} \end{cases}$$

Cette requête a été présentée dans [7] sans méthode de résolution associée.

Les motifs pics peuvent être recherchés relativement à diverses mesures comme l'aire, le taux de croissance des motifs émergents, etc. Les motifs émergents [9] sont des motifs dont la fréquence varie fortement entre deux classes. Ils caractérisent les classes de manière quantitative et qualitative. De par leur capacité à faire ressortir les distinctions entre classes, les motifs émergents permettent de construire des classifieurs ou de proposer une aide au diagnostic. L'émergence d'un motif est quantifiée par la mesure de son taux de croissance (*growth rate*) entre deux classes.

Définition 2.4 (Taux de croissance d'un motif)

Soient D_1et $D_2 \subseteq \mathcal{T}$ les ensembles de transactions correspondantes respectivement aux classes c_1 et c_2 et soit $\texttt{freq'}(X, D_j)$ la fréquence du motif X dans D_j, le taux de croissance de X dans D_1 est :

$$\texttt{growth-rate}_{D_1}(X) = \frac{|D_2| \times \texttt{freq'}(X,D_1)}{|D_1| \times \texttt{freq'}(X,D_2)}$$

Définition 2.5 (Motif émergent)

Soient un taux de croissance minimum ρ et une base de données $\mathbf{r}$ partitionnée en deux sous ensembles D_1 et D_2. Le motif X est émergent de D_2 vers D_1 ssi $\texttt{growth-rate}_{D_1}(X) \geq \rho$

Exemple 2.1 (pic d'émergence)

Soit $m(X) = \texttt{growth-rate}_{D_1}(X)$, $\delta = 1$, $\rho = 2$ et $d(X,Y) = |X \backslash Y| + |Y \backslash X|$, et considérons le jeu de données décrit par la table 1.

Le motif **AB** *est un peak puisque $\texttt{growth-rate}_{D_1}(AB) = 3$ est au moins 2 fois plus grand que les taux de croissance de ses 4 voisins A, B, ABC et ABD qui sont respectivement* 1.5, 1.5, 0.75 *et* 1.5.

2.4 Groupes de synexpression

Le domaine de l'analyse d'expression de gènes fournit un autre exemple de contrainte n-aire.

En effet, les motifs locaux composés de tags (ou gènes) et satisfaisants la contrainte d'aire sont au cœur de la recherche de groupes de synexpression [15]. Néanmoins, dans des données réelles telles que celles du transcriptome, la recherche de motifs tolérants aux erreurs [2] y est capitale afin de tenir compte de l'incertain qui est présent dans les données [2]. Les contraintes n-aires sont une façon naturelle de concevoir des motifs tolérants aux erreurs candidats à être des groupes de synexpression : ceux-ci sont définis par la combinaison de plusieurs motifs locaux satisfaisant la contrainte d'aire et ayant un fort recouvrement entre eux. Plus précisément, à partir de deux motifs locaux X et Y, on définit la contrainte n-aire suivante $\texttt{area}(X) > min_{\texttt{area}} \wedge \texttt{area}(Y) > min_{\texttt{area}} \wedge (\texttt{area}(X \cap Y) > \alpha \times min_{\texttt{area}})$ où $min_{\texttt{area}}$ est le seuil d'aire minimale et α est un paramètre donné par l'utilisateur pour fixer le recouvrement minimal entre motifs locaux. Cette contrainte n-aire peut être étendue à k motifs ($k \geq 2$) comme suit :

2. Un concept formel associe un ensemble maximal d'items à un ensemble maximal de transactions, des motifs tolérants aux erreurs constituent une relaxation de cette association en cherchant à extraire des rectangles d'ensembles denses en valeur 1 mais acceptant aussi un nombre contrôlé de 0.

$Synexpression(X_1, ..., X_k) \equiv$

$$\begin{cases} \forall\ 1 \leq i < j \leq k, \\ \texttt{area}(X_i) > min_{\texttt{area}} \wedge \\ \texttt{area}(X_j) > min_{\texttt{area}} \wedge \\ \texttt{area}(X_i \cap X_j) > \alpha \times min_{\texttt{area}} \wedge \\ \forall Z\ tq\ [\texttt{freq}(Z) > min_{\texttt{freq}}\ et\ \texttt{area}(Z) > min_{\texttt{area}}], \\ \exists i \in [1..k], \texttt{area}(Z \cap X_i) < \alpha \times min_{\texttt{area}} \end{cases}$$

L'utilisation de la quantification universelle dans la dernière contrainte permet de vérifier que le regroupement trouvé est maximal.

3 La PPC pour l'extraction de motifs

3.1 État de l'art

En 2008, [8] a proposé une approche permettant de traiter, dans un cadre unifié, un large ensemble de motifs locaux et de contraintes telles que la fermeture, la maximalité, les contraintes monotones ou anti-monotones et des variations de ces contraintes. Cette approche, qui marque un cadre fondateur, est malgré tout limitée aux motifs locaux.

Dans [14, 13], nous avons proposé une approche générique pour l'extraction de motifs sous contraintes n-aires. Cette approche (appelée *Pure-CP*) repose uniquement sur l'utilisation d'un solveur de CSP. Elle étend la modélisation proposée dans [8].

[21] propose une comparaison entre les performances des solveurs des CSP génériques et les extracteurs de motifs classiques. Il présente aussi un nouvel algorithme d'extraction de motifs sous contraintes basé sur les techniques de la PPC. [11] présente une méthode pour modéliser et résoudre des problèmes d'extraction portant sur k motifs ce qui est similaire aux contraintes n-aires.

Dans [19], nous avons proposé un langage de contraintes permettant d'écrire des requêtes n-aires de manière déclarative. L'ensemble des contraintes primitives composant ce langage peut être modélisé et résolu utilisant *Pure-CP*.

3.2 Approche Pure-CP

Nous avons proposé dans [14] *Pure-CP*, une approche générique pour l'extraction de motifs sous contraintes n-aires utilisant uniquement un solveur de CSP. Cette section décrit les principales étapes de cette approche.

a) Modélisation

Soit $\boldsymbol{r}$ un contexte transactionnel où $\mathcal{I}$ est l'ensemble de ses n items et $\mathcal{T}$ l'ensemble de ses m transactions. Soit $\boldsymbol{d}$ la matrice 0/1 de dimension (m, n) telle que $\forall t \in \mathcal{T}, \forall i \in \mathcal{I}$, $(d_{t,i} = 1)$ ssi $(i \in t)$.

Soient $X_1, X_2, ..., X_k$ les k motifs recherchés. Chaque motif inconnu X_j est modélisé par n variables de décision $\{X_{1,j}, X_{2,j}, ..., X_{n,j}\}$ (de domaine $\{0, 1\}$) telles que $(X_{i,j} = 1)$ ssi l'item i appartient au motif X_j. Le support T_{X_j} de chaque motif X_j est représenté par m variables de décision $\{T_{1,j}, T_{2,j}, ..., T_{m,j}\}$ (de domaine $\{0, 1\}$) qui sont associées à X_j comme suit : $(T_{t,j} = 1)$ ssi $(X_j \subseteq t)$.

b) Liens entre les variables et la base de données

La relation entre chaque motif recherché X_j ($1 \leq j \leq k$), son support T_{X_j} et la base de données $\boldsymbol{r}$est établie via des contraintes réifiées imposant que, pour chaque transaction t, $(T_{t,j} = 1)$ ssi $(X_j \subseteq t)$.

Ceci est formulé par l'équation suivante :

$$\forall j \in [1..k], \forall t \in \mathcal{T}, (T_{t,j} = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} X_{i,j} \times (1 - d_{t,i}) = 0 \quad (1)$$

c) Reformulation des contraintes

Considérons un opérateur $\texttt{op} \in \{<, \leq, >, \geq, =, \neq\}$; les contraintes numériques se reformulent comme suit :

- $\texttt{freq}(X_p)\ \texttt{op}\ \alpha \rightarrow \sum_{t \in \mathcal{T}} T_{t,p}\ \texttt{op}\ \alpha$
- $\texttt{size}(X_p)\ \texttt{op}\ \alpha \rightarrow \sum_{i \in \mathcal{I}} X_{i,p}\ \texttt{op}\ \alpha$

Certaines contraintes ensemblistes (telles que égalité, inclusion, appartenance,...) se reformulent directement à l'aide de contraintes linéaires :

- $X_p = X_q \rightarrow \forall i \in \mathcal{I}, X_{i,p} = X_{i,q}$
- $X_p \subseteq X_q \rightarrow \forall i \in \mathcal{I}, X_{i,p} \leq X_{i,q}$
- $i_o \in X_p \rightarrow X_{i_0,p} = 1$

Les autres contraintes ensemblistes (telles que intersection, union, différence,...) se reformulent aisément à l'aide de contraintes booléennes en utilisant la fonction de conversion (b::$\{0, 1\} \rightarrow \{False, True\}$) et les opérateurs booléens usuels :

- $X_p \cap X_q = X_r \rightarrow \forall i \in \mathcal{I}, b(X_{i,r}) = b(X_{i,p}) \wedge b(X_{i,q})$
- $X_p \cup X_q = X_r \rightarrow \forall i \in \mathcal{I}, b(X_{i,r}) = b(X_{i,p}) \vee b(X_{i,q})$
- $X_p \backslash X_q = X_r \rightarrow \forall i \in \mathcal{I}, b(X_{i,r}) = b(X_{i,p}) \wedge \neg b(X_{i,q})$

3.3 Discussion

Les variables d'un CSP sont toutes quantifiées existentiellement. Dans le cadre d'extraction de motifs sous contraintes utilisant une modélisation CSP, les seuls cas envisageables sont :

- **Existe-il** un motif vérifiant une propriété locale P (motifs locaux) ?
- **Existe-il** un motif vérifiant une propriété P' définie par rapport à un ensemble de motifs préalablement connus ou faisant partie des inconnues du problème (motifs n-aires) ?

Mais, dans les problèmes d'extraction de motifs sous contraintes, on trouve souvent des contraintes du type :

– **Existe-il** un motif X vérifiant une propriété P **quel que soit** un motif Y vérifiant une propriété P' (P' est généralement définie en fonction de X) ?

Pour la modélisation de telles requêtes, nous avons besoin d'utiliser la *quantification universelle* ($\forall$) que les CSPs classiques ne peuvent pas gérer. D'où le recours à l'utilisation des Problèmes de Satisfaction de Contraintes Quantifiées (QCSP) qui sont certes plus difficiles à résoudre mais strictement plus expressifs que les CSPs.

4 CSPs Quantifiés

4.1 QCSP

La formule logique associée à un CSP ayant pour variables $x_1, x_2, ..., x_n$ est de la forme :

$$\exists x_1 \in D_1, \exists x_2 \in D_2, ..., \exists x_n \in D_n \; c_1 \wedge c_2 \wedge ... \wedge c_k$$

Cette formule est en *forme normale prénexe*. Toutes les variables apparaissent ainsi quantifiées en début de formule, constituant le *préfixe*, chaque quantification liant toutes les occurrences de la variable dans le reste de la formule. Les QCSP étendent les CSP en introduisant dans cette formule des quantificateurs universels (i.e., $\forall$) en lieu et place de certains des quantificateurs existentiels (i.e., $\exists$) présents[4]. Par exemple, on peut représenter la formule suivante :

$$\forall x_1 \in D_1, \exists x_2 \in D_2, ..., \exists x_n \in D_n \; c_1 \wedge c_2 \wedge ... \wedge c_k$$

Cette formule reste en forme normale prénexe. Cependant, l'introduction de quantificateurs universels implique par ailleurs une relation d'ordre sur les variables qui n'était pas présente dans les CSP : la satisfiabilité de la formule logique précédente est susceptible de changer si l'on inverse dans le préfixe deux variables quantifiées de manière différente ou même deux variables séparées par une alternance de quantificateurs. Cette relation d'ordre sera donc aussi précisée dans la définition formelle d'un QCSP. Cependant, si deux variables qui se suivent dans le préfixe sont quantifiées de la même manière, leur ordre relatif n'influe aucunement sur la sémantique de la formule. Une sous séquence du préfixe où toutes les variables sont quantifiées de la même manière est appelée *bloc de quantification* ou *qset*.

Dans les définitions qui suivent, soit p un entier tel que $p \leq n$ (où n est le nombre de variables du problème).

Définition 4.1 (qset)
Un qset est un couple (qt, W) avec W un ensemble de variables et $qt \in \{\forall, \exists\}$ un quantificateur.

Le préfixe d'un QCSP peut donc être formellement défini comme suit :

Définition 4.2 (préfixe de QCSP)
Un préfixe de QCSP est une séquence de qsets $((qt_1, W_1), ..., (qt_p, W_p))$ dont les ensembles de variables sont deux à deux disjoints ($\forall i, j,\ i \neq j \rightarrow (W_i \cap W_j = \emptyset)$).

Le QCSP se définit alors comme un préfixe accompagné d'un ensemble de contraintes dont les variables doivent faire partie du préfixe :

Définition 4.3 (QCSP)
Un QCSP est un couple (P, G) avec $P = ((qt_1, W_1), ..., (qt_p, W_p))$ un préfixe et G un ensemble de contraintes appelé Goal tel que $var(G) \subseteq \bigcup_{i \in \{1,...,p\}} W_i$.

Exemple 4.1
Le problème :

$\exists X \in \{0,1,2,3\}$, $\forall Y \in \{0,1,2\}$, $\exists Z \in \{0,\ldots,6\}$
$X + Y = Z$

est représenté par le QCSP suivant :
$Q = ([(\exists, X), (\forall, Y), (\exists, Z,)], \{X + Y = Z\})$

4.2 QCSP$^+$

Le formalisme des QCSP est étendu dans [1] pour permettre de modéliser explicitement des restrictions propres à chaque variable quantifiée. Chaque restriction est définie sous forme d'un ensemble de contraintes : le quantificateur ne porte alors que sur les tuples satisfaisant ces contraintes.

Un QCSP$^+$ doit donc intégrer des contraintes dont le but est de réduire la portée des quantificateurs. Ces contraintes sont donc ajoutées directement dans les qset.

Définition 4.4 (rqset)
Un rqset est un triplet (qt, W, C) avec W un ensemble de variables, $qt \in \{\forall, \exists\}$ un quantificateur et C un ensemble de contraintes.

Le préfixe d'un QCSP$^+$ peut donc être formellement défini comme suit :

Définition 4.5 (préfixe de QCSP$^+$)
Un préfixe de QCSP est une séquence de rqsets $((qt_1, W_1, C_1), ..., (qt_p, W_p, C_p))$ telle que :
– $\forall i, j \in \{1, .., p\}, i \neq j \rightarrow W_i \cap W_j = \emptyset$
– $\forall i \in \{1, .., p\}, var(C_i) \subseteq (W_1 \cup ... \cup W_p)$

Le QCSP$^+$ se définit alors de manière analogue au QCSP comme suit :

Définition 4.6 (QCSP$^+$)
Un QCSP$^+$ est un couple (P, G) avec $P = ((qt_1, W_1, C_1), ..., (qt_p, W_p, C_p))$ un préfixe de QCSP$^+$ et G un ensemble de contraintes appelé Goal tel que $var(G) \subseteq \bigcup_{i \in \{1,...,p\}} W_i$.

Exemple 4.2
Le problème :

$\exists X \in \{0, 1, 2, 3\}$
$\forall Y \in \{0, 1, 2\}$ *tel que* $Y \neq X$,
$\exists Z \in \{0, \ldots, 6\}$ *tel que* $Z < 2 * X$,
$X + Y = Z$

est représenté par le QCSP$^+$ suivant :
$Q = \big([(\exists, X), (\forall, Y, \{X \neq Y\}),$
$(\exists, Z, \{Z < 2 * X\})], Goal = \{X + Y = Z\}\big)$

4.3 *QeCode*

QeCode[3] est un solveur de QCSP écrit en C++ et basé sur le solveur de contraintes *Gecode*[4]. Il intègre aussi les procédures de recherche des QCSP$^+$ proposées dans [24]. L'ensemble des contraintes implémentées dans *Gecode* (y compris par les utilisateurs, *Gecode* permettant l'utilisation de propagateurs personnalisés) peut donc être utilisé dans *QeCode*. Ceci est à l'origine du choix de *QeCode* pour la mise en œuvre de l'approche que nous proposons dans cet article.

5 QCSP$^+$ pour l'extraction de motifs

L'approche d'extraction de motifs sous contraintes quantifiées que nous proposons (approche QCSP) repose sur la modélisation proposée pour *Pure-CP*. Elle utilise les mêmes contraintes réifiées pour établir le lien entre les motifs recherchés et les transactions les supportant (cf. équation 1). Elle partage aussi la modélisation des motifs recherchés ainsi que l'implantation et la reformulation des contraintes (cf. section 3.2).

Ainsi, soit $\boldsymbol{r}$ un contexte transactionnel où $\mathcal{I}$ est l'ensemble de ses n items et $\mathcal{T}$ l'ensemble de ses m transactions et soient $X_1, X_2, ..., X_k$ les k motifs recherchés. Chaque motif inconnu X_j est modélisé par n variables de décision $\{X_{1,j}, X_{2,j}, ..., X_{n,j}\}$ et son support T_{X_j} est représenté par m variables de décision $\{T_{1,j}, T_{2,j}, ..., T_{m,j}\}$.

Si un motif X_J est quantifié existentiellement, alors toutes ses variables de décision le seront aussi (comme dans *Pure-CP*). Par contre, si un motif X_j est quantifié universellement, alors toutes ses variables de décision seront quantifiées universellement.

3. `http://www.univ-orleans.fr/lifo/software/qecode/QeCode.html`
4. `http://http://www.gecode.org/`

La modélisation s'effectue par niveaux dépendant de la portée de chaque quantificateur, le dernier niveau étant consacré aux contraintes du *Goal*.

Remarque 5.1 *Toutes les variables que nous introduisons dans les modélisations qui suivent et pour lesquelles les domaines ne sont pas définis explicitement sont des variables représentant les motifs de type X_j et représentent donc en variables de décision de domaine $\{0, 1\}$ (cf. section 3.2).*

Exemple 5.1 (peak)
Cet exemple détaille la modélisation de la requête peak (cf. section 2.3).

Nous notons par $\boldsymbol{m}$ une mesure d'intérêt (taux d'émergence, aire, ...). Nous fixons à 1 la valeur de la distance maximale δ délimitant le voisinage.

Pour modéliser ce problème, on introduit les variables suivantes :
- *X_1 représentant le motif pic X, cette variable est quantifiée existentiellement.*
- *X_2 représentant le motif Y représentant le voisinage de X, cette variable est quantifiée universellement.*

$Q = \Big(\big[(\exists, X_1, C_1),\ (\forall, X_2, C_2)\big],\ Goal = C_3\Big)$

Où :
- $C_1 = \emptyset$
- $C_2 = d(X_1, X_2) \leq 1$
- $C_3 = \boldsymbol{m}(X_1) \geq \rho \times \boldsymbol{m}(X_2)$

Exemple 5.2 (Groupes de synexpression)
Nous présentons dans cet exemple la modélisation sous forme de QCSP$^+$ de la requête permettant la découverte de groupes de synexpression présentée dans la section 2.4. La quantification universelle apporte l'attrait supplémentaire de pouvoir extraire les groupements maximaux, renforçant ainsi l'hypothèse que l'ensemble des motifs agglomérés forme un groupe de synexpression.

Pour la modélisation de cette requête, nous introduisons les variables suivantes :
- *$\{X_1, X_2, ..., X_k, Z\}$ variables représentant les motifs,*
- *$\{A_2, A_3, ..., A_k, B\}$ variables représentant les intersections entre les motifs dans la perspective du calcul de l'aire de ces intersections,*
- *$\{i_3, i_4, ..., i_k, i\}$ variables compteurs. Ces variables permettent d'éviter d'associer une variable supplémentaire à chaque intersection deux à deux possibles entre les k motifs.*

La requête des groupes de synexpression permettant l'extraction de groupements de taille k $Synexpression(X_1, ..., X_k)$ peut ainsi être modélisée par la formule suivante :

$Synexpression(X_1, ..., X_k) \equiv$

$$
\begin{cases}
\exists X_1,\ [\mathit{freq}(X_1) > min_{freq}, \mathit{area}(X_1) > min_{area}] \\
\exists X_2, A_2\ tq\ \mathit{freq}(X_2) > min_{freq} \wedge \\
\qquad \mathit{area}(X_2) > min_{area} \wedge \\
\qquad X_1 \neq X_2\ \wedge\ A_2 = X_1 \cap X_2\ \wedge \\
\qquad \mathit{area}(A_2) > \alpha \times min_{area} \\
\quad \exists X_3\ tq\ \mathit{freq}(X_3) > min_{freq} \wedge \\
\qquad \mathit{area}(X_3) > min_{area} \wedge X_1 \neq X_3 \wedge \\
\qquad X_2 \neq X_3 \\
\quad \forall i_3 \in [1..2], A_3\ tq\ A_3 = X_3 \cap X_{i_3} \wedge \\
\qquad \mathit{area}(X_3) > \alpha \times min_{area} \\
\quad ... \\
\quad \exists X_k\ tq\ \mathit{freq}(X_k) > min_{freq} \wedge \\
\qquad \mathit{area}(X_k) > min_{area} \wedge \\
\qquad X_1 \neq X_k\ \wedge\ ...\ \wedge\ X_{k-1} \neq X_k \\
\quad \forall i_k \in [1..k-1], A_k\ tq\ A_k = X_k \cap X_{i_k} \wedge \\
\qquad \mathit{area}(A_k) > \alpha \times min_{area} \\
\quad \forall Z\ tq\ \mathit{freq}(Z) > min_{freq} \wedge \\
\qquad \mathit{area}(Z) > min_{area} \wedge \\
\qquad X_1 \neq Z\ \wedge\ ...\ \wedge\ X_k \neq Z \\
\quad \exists i \in [1..k], B\ \ tq\ B = Z \cap X_i, \\
\quad \mathit{area}(B) < \alpha \times min_{area}
\end{cases}
$$

Nous considérons par exemple le cas où on veut spécifier un recouvrement composé de trois motifs X_1, X_2, X_3. La modélisation de la requête des synexpressions dans ce cas demande l'utilisation des variables suivantes :

- *$\{X_1, X_2, X_3, Z\}$ variables représentant les motifs.*
- *$\{A_2, A_3, B\}$ variables représentant les intersections entre les motifs,*
- *$\{i_3, i\}$ variables compteurs.*

La requête est finalement modélisée par le QCSP$^+$ suivant :

$Q = \Big(\big[(\exists, X_1, C_1), (\exists, (X_2, A_2), C_2), (\exists, (X_3), C_3),$
$(\forall, (i_3 \in [1..2], A_3), C_4), (\forall, (Z), C_5),$
$(\exists, (i \in [1..3], B), C_6) \big], Goal = C_7 \Big)$

où :

- $C_1 = \mathit{freq}(X_1) > min_{freq} \wedge \mathit{area}(X_1) > min_{area}$
- $C_2 = \mathit{freq}(X_2) > min_{freq} \wedge \mathit{area}(X_2) > min_{area} \wedge X_1 \neq X_2\ \wedge A_2 = X_1 \cap X_2\ \wedge \mathit{area}(A_2) > \alpha \times min_{area}$
- $C_3 = \mathit{freq}(X_3) > min_{freq} \wedge \mathit{area}(X_3) > min_{area} \wedge X_1 \neq X_2\ \wedge X_2 \neq X_3$
- $C_4 = A_3 = X_3 \cap X_{i_3}\ \wedge \mathit{area}(X_3) > \alpha \times min_{area}$
- $C_5 = \mathit{freq}(Z) > min_{freq} \wedge \mathit{area}(Z) > min_{area} \wedge X_1 \neq Z\ \wedge X_2 \neq Z\ \wedge X_3 \neq Z$
- $C_6 = B = Z \cap X_i$
- $C_7 = \mathit{area}(B) < \alpha \times min_{area}$

6 Expérimentations

Nous montrons, dans cette section, la faisabilité et les apports pratiques de l'approche basée sur les QCSPs présentée dans cet article.

6.1 Protocole expérimental

Nous avons mené différentes expérimentations sur plusieurs jeux de données de l'UCI *repository*[5] ainsi que sur le jeu de données réelles `Meningitis`[6]. La table 2 résume les différentes caractéristiques des jeux de données utilisés.

Les expérimentations ont été menées principalement sur la requête *peak* appliquée à la mesure du taux de croissance d'un motif émergent (`growth-rate`).

La machine utilisée est un PC 2.83 GHz Intel Core 2 Duo processor (4 GB de RAM) sous Ubuntu Linux.

Enfin nous avons utilisé le solveur de QCSP *QeCode*.

Jeu de données	#trans	#items	densité
`Australian`	690	55	0.25
`German`	1000	76	0.28
`Meningitis`	329	84	0.27
`Posto`	90	23	0.39

TABLE 2 – Description des jeux de données

6.2 Solveur utilisé

Nous utilisons le solveurs de QCSP *QeCode* (cf. section 4.3). Initialement, pour une instance QCSP donnée, *QeCode* retourne une seule stratégie gagnante (ou solution), si elle existe. Nous avons donc introduit quelques modifications sur le solveur afin d'obtenir une version retournant l'intégralité des solutions.

Ainsi, notre approche permet d'extraire l'ensemble correct et complet des motifs satisfaisant les requêtes contenant des motifs quantifiés universellement.

6.3 Résultats expérimentaux

Les figures 1 et 2 décrivent la variation du nombre de motifs pics détectés par rapport à la mesure du taux croissance `growth-rate` en fonction des paramètres ρ (cf. exemple 5.1) et $minfr$ (seuil minimal de fréquence des motifs émergents) pour les jeux de données `German`, `Meningitis`, `Postoperative-patient-data` (`Posto`) et `Australian`.

On constate que cette requête conduit à très peu de motifs (alors qu'il est bien commun que le nombre de motifs émergents soit très grand [9]). Ces résultats mettent en valeur l'efficacité de l'utilisation des contraintes contenant des quantifications universelles

5. `http://www.ics.uci.edu/~mlearn/MLRepository.html`
6. `Meningitis` recense les pathologies des enfants atteints d'une méningite virale ou bactérienne.

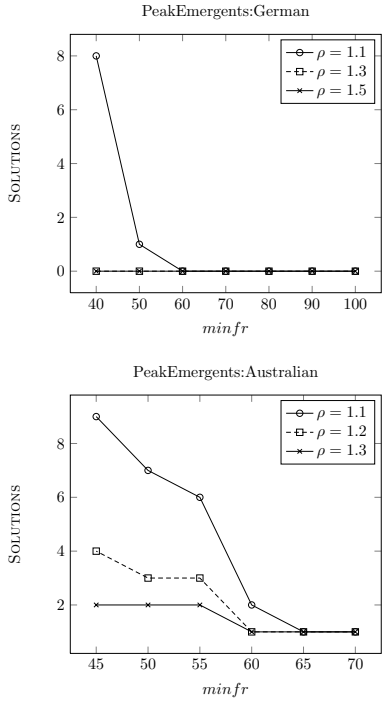

FIGURE 1 – Évolution du nombre de pics d'émergence extraits en fonction de $minfr$ (1/2)

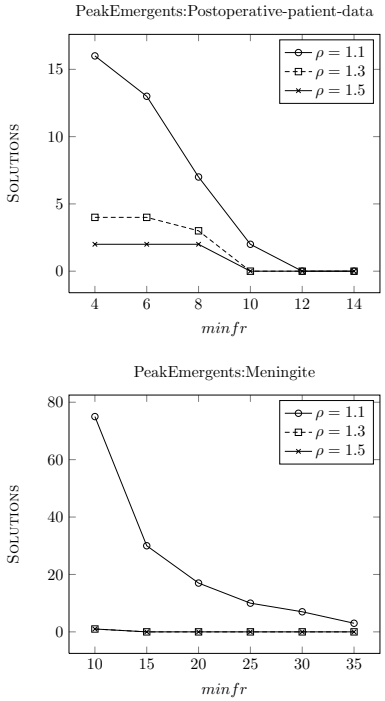

FIGURE 2 – Évolution du nombre de pics d'émergence extraits en fonction de $minfr$ (2/2)

dans la réduction du nombre de motifs extraits en définissant des propriétés fortes sur les motifs recherchés. Ceci ouvre une piste intéressante vers la définition de requêtes permettant de cibler des motifs de haut niveau et en faible nombre.

6.4 Efficacité

Les expérimentations menées permettent aussi de quantifier les temps de calcul consommés par notre approche (cf. figures 3 et 4). Naturellement, les temps de calcul varient en fonction de la taille des jeux de données et de la dureté des contraintes.

Les résultats expérimentaux montrent aussi que la difficulté à fouiller les jeux de données augmente en fonction du nombre d'items composant le jeu de données. Ceci est expliqué par le fait que, pour les requêtes utilisées dans les expérimentations, la quantification universelle porte sur les variables représentant les motifs. La figure 5 met l'accent sur ce comportement que nous expliquons plus en détail dans la section suivante.

7 Discussion

Dans le cas des QCSP, une simple affectation de toutes les variables ne permet pas de vérifier la viabilité de l'instanciation comme c'est le cas pour les CSP classiques : en effet, le *Goal* doit être vérifié pour toutes les valeurs possibles des variables universellement quantifiées du problème. Une solution d'un QCSP doit donc exprimer tous ces cas possibles. De plus, la formule logique associée (qui est une formule du premier ordre sous forme prénexe) lie les variables dans un ordre donné par le préfixe. Ainsi, la valeur d'une variable existentielle postérieure (selon l'ordre du préfixe) à une variable universelle sera fonction de la valeur de celle-ci, alors que dans l'ordre inverse, il est nécessaire d'avoir une valeur d'une variable existentielle consistante avec toutes les valeurs de la variable universelle placée postérieurement.

Ceci a été traduit dans nos expérimentations par le fait qu'on a beaucoup plus de difficultés à gérer les jeux de données quand le nombre d'items augmente. Dans les requêtes présentées dans cet article et celles utilisées dans les expérimentations, la quantification universelle porte sur les variables représentant les motifs. Il est ainsi plus coûteux de vérifier tous les motifs possibles quand le nombre d'items augmente. C'est ce qui explique l'absence d'expérimentations présentées sur les groupes de synexpression où l'extraction s'effectue généralement dans des bases de données comportant plusieurs milliers d'items.

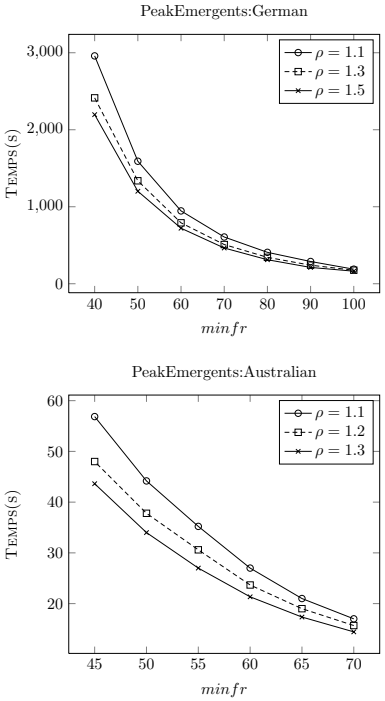

FIGURE 3 – Évolution du temps d'exécution de la requête d'extraction de pics d'émergence en fonction de $minfr$ (1/2)

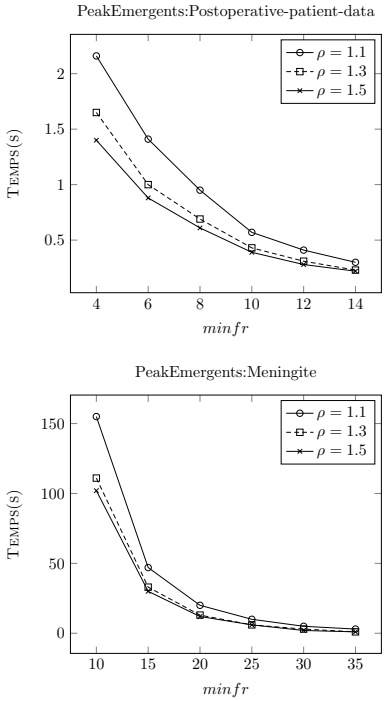

FIGURE 4 – Évolution du temps d'exécution de la requête d'extraction de pics d'émergence en fonction de $minfr$ (2/2)

8 Conclusion

L'approche proposée a montré l'élégance de la modélisation de certaines requêtes de fouilles de données utilisant les QCSPs. Acceptant l'intégralité de l'ensemble des contraintes primitives composant le langage de contraintes proposé dans [19], elle ouvre la porte vers la modélisation des requêtes cherchant à vérifier des propriété plus fortes sur les motifs et conduisant à l'extraction de motifs plus pertinent et facilement interprétables par l'utilisateur. Les résultats expérimentaux ont montré la faisabilité de l'approche même s'il n'est pas possible pour le moment de traiter des bases de données de tailles réelles (i.e, plusieurs milliers d'items) pour la recherche de groupes de synexpression. Nos perspectives s'inscrivent dans cette direction.

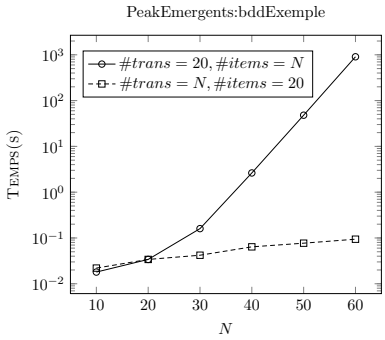

FIGURE 5 – Évolution du temps d'exécution de la requête d'extraction de pics d'émergence en fonction de la taille du jeu de données (densité fixée à 0.4)

Références

[1] Marco Benedetti, Arnaud Lallouet, and Jérémie Vautard. QCSP Made Practical by Virtue of Restricted Quantification. In *IJCAI*, pages 38–43. AAAI Press, 2007.

[2] J. Besson, C. Robardet, and J-F. Boulicaut. Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In *ICCS'06*, pages 144–157, Aalborg, Denmark, 2006.

[3] F. Bonchi, F. Giannotti, C. Lucchese, S. Orlando, R. Perego, and R. Trasarti. A constraint-based querying system for exploratory pattern discovery. *Inf. Syst.*, 34(1) :3–27, 2009.

[4] Lucas Bordeaux, Marco Cadoli, and Toni Mancini. CSP properties for quantified constraints : Definitions and complexity. In *National Conference on Artificial Intelligence*, pages 360–365. AAAI Press, 2005.

[5] Björn Bringmann and Albrecht Zimmermann. The chosen few : On identifying valuable patterns. In *ICDM*, pages 63–72. IEEE Computer Society, 2007.

[6] Björn Bringmann and Albrecht Zimmermann. One in a million : picking the right patterns. *Knowl. Inf. Syst.*, 18(1) :61–81, 2009.

[7] Bruno Crémilleux and Arnaud Soulet. Discovering knowledge from local patterns with global constraints. In *ICCSA (2)*, pages 1242–1257, 2008.

[8] Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint Programming for Itemset Mining. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining edition :14 location*, Las Vegas, Nevada, USA, 2008.

[9] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns : Discovering trends and differences. In *KDD*, pages 43–52, 1999.

[10] Arnaud Giacometti, Eynollah Khanjari Miyaneh, Patrick Marcel, and Arnaud Soulet. A framework for pattern-based global models. In *IDEAL*, volume 5788 of *Lecture Notes in Computer Science*, pages 433–440. Springer, 2009.

[11] Tias Guns, Siegfried Nijssen, and Luc De Raedt. k-Pattern set mining under constraints. *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[12] Pascal Van Hentenryck, Yves Deville, and Choh-Man Teng. A generic arc-consistency algorithm and its specializations. *Artif. Intell.*, 57(2-3) :291–321, 1992.

[13] M. Khiari, P. Boizumault, and B. Crémilleux. Extraction de motifs n-aires utilisant la PPC. In *6èmes Journées Francophones de Programmation par Contraintes (JFPC'10)*, pages 167–176, Caen, 2010.

[14] Mehdi Khiari, Patrice Boizumault, and Bruno Crémilleux. Constraint programming for mining n-ary patterns. In David Cohen, editor, *CP*, volume 6308 of *Lecture Notes in Computer Science*, pages 552–567. Springer, 2010.

[15] J. Kléma, S. Blachon, A. Soulet, B. Crémilleux, and O. Gandrillon. Constraint-based knowledge discovery from sage data. *In Silico Biology*, 8(0014), 2008.

[16] A. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz. From local patterns to global models : The lego approach to data mining. In *Int. Workshop LeGo co-located with ECML/PKDD'08*, pages 1–16, Antwerp, Belgium, 2008.

[17] Arno J. Knobbe and Eric K. Y. Ho. Pattern teams. In *PKDD*, pages 577–584, 2006.

[18] Laks V. S. Lakshmanan, Raymond T. Ng, Jiawei Han, and Alex Pang. Optimization of constrained frequent set queries with 2-variable constraints. In *SIGMOD Conference*, pages 157–168. ACM Press, 1999.

[19] J.-P. Métivier, P. Boizumault, B. Crémilleux, M. Khiari, and S. Loudni. A constraint-based language for declarative pattern discovery. In *27th annual ACM Symposium on Applied Computing (SAC'12)*, pages 1–7, Riva del Garde (Trento), Italy, March 2012.

[20] R. T. Ng, V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *proceedings of ACM SIGMOD'98*, pages 13–24. ACM Press, 1998.

[21] Siegfried Nijssen and Tias Guns. Integrating constraint programming and itemset mining. In *ECML/PKDD (2)*, pages 467–482, 2010.

[22] Luc De Raedt and Albrecht Zimmermann. Constraint-based pattern set mining. In *SDM*. SIAM, 2007.

[23] Einoshin Suzuki. Undirected Discovery of Interesting Exception Rules. *IJPRAI*, 16(8) :1065–1086, 2002.

[24] Jérémie Vautard. *Modélisation et résolution de problèmes de décision et d'optimisation hiérarchiques en utilisant des contraintes quantifiées.* These, Université d'Orléans, April 2010.

[25] Dong Xin, Hong Cheng, Xifeng Yan, and Jiawei Han. Extracting redundancy-aware top-k patterns. In *KDD*, pages 444–453. ACM, 2006.

[26] X. Yin and J. Han. CPAR : classification based on predictive association rules. In *proceedings of the 2003 SIAM Int. Conf. on Data Mining (SDM'03)*, San Fransisco, CA, May 2003.