HAL open science

Extraction de concepts sous contraintes dans des données d'expression de gènes

Baptiste Jeudy, François Rioult

To cite this version:

Baptiste Jeudy, François Rioult. Extraction de concepts sous contraintes dans des données d'expression de gènes. Conférence sur l'apprentissage automatique, Jun 2005, Nice, France. pp.265-280. hal-00359222

HAL Id: hal-00359222

https://hal.science/hal-00359222v1

Submitted on 6 Feb 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Extraction de concepts sous contraintes dans des données d'expression de gènes*

Baptiste Jeudy[1], François Rioult[2]

[1] Équipe Universitaire de Recherche en Informatique de St-Etienne (EURISE),
Université de St-Etienne.
baptiste.jeudy@univ-st-etienne.fr

[2] GREYC - CNRS UMR 6072,
Université de Caen Basse-Normandie
francois.rioult@info.unicaen.fr

Abstract : L'une des activités les plus importantes en biologie est l'analyse des données d'expression de gènes. Les biologistes espèrent ainsi mieux comprendre les fonctions des gènes et leurs interactions. Nous étudions dans cet article une technique permettant d'aider à l'analyse de ces données d'expression : l'extraction de concepts sous contraintes. Pour cela, nous proposons d'extraire des fermés sous contraintes dans les données "transposées" en utilisant des algorithmes classiques. Ceci nous amène a étudier la "transposition" des contraintes dans les données transposées de manière à pouvoir les utiliser dans ces algorithmes.

Mots-clés : Extraction de connaissances, Data-mining, Concepts Formels, Itemsets Fermés, Contraintes.

1 Motivations

Maintenant que le décodage du génome est terminé pour de nombreuses espèces animales et végétales, il reste encore un formidable défi pour la biologie moderne : comprendre la fonction de tous ces gènes et la manière dont ils interagissent entre-eux. Pour cela, les biologistes mènent des expériences de mesure de l'expression de gènes. Celles-ci ont pour but de leur fournir des données leur permettant de faire des hypothèses sur ces fonctions et ces interactions.

Les données d'expression de gènes se présentent typiquement sous la forme d'une matrice binaire. Chaque colonne représente un gène et chaque ligne donne les résultats d'une expérience de mesure du niveau d'expression des gènes. Chacune de ces expériences consiste à déterminer, pour une cellule donnée issue d'une situation biologique donnée (par exemple un organe spécifique, une culture cellulaire), quels sont les gènes qui sont

*Ce travail a été partiellement financé par l'ACI masse de données (MD 46, Bingo)

sur-exprimés, c'est-à-dire ceux qui ont une activité biologique importante au moment de la mesure. Dans la matrice, les gènes qui sont sur-exprimés[1] dans une situation biologique sont codés par un 1. Ceux qui ne le sont pas sont codés par un 0. La table 1 donne un exemple d'une telle matrice.

	Gène 1	Gène 2	Gène 3	Gène 4
cellule 1	1	1	1	0
cellule 2	1	1	1	0
cellule 3	0	1	1	1

Table 1: Exemple de matrice d'expression de gènes

Dans cet article, nous étudions une technique de fouille de données permettant d'aider le biologiste à faire des hypothèses sur les fonctions des gènes et la manière dont ils interagissent. Pour cela, les techniques d'extraction de motifs semblent particulièrement adaptées. Il existe cependant de nombreux types de motifs : les itemsets, les itemsets fermés ou libres, les règles d'association ou encore les concepts formels. Nous avons choisi ici d'étudier l'extraction des concepts.

Dans ce cadre, un concept formel est une paire (G, E) où G est un ensemble de gènes (i.e., un ensemble de colonnes de la matrice) appelé intension du concept et E un ensemble d'expériences (i.e., un ensemble de lignes) appelé extension du concept. Ces ensembles sont tels que si $g \in G$ et $e \in E$, alors le gène g est sur-exprimé dans l'expérience e (il y a un 1 dans la ligne e colonne g). De plus, les deux ensembles G et E sont maximaux, i.e., ils ne peuvent pas grossir sans perdre la propriété précédente (une définition plus formelle des concepts est donnée dans la section 2). Autrement dit, un concept est une sous-matrice maximale ne contenant que des 1. Dans notre matrice exemple, ($\{$Gène 1, Gène 2, Gène 3$\}$, $\{$cel 1, cel 2 $\}$) est un concept.

Du point de vue du biologiste, les concepts sont très intéressants. En effet, un concept (G, E) regroupe des gènes qui sont sur-exprimés dans les mêmes expériences. Si la fonction de certains de ces gènes est connue, cela peut permettre de faire des hypothèses sur la fonction de ceux qui sont inconnus. De plus, si les expériences apparaissant dans l'extension E partagent des propriétés communes (par exemple, elles concernent toutes des cellules du foie ou des cellules cancéreuses), cela permet encore une fois de faire des hypothèses sur les gènes. Le fait que les concepts associent à la fois des gènes et des expériences est donc un avantage par rapport à d'autres motifs comme les itemsets ou les règles d'association qui ne portent que sur les gènes. De plus, un gène (ou une expérience) peut apparaître dans plusieurs concepts (par opposition à ce qui se passe dans le cas du clustering). Si le biologiste s'intéresse à un gène particulier, il peut donc étudier quels sont les gènes liés à celui-ci (i.e., apparaissant dans les mêmes concepts) suivant les situations biologiques. Cela est très important car il s'avère en effet qu'un gène peut intervenir dans plusieurs fonctions biologiques différentes. Enfin, les concepts sont beaucoup moins nombreux que les itemsets tout en représentant la même information : ils sont donc plus simples à exploiter.

Pour simplifier encore l'exploitation de ces concepts par le biologiste, l'utilisation de

[1] dont l'activité biologique dépasse un seuil fixé par le biologiste

contraintes semble pertinente : le biologiste peut indiquer une contrainte qui doit être satisfaite par tous les concepts extraits. Par exemple, il peut imposer qu'un gène particulier (ou ensemble de gènes) apparaisse (ou pas) dans les concepts extraits. Il peut aussi se restreindre aux concepts impliquant des expériences sur des cellules cancéreuses ou contenant au moins 5 gènes. L'utilisation des contraintes permet finalement au biologiste de mieux cibler sa recherche.

1.1 Notre contribution

Nous proposons dans cet article d'étudier l'extraction de concepts sous contraintes dans des données d'expression de gènes. Cette extraction pose deux problèmes principaux :

1. utilisation des contraintes : nous laissons la possibilité à l'utilisateur de spécifier une contrainte portant à la fois sur l'intension et l'extension du concept. Ces contraintes sont utiles pour l'utilisateur pour préciser sa recherche mais elles sont aussi parfois indispensables pour rendre l'extraction faisable. En effet, il est généralement impossible d'extraire tous les concepts. Il faut donc dans ce cas utiliser les contraintes *pendant* l'extraction (et non pas seulement dans une phase de filtrage des concepts *après* l'extraction) pour diminuer la complexité celle-ci.

2. taille des données : la complexité des algorithmes d'extraction est généralement linéaire par rapport au nombre de lignes et exponentielle par rapport au nombre de colonnes. Or dans le cas des données d'expression de gènes, le nombre de colonnes est souvent très important : l'utilisation de techniques comme les puces à ADN permet d'obtenir l'expression de milliers de gènes en une seule expérience. D'un autre coté, le nombre d'expériences est souvent réduit du fait du temps nécessaire à leur mise en place et de leur coût. Ceci amène à des matrices comportant beaucoup de colonnes (jusqu'à plusieurs milliers) et relativement peu de lignes (quelques dizaines ou centaines) ce qui est plutôt atypique dans le domaine du data-mining. Les algorithmes classiques ne sont donc pas bien adaptés à ce type de données.

L'extraction de motifs sous contrainte est un thème de recherche qui a été très étudié ces dernières années (Srikant *et al.*, 1997; Ng *et al.*, 1998; Garofalakis *et al.*, 1999; Boulicaut & Jeudy, 2000; Pei & Han, 2000; Zaki, 2000; Boulicaut & Jeudy, 2001; Bucila *et al.*, 2003; Albert-Lorincz & Boulicaut, 2003; Bonchi *et al.*, 2003; Bonchi & Lucchese, 2004)... De nombreux algorithmes ont été proposés et tentent d'utiliser efficacement les contraintes pour diminuer les temps d'extraction en élaguant le plus tôt possible l'espace de recherche. L'extraction de concepts est fortement liée à l'extraction d'itemsets libres ou fermés dont l'étude a également donné lieu à de nombreux travaux (Pasquier *et al.*, 1999; Boulicaut *et al.*, 2000; Pei *et al.*, 2000; Zaki & Hsiao, 2002; Boulicaut *et al.*, 2003)...

Cependant, ces travaux ne font pas d'extraction de concepts sous contrainte et ne sont pas adaptés à des données ayant plus de colonnes que de lignes. En ce qui concerne l'extraction de concepts sous contraintes, une proposition récente à été faite dans (Besson *et al.*, 2004). Cependant, l'algorithme proposé, D-Miner, ne permet que

de traiter un type particulier de contraintes, les contraintes monotones. Nous verrons dans la section 4 comment l'étude que nous proposons ici va nous permettre de traiter aussi les contraintes anti-monotones avec cet algorithme.

En ce qui concerne le second problème, plusieurs propositions ont été faites récemment pour le résoudre : l'algorithme CARPENTER (Pan *et al.*, 2003) est conçu pour extraire les fermés fréquents dans une base de données avec plus de colonnes que de lignes. Dans (Rioult *et al.*, 2003; Rioult & Crémilleux, 2003), les auteurs utilisent des algorithmes classiques mais au lieu de faire l'extraction dans les données originales, ils travaillent sur la matrice transposée. Dans ce cas, la matrice transposée comporte beaucoup de lignes et peu de colonnes, ce qui permet d'utiliser les techniques habituelles efficacement. Cependant, ces travaux ne traitent que du cas de la contrainte de fréquence ou de contraintes simples sur les itemsets. Le cas général où la contrainte est une formule booléenne construite à partir de contraintes simples, portant à la fois sur l'intension et l'extension, n'est pas abordé.

Notre proposition est donc d'utiliser des algorithmes classiques (éventuellement légèrement modifiés) dans la matrice transposée, afin de travailler sur des données au format plus classique (peu de colonnes, beaucoup de lignes). Pour pouvoir traiter des contraintes complexes portant sur les concepts, nous allons présenter ici une étude théorique sur les contraintes et sur la manière de les "transposer" (en fait, il s'agira plutôt d'une projection) de façon à pouvoir les utiliser dans la matrice transposée.

Cet article est organisé de la manière suivante : dans la section 2, nous rappelons quelques définitions à propos de l'extraction d'itemsets et de la correspondance de Galois. Nous présentons ensuite formellement le problème que nous cherchons à résoudre. Dans la section 3, nous présentons la projection des contraintes simples et composées. Ensuite, la section 4 montre comment utiliser la projection de contraintes et l'extraction dans la matrice transposée pour résoudre notre problème. Finalement, nous concluons dans la section 5.

2 Définitions

Pour éviter les confusions entre les lignes (ou colonnes) de la base de données originale et les lignes (ou colonnes) de base de données "transposée", nous définissons une base de données comme une relation entre deux ensembles : un ensemble d'attributs et un ensemble d'objets. L'ensemble des **attributs** (ou items) est noté $\mathcal{A}$ et correspond, dans notre application biologique, à l'ensemble des gènes. L'ensemble des **objets** est noté $\mathcal{O}$ et représente les situations biologiques. L'**espace des attributs**, $2^{\mathcal{A}}$, est la collection des sous-ensembles de $\mathcal{A}$, appelés **itemsets** et l'**espace des objets**, $2^{\mathcal{O}}$, est la collection des sous-ensembles de $\mathcal{O}$. Lorsqu'on considère l'ordre défini par l'inclusion ensembliste, chacun des espaces $2^{\mathcal{A}}$ et $2^{\mathcal{O}}$ est naturellement muni d'une structure de treillis.

Une base de données est une relation binaire de $\mathcal{A} \times \mathcal{O}$ et peut être représentée par une matrice booléenne dont les colonnes sont les attributs et les lignes sont les objets. Cette matrice constitue la représentation originale de la base. Au cours de cet article, nous considérerons que la base de données a plus d'attributs que d'objets et nous utiliserons également la représentation transposée des données, où les attributs de la base sont portés sur les lignes et les objets sur les colonnes (cf. Table 2).

	a_1	a_2	a_3	a_4
o_1	1	1	1	0
o_2	1	1	1	0
o_3	0	1	1	1

	o_1	o_2	o_3
a_1	1	1	0
a_2	1	1	1
a_3	1	1	1
a_4	0	0	1

Table 2: Représentation originale et transposée de la base de données présentée table 1. Les attributs sont $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$ et les objets sont $\mathcal{O} = \{o_1, o_2, o_3\}$. Nous utilisons une notation sous forme de chaîne pour les ensembles, par exemple $a_1a_3a_4$ désigne l'ensemble d'attributs $\{a_1, a_3, a_4\}$ et o_2o_3 désigne l'ensemble d'objets $\{o_2, o_3\}$. Cette base de données sera utilisée dans tous les exemples.

2.1 Correspondance de Galois

L'idée principale qui fonde notre travail est d'utiliser la correspondance forte entre les treillis des $2^{\mathcal{A}}$ et $2^{\mathcal{O}}$, appelée **correspondance de Galois**. Cette correspondance a été utilisée la première fois en fouille de données quand des algorithmes d'extraction des itemsets fermés fréquents ont été proposés (Pasquier *et al.*, 1999) et elle est aussi utilisée dans de nombreux travaux en apprentissage conceptuel (Wille, 1992; Nguifo & Njiwoua, 2000).
Étant donnée une base de données bd, les opérateurs f et g de Galois sont définis par :

- f, appelé *intension*, est une fonction de $2^{\mathcal{O}}$ vers $2^{\mathcal{A}}$ définie par

$$f(O) = \{a \in \mathcal{A} \mid \forall o \in O,\ (a, o) \in bd\}\,,$$

- g, appelé *extension*, est une fonction de $2^{\mathcal{A}}$ vers $2^{\mathcal{O}}$ définie par

$$g(A) = \{o \in \mathcal{O} \mid \forall a \in A,\ (a, o) \in bd\}\,.$$

Pour un ensemble A, $g(A)$ est aussi appelé **l'ensemble support** de A dans bd. C'est l'ensemble des objets qui sont en relation avec tous les attributs de A. La **fréquence** de A dans bd, notée $\mathsf{Freq}(A, bd)$ (ou plus simplement $\mathsf{Freq}(A)$), est définie par $\mathsf{Freq}(A) = |g(A)|$.

Ces deux fonctions créent un lien entre l'espace des attributs et l'espace des objets. Pourtant, comme les deux espaces n'ont a priori pas le même cardinal, aucune bijection n'est possible entre eux. Cela signifie que plusieurs ensembles d'attributs ont la même image par g dans l'espace des objets et vice-versa. On peut donc définir deux relations d'équivalence r_a et r_o sur $2^{\mathcal{O}}$ et $2^{\mathcal{A}}$:

- si A et B sont deux ensembles d'attributs, $A\, r_a\, B$ si $g(A) = g(B)$,
- si O et P sont deux ensembles d'objets, $O\, r_o\, P$ si $f(O) = f(P)$.

Dans chaque classe d'équivalence, il y a un élément particulier : le plus grand élément d'une classe, au sens de l'inclusion, est unique et appelé **ensemble d'attributs fermé**

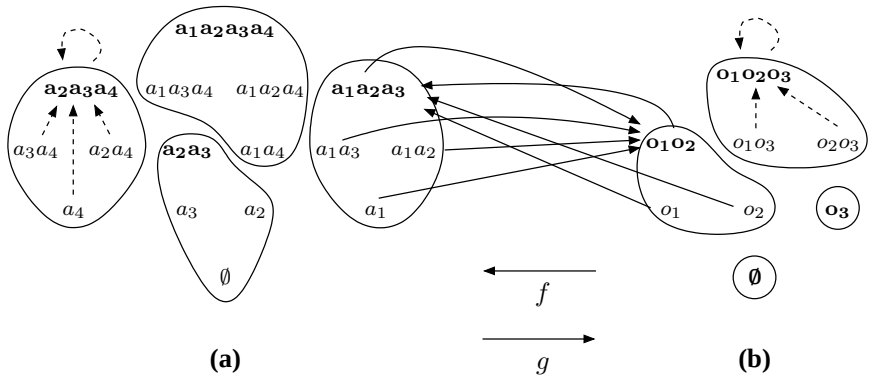

Figure 1: Les classes d'équivalence pour r_a dans le treillis des attributs (a) et pour r_o dans celui des objets (b). Les ensembles fermés sont en gras. Les flèches représentent les opérateurs f et g entre les classes de $a_1a_2a_3$ et o_1o_2. Les flèches en pointillés représentent les opérateurs de clôture h et h'.

pour r_a ou **ensemble d'objets fermé** pour r_o. Les opérateurs f et g de Galois fournissent, par composition, deux opérateurs de **fermeture** notés $h = f \circ g$ et $h' = g \circ f$. Les ensembles fermés sont les points fixes des opérateurs de fermeture et la fermeture d'un ensemble est l'ensemble fermé de sa classe d'équivalence. Dans la suite, nous évoquerons indifféremment h ou h' avec la notation cl.

Une paire (A, O) constituée d'un ensemble d'attributs fermé A et de l'ensemble d'objets fermé correspondant O est appelée un **concept formel**. L'ensemble des concepts de la base de données bd est noté :

$$\mathsf{Concepts}(bd) = \{(A, O) \mid f(O) = A \wedge g(A) = O\}\,.$$

Exemple 1
Dans la figure 1, les ensembles d'objets fermés sont $\emptyset$, o_3, o_1o_2, et $o_1o_2o_3$. Les ensembles d'attributs fermés sont a_2a_3, $a_2a_3a_4$, $a_1a_2a_3$ et $a_1a_2a_3a_4$. Comme $g(o_1o_2) = a_1a_2a_3$ et $f(a_1a_2a_3) = o_1o_2$, $(a_1a_2a_3, o_1o_2)$ est un concept. Les autres concepts sont $(a_2a_3, o_1o_2o_3)$, $(a_2a_3a_4, o_3)$, $(a_1a_2a_3a_4, \emptyset)$.

Propriété 1
A et B sont des ensembles d'attributs, O et P des ensembles d'objets et E un ensemble d'attributs ou d'objets.

- *f sont g sont décroissantes par rapport à l'inclusion : si $A \subseteq B$ alors $g(B) \subseteq g(A)$ et si $O \subseteq P$, $f(P) \subseteq f(O)$;*
- *$f \circ g \circ f = f$;*
- *E est fermé si et seulement si $\mathsf{cl}(E) = E$ et sinon $E \subseteq \mathsf{cl}(E)$;*
- *(A, O) est un concept si et seulement si O est fermé et $A = f(O)$*

2.2 Contraintes

Afin de permettre au biologiste de focaliser son étude sur les concepts qui l'intéressent réellement, nous lui laissons la possibilité de définir une contrainte qui devra être satisfaite par tous les concepts extraits.

Si on note $\mathcal{B}$ l'ensemble des bases de données booléennes (i.e., des matrices booléennes), on appelle **contrainte sur les concepts** une fonction booléenne $\mathcal{C}$ de $2^{\mathcal{A}} \times 2^{\mathcal{O}} \times \mathcal{B}$.

Outre le fait qu'une contrainte permet de mieux cibler les ensembles extraits, leur utilisation, lorsqu'elles sont efficacement intégrées à l'algorithme d'extraction, permet également de réduire considérablement le temps de calcul. C'est ce qui explique l'intérêt croissant ces dernières années pour l'étude des algorithmes d'extraction sous contraintes. Cependant, les contraintes utilisées dans ces algorithmes ne portent généralement que sur les itemsets (et pas simultanément sur les itemsets et les ensembles d'objets). Mais, dans la section suivante, nous verrons comment projeter une contrainte sur les concepts pour obtenir une contrainte ne portant plus que sur les objets, et ainsi pouvoir utiliser des techniques classiques d'extraction sous contraintes (sauf que nous les utiliserons dans les données transposées).

Parmi les contraintes portant sur les itemsets, la plus utilisée est sans doute la contrainte de fréquence minimale $\mathcal{C}_{\gamma\text{-freq}}$. Cette contrainte est satisfaite par les itemsets dont la fréquence est supérieure à un seuil $gamma$ fixé par l'utilisateur : $\mathcal{C}_{\gamma\text{-freq}}(X) = (\mathsf{Freq}(X) > \gamma)$. On peut également être intéressé par sa négation : c'est-à-dire chercher des itemsets suffisamment rares et donc utiliser une contrainte de fréquence maximale. Il existe également de nombreuses contraintes syntaxiques. Une contrainte est syntaxique lorsqu'elle ne dépend pas de la matrice des données bd. Par exemple, la contrainte[2] $\mathcal{C}(A) = a_1 \in A$ est syntaxique, alors que la contrainte de fréquence ne l'est pas (en effet, la fréquence d'un itemset dépend des données).

Parmi les contraintes syntaxiques, les contraintes de "sur-ensemble" et de "sous-ensemble" permettent par combinaison (conjonction, disjonction, négation) de construire les autres contraintes syntaxiques (cf. table 3). Étant donné un ensemble constant E, la contrainte de sous-ensemble $\mathcal{C}_{\subseteq E}$ est définie par : $\mathcal{C}_{\subseteq E}(X) = (X \subseteq E)$. La contrainte de sur-ensemble $\mathcal{C}_{\supseteq E}$ est définie par : $\mathcal{C}_{\supseteq E}(X) = (X \supseteq E)$. Remarquons que comme nous allons ensuite utiliser des contraintes sur les itemsets et les ensembles d'objets, les ensembles X et E peuvent soit être tous les deux des itemsets soit tous les deux des ensembles d'objets.

Lorsqu'une valeur numérique $a.v$ est associée à chaque attribut a (par exemple un coût), on peut définir d'autres contraintes syntaxiques du type (Ng *et al.*, 1998) $\text{MAX}(X)\,\theta\,\alpha$ (où $\theta \in \{<, >, \leq, \geq\}$) pour différents opérateurs d'agrégation tels que MAX, MIN, SOM (la somme), MOY (la moyenne). Parmi ces contraintes, celles qui utilisent les opérateurs MIN et MAX peuvent être récrites simplement en utilisant les contraintes $\mathcal{C}_{\supseteq E}$ et $\mathcal{C}_{\subseteq E}$ en utilisant l'ensemble $\mathsf{sup}_\alpha = \{a \in \mathcal{A} \mid a.v > \alpha\}$ comme indiqué dans la table 3.

Le fait de récrire toutes ces contraintes syntaxiques en utilisant uniquement les contraintes $\mathcal{C}_{\subseteq E}$ et $\mathcal{C}_{\supseteq E}$ nous permettra de limiter le nombre de contraintes à étudier dans la section 3 sur la projection des contraintes.

[2] On notera $\mathcal{C}(A)$ au lieu de $\mathcal{C}(A, O, bd)$ lorsque l'expression de la contrainte $\mathcal{C}$ n'utilise pas O et bd.

$X \not\subseteq E \equiv \neg\mathcal{C}_{\subseteq E}(X)$	$X \cap E = \emptyset \equiv X \subseteq \overline{E}$
$X \not\supseteq E \equiv \neg\mathcal{C}_{\supseteq E}(X)$	$X \cap E \neq \emptyset \equiv \neg(X \subseteq \overline{E})$
$\text{MIN}(X) > \alpha \equiv X \subseteq sup_\alpha$	$\text{MAX}(X) > \alpha \equiv X \cap sup_\alpha \neq \emptyset$
$\text{MIN}(X) \leq \alpha \equiv X \not\subseteq sup_\alpha$	$\text{MAX}(X) \leq \alpha \equiv X \cap sup_\alpha = \emptyset$

$$|X \cap E| \geq 2 \equiv \bigvee_{1 \leq i < j \leq n} e_i e_j \subseteq X$$

Table 3: Exemples de contraintes obtenues par combinaison des contraintes de sur-ensemble et de sous-ensemble. $E = \{e_1, e_2, ..., e_n\}$ est un ensemble constant et X un ensemble variable. Le complémentaire de E dans $\mathcal{A}$ ou dans $\mathcal{O}$ est noté $\overline{E}$.

Finalement, toutes ces contraintes peuvent être combinées pour construire une contrainte sur les concepts, par exemple $\mathcal{C}(A, O) = (a_1 a_2 \subseteq A \wedge (O \cap o_4 o_5 = \emptyset))$.

Pour pouvoir utiliser efficacement les contraintes dans les algorithmes d'extraction, il est nécessaire d'étudier leurs propriétés. Ainsi, deux types de contraintes importantes ont été mises en évidence : les contraintes monotones et les contraintes anti-monotones. Une contrainte $\mathcal{C}$ est **anti-monotone** si $\forall A, B \ (A \subseteq B \wedge \mathcal{C}(B)) \Longrightarrow \mathcal{C}(A)$. $\mathcal{C}$ est **monotone** si $\forall A, B \ (A \subseteq B \wedge \mathcal{C}(A)) \Longrightarrow \mathcal{C}(B)$. Dans les deux définitions, A et B peuvent être des ensembles d'attributs ou d'objets. La contrainte de fréquence est anti-monotone. L'anti-monotonicité est une propriété importante, parce que les algorithmes d'extraction par niveaux l'utilisent la plupart du temps pour élaguer l'espace de recherche. En effet, quand un ensemble ne satisfait pas la contrainte, ses spécialisations non plus et elles peuvent donc être élaguées (Agrawal *et al.*, 1996).

Les compositions élémentaires de telles contraintes ont les mêmes propriétés : la conjonction ou la disjonction de deux contraintes anti-monotones (resp. monotones) est anti-monotone (resp. monotone). La négation d'une contrainte anti-monotone est monotone, et vice-versa.

2.3 Définition du problème

Nous définissons la tâche d'extraction de concepts sous contraintes de la manière suivante : étant donnés une base de données bd et une contrainte $\mathcal{C}$ sur les concepts, nous voulons extraire l'ensemble des concepts qui satisfont $\mathcal{C}$, c'est-à-dire la collection $\{(A, O) \in \mathsf{Concepts}(bd) \mid \mathcal{C}(A, O, bd)\}$.

3 Projections de contraintes

Pour extraire les concepts sous contraintes, nous proposons d'utiliser des techniques classiques d'extraction de fermés sous contraintes dans la matrice transposée. Cependant, dans ces algorithmes, les contraintes possibles portent uniquement sur les itemsets. Donc, s'ils sont utilisés dans la matrice transposée, les contraintes porteront sur les ensembles d'objets.

Pour permettre leur utilisation, nous allons donc étudier dans cette section un mécanisme de "projection" de contrainte : étant donné une contrainte $\mathcal{C}$ portant sur les concepts (c'est-à-dire à la fois sur l'intension et l'extension), nous voulons calculer une contrainte $p(\mathcal{C})$ portant uniquement sur les ensembles d'objets et telle que la collection des ensembles fermés d'objets satisfaisant cette contrainte soit exactement la collection des extensions des concepts satisfaisant $\mathcal{C}$. De cette manière, La collection des concepts satisfaisant $\mathcal{C}$ est exactement la collection des $(f(O), O)$ tels que O est fermé et satisfait la contrainte projetée $p(\mathcal{C})$:

$$\{(A, O) \in \mathsf{Concepts}(bd) \mid \mathcal{C}(A, O, bd)\} = \\ \{(f(O), O) \in \mathcal{A} \times \mathcal{O} \mid p(\mathcal{C})(O, bd) \wedge O \in \mathsf{Fermés}(bd)\}.$$

Ainsi, pour résoudre notre problème, il suffira d'extraire les ensembles fermés d'objets O satisfaisant $p(\mathcal{C})$ et de générer tous les concepts de la forme $(f(O), O)$.

3.1 Définitions et propriétés

Cela signifie que nous voulons extraire des ensembles fermés d'objets O tels que le concept $(f(O), O)$ satisfasse la contrainte $\mathcal{C}$. Par conséquent, une définition naturelle de la projection de la contrainte $\mathcal{C}$ est :

Définition 1 (Contrainte projetée)
Étant donnée une contrainte $\mathcal{C}$ sur les concepts, nous définissons la contrainte projetée de $\mathcal{C}$ de la façon suivante : $p(\mathcal{C})(O, bd) = \mathcal{C}(f(O), O, bd)$.

La proposition suivante assure que l'on obtient bien le résultat voulu :

Proposition 1
Soit $\mathcal{C}$ une contrainte sur les concepts, bd une base de donnée et $p(\mathcal{C})$ la projection de la contrainte $\mathcal{C}$. Alors :

$$\{(A, O) \in \mathsf{Concepts}(bd) \mid \mathcal{C}(A, O, bd)\} = \\ \{(f(O), O) \in \mathcal{A} \times \mathcal{O} \mid p(\mathcal{C})(O, bd) \wedge O \in \mathsf{Fermés}(bd)\}.$$

Preuve : $(A, O) \in \mathsf{Concepts}(bd) \wedge \mathcal{C}(A, O, bd) \Leftrightarrow O \in \mathsf{Fermés}(bd) \wedge A = f(O) \wedge \mathcal{C}(A, O, bd) \Leftrightarrow O \in \mathsf{Fermés}(bd) \wedge \mathcal{C}(f(O), O, bd) \Leftrightarrow O \in \mathsf{Fermés}(bd) \wedge p(\mathcal{C})(O, bd)$. □

Exemple 2
Soit la contrainte $\mathcal{C}(A) = (a_4 \notin A)$. Sa projection est (par définition) : $p(\mathcal{C})(O) = (a_4 \notin f(O))$. Dans la matrice de la table 2, les ensembles fermés d'objets qui satisfont $p(\mathcal{C})$ sont o_1o_2 et $o_1o_2o_3$. Si on calcule les paires $(f(O), O)$ pour ces deux ensembles d'objets, on trouve : $(a_1a_2a_3, o_1o_2)$ et $(a_2a_3, o_1o_2o_3)$ qui sont bien les concepts satisfaisant $\mathcal{C}$.

Par conséquent, pour extraire la collection des concepts qui satisfont $\mathcal{C}$, nous pouvons utiliser des algorithmes classiques d'extraction de fermés dans la matrice transposée

avec la contrainte $p(\mathcal{C})$. Cependant, il faut vérifier que cette contrainte $p(\mathcal{C})$ est effectivement utilisable dans ces algorithmes.

Nous allons commencer par étudier les contraintes complexes, c'est-à-dire des contraintes construites à partir de contraintes plus simples en utilisant des opérateurs booléens comme la conjonction, la disjonction ou la négation.

Proposition 2
Si $\mathcal{C}$ et $\mathcal{C}'$ sont deux contraintes sur les concepts, alors :
$p(\mathcal{C} \wedge \mathcal{C}') = p(\mathcal{C}) \wedge p(\mathcal{C}')$, $\quad p(\mathcal{C} \vee \mathcal{C}') = p(\mathcal{C}) \vee p(\mathcal{C}')$ *et* $\quad p(\neg\mathcal{C}) = \neg p(\mathcal{C})$.

Preuve : Pour la conjonction : $p(\mathcal{C} \wedge \mathcal{C}')(O) = (\mathcal{C} \wedge \mathcal{C}')(f(O), O) = \mathcal{C}(f(O), O) \wedge \mathcal{C}'(f(O), O) = (p(\mathcal{C}) \wedge p(\mathcal{C}'))(O)$. La preuve est similaire pour la disjonction et la négation. □

Cette proposition permet de "pousser" la projection dans les contraintes complexe. L'étape suivante est donc d'étudier ce qui se passe au niveau des contraintes élémentaires.

Exemple 3
Si $\mathcal{C}(A) = (|A| > 4 \wedge \mathsf{Freq}(A) > 2) \vee (A \cap \{a_1 a_4\} \neq \emptyset)$ alors, d'après cette proposition, la projection $p(\mathcal{C})$ de $\mathcal{C}$ est égale à $p(\mathcal{C}) = (p(\mathcal{C}_1) \wedge p(\mathcal{C}_2)) \vee p(\mathcal{C}_3)$ avec $\mathcal{C}_1(A) = |A| > 4$, $\mathcal{C}_2(A) = \mathsf{Freq}(A) > 2$ et $\mathcal{C}_3(A) = (A \cap \{a_1 a_4\} \neq \emptyset)$. Nous verrons dans la section suivante comment calculer les projections de $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$.

Ces contraintes élémentaires peuvent porter sur l'intension du concept (ex : $\mathcal{C}(A, O) = (a_1 \in A)$) ou sur son extension (ex : $\mathcal{C}(A, O) = (|O \cap o_1 o_3 o_5| \geq 2)$. ou enfin sur les deux (Par exemple, la contrainte d'aire minimale sur les concepts : $\mathcal{C}(A, O) = (|A| \,.\, |O|) > \alpha$). Les contraintes élémentaires qui ne portent que sur l'extension des concepts ne sont pas modifiées par la projection, nous allons donc nous focaliser sur les contraintes portant sur les itemsets.

Les contraintes les plus efficacement prises en compte par les algorithmes d'extraction sous contrainte sont les contraintes monotones et anti-monotones. Il est donc important d'étudier comment se comporte la projection de contraintes par rapport à ces propriétés :

Proposition 3
Soit $\mathcal{C}$ une contrainte sur les itemsets :

- *si $\mathcal{C}$ est anti-monotone alors $p(\mathcal{C})$ est monotone ;*
- *si $\mathcal{C}$ est monotone alors $p(\mathcal{C})$ est anti-monotone.*

Preuve : Si O est un ensemble d'objet, $p(\mathcal{C})(O) = \mathcal{C}(f(O))$ par définition de la projection. Or f est décroissante par rapport à l'inclusion (cf. prop. 1) d'où les propriétés. □

Contrainte $\mathcal{C}(A)$	Contrainte projetée $p(\mathcal{C})(O)$
$\mathsf{Freq}(A)\,\theta\,\alpha$	$\lvert O\rvert\,\theta\,\alpha$
$\lvert A\rvert\,\theta\,\alpha$	$\mathsf{Freq}(O)\,\theta\,\alpha$
$A \subseteq E$	si E est fermé : $g(E) \subseteq O$
	sinon : $O \not\subseteq g(f_1) \wedge ... \wedge O \not\subseteq g(f_m)$
$E \subseteq A$	$O \subseteq g(E)$
$A \not\subseteq E$	si E est fermé : $g(E) \not\subseteq O$
	sinon : $O \subseteq g(f_1) \vee ... \vee O \subseteq g(f_m)$
$E \not\subseteq A$	$O \not\subseteq g(E)$
$A \cap E = \emptyset$	si $\overline{E}$ est fermé : $g(\overline{E}) \subseteq O$
	sinon : $O \not\subseteq g(e_1) \wedge ... \wedge O \not\subseteq g(e_n)$
$A \cap E \neq \emptyset$	si $\overline{E}$ est fermé : $g(\overline{E}) \not\subseteq O$
	sinon : $O \subseteq g(e_1) \vee ... \vee O \subseteq g(e_n)$
$\mathrm{SOM}(A)\,\theta\,\alpha$	$\mathsf{Freq}_p(O)\,\theta\,\alpha$
$\mathrm{MOY}(A)\,\theta\,\alpha$	$\mathsf{Freq}_p(O)/\mathsf{Freq}(O)\,\theta\,\alpha$
$\mathrm{MIN}(A) > \alpha$	$p(A \subseteq sup_\alpha)$
$\mathrm{MIN}(A) \leq \alpha$	$p(A \not\subseteq sup_\alpha)$
$\mathrm{MAX}(A) > \alpha$	$p(A \cap sup_\alpha \neq \emptyset)$
$\mathrm{MAX}(A) \leq \alpha$	$p(A \cap sup_\alpha = \emptyset)$
	$\theta \in \{<, >, \leq, \geq\}$

Table 4: Contraintes projetées. A est un ensemble variable d'attributs, $E = \{e_1, e_2, ..., e_n\}$ un ensemble fixé d'attributs, $\overline{E} = \mathcal{A} \setminus E = \{f_1, f_2, ..., f_m\}$ son complémentaire et O un ensemble d'objets fermé.

3.2 Projection de contraintes classiques

Dans la section précédente, nous avons donné la définition de la projection de contrainte. Cette définition fait intervenir $f(O)$. Cela signifie que pour tester la contrainte projetée, il est nécessaire, pour chaque ensemble d'objets O, de calculer son intension $f(O)$. Certains algorithmes, tels que CHARM (Zaki & Hsiao, 2002), utilisent une structure de données particulière –la représentation verticale des données– et par conséquent calculent pour chaque ensemble O l'ensemble $f(O)$. Cependant, beaucoup d'autres algorithmes n'utilisent pas cette structure et ne peuvent donc directement utiliser les contraintes projetées. C'est pour cette raison que dans cette section nous étudions les contraintes projetées de contraintes classiques et nous calculons une expression de ces contraintes ne faisant plus intervenir $f(O)$.

Nous allons d'abord étudier la contrainte de fréquence minimale (qui est la contrainte la plus courante) : $\mathcal{C}_{\gamma\text{-freq}}(A) = (\mathsf{Freq}(A) > \gamma)$. Par définition, sa contrainte projetée est : $p(\mathcal{C}_{\gamma\text{-freq}})(O) = (\mathsf{Freq}(f(O)) > \gamma)$. Par définition de la fréquence, $\mathsf{Freq}(f(O)) = |g(f(O))| = |\mathsf{cl}(O)|$ et si O est un ensemble fermé d'objets, $\mathsf{cl}(O) = O$ et par conséquent $p(\mathcal{C}_{\gamma\text{-freq}})(O) = (|O| > \gamma)$. Finalement, la projection de la contrainte de fréquence minimale est une contrainte de taille minimale. Si on avait considéré la contrainte de fréquence maximale, on aurait évidement trouvé comme projection une

contrainte de taille maximale.

De par la symétrie du problème, il découle que la projection de la contrainte de taille maximale (resp. minimale) est la contrainte de fréquence : si $\mathcal{C}(A) = (|A| \, \theta \, \alpha)$ alors $p(\mathcal{C})(O) = (|f(O)| \, \theta \, \alpha)$. Or $|f(O)|$ est exactement la fréquence de O si on se place dans la matrice transposée.

Les deux propositions suivantes donnent l'expression de la projection des contraintes de sur-ensemble et de sous-ensemble :

Proposition 4
Soit E un itemset, alors :

$$p(\mathcal{C}_{\supseteq E})(O) \equiv g(E) \supseteq \mathsf{cl}(O).$$

Preuve : $p(\mathcal{C}_{\supseteq E})(O) \Leftrightarrow (E \subseteq f(O)) \Rightarrow (g(E) \supseteq g \circ f(O)) \Leftrightarrow (g(E) \supseteq \mathsf{cl}(O))$. Réciproquement, $(g \circ f(O) \subseteq g(E)) \Rightarrow (f \circ g \circ f(O) \supseteq f \circ g(E)) \Rightarrow (f(O) \supseteq \mathsf{cl}(E)) \Rightarrow f(O) \supseteq E$. □

Proposition 5
Soit E un itemset, alors, si E est fermé :

$$p(\mathcal{C}_{\subseteq E})(O) \equiv g(E) \subseteq \mathsf{cl}(O),$$

si E n'est pas fermé, on pose $\overline{E} = \mathcal{A} \setminus E = \{f_1, ..., f_m\}$ et :

$$p(\mathcal{C}_{\subseteq E})(O) \equiv (\mathsf{cl}(O) \not\subseteq g(f_1) \wedge \mathsf{cl}(O) \not\subseteq g(f_2) \wedge ... \wedge \mathsf{cl}(O) \not\subseteq g(f_m).$$

Preuve : $p(\mathcal{C}_{\subseteq E})(O) \Leftrightarrow \mathcal{C}_{\subseteq E}(f(O)) \Leftrightarrow (f(O) \subseteq E) \Rightarrow (g \circ f(O) \supseteq g(E)) \Leftrightarrow (\mathsf{cl}(O) \supseteq g(E))$. Réciproquement, (si E est fermé): $(g(E) \subseteq g \circ f(O)) \Rightarrow (f \circ g(E) \supseteq f \circ g \circ f(O)) \Rightarrow (\mathsf{cl}(E) \supseteq f(O)) \Rightarrow (E \supseteq f(O))$. Si E n'est pas fermé, on récrit la contrainte : $(A \subseteq E) = f_1 \notin A \wedge ... \wedge f_m \notin A$ et on utilise les propositions 2 et 4. □

La table 4 récapitule les contraintes projetées de contraintes classiques. Les contraintes de fréquence et de taille ont été traitées plus haut. Les deux propriétés précédentes, avec l'aide de la table 3 et de la proposition 2 nous permettent de calculer la projection des contraintes syntaxiques, exceptées les contraintes utilisant les opérateurs d'agrégation MOY et SOM. Dans cette table, on suppose que l'ensemble d'objets O est fermé. Cela n'est pas une restriction importante dans la mesure où nous ne nous intéressons qu'à des algorithmes d'extraction de fermés (ces fermés serviront à générer les concepts).

Examinons maintenant les contraintes utilisant les opérateurs d'agrégation MOY et SOM. Par définition, les contraintes projetées sont : $\text{MOY}(f(O)) \, \theta \, \alpha$ et $\text{SOM}(f(O)) \, \theta \, \alpha$. Il faut donc trouver une expression de $\text{MOY}(f(O))$ et $\text{SOM}(f(O))$ ne faisant plus intervenir f. En fait, il suffit d'étudier l'opérateur SOM car $\text{MOY}(f(O)) = \text{SOM}(f(O)) / |f(O)| = \text{SOM}(f(O)) / \mathsf{Freq}(O)$ donc si nous trouvons une expression de $\text{SOM}(f(O))$ dans la base projetée, nous obtiendrons aussi une expression pour $\text{MOY}(f(O))$.

L'ensemble $f(O)$ est un ensemble d'attribut, donc dans la matrice transposée, c'est un ensemble de lignes. Les valeurs $a.v$ sur lesquelles la somme est calculée sont attachées aux attributs a et donc aux lignes de la matrice transposée. La valeur $\text{SOM}(f(O))$ est

donc la somme de ces valeurs v sur toutes les lignes de $f(O)$, c'est-à-dire les lignes contenant O. Autrement dit, $\text{SOM}(f(O))$ est une fréquence pondérée par les valeurs v (nous notons cette fréquence pondérée Freq_p). Celle-ci peut être facilement calculée par les algorithmes en plus de la fréquence "classique" Freq. Il suffit pour cela, lors de la passe sur les données, d'incrémenter cette fréquence pondérée de $a.v$ pour chaque ligne a contenant O.

Ces expressions de la contrainte projetée sont intéressantes car elles n'impliquent plus le calcul de $f(O)$ pour chaque ensemble devant être testé. Les ensembles $g(\overline{E})$ or $g(e_i)$ qui apparaissent dans ces contraintes peuvent quant à eux être calculés une fois pour toute lors de la première passe sur les données (en effet, l'ensemble E est constant).

Exemple 4
Considérons la contrainte $\mathcal{C}_3(A) = (A \cap \{a_1a_4\} \neq \emptyset)$ de l'exemple précédent. Dans la table 2, l'itemset $\overline{a_1a_4} = a_2a_3$ est fermé. Par conséquent, la contrainte projetée est $p(\mathcal{C}_3)(O) = (g(a_2a_3) \not\subseteq O)$. Comme $g(a_2a_3) = o_1o_2o_3$, $p(\mathcal{C}_3)(O) = (o_1o_2o_3 \not\subseteq O)$. La projection de la contrainte $\mathcal{C}(A) = (|A| > 4 \wedge \text{Freq}(A) > 2) \vee (A \cap \{a_1a_4\} \neq \emptyset)$ de l'exemple 3 est donc : $p(\mathcal{C})(O) = (\text{Freq}(O) > 4 \wedge |O| > 2) \vee (o_1o_2o_3 \not\subseteq O)$.

4 Utilisation de la projection de contraintes

Dans cette section, nous présentons deux stratégies pour extraire les concepts satisfaisant une contrainte $\mathcal{C}$ et ainsi résoudre le problème posé dans la section 2.3.

La première stratégie utilise les algorithmes classiques d'extraction de fermés :

1. Calculer la contrainte projetée $p(\mathcal{C})$ de $\mathcal{C}$ en utilisant la table 4 et la propriété 2 ;
2. Utiliser un algorithme pour l'extraction de fermés sous contraintes dans la matrice transposée (comme par exemple, ceux proposés dans (Bonchi & Lucchese, 2004) ou (Boulicaut & Jeudy, 2001)) avec la contrainte $p(\mathcal{C})$. Il est aussi possible d'utiliser des algorithmes d'extraction de fermés fréquent tels que CHARM (Zaki & Hsiao, 2002), CARPENTER (Pan *et al.*, 2003) ou CLOSET (Pei *et al.*, 2000) en leur rajoutant une étape d'élagage supplémentaire pour traiter la contrainte (à la manière de ce qui est fait dans (Pei & Han, 2000)).
3. Ces algorithmes extraient des ensembles fermés. Cela signifie qu'ils vont retourner les ensembles d'objets fermés (car nous travaillons dans la matrice transposée) qui satisfont la contrainte $p(\mathcal{C})$. Il faut alors pour chacun de ces fermés calculer son intension $f(O)$, d'après la proposition 1, les paires $(f(O), O)$ ainsi formées seront exactement les concepts qui satisfont la contrainte $\mathcal{C}$. Le calcul de $f(O)$ peut être fait lors d'une dernière passe sur les données ou alors intégré dans les algorithmes. En fait, ces algorithmes calculent les intensions lors du calcul de la fréquence des ensembles (la fréquence de O est $|f(O)|$). Il suffit donc de les modifier pour qu'ils stockent ces intensions.

Exemple 5
Imaginons que nous voulions extraire les concepts satisfaisant la contrainte $\mathcal{C}(A) = (A \cap \{a_1a_4\} \neq \emptyset)$ avec cette stratégie. La projection de $\mathcal{C}$ est (cf. exemple 4) :

$p(\mathcal{C})(O) = (o_1o_2o_3 \not\subseteq O)$. *Les ensembles fermés d'objets qui satisfont cette contrainte sont* $T = \{\emptyset, o_1o_2, o_3\}$ *(calculés dans la matrice transposée avec un algorithme d'extraction de fermés sous contraintes). Nous pouvons ensuite calculer les concepts correspondants qui sont :* $(a_1a_2a_3a_4, \emptyset)$, $(a_1a_2a_3, o_1o_2)$ *et* $(a_2a_3a_4, o_3)$.

La seconde stratégie est basée sur le nouvel algorithme D-Miner (Besson *et al.*, 2004). Cet algorithme extrait des concepts sous une contrainte $\mathcal{C}$ qui est la conjonction d'une contrainte monotone sur les attributs et d'une contrainte monotone sur les objets. Il ne peut cependant pas traiter le cas où des contraintes anti-monotones sont utilisées.

Notre stratégie consiste alors à projeter les contraintes anti-monotones définies dans l'espace des attributs sur l'espace des objets et à projeter les contraintes anti-monotones définies dans l'espace des objets sur l'espace des attributs. En effet, d'après la proposition 3, la projection transforme une contrainte anti-monotone en une contrainte monotone. Cela permet donc d'utiliser D-Miner avec des contraintes monotones et anti-monotones. Nous n'avons présenté que la projection des contraintes de l'espace des attributs sur l'espace des objets. Cependant, la projection dans l'autre sens est similaire. En fait, il suffit de remplacer la fonction f par la fonction g.

5 Conclusion

L'analyse des données d'expression de gènes pose un problème spécifique pour l'extraction de motifs : les données contiennent beaucoup plus de colonnes que de lignes, ce qui rend les algorithmes d'extraction classiques inopérants. Dans ce cas, extraire les motifs dans la matrice transposée permet de s'affranchir de ce problème.

La transposition a déjà été étudiée dans le cas de la contrainte de fréquence, mais l'étude générale de ce qui se passe dans le cas d'une contrainte complexe restait à faire. Cette étude nous a permis de proposer des stratégies pour extraire des concepts sous contraintes. Ces stratégies, plutôt que de proposer un nouvel algorithme, se fondent sur l'utilisation d'algorithmes classiques et éprouvés d'extraction de fermés ou de concepts. Afin de rendre leur utilisation possible, nous avons défini une opération de projection des contraintes et nous avons étudié ses propriétés ainsi que les projections de contraintes classiques.

References

AGRAWAL R., MANNILA H., SRIKANT R., TOIVONEN H. & VERKAMO A. I. (1996). Fast discovery of association rules. In U. M. FAYYAD, G. PIATETSKY-SHAPIRO, P. SMYTH & R. UTHURUSAMY, Eds., *Advances in Knowledge Discovery and Data Mining*, p. 307–328. Menlo Park : AAAI Press.

ALBERT-LORINCZ H. & BOULICAUT J.-F. (2003). Mining frequent sequential patterns under regular expressions: a highly adaptative strategy for pushing constraints. In *Third SIAM International Conference on Data Mining (SIAM DM'03)*, p. 316–320.

BESSON J., ROBARDET C. & BOULICAUT J.-F. (2004). Constraint-based mining of formal concepts in transactional data. In H. DAI, R. SRIKANT & C. ZHANG, Eds., *Proceedings of the*

8th Pacif-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04), volume 3056 of *Lecture Notes in Computer Science*, p. 615–624, Sydney, Australia.

BONCHI F., GIANNOTTI F., MAZZANTI A. & PEDRESCHI D. (2003). Exante: Anticipated data reduction in constrained pattern mining. In *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, volume 2838 of *Lecture Notes in Artificial Intelligence*, Cavtat-Dubrovnik, Croatia.

BONCHI F. & LUCCHESE C. (2004). On closed constrained frequent pattern mining. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, Brighton, UK.

BOULICAUT J.-F., BYKOWSKI A. & RIGOTTI C. (2000). Approximation of frequency queries by mean of free-sets. In D. ZIGHED, J. KOMOROWSKI & J. M. ZYTKOW, Eds., *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00)*, volume 1910 of *Lecture Notes in Artificial Intelligence*, p. 75–85, Lyon, France: Springer-Verlag.

BOULICAUT J.-F., BYKOWSKI A. & RIGOTTI C. (2003). Free-sets : a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, **7**(1), 5–22.

BOULICAUT J.-F. & JEUDY B. (2000). Using constraint for itemset mining: should we prune or not? In A. DOUCET, Ed., *Actes des 16e Journées Bases de Données Avancées (BDA'00)*, p. 221–237, Blois, France: Université de Tours.

BOULICAUT J.-F. & JEUDY B. (2001). Mining free-sets under constraints. In M. E. ADIBA, C. COLLET & B. C. DESAI, Eds., *Proceedings of the International Database Engineering & Applications Symposium (IDEAS'01)*, p. 322–329, Grenoble, France: IEEE Computer Society.

BUCILA C., GEHRKE J. E., KIFER D. & WHITE W. (2003). Dualminer: A dual-pruning algorithm for itemsets with constraints. *Data Mining and Knowledge Discovery*, **7**(4), 241–272.

GAROFALAKIS M. M., RASTOGI R. & SHIM K. (1999). SPIRIT: Sequential pattern mining with regular expression constraints. In M. P. ATKINSON & OTHERS, Eds., *Proceedings of the 25nd International Conference on Very Large Data Bases (VLDB'99)*, p. 223–234, Edinburgh, UK: San Francisco : Morgan Kaufmann.

NG R., LAKSHMANAN L. V., HAN J. & PANG A. (1998). Exploratory mining and pruning optimizations of constrained associations rules. In L. M. HAAS & A. TIWARY, Eds., *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'98)*, volume 27(2) of *SIGMOD Record*, p. 13–24, Seattle, Washington, USA: New York : ACM Press.

NGUIFO E. M. & NJIWOUA P. (2000). GLUE: a lattice-based constructive induction system. *Intelligent Data Analysis*, **4**(4), 1–49.

PAN F., CONG G., TUNG A. K. H., YANG J. & ZAKI M. J. (2003). CARPENTER: Finding closed patterns in long biological datasets. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington DC: New York : ACM Press.

PASQUIER N., BASTIDE Y., TAOUIL R. & LAKHAL L. (1999). Efficient mining of association rules using closed itemset lattices. *Information Systems*, **24**(1), 25–46.

PEI J. & HAN J. (2000). Can we push more constraints into frequent pattern mining? In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining (KDD'00)*, p. 350–354, Boston, USA: New York : ACM Press.

PEI J., HAN J. & MAO R. (2000). CLOSET an efficient algorithm for mining frequent closed itemsets. In D. GUNOPULOS & R. RASTOGI, Eds., *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'00)*, Dallas, Texas,

USA. Disp. en ligne (sept. 2002) `http://www.cs.ucr.edu/~dg/DMKD.html`, 10 pages.

RIOULT F., BOULICAUT J.-F., CRÉMILLEUX B. & BESSON J. (2003). Using transposition for pattern discovery from microarray data. In *8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, p. 73–79, San Diego, USA.

RIOULT F. & CRÉMILLEUX B. (2003). Optimisation d'extraction de motifs : une nouvelle méthode fondée sur la transposition de données. In *Conférence d'Apprentissage, CAp'03*, p. 299–313.

SRIKANT R., VU Q. & AGRAWAL R. (1997). Mining association rules with item constraints. In D. HECKERMAN, H. MANNILA, D. PREGIBON & R. UTHURUSAMY, Eds., *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*, p. 67–73, Newport Beach, California, USA: Menlo Park : AAAI Press.

WILLE R. (1992). Concept lattices and conceptual knowledge systems. *Computer mathematic applied*, **23**((6-9)), 493–515.

ZAKI M. J. (2000). Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the 9th ACM International Conference on Information and Knowledge Management (CIKM'00)*, p. 422–429, Washington DC, USA: New York : ACM Press.

ZAKI M. J. & HSIAO C.-J. (2002). CHARM: An efficient algorithm for closed itemset mining. In R. GROSSMAN, J. HAN, V. KUMAR, H. MANNILA & R. MOTWANI, Eds., *2nd SIAM International Conference on Data Mining (SIAM DM'02)*, Arlington, USA. Disp. en ligne (sept. 2002) `http://www.siam.org/meetings/sdm02/`, 17 pages.