Draft Revue Information Interaction Intelligence I3. J-M. Petit et P. Poncelet (Eds).
Cepadues Edititions. Numéro Hors-Série 2007, pp. 141-160.

Un algorithme générique d'extraction de bi-ensembles sous contraintes dans des données booléennes

Jérémy Besson[⋆†] Céline Robardet[⋆] Jean-François Boulicaut[⋆]

[⋆] INSA Lyon, LIRIS CNRS UMR 5205, F-69621 Villeurbanne
[†] UMR INRA/INSERM 1235, F-69372 Lyon cedex 08

Résumé

L'existence d'algorithmes complets pour l'extraction sous contraintes de concepts formels (e.g., tous les concepts fréquents) permet d'exploiter de grands jeux de données et a de nombreuses applications. Cependant, lorsque les données traitées sont bruitées, il semble important d'introduire une tolérance aux exceptions. Pour cela, nous avons plongé le calcul de concepts formels dans un cadre plus général d'extraction de bi-ensembles sous contraintes, i.e., un calcul d'associations d'ensembles d'objets et d'ensembles d'attributs satisfaisant des contraintes. Nous proposons un algorithme original, correct, et complet pour calculer des collections de bi-ensembles sous contraintes. Sa généricité permet de discuter de mécanismes fondamentaux comme la spécification déclarative des propriétés des motifs, l'énumération ou encore la propagation des contraintes. Deux instances concrètes de l'algorithme sont présentées et évaluées.

Mots-clés *: Fouille de données, concepts formels, tolérance aux exceptions*

Abstract

The availability of complete algorithms for constraint-based mining of formal concepts (e.g., frequent ones) enables to process large boolean datasets and can be used in many application domains. Considering noisy data, it seems important to extend formal concepts towards fault-tolerance. We embed formal concept extraction into a more general framework for constraint-based mining of bi-sets, i.e., computing associations of sets of objects with sets of attributes that satisfy some constraints. We propose an original, correct and complete algorithm for constraint-based bi-set mining. Its genericity supports the discussion of fundamental mechanisms like the declarative specification of the pattern properties, the enumeration and the constraint propagation strategies. Two concrete instances of this algorithm are presented and evaluated.

Key-words*: Data mining, formal concepts, fault-tolerance*

1 INTRODUCTION

De nombreux processus de découverte de connaissances peuvent s'appuyer sur la fouille de données booléennes, c'est-à-dire l'analyse de très grandes matrices codant l'association entre des objets et des propriétés. La découverte de connaissances à partir de motifs ensemblistes (e.g., des ensembles fréquents, des règles d'association à forte confiance, des concepts formels) dans de telles données a été très étudiée (voir par exemple les articles de synthèse dans le numéro spécial [10]). Les chercheurs en fouille de données ont proposé de nombreux algorithmes complets pour le calcul de tous les motifs ensemblistes satisfaisant certaines contraintes. C'est notamment le cas des extractions sous contraintes de concepts formels (voir, par exemple, [24, 7, 5]). Intuitivement, les concepts formels sont des rectangles maximaux de valeurs vraies (modulo des permutations des lignes et des colonnes). Formellement, ce sont des ensembles fermés d'objets associées à des ensembles fermés de propriétés. Dans le domaine d'application que nous privilégions, i.e., la biologie moléculaire, les concepts formels peuvent être vus comme des modules de transcription putatifs dans des données qui codent la sur-expression de gènes dans certaines situations biologiques [4, 5]. Il existe d'autres applications à la biologie moléculaire comme, par exemple, [18].

Lorsque les relations codées sont susceptibles d'être bruitées, le nombre de concepts formels explose exponentiellement. C'est le cas lorsque les valeurs booléennes codées sont le résultat de traitements sur des données numériques issues de processus expérimentaux complexes, par exemple la mesure de niveaux d'expression de gènes par une technologie Puce ADN.

Pour remédier à ces difficultés, il nous a semblé important de plonger le calcul de concepts formels dans un cadre plus général d'extraction de bi-ensembles sous contraintes, i.e., le calcul d'associations d'ensembles d'objets et d'ensembles de propriétés qui satisfont certaines contraintes. On peut alors considérer l'extension des concepts formels vers une tolérance aux exceptions (voir notamment [6]). En effet, si les concepts formels ne correspondent qu'à des relations enregistrées (présence de valeurs vraies uniquement), nous pouvons relâcher cette condition pour tolérer certaines absences de relations.

Dans cet article, nous proposons un algorithme générique pour calculer des bi-ensembles sous contraintes. C'est un algorithme original inspiré du cadre mono-dimensionnel appelé DUALMINER [12]. Nous sommes donc dans la lignée des travaux comme [17] qui introduisait en 1997 l'algorithme LEVELWISE. Il s'agissait d'une abstraction de plusieurs algorithmes déjà présentés pour l'extraction d'ensembles fréquents (typiquement APRIORI [2]) mais aussi de dépendances d'inclusion ou de dépendances fonctionnelles. La généricité de LEVELWISE a inspiré de très nombreux travaux, qu'il s'agisse des usages multiples du concept de frontière et ses relations avec les classiques espaces des versions [19] ou bien de l'identification de mécanismes fondamentaux concernant les processus corrects et complets

d'énumération et/ou d'élagage dans de grands espaces de recherche. Dans le même ordre d'idée, nous pensons que la généricité de l'algorithme présenté ici permet de discuter de mécanismes fondamentaux comme la spécification déclarative des propriétés des motifs (e.g., le type de tolérance aux exceptions), l'énumération ou la propagation des contraintes dans le cadre plus complexe des bi-ensembles. Nous pouvons donc mieux comprendre les développements actuels ou les algorithmes ad-hoc, par exemple [23]. L'algorithme présenté résulte d'ailleurs d'une abstraction de D-MINER [7] pour l'extraction de concepts formels satisfaisant des contraintes de taille minimale sur les deux dimensions et DR-MINER [6] pour l'extraction de motifs tolérants aux exceptions.

La section 2 présente les types de motifs utilisés et formalise le problème traité. La section 3 introduit le cadre et les mécanismes d'élagage qui seront utilisés. La section 4 concerne la définition de l'algorithme générique. Dans la section 5, nous discutons des instances D-MINER et DR-MINER. Enfin, la section 6 est une brève conclusion.

2 DOMAINES DE MOTIFS

Soient G et M deux ensembles, appelés respectivement l'ensemble des objets et l'ensemble des attributs. Soit $\mathbf{r}$ une relation binaire entre objets et attributs, $\mathbf{r} \subseteq G \times M$ telle que $g \in G$, $m \in M$, $\mathbf{r}(g, m)$ si et seulement si l'objet g possède l'attribut m. $K = (G, M, \mathbf{r})$ est classiquement employé pour désigner le contexte d'extraction. Il peut être représenté sous la forme d'une matrice booléenne : les lignes correspondent aux objets et les colonnes aux attributs. On parle aussi de données transactionnelles par référence aux applications pour lequelles les objets sont des transactions et les attributs codent la présence ou l'absence d'items dans ces transactions.

La Figure 1 est un exemple de contexte concernant 5 objets et 4 attributs, i.e., $(\{g_1, g_2, g_3, g_4, g_5\}, \{m_1, m_2, m_3, m_4\}, \mathbf{r}_1)$.

	m_1	m_2	m_3	m_4
g_1	1	1	1	1
g_2	1	1	0	0
g_3	0	1	1	0
g_4	0	0	1	0
g_5	0	0	0	0

FIG. 1 – Contexte booléen $\mathbf{r}_1$

Dans un contexte $K = (G, M, \mathbf{r})$, un bi-ensemble (X, Y) est un couple d'ensembles composé d'un ensemble d'objets $X \subseteq G$ et d'un ensemble d'attributs $Y \subseteq M$. Certains types de bi-ensembles ont été très étudiés comme les itemsets et leurs ensembles d'objets supports (voir, e.g., [1, 2]),

les concepts formels (voir, e.g., [14, 15]), ou encore certains types de motifs tolérants aux exceptions (voir, e.g., [6, 22]).

Donnons des définitions originales de ces domaines de motifs. Nous utilisons les symboles $\bigwedge_{i\in E}$, $\bigvee_{i\in E}$, $\sum_{i\in E}$ et $Max_{i\in E}$ pour désigner les fonctions d'agrégation sur un ensemble E associées respectivement à la conjonction et à la disjonction booléenne, à l'addition et à la valeur maximum sur les réels. Ainsi, $\bigwedge_{i\in\{a,b,c,d\}} p(i) = p(a) \wedge p(b) \wedge p(c) \wedge p(d)$.

Définition 1 (Itemset et ensemble support)
Un bi-ensemble (X, Y) est un itemset avec son ensemble support ssi la contrainte $\mathcal{C}_{IS}(X, Y) \equiv \bigwedge_{i\in X} \bigwedge_{j\in Y} \mathbf{r}(i, j) \wedge \bigwedge_{i\in G\setminus X} \bigvee_{j\in Y} \neg\mathbf{r}(i, j)$ est satisfaite.

Exemple. $(\{g_1, g_2\}, \{m_1\})$ et $(\{g_1\}, \{m_2, m_3, m_4\})$ sont des itemsets avec ensembles supports dans $\mathbf{r}_1$.

Le problème classique de l'extraction des itemsets fréquents revient à calculer tous les itemsets dont l'ensemble support a une taille minimale.

Définition 2 (Concept formel)
Un bi-ensemble (X, Y) est un concept formel ssi il satisfait la contrainte

$$\mathcal{C}_{CF}(X, Y) \equiv \begin{cases} \bigwedge_{i\in X} \bigwedge_{j\in Y} \mathbf{r}(i, j) \\ \bigwedge_{i\in G\setminus X} \bigvee_{j\in Y} \neg\mathbf{r}(i, j) \\ \bigwedge_{j\in M\setminus Y} \bigvee_{i\in X} \neg\mathbf{r}(i, j) \end{cases}$$

Exemple. $(\{g_1, g_2\}, \{m_1, m_2\})$ et $(\{g_1, g_3\}, \{m_2, m_3\})$ sont des concepts formels dans $\mathbf{r}_1$.

Dans ce cas, les concepts formels sont définis comme des rectangles maximaux de "1" [20]. Plus classiquement, ils sont définis en termes d'ensembles fermés associés par une connexion de Galois [26].

Définition 3 (Motif tolérant au bruit)
Un bi-ensemble (X, Y) est un motif tolérant au bruit ssi il satisfait la contrainte $\mathcal{C}_{DRBS}$:
$\mathcal{Z}_G(x, Y) = \sum_{i\in Y} \delta(\neg\mathbf{r}(x, i))$
$\mathcal{Z}_M(y, X) = \sum_{i\in X} \delta(\neg\mathbf{r}(i, y))$ *Avec* $\delta(\text{TRUE}) = 1$ *et* $\delta(\text{FALSE}) = 0$

$$\mathcal{C}_{DRBS}(X, Y) \equiv \begin{cases} \bigwedge_{i\in X} \mathcal{Z}_G(i, Y) \leq \alpha \\ \bigwedge_{i\in Y} \mathcal{Z}_M(i, X) \leq \alpha \\ \bigwedge_{i\in G\setminus X} (Max_{j\in X}(\mathcal{Z}_G(j, Y)) \leq \mathcal{Z}_G(i, Y)) \\ \bigwedge_{i\in M\setminus Y} (Max_{j\in Y}(\mathcal{Z}_M(j, X)) \leq \mathcal{Z}_M(i, X)) \end{cases}$$

α représente le nombre maximum de 0 admis par ligne et par colonne dans le bi-ensemble. Il faut noter que des valeurs différentes de α peuvent être utilisées pour les deux dimensions.

Les deux dernières contraintes imposent que les lignes et les colonnes à l'extérieur des bi-ensembles contiennent plus de valeurs "0" qu'à l'intérieur et ainsi améliorent sensiblement la pertinence des motifs. En revanche, ces contraintes n'imposent pas la maximalité des bi-ensembles. Il faut post-traiter les collections afin d'obtenir cette propriété [6].

Exemple. $(\{g_1, g_2, g_3\}, \{m_1, m_2, m_3\})$ et $(\{g_1, g_2, g_3, g_4\}, \{m_2, m_3\})$ sont des motifs tolérants au bruit dans $\mathbf{r}_1$ lorsque $\alpha = 1$.

Ces motifs ont été utilisés à la fois comme vecteurs de découverte d'information dans de nombreux domaines d'application (Web, post-génomique,...) mais aussi pour aider à la construction d'autres motifs plus complexes. La justesse et la complétude des extractions nécessite l'utilisation de contraintes pour sélectionner les motifs pertinents pour la tâche d'analyse des données. L'utilisation active des contraintes au cours de l'extraction est cruciale pour la faisabilité des extractions, permettant de ne parcourir qu'une petite partie de l'espace de recherche sans manquer de motifs solutions.

Les contraintes monotones et anti-monotones (voir Définition 4) peuvent être utilisées pour élaguer l'espace de recherche.

Définition 4 (Contraintes monotones et anti-monotones)
Une contrainte $\mathcal{C} : E \rightarrow \{\text{TRUE,FALSE}\}$ est anti-monotone par rapport à une relation d'ordre $\preceq$ sur E ssi $\forall a, b \in E$ tel que $a \preceq b$, $\mathcal{C}(b) \Rightarrow \mathcal{C}(a)$, la contraposée étant $\neg\mathcal{C}(a) \Rightarrow \neg\mathcal{C}(b)$. De façon duale, $\mathcal{C} : E \rightarrow \{\text{TRUE,FALSE}\}$ est monotone par rapport à $\preceq$ ssi $\forall a, b \in E$ tel que $a \preceq b$, $\mathcal{C}(a) \Rightarrow \mathcal{C}(b)$, la contraposée étant $\neg\mathcal{C}(b) \Rightarrow \neg\mathcal{C}(a)$. On utilisera l'expression "(anti-)monotone" pour désigner une contrainte monotone ou anti-monotone. Une contrainte à plusieurs paramètres est dite monotone (resp. anti-monotone) ssi elle est monotone (resp. anti-monotone) sur chacun de ses paramètres.

Exemple. $\mathcal{C}_\alpha \equiv |X| > \alpha$ est monotone, par exemple comme $\{g_1, g_2, g_3\}$ satisfait $\mathcal{C}_{\alpha=2}$ alors tous les sur-ensembles de $\{g_1, g_2, g_3\}$ la satisfont aussi. La Figure 2 présente des exemples de contraintes (anti-)monotones suivant l'inclusion ensembliste.

Contrainte $\mathcal{C}(X)$	
monotone	anti-monotone
$\lvert X\rvert > \alpha$	$\lvert X\rvert < \alpha$
$x \in X$	$x \notin X$
$E \subseteq X$	$X \subseteq E$

FIG. 2 – Exemples de contraintes monotones et anti-monotones selon $\subseteq$

Propriété 1
Une conjonction et disjonction de contraintes (anti-)monotones est une contrainte (anti-)monotone.

Les contraintes dites convertibles [21] (voir Définition 5) peuvent aussi être exploitées pour réduire l'espace de recherche.

Définition 5 (Contraintes convertibles)
Une contrainte $\mathcal{C} : E \rightarrow \{\text{TRUE,FALSE}\}$ est anti-monotone convertible ssi (i) $\mathcal{C}$ n'est pas anti-monotone (ii) il existe un ordre total sur E tel que si un motif satisfait la contrainte alors tous ses préfixes la satisfont aussi. La contraposée indique que si un motif E ne satisfait pas la contrainte alors tous les motifs ayant E comme préfixe ne la satisfont pas non plus. Par la suite, nous emploierons le terme convertible pour désigner les contraintes anti-monotones convertibles.

Exemple. Soit $val : G \rightarrow \mathbb{R}$ une fonction, $\mathcal{C}(X) \equiv \sum_{i \in X} val(i)/|X| > \alpha$ est une contrainte convertible. Soit la relation d'ordre $\subseteq_{conv}$ telle que $\forall\ i,\ j \in G,\ j \subseteq_{conv} i \Leftrightarrow val(i) \leq val(j)$. Alors si $\{g_i, g_j, ..., g_k\}$, un ensemble ordonné par $\subseteq_{conv}$ sur G, ne satisfait pas $\mathcal{C}$ alors tous les ensembles $\{g_i, g_j, ..., g_k, g_l\}$ tels que $g_k \subseteq_{conv} g_l$ ne satisfont pas $\mathcal{C}$.

De nombreux algorithmes d'extraction d'itemsets fréquents et de concepts formels capables d'exploiter les contraintes anti-monotones et convertibles suivant l'inclusion ensembliste ont été proposés comme APRIORI [2], Ganter [13], PASCAL [3], CHARM [27], AC-MINER [9] ou encore CLOSET [25].

Tous ces algorithmes, et bien d'autres, exploitent le même principe algorithmique :

- Énumération des ensembles d'objets ou d'attributs suivant une relation de spécialisation
- Élagage de l'espace de recherche avec les contraintes anti-monotones et convertibles vis-à-vis de cette relation de spécialisation
- Génération de l'ensemble de l'autre dimension par la connexion de Galois [26]
- Post-traitement de la collection finale pour exploiter les contraintes qui ne sont ni anti-monotones ni convertibles.

Pour les itemsets, l'énumération est réalisée sur les attributs alors que pour les concepts formels elle est effectuée sur la plus petite des deux dimensions. Généralement, l'inclusion ensembliste est utilisée comme relation d'ordre pour effectuer l'énumération, i.e., comme relation de spécialisation structurant l'espace de recherche. Cette stratégie pose un problème important puisque seules les contraintes anti-monotones ou convertibles sont exploitées pour réduire l'espace de recherche. Ces classes de contraintes sont trop restreintes en pratique. Ainsi, pour les itemsets et les concepts formels, lorsque l'ensemble énuméré diminue, on observe que la taille de l'ensemble associé dans l'autre dimension augmente. Ainsi, une contrainte qui est anti-monotone pour les ensembles d'une dimension (e.g., les attributs) devient

monotone pour les ensembles de l'autre dimension (e.g., les objets). C'est le cas de l'énumération de l'algorithme APRIORI avec la classique contrainte de fréquence minimale (contrainte de taille sur les ensembles d'objets support pour les itemsets) qui est bien connue comme anti-monotone alors que sa contrepartie sur la taille minimale des ensembles d'attributs est monotone. Clairement, APRIORI ne peut exploiter efficacement que la contrainte de taille minimale sur les objets mais pas celle sur les attributs.

L'utilisation des contraintes convertibles impose un ordre d'énumération des éléments de G ou de M. Or, cet ordre d'énumération a une grande influence sur l'efficacité des algorithmes d'extraction de bi-ensembles. Chaque algorithme adopte sa propre stratégie d'énumération, et, par exemple, commence par les attributs les plus fréquents. Imposer un ordre d'énumération apriori pour exploiter les contraintes convertibles réduit considérablement l'efficacité même de l'algorithme ainsi que le bénéfice d'une utilisation active des contraintes convertibles. De plus, une conjonction de contraintes convertibles n'est pas forcement convertible. Cet ordre d'énumération impose aussi de ne réaliser aucune propagation de contraintes qui aurait comme effet d'ajouter des éléments de la dimension énumérée dans le bi-ensemble, ce qui peut encore une fois réduire l'efficacité des calculs.

Nous présentons maintenant un cadre générique permettant d'extraire efficacement des itemsets, des concepts formels, mais aussi d'autres types de bi-ensembles satisfaisant des contraintes sur les deux dimensions. Nous souhaitons pouvoir extraire les collections de bi-ensembles suivantes :

$$\theta = \{(X, Y) \mid \mathfrak{C}(X, Y, \mathbf{r})\}$$

où $\mathfrak{C}$ est une conjonction de disjonctions de contraintes primitives. Nous verrons par la suite les propriétés que doit satisfaire $\mathfrak{C}$.

3 UN CADRE GÉNÉRIQUE

Nous utilisons une relation d'ordre $\preceq_{BE}$ sur les bi-ensembles telle que $(X_1, Y_1) \preceq_{BE} (X_2, Y_2) \iff X_1 \subseteq X_2$ et $Y_1 \subseteq Y_2$. Notons que ce n'est pas la relation d'ordre traditionnellement employée.

La collection des bi-ensembles ordonnés par $\preceq_{BE}$ forme un treillis avec comme borne inférieure $(\bot_G, \bot_M) = (\emptyset, \emptyset)$ et comme borne supérieure $(\top_G, \top_M) = (G, M)$.

Définition 6
$ER = \langle(\bot_G, \bot_M), (\top_G, \top_M)\rangle$ est un treillis complet de bi-ensembles ssi $(X, Y) \in ER \Leftrightarrow (\bot_G, \bot_M) \preceq_{BE} (X, Y) \preceq_{BE} (\top_G, \top_M)$. $ER' = \langle(\bot'_G, \bot'_M), (\top'_G, \top'_M)\rangle$ est un sous-treillis de ER ssi ER' est un treillis complet de bi-ensembles tel que $\bot_G \preceq_{BE} \bot'_G$, $\bot_M \preceq_{BE} \bot'_M$, $\top'_G \preceq_{BE} \top_G$ et $\top'_M \preceq_{BE} \top_M$.

Désignons par $\mathcal{B}$ l'ensemble des sous-treillis de $\langle(\emptyset,\emptyset),(G,M)\rangle$, i.e., $\mathcal{B} = \{\langle(X_1,Y_1),(X_2,Y_2)\rangle$ t.q. $X_1, X_2 \in 2^G, Y_1, Y_2 \in 2^M$ et $X_1 \subseteq X_2,\ Y_1 \subseteq Y_2\}$. Le premier (resp. second) bi-ensemble est la borne inférieure (resp. supérieure) du treillis.

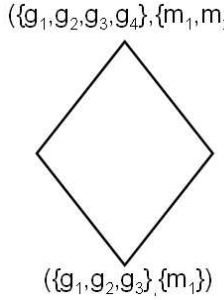
La Figure 3 présente un exemple de treillis contenant quatre bi-ensembles $(\{g_1,g_2,g_3\},\{m_1\})$, $(\{g_1,g_2,g_3\},\{m_1,m_2\})$, $(\{g_1,g_2,g_3,g_4\},\{m_1\})$ et $(\{g_1,g_2,g_3,g_4\},\{m_1,m_2\})$.

Afin de simplifier les formules à venir, nous adoptons les notations suivantes avec $ER = \langle(\bot_G,\bot_M),(\top_G,\top_M)\rangle$:

$$ER \cup \{e\} = \begin{cases} \langle(\bot_G \cup \{e\},\bot_M),(\top_G \cup \{e\},\top_M)\rangle \text{ si } e \in G \\ \langle(\bot_G,\bot_M \cup \{e\}),(\top_G,\top_M \cup \{e\})\rangle \text{ si } e \in M \end{cases}$$

$$ER \setminus \{e\} = \begin{cases} \langle(\bot_G \setminus \{e\},\bot_M),(\top_G \setminus \{e\},\top_M)\rangle \text{ si } e \in G \\ \langle(\bot_G,\bot_M \setminus \{e\}),(\top_G,\top_M \setminus \{e\})\rangle \text{ si } e \in M \end{cases}$$

Pour extraire les bi-ensembles sous contraintes, l'algorithme proposé parcours le treillis des bi-ensembles suivant $\preceq_{BE}$ en élaguant des sous-treillis ne contenant aucun bi-ensemble satisfaisant $\mathfrak{C}$. Ce cadre générique est inspiré de DUAL-MINER [12]. Avant de décrire l'algorithme, nous allons d'abord étudier quelle classe de contraintes nous allons pouvoir utiliser pour élaguer l'espace de recherche et ensuite étudier le problème des grands volumes de données.



$Upper(\mathcal{C}_{area}) = |\{g_1,g_2,g_3,g_4\}| \times |\{m_1,m_2\}| > 4 = TRUE$
Le treillis est conservé.

$Upper(\mathcal{C}_{size}) = |\{g_1,g_2,g_3,g_4\}|/|\{m_1\}| > 4 = False$
Le treillis est élagué.

FIG. 3 – Exemple de treillis de bi-ensembles et de bornes supérieures de contraintes

3.1 Borne supérieure de contraintes

Nous allons utiliser des bornes supérieures des contraintes dans les sous-treillis pour pouvoir élaguer les espaces de recherche.

Définition 7 (Borne supérieure)
Soit $ER \in \mathcal{B}$ et $\mathcal{C}$ une contrainte sur $\mathcal{B}$ alors une borne supérieure de $\mathcal{C}$ sur ER notée $Upper(\mathcal{C}(ER))$ est telle que $\forall(X,Y) \in ER\ \mathcal{C}(X,Y) \Rightarrow Upper(\mathcal{C}(ER))$ et la contraposée $\neg Upper(\mathcal{C}(ER)) \Rightarrow \neg\mathcal{C}(X,Y)$. Si la

borne supérieure d'une contrainte $\mathcal{C}$ pour un sous-treillis n'est pas satisfaite, alors aucun des bi-ensembles dans ce sous-treillis ne satisfait $\mathcal{C}$.

Exemple. Soit $\mathcal{C}_{area}(X,Y) \equiv |X| * |Y| > \alpha$ et $\mathcal{C}_{size}(X,Y) \equiv |X|/|Y| > \alpha$, $Upper(\mathcal{C}_{area}(ER)) = |\top_G| \times |\top_M| > \alpha$ et $Upper(\mathcal{C}_{size}(ER)) = |\top_G|/|\bot_M| > \alpha$ sont respectivement leur borne supérieure. La Figure 3 présente un exemple pour ces deux contraintes.

Pour les contraintes (anti-)monotones sur $\subseteq$, il est facile de calculer des bornes supérieures pertinentes.

Définition 8 (Borne de contraintes (anti-)monotones)
Soit $\mathcal{C}(X,Y)$ une contrainte (anti-)monotone selon $\subseteq$ sur les deux dimensions, i.e., $\mathcal{C}$ est monotone ou anti-monotone selon l'inclusion ensembliste pour X et Y, alors on peut définir une borne supérieure de $\mathcal{C}$ pour $ER \in \mathcal{B}$ telle que $Upper(\mathcal{C})(ER) = \mathcal{C}(X_1, Y_1)$

$$\begin{array}{rl} \text{avec } X_1 = \bot_G & \text{si } \mathcal{C} \text{ est monotone pour } X \\ X_1 = \top_G & \text{si } \mathcal{C} \text{ est anti-monotone pour } X \\ Y_1 = \bot_M & \text{si } \mathcal{C} \text{ est monotone pour } Y \\ Y_1 = \top_M & \text{si } \mathcal{C} \text{ est anti-monotone pour } Y \end{array}$$

Grace à ce cadre générique, nous pouvons utiliser d'autres contraintes que les contraintes (anti-)monotones et convertibles. Nous proposons une nouvelle classe de contraintes appelée contraintes (anti-)monotone par morceaux pour lesquelles des bornes supérieures non triviales peuvent être calculées. Ainsi, ce cadre permet d'utiliser plus de contraintes exploitables efficacement au cours de l'extraction. Par ce faire, nous allons d'abord définir une fonction $\mathcal{R}(\mathcal{C})$ qui est une simple réécriture de la contrainte $\mathcal{C}$.

Définition 9 ($\mathcal{R}(\mathcal{C})$)
Soit $\mathcal{C}(X,Y)$ une contrainte sur $2^G \times 2^M$, on note $\mathcal{R}(\mathcal{C})$ la contrainte obtenue à partir de $\mathcal{C}$ en remplaçant chaque instance de X (resp. Y) dans $\mathcal{C}$ par un nouveau paramètre X_i (resp. Y_i) dans la contrainte $\mathcal{R}(\mathcal{C})$.

Exemple. Soit $Val^+ : E \rightarrow \mathbb{R}^+$, $\mathcal{C}_1(X) \equiv \sum_{i \in X} Val^+(i)/|X| > \alpha$ et $\mathcal{C}_2(X,Y) \equiv |X \cup E| * |Y|/|X| > \alpha$ deux contraintes. On a $\mathcal{R}(\mathcal{C}_1) \equiv \mathcal{C}'(X_1, X_2) \equiv \sum_{i \in X_1} Val^+(i)/|X_2| > \alpha$ et $\mathcal{R}(\mathcal{C}_2) \equiv \mathcal{C}''(X_1, X_2, Y_1) \equiv |X_1 \cup E| * |Y_1|/|X_2| > \alpha$.

Définition 10 (Contraintes strictement (anti-)monotone par morceaux)
On appelle contrainte strictement (anti-)monotone par morceaux une contrainte $\mathcal{C}$ telle que $\mathcal{R}(\mathcal{C})$ est (anti-)monotone pour chacun de ses paramètres. On notera $\mathcal{X}_m$ (resp. $\mathcal{Y}_m$) l'ensemble des paramètres X_i (resp. Y_i) de $\mathcal{R}(\mathcal{C})$ pour lesquels $\mathcal{R}(\mathcal{C})$ est monotone et $\mathcal{X}_{am}$ (resp. $\mathcal{Y}_{am}$) l'ensembles des paramètres X_i (resp. Y_i) pour lesquels $\mathcal{R}(\mathcal{C})$ est anti-monotone, i.e., $\mathcal{X}_{am} \cup \mathcal{X}_m$ est égal à l'ensemble des paramètres X_i.

Exemple. La contrainte $\mathcal{C}_1(X) \equiv \sum_{i \in X} Val^+(i)/|X| > \alpha$, qui n'est pas anti-monotone, est strictement (anti-)monotone par morceaux. En effet, la contrainte $\mathcal{R}(\mathcal{C}_1) \equiv \sum_{i \in X_1} Val^+(i)/|X_2| > \alpha$ est (anti-)monotone pour chacun de ses paramètres, i.e., X_1 et X_2.

Nous allons montrer comment définir des bornes supérieures pour des éléments de $\mathcal{B}$ pour les contraintes strictement (anti-)monotones par morceaux.

Propriété 2 (Bornes pour $\mathcal{R}(\mathcal{C})$)
Soit $\mathcal{C}$ une contrainte strictement (anti-)monotone par morceaux et $ER = \langle (\bot_G, \bot_M), (\top_G, \top_M) \rangle \in \mathcal{B}$, on peut définir une borne supérieure de $\mathcal{C}$ sur ER telle que $Upper(\mathcal{C})(ER) = \mathcal{R}(\mathcal{C})$ avec $\mathcal{X}_m^i = \top_G$, $\mathcal{X}_{am}^j = \bot_G, \mathcal{Y}_m^k = \top_M$ et $\mathcal{Y}_{am}^l = \bot_M$.

Exemple. On a $\mathcal{R}(\mathcal{C}_1) \equiv \sum_{i \in X_1} Val^+(i)/|X_2| > \alpha$ avec $\mathcal{X}_m = \{X_1\}$ et $\mathcal{X}_{am} = \{X_2\}$. Ainsi on obtient $Upper(\mathcal{C}_1) \equiv \sum_{i \in \top_G} Val^+(i)/|\bot_G| > \alpha$. Pour $\mathcal{R}(\mathcal{C}_2) \equiv |X_1 \cup E| * |Y_1|/|X_2| > \alpha$, on a $\mathcal{X}_m = \{X_1\}$, $\mathcal{Y}_m = \{Y_1\}$ et $\mathcal{X}_{am} = \{X_2\}$. Ainsi une borne supérieure pour $\mathcal{C}_2$ est $Upper(\mathcal{C}_2) \equiv |\top_G \cup E| * |\top_M|/|\bot_G| > \alpha$.

Certaines contraintes (anti-)monotones ne sont pas strictement (anti-)monotones par morceaux, e.g. , $\mathcal{C}(X) \equiv \sum_{i \in X} |Val(i) - \alpha|/|\sum_{i \in X} Val(i) - \alpha| > \alpha$. En effet, la fonction $\sum_{i \in X} |Val(i) - \alpha|/|\sum_{i \in X} Val(i) - \alpha|$ est croissante selon X pour $\subseteq$. En revanche, $\mathcal{R}(\mathcal{C}) \equiv \sum_{i \in X_1} |Val(i) - \alpha|/|\sum_{i \in X_2} Val(i) - \alpha|$ est ni monotone ni anti-monotone selon X_2.

Définition 11 (Contrainte (anti-)monotone par morceaux et Borne)
On appelle contrainte (anti-)monotone par morceaux une contrainte $\mathcal{C}$ qui est une conjonction de disjonctions de contraintes $\mathcal{C}_{ij}$ (anti-)monotones ou strictement (anti-)monotones par morceaux :

$$\mathcal{C} \equiv \bigwedge_{i \in I} \bigvee_{j \in J} \mathcal{C}_{ij}$$

La borne supérieure de $\mathcal{C}$ est telle que :

$$Upper(\mathcal{C}) \equiv \bigwedge_{i \in I} \bigvee_{j \in J} Upper(\mathcal{C}_{ij})$$

L'article [23] présente une méthode pour obtenir automatiquement ces bornes supérieures pour des contraintes construites à partir d'un ensemble fixé de primitives. La Figure 4 présente quelques exemples de contraintes (anti-)monotones par morceaux et leur borne supérieure associée.

$\mathcal{C}(X,Y)$	$Upper(\mathcal{C})$
$\|X\| * \|Y\| > \alpha$	$\|\top_G\| * \|\top_M\| > \alpha$
$\sum_{i\in X} Val^+(i)/\|X\| > \alpha$	$\sum_{i\in \top_G} Val^+(i)/\|\bot_M\| > \alpha$
$a \in X \wedge b \in Y$	$a \in \top_G \wedge b \in \top_M$
$X \subseteq E \wedge \|Y\| > \alpha$	$\bot_G \subseteq E \wedge \|\top_M\| > \alpha$
$\|X \cap E\| > \alpha \wedge \|Y \cap E'\| < \beta$	$\|\top_G \cap E\| > \alpha \wedge \|\bot_M \cap E'\| < \beta$

FIG. 4 – Exemples de contraintes (anti-)monotones par morceaux

3.2 Grands volumes de données

Il convient maintenant de regarder les problèmes liés aux grands jeux de données. La contrainte $\mathfrak{C}$ peut être définie sous la forme d'une conjonction de deux contraintes, l'une spécifiant le type de motifs à extraire $\mathcal{C}_{motif}$ et l'autre étant un prédicat de sélection $\mathcal{C}_{select}$: $\mathfrak{C} \equiv \mathcal{C}_{motif}(X,Y,\mathbf{r}) \wedge \mathcal{C}_{select}(X,Y)$.

La contrainte $\mathcal{C}_{motif}$ accède aux données contrairement à $\mathcal{C}_{select}$. Nous allons imposer que la contrainte $\mathcal{C}_{motif}$ soit construite à partir de fonctions d'agrégation de données. La borne supérieure de $\mathcal{C}_{motif}$ peut alors être représentée sous la forme :

$$Upper(\mathcal{C}_{motif})(ER,\mathbf{r}) \equiv \mathcal{C}_{eval}(\mathfrak{A}_1(ER,\mathbf{r}),\ldots,\mathfrak{A}_k(ER,\mathbf{r}),ER)$$

où les $\mathfrak{A}_i$ sont des fonctions d'agrégation et $\mathcal{C}_{eval}$ est une contrainte paramétrée par ER et les fonctions d'agrégation calculées sur ER.

La section 5 en présente deux exemples. L'un des points clés de la recherche de bi-ensembles dans des données booléennes est la prise en compte des grands volumes de données. En effet, il faut réduire au maximum les accès aux données au cours de l'extraction. Ainsi, il est nécessaire de pouvoir calculer incrémentalement les agrégations $\mathfrak{A}_i$ au fur et à mesure de la réduction de l'espace de recherche et ceci en accédant le moins possible aux données. La Propriété 3 formalise cette idée.

Propriété 3
$\mathcal{C}_{motif}$ est efficace sur des grands volumes de données s'il existe des fonctions f_{IN} et f_{OUT} calculables incrémentalement telles que :

$$Upper(\mathcal{C}_{motif})(ER \setminus \{e\},\mathbf{r}) \equiv$$
$$\mathcal{C}_{eval}(f_{OUT}(\mathfrak{A}_1(ER,\mathbf{r}),\ldots,\mathfrak{A}_k(ER,\mathbf{r}),ER,\mathbf{r}[e]),ER)$$
$$Upper(\mathcal{C}_{motif})(ER \cup \{e\},\mathbf{r}) \equiv$$
$$\mathcal{C}_{eval}(f_{IN}(\mathfrak{A}_1(ER,\mathbf{r}),\ldots,\mathfrak{A}_k(ER,\mathbf{r}),ER,\mathbf{r}[e]),ER)$$

où $e \in M \cup G$ et $\mathbf{r}[e]$ désigne la projection de $\mathbf{r}$ sur e.

La contrainte $\mathcal{C}_{eval}$ peut alors être calculée à partir des valeurs des fonctions d'agrégation précédemment calculées en n'accédant qu'à une petite partie de $\mathbf{r}$.

Il est important d'utiliser une contrainte $\mathcal{C}_{motif}$ satisfaisant la Propriété 3. Ainsi, pour chaque nouveau bi-ensemble candidat considéré, les fonctions d'agrégation peuvent être calculées à partir de son bi-ensemble père. Ce mécanisme est utile à la fois lors de l'énumération et lors de la propagation des contraintes qui génére (un) de nouveau(x) candidats à partir d'un bi-ensemble.

Finalement, le cadre générique d'extraction de bi-ensembles sous contraintes que nous proposons permet de calculer les collections :

$$\theta = \{(X, Y) \mid \mathfrak{C} \equiv \mathcal{C}_{motif}(X, Y, \mathbf{r}) \wedge \mathcal{C}_{select}(X, Y)\}$$

avec $\mathfrak{C}$ une conjonction de contraintes (anti-)monotones par morceaux et $\mathcal{C}_{motif}(X, Y, \mathbf{r})$ qui satisfait la Propriété 3.

Nous allons maintenant décrire plus précisément l'algorithme générique permettant de calculer θ.

4 ALGORITHME GÉNÉRIQUE

Pour extraire la collection $\{(X, Y) \mid \mathfrak{C}(X, Y, \mathbf{r})\}$, l'algorithme n'explore que certains sous-treillis de $\mathcal{B}$ à l'aide de trois mécanismes fondamentaux : l'énumération, la propagation et l'élagage. L'algorithme débute avec le treillis complet $\langle(\emptyset, \emptyset), (G, M)\rangle$, récursivement propage la contrainte $\mathfrak{C}$ et finalement, si le sous-treillis ainsi généré est consistant avec $\mathfrak{C}$ alors il est partitionné en deux nouveaux sous-treillis. Trois fonctions principales sont nécessaires : $Prop : \mathcal{B} \rightarrow \mathcal{B}$ pour la propagation de contraintes, $Prune : \mathcal{B} \rightarrow \{\text{TRUE,FALSE}\}$ pour l'élagage et $Enum : \mathcal{B} \times G \cup M \rightarrow \mathcal{B}^2$ pour l'énumération.

4.1 Énumération

Soit $Enum : \mathcal{B} \times G \cup M \rightarrow \mathcal{B}^2$ telle que

$$Enum(ER, e) = (ER \cup \{e\}, ER \setminus \{e\})$$

avec $e \in \top_G \setminus \bot_G$ ou $e \in \top_M \setminus \bot_M$. $Enum$ génère deux sous-treillis qui sont une partition du treillis ER.

Soit $Choose : \mathcal{B} \rightarrow G \cup M$ une fonction qui retourne un élément de $\top_G \setminus \bot_G \cup \top_M \setminus \bot_M$. En fonction du type de motif extrait, une heuristique permettant d'accélérer l'extraction est utilisée pour choisir l'élément parmi $\top_G \setminus \bot_G \cup \top_M \setminus \bot_M$.

4.2 Élagage

Un sous-treillis est élagué si aucun des bi-ensembles qu'il contient ne satisfait la contrainte $\mathfrak{C}$. $Prune : \mathcal{B} \rightarrow \{\text{TRUE,FALSE}\} = Upper(\mathfrak{C})$

$K = (G, M, I)$ un contexte et $\mathfrak{C}$ une contrainte (anti-)monotone sur $2^G \times 2^M$.
DÉBUT ALGORITHME
    GENERATE$\langle(\emptyset, \emptyset), (G, M)\rangle$
Fin Algorithme

GENERATE$(\mathcal{L})$
    Soit $\mathcal{L} = \langle(\bot_G, \bot_M), (\top_G, \top_M)\rangle$
    $\mathcal{L} \leftarrow$ **Prop**$(\mathcal{L})$
    Si **Prune**$(\mathcal{L})$ alors
        Si $(\bot_G, \bot_M) \neq (\top_G, \top_M)$ alors
            $(\mathcal{L}_1, \mathcal{L}_2) \leftarrow$ **Enum**$(\mathcal{L}$,**Choose**$(\mathcal{L}))$
            GENERATE$(\mathcal{L}_1)$
            GENERATE$(\mathcal{L}_2)$
        Sinon Enregistrer$(\bot_G, \bot_M)$
        Fin Si
    Fin Si
Fin Generate

FIG. 5 – Pseudo-code

4.3 Propagation

$\mathfrak{C}$ peut être utilisée pour réduire la taille du treillis considéré. Avec $ER = \langle(\bot_G, \bot_M), (\top_G, \top_M)\rangle$, on a :

$$
\begin{aligned}
&Prop(ER) = \{\langle(\bot'_G, \bot'_M), (\top'_G, \top'_M)\rangle \in \mathcal{B} \mid \\
&\bot'_G = \bot_G \cup \{a \in \top_G \setminus \bot_G \mid \neg Upper(\mathfrak{C})(ER \setminus \{a\})\} \\
&\bot'_M = \bot_M \cup \{a \in \top_M \setminus \bot_M \mid \neg Upper(\mathfrak{C})(ER \setminus \{a\})\} \\
&\top'_G = \top_G \setminus \{a \in \top_G \setminus \bot_G \mid \neg Upper(\mathfrak{C})(ER \cup \{a\})\} \\
&\top'_M = \top_M \setminus \{a \in \top_M \setminus \bot_M \mid \neg Upper(\mathfrak{C})(ER \cup \{a\})\}\}
\end{aligned}
$$

La Figure 6 présente le fonctionnement des trois mécanismes précédemment décrits. On débute avec un treillis B. $Prop$ est utilisée pour réduire la taille de B, i.e. , enlever des éléments $e \in \top \setminus \bot$ à la borne supérieure de B et ajouter des éléments de $e \in \top \setminus \bot$ à la borne inférieure de B. Ensuite, on vérifie que le nouveau treillis B_1 ainsi obtenu satisfait $Upper(\mathfrak{C})$. Si c'est le cas, deux nouveaux treillis sont générés B_2 et B_3 et les mêmes étapes sont appliquées récursivement sur ces deux nouveaux candidats. La Figure 5 présente l'algorithme.

Suivant les caractéristiques du jeu de données utilisé en entrée, il n'est pas toujours judicieux d'utiliser $\mathfrak{C}$ pour propager les contraintes. En effet, certaines parties de $\mathfrak{C}$ peuvent être coûteuses à évaluer par rapport au gain qu'elles apportent en terme de réduction de l'espace de recherche. Ainsi, il

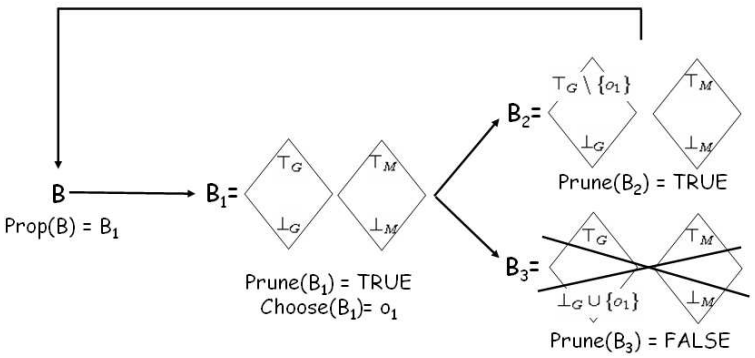


FIG. 6 – Exemple d'exécution de l'algorithme

peut être utile d'utiliser dans $Prop$ non pas $\mathfrak{C}$ mais une autre contrainte $\mathcal{C}$ telle que $\neg\mathcal{C} \Rightarrow \neg\mathfrak{C}$. La Section 5.1 donne un exemple.

5 INSTANCES DE L'ALGORITHME

Nous allons considérer deux instances concrètes de cet algorithme, l'une pour extraire les concepts formels et l'autre pour calculer des motifs tolérants au bruits. Ces deux instances correspondent à D-MINER [7] et DR-MINER [6]. Les tâches d'extraction associées sont respectivement :

$$\theta_1 = \{(X,Y) \mid \mathfrak{C} \equiv \mathcal{C}_{CF}(X,Y,\mathbf{r}) \wedge \mathcal{C}_{select}(X,Y)\}$$

$$\theta_2 = \{(X,Y) \mid \mathfrak{C} \equiv \mathcal{C}_{DRBS}(X,Y,\mathbf{r}) \wedge \mathcal{C}_{select}(X,Y)\}$$

pour lesquelles $\mathcal{C}_{select}$ est une contrainte (anti-)monotone par morceaux.

Nous montrons que les contraintes $\mathcal{C}_{CF}$ et $\mathcal{C}_{DRBS}$ satisfont les propriétés demandées, i.e. , que $\mathcal{C}_{CF}$ et $\mathcal{C}_{DRBS}$ sont (anti-)monotones par morceaux et satisfont la Propriété 3.

5.1 Concepts formels

Pour la contrainte $\mathcal{C}_{CF}$, nous utilisons les agrégations suivantes :
- $\mathfrak{A}_1(X,Y) = \bigwedge_{i\in X} \bigwedge_{j\in Y} \mathbf{r}(i,j)$
- $\mathfrak{A}_2(X,Y) = \bigwedge_{i\in G\setminus X} \bigvee_{j\in Y} \neg\mathbf{r}(i,j)$
- $\mathfrak{A}_3(X,Y) = \bigwedge_{j\in M\setminus Y} \bigvee_{i\in X} \neg\mathbf{r}(i,j)$

Ainsi, nous avons :

$$\begin{aligned} \mathcal{C}_{CF} &\equiv \mathfrak{A}_1 \wedge \mathfrak{A}_2 \wedge \mathfrak{A}_3 \equiv \mathcal{C}_{eval}(\mathfrak{A}_1,\mathfrak{A}_2,\mathfrak{A}_3) \\ \mathcal{R}(\mathcal{C}_{CF}) &\equiv \mathcal{C}_{eval}(\mathfrak{A}_1(X_1,Y_1),\mathfrak{A}_2(X_2,Y_2),\mathfrak{A}_3(X_3,Y_3)) \end{aligned}$$

Propriété 4
$\mathcal{C}_{CF}$ est (anti-)monotone par morceaux.

Preuve. $\mathcal{R}(\mathcal{C}_{CF})$ est (anti-)monotone selon chacun de ses 6 paramètres. D'après la Définition 11, $\mathcal{C}_{CF}$ est (anti-)monotone par morceaux.□

D'après les propriétés 2 et 4, on a :

$$Upper(\mathcal{C}_{CF})(ER) = \mathcal{C}_{eval}(\mathfrak{A}_1(\bot_G, \bot_M), \mathfrak{A}_2(\top_G, \bot_M), \mathfrak{A}_3(\bot_G, \top_M))$$

Propriété 5
La contrainte $\mathcal{C}_{CF}$ satisfait la Propriété 3.

Preuve. L'algorithme n'énumère un sous-treillis ER que si $Upper(\mathcal{C}_{CF})(ER)$ est satisfait, c'est-à-dire $\mathfrak{A}_1(\bot_G, \bot_M) = \mathfrak{A}_2(\top_G, \bot_M) = \mathfrak{A}_3(\bot_G, \top_M) = TRUE$ pour ER.
$\mathfrak{A}_1(\bot_G \cup \{e\}, \bot_M) = \mathfrak{A}_1(\bot_G, \bot_M) \wedge \bigwedge_{j \in \bot_M} \mathbf{r}(e, j) = \bigwedge_{j \in \bot_M} \mathbf{r}(e, j)$
$\mathfrak{A}_1(\bot_G, \bot_M \cup \{e\}) = \mathfrak{A}_1(\bot_G, \bot_M) \wedge \bigwedge_{j \in \bot_G} \mathbf{r}(e, j) = \bigwedge_{j \in \bot_G} \mathbf{r}(e, j)$
$\mathfrak{A}_2(\top_G \backslash \{e\}, \bot_M) = \mathfrak{A}_2(\top_G, \bot_M) \wedge \bigvee_{j \in \bot_M} \neg\mathbf{r}(e, j) = \bigvee_{j \in \bot_M} \neg\mathbf{r}(e, j)$
$\mathfrak{A}_2(\top_G, \bot_M \cup \{e\}) = \bigwedge_{i \in G \backslash \top_G} (\bigvee_{j \in \bot_M} \neg\mathbf{r}(i, j) \vee \neg\mathbf{r}(i, e)) = TRUE$
$\mathfrak{A}_3$ est identique à $\mathfrak{A}_2$ □

Avec D-MINER, nous nous sommes intéressés aux contextes d'extraction contenant beaucoup de concepts formels, c'est-à-dire des données représentées par une relation très dense. Pour optimiser D-MINER, nous avons adopté une stratégie de propagation qui n'utilise pas entièrement $\mathcal{C}_{CF}$ pour réduire l'espace de recherche (propagation de contraintes par $Prop$) mais plutôt la contrainte $\mathcal{C}'_{CF} \equiv \mathfrak{A}_1$. En effet, comme $\neg\mathcal{C}'_{CF} \Rightarrow \neg\mathcal{C}_{CF}$, la contrainte $\mathcal{C}'_{CF}$ peut être utilisée à la place de $\mathcal{C}_{CF}$ dans la fonction $Prop$ tout en conservant la correction et la complétude de l'algorithme. En revanche, $\mathcal{C}_{CF}$ doit être utilisée dans $Prune$. Ce choix a été fait car la partie de $\mathcal{C}_{CF}$ que l'algorithme n'utilise pas dans $Prop$, i.e., $\mathfrak{A}_2 \wedge \mathfrak{A}_3$ est coûteuse à calculer alors que dans des contextes contenant beaucoup de concepts formels, elle est peu sélective.

Propriété 6 (Délai en moyenne)
Le délai de D-MINER *a été étudié dans [4]. Il vaut*
- $O(n^2 * m)$ *dans le pire cas.*
- $(n - log2(|C|) + 1)O(n * m)$ *en moyenne.*
  Avec $n = |M|$ et $m = |G|$.

Le délai est le coût de calcul pour passer d'une solution à une autre [16]. Le délai dans le pire des cas est identique aux meilleurs algorithmes d'extraction de concepts formels. En revanche, lorsque la taille de la collection de concepts formels augmente, le délai en moyenne diminue jusqu'à tendre vers $O(n*m)$ quand $|\theta| = 2^n$. Ce point illustre bien l'efficacité de la stratégie de propagation de $\mathfrak{C}$ adoptée pour les jeux de données contenant beaucoup de concepts formels.

5.2 Bi-ensembles tolérants au bruit

Pour les bi-ensembles tolérants au bruit (voir Définition 3), nous utilisons les fonctions d'agrégations suivantes :
- $\mathfrak{A}_1(X,Y) = Max_{i\in X}(\mathcal{ZG}_i(Y))$
- $\mathfrak{A}_2(X,Y) = Max_{i\in Y}(\mathcal{ZM}_i(X))$
- $\mathcal{Z}_G(i,Y)$ et $\mathcal{Z}_M(i,X)$

Ainsi, nous avons :
$\mathcal{R}(\mathcal{C}_{DRBS})(X_1,X_2,X_3,X_4,X_5,X_6,Y_1,Y_2,Y_3,Y_4,Y_5,Y_6) \equiv$

$$\begin{cases} \bigwedge_{i\in X_1} \mathcal{Z}_G(i,Y_1) \leq \alpha \\ \bigwedge_{i\in Y_2} \mathcal{Z}_M(i,X_2) \leq \alpha \\ \bigwedge_{i\in G\setminus X_3}(\mathfrak{A}_1(X_4,Y_3) \leq \mathcal{Z}_G(i,Y_4)) \\ \bigwedge_{i\in M\setminus Y_5}(\mathfrak{A}_2(X_5,Y_6) \leq \mathcal{Z}_M(i,X_6)) \end{cases}$$

Propriété 7
$\mathcal{C}_{DRBS}$ est (anti-)monotone par morceaux.

Preuve. $\mathcal{R}(\mathcal{C}_{DRBS})$ est (anti-)monotone pour ses 12 paramètres. D'après Définition 11, $\mathcal{C}_{DRBS}$ est (anti-)monotone par morceaux.□

D'après les Propriétés 2 et 8, la borne supérieure de $\mathcal{C}_{DRBS}$ est :
$Upper(\mathcal{C}_{DRBS})(\bot_G,\bot_G,\bot_G,\bot_G,\bot_G,\top_G,\bot_M,\bot_M,\bot_M,\top_M,\bot_M,\bot_M)$

Elle peut être exprimée avec les fonctions d'agrégations :
$Upper(\mathcal{C}_{DRBS})(ER = \langle(\bot_G,\bot_M),(\top_G,\top_M)\rangle,\mathbf{r}) \equiv \mathcal{C}_{eval}(\mathcal{ZG}_i(\bot_M),$
$\mathcal{ZG}_i(\top_M), \mathcal{ZM}_i(\bot_G), \mathcal{ZM}_i(\top_G), \mathfrak{A}_1(\bot_G,\bot_M)\ \mathfrak{A}_2(\bot_G,\bot_M), ER) \equiv$

$$\begin{cases} \bigwedge_{i\in\bot_G} \leq \mathcal{ZG}_i(\bot_M)\alpha \\ \bigwedge_{i\in\bot_M} \mathcal{ZM}_i(\bot_G) \leq \alpha \\ \bigwedge_{i\in G\setminus\bot_G}(\mathfrak{A}_1(\bot_G,\bot_M) \leq \mathcal{ZG}_i(\top_M)) \\ \bigwedge_{i\in M\setminus\bot_M}(\mathfrak{A}_2(\bot_G,\bot_M) \leq \mathcal{ZM}_i(\top_G)) \end{cases}$$

Propriété 8
La contrainte $\mathcal{C}_{DRBS}$ satisfait la Propriété 3.

Preuve. $\mathcal{ZG}_i(\bot_M \cup \{e\}) = \mathcal{ZG}_i(\bot_M) + \mathbf{r}(i,e)$
$\mathcal{ZG}_i(\bot_M \setminus \{e\}) = \mathcal{ZG}_i(\bot_M) - \mathbf{r}(i,e)$
$\mathfrak{A}_1(\bot_G \cup \{e\},\bot_M) = Max(\mathfrak{A}_1(\bot_G,\bot_M), \mathcal{ZG}_e(Y))$
$\mathfrak{A}_1(\bot_G,\bot_M \cup \{e\}) = Max_{i\in\bot_M}(\mathcal{ZG}_i(\bot_M) + \mathbf{r}(i,e))$.
La démonstration est identique pour $\mathcal{ZM}_i$ et $\mathfrak{A}_2$ □

5.3 Expérimentations

Pour montrer l'intérêt du cadre générique que nous proposons, nous avons comparé deux stratégies pour extraire les concepts formels (X,Y) satisfaisant la contrainte $\mathcal{C}_{moy}(E) \equiv \sum_{i\in E} val^+(i)/|E| > \alpha$ avec $E = X$

et/ou $E = Y$. Nous utilisons le jeu de données malaria [11] qui concerne le transcriptome du cycle de développement intraerythrocytique du Plasmodium Falciparum, i.e., un agent responsable de la malaria humaine. Les données fournissent le profil d'expression de 3 719 gènes dans 46 échantillons biologiques. $\mathcal{C}_{moy}(E)$ est une contrainte à la fois convertible et strictement (anti-)monotone par morceaux. La première stratégie énumère les éléments i de E par ordre décroissant de $val(i)$. Dans ce cas, on exploite le fait que $\mathcal{C}_{moy}$ est convertible. La deuxième stratégie utilise le cadre générique proposé, i.e, une borne supérieure $Upper(\mathcal{C}_{moy})$. Pour ces expérimentations, les valeurs des $val(i)$ sont tirées aléatoirement entre 0 et 10. Nous allons faire varier α de 0 à 10 et comparer les deux stratégies en fonction du nombre d'énumérations réalisées pour extraire la collection de concepts formels. En effet, plus le nombre d'énumération est faible et plus la stratégie employée est efficace pour exploiter la contrainte. $\alpha = 0$ correspond à la collection complète des concepts formels alors que pour $\alpha = 10$, seuls ceux contenant seulement des éléments i pour lesquels $val(i) = 10$ sont calculés. La première stratégie n'est pas satisfaisante si la contrainte est utilisée sur les attributs ($\mathcal{C}_{moy}(Y)$). En effet, cette stratégie nécessite l'utilisation d'une énumération sur les attributs ($|M| = 3719$) au lieu d'énumérer les candidats avec les objets ($|G| = 47$), i.e., la plus petite des deux dimensions. Le gain de calcul espéré grâce à la contrainte est annulée par le fort accroissement de la taille de l'espace de recherche à parcourir. Finalement, l'extraction est infaisable quelque soit la valeur de α alors qu'elle est faisable avec la seconde stratégie pour toutes les valeurs de α. Considérons maintenant la contrainte $\mathcal{C}_{moy}(X)$, i.e., la contrainte appliquée sur les objets. La courbe de la Figure 7 montre le rapport entre le nombre d'énumérations pour la première stratégie et celui pour la seconde stratégie en fonction de α.

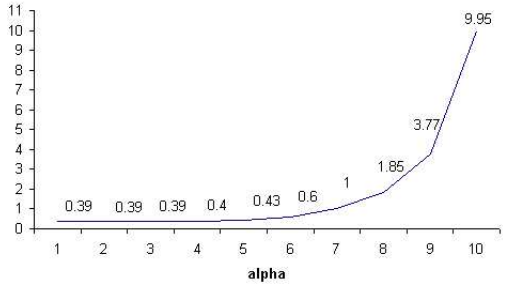


FIG. 7 – Rapport entre le nombre d'énumérations pour la première stratégie et celui de la seconde stratégie pour la contrainte $\mathcal{C}_{moy}(X)$ en fonction de α

Il apparaît clairement que pour $\alpha < 6$, la seconde stratégie est bien plus efficace que la première. Au delà, la première stratégie devient plus efficace. Au début, la contrainte $\mathcal{C}_{CF}$ est plus sélective que $\mathcal{C}_{moy}$, il est alors préférable

de favoriser l'heuristique d'énumération liée à $\mathcal{C}_{CF}$, ce que réalise la seconde stratégie, au dépend de celle liée à $\mathcal{C}_{moy}$. Dès lors que la contrainte $\mathcal{C}_{CF}$ est plus sélective que $\mathcal{C}_{moy}$, ce qui est généralement le cas, le cadre générique proposé est plus efficace. La première stratégie ne permet pas d'exploiter complètement la contrainte $\mathcal{C}_{moy}(X) \wedge \mathcal{C}_{moy}(Y)$, la contrainte $\mathcal{C}_{moy}(Y)$ devant être utilisée en post-traitement. En effet, si l'énumération est effectuée avec les attributs, rien n'indique que l'ensemble des objets sera instancié selon l'ordre croissant des valeurs des objets. La seconde stratégie permet, en revanche, d'exploiter cette contrainte. La Figure 8 montre le log_{10} de la taille de la collection extraite en fonction de α pour la seconde stratégie.

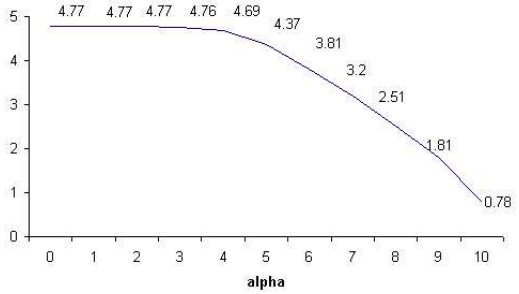


FIG. 8 – log_{10} de la taille de la collection extraite en fonction de α pour la contrainte $\mathcal{C}_{moy}(X) \wedge \mathcal{C}_{moy}(Y)$ en fonction de α

6 CONCLUSION

Cet article est le résultat de plusieurs années de recherche sur l'extraction de bi-ensembles sous contraintes. Après avoir proposé des algorithmes pour extraire des concepts formels [7, 5] et certaines de leurs extensions vers la tolérance aux exceptions comme les DR-bi-ensembles [6], nous nous sommes intéressés à une abstraction de ces travaux dans un algorithme générique. Ce cadre permet de revisiter le calcul de concepts formels et de discuter de mécanismes fondamentaux comme la spécification déclarative des propriétés des bi-ensembles, l'énumération ou la propagation des contraintes. Comme tout algorithme générique, il permet donc de mieux comprendre les degrés de liberté dont nous disposons pour calculer des motifs bi-dimensionnels. Nous venons d'ailleurs d'étudier une nouvelle instance dédiées à des bi-ensembles dans des données numériques [8]. Une des perspectives consiste à mieux comprendre les multiples relations qui existent entre des motifs locaux comme les bi-ensembles et des motifs globaux comme des bi-partitions. En effet, ces dernières apparaissent comme des collections de bi-ensembles satisfaisant de nouveaux types de contraintes.

Remerciements. Les recherches post-doctorales de Jérémy Besson sont financées par l'INRA (ASC). Ce travail a été partiellement financé par le contrat IQ (IST FET FP6-516169) et l'ACI Bingo MD46.

RÉFÉRENCES

[1] R. Agrawal, T. Imielinski et A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, Washington, D.C., USA, June 1993. ACM Press.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen et A. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.

[3] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme et L. Lakhal. PASCAL : un algorithme d'extraction des motifs fréquents. *Technique et Science Informatiques*, 21 :65–95, 2002.

[4] J. Besson. *Découvertes de motifs pertinents pour l'analyse du transcriptome : application à l'insulio-résistance*. PhD thesis, INSA Lyon, LIRIS CNRS UMR 5205, F-69621 Villeurbanne, 2005.

[5] J. Besson, C. Robardet et J-F. Boulicaut. Constraint-based mining of formal concepts in transactional data. In *PaKDD*, volume 3056 of *LNCS*, pages 615–624, Sydney, Australia, May 2004. Springer-Verlag.

[6] J. Besson, C. Robardet et J-F. Boulicaut. Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In *ICCS*, volume 4068 of *LNCS*, pages 144–157, Aalborg, Denmark, July 2006. Springer-Verlag.

[7] J. Besson, C. Robardet, J-F. Boulicaut et S. Rome. Constraint-based bi-set mining for biologically relevant pattern discovery in microarray data. *Intelligent Data Analysis*, 9(1) :59–82, 2004.

[8] J. Besson, C. Robardet, Luc De Raedt et J-F. Boulicaut. Mining numerical bi-sets. In *KDID co-located with ECML/PKDD 2006*, pages 9–19, Berlin, Germany, 2006.

[9] J-F. Boulicaut, A. Bykowski et C. Rigotti. Approximation of frequency queries by mean of free-sets. In *PKDD*, volume 1910 of *LNCS*, pages 75–85, Lyon, France, 2000. Springer-Verlag.

[10] J-F. Boulicaut et B. Crémilleux (Coordinateurs). *Extraction de motifs dans des bases de données RSTI ISI 9(3/4)*. Hermes-Lavoisier, 2004.

[11] Z. Bozdech, M. Llinás, B.L. Pulliam, E. Wong, J. Zhu et J. DeRisi. The transcriptome of the intraerythrocytic developmental cycle of plasmodium falciparum. *PLoS Biology*, 1(1) :1–16, Octobre 2003.

[12] C. Bucila, J. E. Gehrke, D. Kifer et W. White. Dualminer : A dual-pruning algorithm for itemsets with constraints. *Data Mining and Knowledge Discovery*, 7(4) :241–272, Oct. 2003.

[13] B. Ganter. Two basic algorithms in concept analysis. Technical report, Technisch Hochschule Darmstadt, Preprint 831, 1984.

[14] B. Ganter et R. Wille. *Formal Concept Analysis : Mathematical Foundations*. Springer-Verlag, 1999.

[15] A. Guénoche. Construction du treillis de galois d'une relation binaire. *Mathématiques, Informatique et Sciences Humaines*, 109 :41–53, 1990.

[16] S.O. Kuznetsov et S. Obiedkov. Comparing performance of algorithms for generating concept lattices. In *Experimental and Theoretical Artificial Intelligence*, volume 14, pages 189–216. 2001.

[17] H. Mannila et H. Toivonen. Levelwise search and borders of theories in knowledge discovery. In *Data Mining and Knowledge Discovery*, volume 1(3), pages 241–258. 1997.

[18] N. Messai, M.-D. Devignes, A. Napoli et M. Smaïl-Tabbone. Querying a bioinformatic data sources registry with concept lattices. In *ICCS*, volume 3596 of *LNCS*, pages 323–336, Kassel, Germany, 2005. Springer-Verlag.

[19] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18 :203–226, 1982.

[20] E. M. Norris. An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées*, 23(2) :243–250, 1978.

[21] J. Pei, J. Han et L. Lakshmanan. Mining frequent itemsets with convertible constraints. In *ICDE*, pages 433–442, Heidelberg, Germany, April 2001. IEEE Computer Society.

[22] J. Pei, A. K. H. Tung et J. Han. Fault-tolerant frequent pattern mining : Problems and challenges. In *Workshop DMKD*, 2001.

[23] A. Soulet et B. Crémilleux. An efficient framework for mining flexible constraints. In *PaKDD*, volume 3518 of *LNCS*, pages 661–671, Hanoi, Vietnam, 2005. Springer-Verlag.

[24] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier et L. Lakhal. Computing iceberg concept lattices with titanic. *Data and Knowledge Engineering*, 42 :189–222, 2002.

[25] J. Wang, J. Han et J. Pei. CLOSET+ : searching for the best strategies for mining frequent closed itemsets. In *SIGKDD*, pages 236–245, Washington, DC, USA, August 2003. ACM Press.

[26] R. Wille. Restructuring lattice theory : an approach based on hierarchies of concepts. In I. Rival, éditeur, *Ordered sets*, pages 445–470. Reidel, Dordrecht, 1982.

[27] M. J. Zaki et C-J. Hsiao. CHARM : An efficient algorithm for closed itemset mining. In *SIAM DM*, Arlington, VA, USA, April 2002. SIAM.