
JEN : un algorithme efficace de construction de générateurs pour l'identification des règles d'association

Amélie Le Floc'h*, **Christian Fisette***, **Rokia Missaoui****, **Petko Valtchev*****, **Robert Godin***

** Département d'informatique, Université du Québec à Montréal*

C.P. 8888, succursale Centre-Ville, Montréal, Québec, Canada, H3C 3P8

*** Département d'informatique et d'ingénierie, Université du Québec en Outaouais*

**** Département d'informatique et recherche opérationnelle, Université de Montréal*

e-mail: rokia.missaoui@uqo.ca

RÉSUMÉ. L'article décrit un algorithme, appelé JEN, d'identification efficace des générateurs à partir du treillis de concepts (Galois) en vue de produire des règles d'association exactes et approximatives. Une analyse comparative empirique illustre la supériorité de JEN sur trois autres procédures de génération de règles et de générateurs, particulièrement lors de l'analyse de données fortement corrélées.

ABSTRACT. This paper describes an algorithm, called JEN, aimed at efficiently extracting generators from a concept (Galois) lattice for mining exact and approximate association rules. An empirical study shows that JEN performs better than three other procedures for rule and generator extraction, specially when data are highly correlated.

MOTS-CLÉS : Data mining, treillis de concepts (Galois), règles d'association, générateurs.

KEYWORDS : Data mining, concept (Galois) lattices, association rules, generators.

1. Introduction

La fouille de données (*data mining*) est une discipline récente, issue de la confluence des statistiques, de l'intelligence artificielle et des bases de données. Elle vise la découverte de régularités (ex. concepts, liens, règles) dans de grands volumes de données. La dernière décennie a enrichi la discipline d'un large spectre d'approches visant à améliorer la performance des méthodes de fouille et à accroître leur capacité de produire un ensemble réduit mais pertinent de concepts et règles.

Cet article traite du problème de l'extraction des connaissances sous forme de règles d'association (Agrawal et al., 1994) en mettant davantage l'accent sur les travaux opérant dans le contexte de l'analyse formelle des concepts (Ganter et al., 1999). Plusieurs études (Godin et al., 1994), (Zaki et al., 1999), (Pasquier, 2000) ont notamment souligné le nombre prohibitif de règles d'association extraites et la nécessité de définir des bases pour les règles d'association. Ces bases constituent des ensembles réduits de règles informatives (prémisse minimale, conséquence maximale) permettant de ne conserver que les règles les plus pertinentes, sans perte d'information. La génération de ces règles d'association informatives peut se faire par une extraction efficace des concepts et de leurs générateurs associés.

Des algorithmes produisant l'ensemble des concepts ainsi que leurs générateurs associés ont été développés avec succès (Godin et al., 1994), (Pasquier, 2000), (Pfaltz et al., 2002), (Bastide et al., 2002). Cependant, peu de travaux font état de l'efficacité et des performances de l'extraction des générateurs dans le processus de production des règles.

Dans cet article, nous proposons l'algorithme *JEN* de construction des générateurs, basé sur la notion de bloqueur minimal (Pfaltz et al., 2002) et offrant des performances significatives, notamment pour des données fortement corrélées.

La section suivante introduit les principales notions du problème de la génération des règles d'association informatives à partir du treillis de concepts et rappelle des notions reliées à l'identification des générateurs (faces et bloqueurs). La section 3 décrit le principe des algorithmes de construction de générateurs et présente notre algorithme *JEN*. Finalement, des résultats expérimentaux impliquant quatre algorithmes, y compris *JEN*, sont présentés en section 4.

2. Treillis de concepts et règles d'association informatives

2.1. Treillis de concepts (Galois)

Un treillis de concept (Galois) correspond à un ordre partiel induit par une relation binaire R entre deux ensembles, un ensemble d'objets O , et un ensemble

d'attributs A . La figure 1 montre un exemple de contexte $K = (O, A, R)$ dans lequel les lignes correspondent aux objets (ex : clients) et les colonnes aux différents attributs (ex : articles) du contexte. La relation oRa indique que l'objet o possède l'attribut a . Deux fonctions f et g permettent d'exprimer les correspondances entre les sous-ensembles d'objets $P(O)$ et les sous-ensembles d'attributs $P(A)$ induits par la relation R .

Définition 1 (Correspondance de Galois)

La fonction f associe l'ensemble des attributs communs à un ensemble d'objets, tandis que g , la fonction duale de f , associe l'ensemble des objets communs à un ensemble d'attributs.

$$f : P(O) \rightarrow P(A), f(X) = X' = \{ a \in A / \forall o \in X, oRa \}$$

$$g : P(A) \rightarrow P(O), g(Y) = Y' = \{ o \in O / \forall a \in Y, oRa \}$$

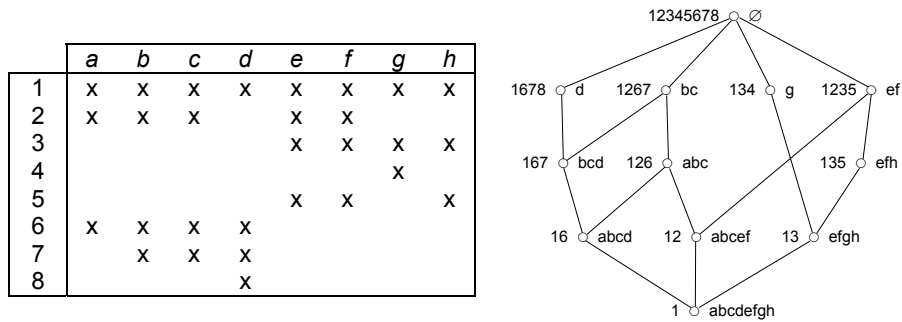


Figure 1. Gauche: contexte K . Droite: Diagramme de Hasse du treillis de Galois induit par le contexte K .

Avec l'exemple ci-haut, la fonction f retournera l'ensemble des articles achetés par un ensemble de clients tandis que la fonction g retournera l'ensemble des clients ayant acheté le même ensemble d'articles: $f(13) = fgh$ et $g(abc) = 126$.

Le couple (g, f) définit la connexion de Galois entre les sous-ensembles d'objets $P(O)$ et d'attributs $P(A)$ du contexte. Les compositions $f \circ g$ et $g \circ f$, constituent des opérateurs de fermeture de la connexion de Galois. L'opérateur $f \circ g$ génère des sous-ensembles fermés d'objets tandis que $g \circ f$ des sous-ensembles fermés d'attributs.

Définition 2 (Concept)

Un concept est un couple complet de la forme (X, Y) dans lequel $X \in P(O)$, $Y \in P(A)$, $X = g(Y)$ et $Y = f(X)$. X est appelé l'*extent* (extension) du concept et Y son *intent* (intention).

L'ensemble de tous les concepts extraits d'un contexte, C_k , est ordonné par une relation de spécialisation/généralisation, notée \leq_k , qui vérifie l'inclusion ensembliste entre les extensions. De plus, l'ordre sur C_k satisfait les propriétés d'un treillis complet.

Théorème 1

L'ordre partiel $L = (C_k, \leq_k)$ est un treillis complet, appelé treillis de concept (Galois) dont le *Meet* et le *Join* s'expriment par les deux relations suivantes (Ganter *et al.*, 1999):

$$\text{Meet} - \bigwedge_{j \in J} (X_j, Y_j) = (\bigcap_{j \in J} X_j, (\bigcup_{j \in J} Y_j)'')$$

$$\text{Join} - \bigvee_{j \in J} (X_j, Y_j) = ((\bigcup_{j \in J} X_j)'', \bigcap_{j \in J} Y_j)$$

2.2. Règles d'association

2.2.1. Caractéristiques

De par sa structure hiérarchique, le treillis de concepts facilite la détection de régularités qui s'expriment sous la forme de règles liant deux ensembles d'attributs. La règle $r : Y_1 \rightarrow Y_2$ indique que la présence de l'ensemble Y_1 dans un objet *entraîne* la présence de l'ensemble Y_2 . Elle peut être de l'un des deux types suivants : une règle d'association exacte, dite d'implication, dont la validité exige que Y_2 soit présent chaque fois que Y_1 l'est, et une règle approximative de forme probabiliste.

Deux mesures de qualité sont habituellement associées aux règles : la confiance et le support. La confiance de la règle $r : Y_1 \rightarrow Y_2$ calcule la probabilité que Y_2 se produise quand Y_1 est présent. Le support (relatif) de la règle r représente la probabilité que Y_1 et Y_2 soient simultanément présents.

En imposant des seuils de support et de confiance élevés lors de la génération des règles, il est possible de restreindre leur nombre et faciliter ainsi leur compréhension par l'analyste. Cependant, le nombre de règles d'association reste important notamment pour les données très fortement corrélées de par le nombre de concepts fréquents obtenus. Pour pallier ce surplus de règles, des bases pour les règles d'association (exactes et approximatives) ont été définies. Celles-ci constituent des ensembles réduits de règles informatives, sans perte d'information.

2.2.2. Règles d'association informatives

Les règles d'association informatives sont des règles d'antécédent minimal et de conséquence maximale. Elles permettent de déduire le maximum d'information avec un minimum d'hypothèses. La maximalité des règles est assurée par l'utilisation d'ensembles d'attributs (*itemsets*) fermés (*intent* des concepts) qui sont par définition maximaux. La minimalité de l'antécédent, quant à elle, s'exprime à l'aide de générateurs, ensembles d'attributs minimaux rattachés à chacun des concepts. Pour un concept c donné, les générateurs correspondent aux *itemsets* minimaux inclus dans l'*intent* de c mais non inclus dans l'*intent* d'un parent de c .

Définition 4 (Règle d'association informative exacte)

Une règle d'association informative exacte (Godin *et al.*, 1994), (Pasquier, 2000) est une règle d'implication de la forme $r : g \rightarrow Y_1 \setminus g$, avec Y_1 un ensemble fermé d'attributs et g un générateur de Y_1 .

Définition 5 (Règle d'association informative approximative)

Une règle d'association informative approximative est une règle d'implication de la forme $r : g \rightarrow Y_2 \setminus g$, où g est un générateur de l'ensemble fermé d'attributs Y_1 , et Y_2 un ensemble fermé prédécesseur immédiat de (i.e., incluant) Y_1 .

Ainsi, pour déterminer l'ensemble des règles informatives exactes et approximatives, il suffit de connaître l'ensemble des générateurs associés à l'*intent* de chacun des concepts du treillis de Galois. La notion de générateur est donc primordiale dans la génération des règles d'association informatives.

À titre d'exemple, l'ensemble des générateurs du concept $(126, abc)$ correspond à l'unique générateur formé de l'attribut a . Ce dernier correspond en effet au plus petit ensemble d'attributs inclus dans l'*intent* de ce concept et non inclus dans l'*intent* de son unique parent. La règle d'association informative exacte $a \rightarrow bc$ de support égal à $3/8$ (support du concept $(126, abc)$) va être générée. Comme le concept courant admet deux prédécesseurs (enfants) immédiats $((16, abcd), (12, abcef))$, deux règles approximatives vont découler de ce concept : $a \rightarrow bcd$ (support = $2/8$ et confiance = $2/3$) et $a \rightarrow bcef$ (support = $2/8$ et confiance = $2/3$).

2.3. Extraction des générateurs

Définition 6 (Face)

Soit $N = (X, Y)$ un concept formé de l'*extent* X et de l'*intent* Y et $\text{Successeur}_i(N)$ le $i^{\text{ème}}$ successeur (parent) immédiat de N dans le treillis de Galois. La $i^{\text{ème}}$ face du concept N correspond à la différence entre son *intent* et l'*intent* de son $i^{\text{ème}}$ successeur.

Soit p le nombre de successeurs immédiats du concept N . La famille des faces F_N du concept N s'exprime alors par la relation suivante :

$$F_N = \{ Y - Intent (Successeur_i(N)), i \in \{1..p\} \}$$

Définition 7 (Bloqueur)

Soit $G = \{ G_1, G_2, \dots, G_n \}$ une famille d'ensembles. Un bloqueur de la famille G est un ensemble dont l'intersection avec tous les ensembles $G_i \in G$ est non vide.

Exemple : si $G = \{AB, AC\}$, alors les ensembles $\{A\}$, $\{AB\}$ et $\{AC\}$ sont des bloqueurs de la famille G .

Définition 8 (Bloqueur minimal)

Un bloqueur B d'une famille d'ensemble $G = \{ G_1, G_2, \dots, G_n \}$ est dit minimal s'il n'existe aucun bloqueur B' de G inclus dans B .

Dans l'exemple précédent, seul l'ensemble $\{A\}$ est un bloqueur minimal de G .

Propriété 1 (Générateur)

Soit $N = (X, Y)$ un concept formé de l'*extent* X et de l'*intent* Y . Soit F_N la famille de ses faces. L'ensemble des générateurs G associé à l'*intent* Y du concept N correspondent aux bloqueurs minimaux de la famille des faces F_N .

À titre d'exemple, la famille des faces du concept $c = (16, abcd)$ correspond à $F_c = \{a, d\}$. L'ensemble des bloqueurs minimaux de la famille F_c est le singleton $\{ad\}$.

3. Construction des générateurs et génération des règles

3.1. Construction des générateurs

Quelques travaux de recherche se sont basés sur la notion de générateur pour la construction des règles d'association. Nous présentons brièvement trois d'entre eux.

3.1.1. Les différentes approches

L'approche proposée par Godin et Missaoui (Godin *et al.*, 1994) est intuitive. À chaque concept fréquent est initialement associé un ensemble de générateurs potentiels. Cet ensemble est formé par toutes les combinaisons croissantes des attributs inclus dans l'*intent* de ce concept. Ensuite, un premier élagage est effectué sur les candidats inclus dans l'*intent* d'un des successeurs immédiats du concept

courant. Enfin, tous les générateurs non minimaux sont supprimés. Le principe est ensuite appliqué à tous les concepts du treillis.

La création des générateurs dans AClose (Pasquier, 2000) est différente, puisque celle-ci est réalisée avant l'extraction des *itemsets* fermés fréquents (*intent* des concepts fréquents). L'ensemble des générateurs est produit par niveau selon la taille de leur *itemset*. Un générateur de taille i est produit en joignant deux générateurs de taille $i-1$ possédant un $i-2$ préfixe commun. L'élagage des candidats s'opère ensuite : les candidats non fréquents et non minimaux sont éliminés. Il est à noter que dans la première approche, la fréquence des générateurs candidats n'est pas à considérer, car ces derniers sont tous des sous-ensembles d'un *intent* fréquent.

Le dernier algorithme, appelé *AGenLocal* (algorithme Apriori pour la génération locale de générateurs), est une approche hybride des deux premiers. Les générateurs potentiels sont construits selon le même principe de jointure qu'AClose, appliqué initialement sur les différents *items* de l'*intent* du concept courant. L'obtention des générateurs se fait ensuite sur les mêmes critères de sélection que Godin et al, avec en supplément un test de vérification de la minimalité des générateurs produits après chaque jointure.

3.1.2. L'algorithme JEN

L'idée des bloqueurs minimaux a été proposée par Pfaltz (Pfaltz, 2002) dans le cadre d'une construction incrémentale des générateurs. Notre approche s'appuie sur la même propriété, à savoir qu'un générateur associé à un concept est un bloqueur minimal de la famille des faces de ce concept. Notre algorithme présente cependant des optimisations non envisageables dans un contexte incrémental.

Nous considérons tout d'abord deux types de concepts : les concepts admettant un seul parent et ceux ayant plusieurs parents. Les générateurs des concepts du premier type correspondent aux différents attributs élémentaires (*items*) appartenant à l'unique face du concept. La définition des bloqueurs minimaux pour les générateurs ne rentre en jeu que pour le 2^{ème} type de concept.

Soit N un concept du treillis de Galois possédant p parents et $F_N = \{F_1, F_2, \dots, F_j, F_{j+1}, \dots, F_p\}$ la famille des faces du concept N . Soit JEN_j l'ensemble des bloqueurs minimaux de la famille des j premières faces à partir de JEN_j et F_{j+1} . Nous cherchons à déterminer l'ensemble des bloqueurs minimaux JEN_{j+1} de la famille des $j+1$ premières faces. Pour cela, nous effectuons l'intersection de la $(j+1)$ ème face F_{j+1} avec chacun des générateurs de l'ensemble JEN_j . Deux cas peuvent alors se produire :

- **Cas 1** : L'intersection est non vide. Dans ce cas, aucun traitement n'est effectué puisque le générateur courant est aussi un bloqueur minimal de la famille formée des $j+1$ faces. Nous rajoutons donc le générateur courant à l'ensemble JEN_{j+1} .

Cas 2 : L'intersection avec le générateur courant g est vide. Cela implique que g n'est plus un bloqueur pour la famille formée des $(j+1)$ èmes faces. Le générateur courant g doit alors être modifié afin qu'il soit un bloqueur minimal de la famille des $j+1$ faces. Nous allons créer autant de nouveaux générateurs qu'il y a d'attributs dans la face F_{j+1} . Ces nouveaux générateurs seront obtenus en rajoutant chacun des attributs de la face courante au générateur courant g . Ces nouveaux générateurs ne seront pas rajoutés directement à l'ensemble JEN_{j+1} , car ceux-ci peuvent être des sur-ensembles des bloqueurs minimaux générés par une intersection vide. Nous les rajoutons donc dans un nouvel ensemble JEN'_{j+1} .

Une fois l'intersection de la face F_{j+1} avec l'ensemble des générateurs de JEN_j effectuée, nous disposons de deux nouveaux ensembles JEN_{j+1} et JEN'_{j+1} . L'étape suivante consiste à insérer tous les bloqueurs minimaux de JEN'_{j+1} dans JEN_{j+1} .

3.1.3. Pseudo - Code

Nous utilisons dans le pseudo-code une structure Nœud qui correspond à un nœud du treillis de concepts. Celle-ci possède plusieurs champs : son intent (**intent**), son extent (**extent**), son support (**support**), sa liste de prédécesseurs immédiats (**predecesseurs**), sa liste de successeur immédiats (**successeurs**) ainsi qu'un ensemble de générateurs (**generateurs**).

La procédure suivante permet de déterminer, pour un nœud donné, l'ensemble des bloqueurs minimaux de la famille des $j+1$ premières faces à partir de JEN_j (ensemble des bloqueurs minimaux de la famille des j premières faces) et de la face associée au $j+1$ ème parent du nœud courant.

Entrées : F_{j+1} : face associée au $j+1$ ème parent du nœud courant; JEN_j : ensemble des bloqueurs minimaux de la famille des j premières faces
Sortie : JEN_{j+1} : ensemble des bloqueurs minimaux de la famille des $j+1$ premières faces
 $JEN'_{j+1} \leftarrow \emptyset$
 $JEN_{j+1} \leftarrow \emptyset$
pour tout generateur $\in JEN_j$ **faire**
 intersection \leftarrow generateur $\cap F_{j+1}$
 si intersection = \emptyset **alors**
 pour tout attribut $\in F_{j+1}$ **faire**
 generateur' = generateur \cup { attribut }

```

        JEN'_{j+1} ← JEN'_{j+1} ∪ { generateur' }
    fin pour
    sinon
        JEN_{j+1} ← JEN_{j+1} ∪ { generateur }
    fin si
fin pour
// Élimination des bloqueurs non minimaux de l'ensemble JEN'_{j+1}
si JEN_{j+1} = ∅ alors
    (1) renvoyer JEN'_{j+1}
sinon
    si JEN'_{j+1} = ∅ alors
        (2) renvoyer JEN_{j+1}
    sinon
        TEMP ← ∅
        pour tout generateur ∈ JEN'_{j+1} faire
            si non ( generateur ⊆ JEN_{j+1} ) alors
                TEMP ← TEMP ∪ { generateur }
            fin si
        fin pour
        (3) renvoyer JEN_{j+1} ∪ TEMP
    fin si
fin si

```

Algorithme 1. *calculBloqueursMinimaux*

Si pour le traitement d'une face F_{j+1} , toutes ses intersections avec l'ensemble JEN_j sont vides, alors tous les bloqueurs créés dans JEN'_{j+1} sont minimaux. De même, si toutes les intersections sont non vides, alors tous les bloqueurs de JEN_{j+1} sont minimaux. Le test de minimalité est alors inutile dans ces deux cas.

```

Entrée :  $N$ : Noeud
Sortie :  $JEN$  : ensemble des générateurs du nœud  $N$ 
ensembleFace = calculFace ( $N$ );
     $JEN$  ← { attributs de la première face };
si  $N$ .successeurs > 1 alors
    On élimine la première face de l'ensemble des faces.
    pour toute face ∈ ensembleFace faire
         $JEN'$  ← calculBloqueursMinimaux (face,  $JEN$ )
         $JEN$  ←  $JEN'$ 
    fin pour
fin si
renvoyer  $JEN$ ;

```

Algorithme 2. *calculGénérateurNoeud*

Entrée : L : ensemble des nœuds du treillis
Sortie : L : ensemble des nœuds dans lequel les générateurs ont été calculé
pour tout noeud $\in L$ **faire**
 noeud.générateurs = calculGénérateurNoeud (noeud)
fin pour
renvoyer L

Algorithme 3. *JEN*

Exemple

Soit le concept courant $c = (12, abcef)$. L'ensemble des faces de ce nœud va être calculé. Comme c admet comme successeurs immédiats $(126, abc)$ et $(1235, ef)$, l'ensemble des faces vaut $\{ef, abc\}$. Ensuite, l'ensemble des générateurs courants de ce nœud, JEN_j est initialisé avec l'ensemble des attributs de la première face : $JEN_j = \{e, f\}$ et la première face est supprimée de l'ensemble des faces : $ensembleFaces = \{abc\}$ (algorithme 2). Puis, l'unique face restante abc de l'ensemble des faces va être intersectée avec le premier bloqueur minimal e de JEN_j . Comme l'intersection est vide, les bloqueurs ae, be et ce vont être créés et rajoutés à l'ensemble JEN'_{j+1} (algorithme 1). Enfin, la face abc va être intersectée avec le dernier bloqueur de l'ensemble $JEN_j : f$. Le même raisonnement est appliqué et JEN'_{j+1} contient finalement les bloqueurs ae, be, ce, af, bf et cf . Et, comme les intersections de la face courante abc avec l'ensemble des bloqueurs courants JEN_j sont toutes vides, tous les bloqueurs créés sont minimaux et l'ensemble $\{ae, be, ce, af, bf, cf\}$ constitue la collection des générateurs de ce nœud (algorithme 3).

3.1.4. *Comparaison avec l'approche de Pfaltz*

Dans l'approche de Pfaltz, l'ajout d'un nouveau successeur (*newc*) à un concept existant du treillis (*covc*) du treillis nécessite deux étapes. Lors de la première étape, l'ensemble des bloqueurs courants du concept *covc* est parcouru une première fois pour déterminer ceux qui resteront inchangés (leur intersection avec la face courante est non vide). Dans la seconde étape, les bloqueurs qui ne seront pas conservés dans l'étape précédente sont reparcourus. Ceux-ci sont en effet susceptibles de devenir minimaux après être augmentés de chacun des *attributs* de la face courante. Finalement, un test de minimalité des nouveaux bloqueurs créés est effectué dans cette deuxième étape.

L'inconvénient de l'approche de Pfaltz réside dans le fait que plusieurs éléments de l'ensemble courant des bloqueurs sont parcourus deux fois. Ils sont notamment

parcourus tous deux fois lorsque toutes les intersections de la face courante avec l'ensemble courant des bloqueurs sont vides, ce qui n'est pas le cas dans notre approche. D'autre part, comme notre algorithme est non incrémental, nous connaissons initialement le nombre de parents du concept courant. Enfin, l'identification du cas de nœuds avec un seul parent évite des calculs inutiles.

3.2. Génération des règles d'association informatives

La génération des règles d'association informatives exactes repose sur la détermination des concepts et des générateurs qui leur sont associés, tandis que la production des règles approximatives se base sur le calcul des générateurs et des prédécesseurs (enfants) immédiats de leur concept associé. Les bases pour les règles exactes (RIE) nécessitent deux balayages: celui de chaque concept et pour chaque concept celui de ses générateurs. Les bases pour les règles approximatives (RIA), quant à elles, demandent un parcours supplémentaire: celui des prédécesseurs immédiats du concept courant. De plus, la confiance de ces règles approximatives doit être supérieure ou égale à la confiance minimale définie par l'utilisateur (*minConfiance*).

4. Résultats expérimentaux

La programmation des algorithmes a été effectuée en Java. Les tests de performance des algorithmes ont été réalisés sur un pentium III (1Ghz) possédant 512 Mo de mémoire. Les différents programmes *AClose*, *Godin et al.*, *AGenLocal* et *Jen* ont été testés à partir de deux bases du domaine public correspondant respectivement à des données fortement corrélées (*mushroom*) et des données faiblement corrélées (10K10K). Les caractéristiques de ces deux bases sont énoncées dans la table ci-dessous.

Bases de données	Nombre d'items	Nombre d'objets	Taille moyenne des objets
<i>mushroom</i>	120	8124	23
10K10K	10000	10000	40

Tableau 1. Jeux de données

Les trois premiers algorithmes *Godin et al.*, *AGenLocal* et *Jen* travaillent à partir du même fichier d'entrée, à savoir l'ensemble des *itemsets* fermés fréquents ordonnés. L'ensemble des *itemsets* fermés fréquents a été préalablement obtenu par CHARM (*Zaki et al.*, 1999). Comme CHARM ne reconstruit que partiellement l'ordre dans le treillis des *itemsets* fermés fréquents, nous lui avons appliqué un algorithme naïf de construction de l'ordre basé sur l'inclusion des *itemsets* fermés

fréquents. Les résultats expérimentaux sont présentés à la page suivante selon une échelle logarithmique. La première colonne concerne les données fortement corrélées, et la seconde les données faiblement corrélées.

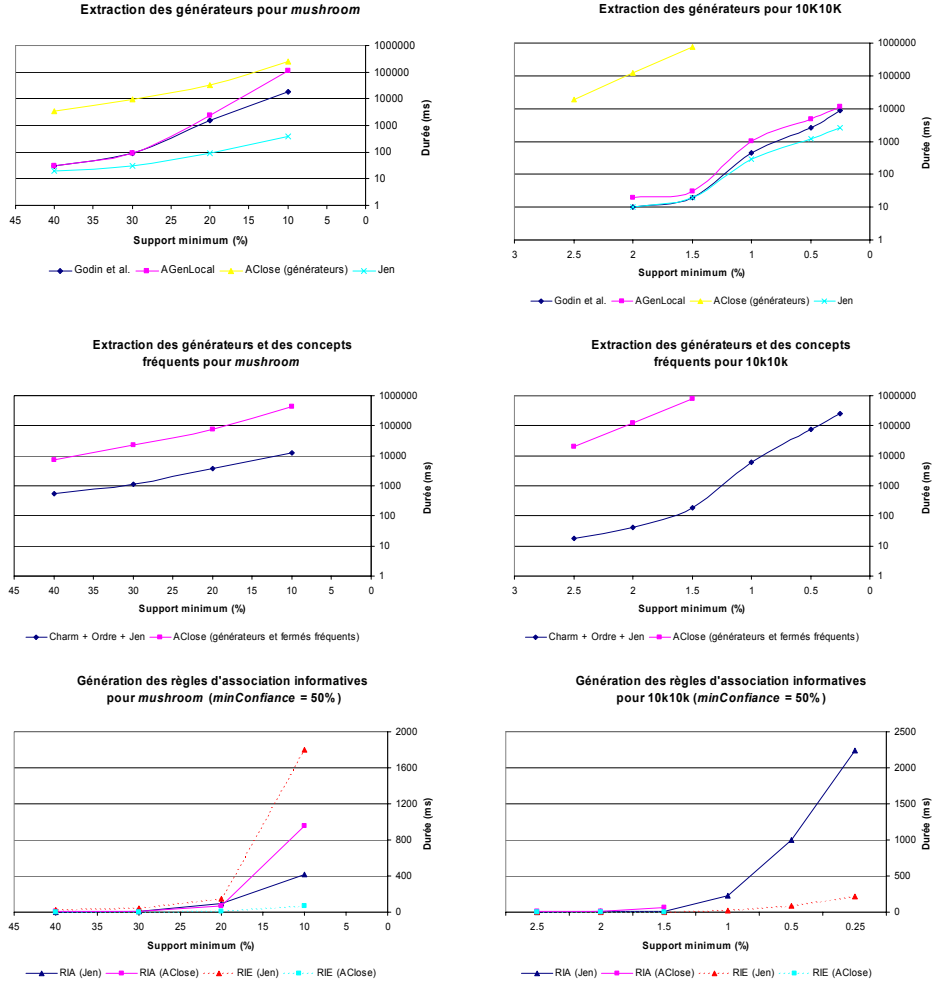


Figure 2. Résultats expérimentaux. Gauche : données corrélées. Droite : données faiblement corrélées

4.1 Données fortement corrélées

Le premier graphique à gauche indique les temps de réponse dans la construction des générateurs pour chacun des algorithmes. L'algorithme JEN s'avère très performant pour les supports faibles. Il est en effet environ 40 fois (resp. 200 fois et 600 fois) plus rapide que Godin *et al.* (resp. AGenLocal et AClose) pour un support de 10%. Les résultats pour des supports avantagent également JEN par rapport aux

trois autres algorithmes. Ainsi, nous observons un gain de 3 par rapport à Godin *et al* et AGenLocal ainsi qu'un facteur multiplicatif de 30 par rapport à AClose. Les performances peu élevées de Godin *et al*, peuvent s'expliquer par la création des générateurs candidats qui est exponentielle avec la taille de l'*intent* des concepts. Quand le support est faible, la taille de l'*intent* de nombreux concepts sera élevé, ce qui n'avantage guère les trois autres algorithmes. La procédure AGenLocal est limitée par les calculs de jointure répétitifs dans la construction des générateurs ainsi que par le test de minimalité des générateurs.

L'algorithme AClose souffre des mêmes difficultés dans le calcul des jointures. Il doit en plus calculer le support des générateurs candidats produits (balayages de la base de données) pour d'une part élaguer les candidats non fréquents et d'autre part éliminer les candidats non minimaux. L'algorithme AClose n'est pas tout à fait comparable avec les trois premiers algorithmes dans le sens où il extrait l'ensemble des générateurs pour ensuite calculer les *itemsets* fermés fréquents. Pour mesurer l'impact de la construction des générateurs de notre algorithme, nous avons tout d'abord comparé notre approche à celle de AClose en terme d'extraction des générateurs et des *itemsets* fermés fréquents (deuxième graphique à gauche). Les performances de CHARM comparées à celles de AClose sont du même ordre de grandeur que celles présentées dans (Zaki *et al.*, 1999). L'ajout du temps de construction de l'ordre dans le treillis et celui de l'algorithme JEN à CHARM diminue cet ordre de grandeur, mais notre extraction des générateurs et des concepts fréquents ordonnés surpasse celle de AClose.

Le troisième graphique à gauche nous indique les performances dans la génération des règles d'association informatives pour AClose et notre approche. Les tests concernant la génération des règles informatives exactes (RIE) sont plus optimaux avec AClose. Cela est dû au fait qu'il suffit de balayer l'ensemble des générateurs pour retrouver leur fermé associé. Dans notre approche, la surcoût est engendré par un double balayage : celui des concepts et pour chaque concept celui de ses générateurs. Par contre, la génération des règles approximatives (RIA) avec une confiance minimale de 50%, est favorable à notre approche. En effet, nous disposons, grâce à la construction de l'ordre dans le treillis, de l'ensemble des prédécesseurs (enfants) immédiats d'un concept fréquent. La génération des règles approximatives avec AClose nécessite plus de traitement puisqu'elle nécessite le calcul des prédécesseurs immédiats pour les fermés associés aux générateurs.

4.2. Données faiblement corrélées

Les résultats des tests de l'algorithme JEN sur l'extraction des générateurs sont moins frappants pour les données faiblement corrélées (deuxième colonne). En effet, notre algorithme est respectivement 3 et 4 fois plus rapide que Godin *et al* et AGenLocal pour les supports faibles (premier graphique). Pour les supports élevés

(supérieurs à 1.5%), les performances des ces trois algorithmes sont presque similaires. Concernant les performances de AClose dans l'extraction des générateurs, celles-ci sont amoindries par le nombre de générateurs extraits et par le nombre de balayages du contexte nécessaires pour mesurer leur support et tester leur minimalité. Du fait que la taille d'*itemset* des générateurs est plus restreinte dans le cas des données faiblement corrélées, le calcul de jointure dans AGenLocal et AClose est moins limitant. Les deux autres graphiques de droite illustrent les mêmes phénomènes que précédemment : optimalité de notre approche non seulement dans l'extraction des concepts fréquents et des générateurs (deuxième graphique) mais aussi dans la génération des règles approximatives (troisième graphique²), et optimalité des performances de AClose dans la génération des règles exactes.

5. Conclusion

Nous proposons un nouvel algorithme *JEN* de construction des générateurs à partir du treillis des concepts. Cet algorithme offre des performances très intéressantes non seulement lors de la construction des générateurs mais aussi pour la génération des règles d'association informatives. Une analyse empirique de *JEN* et trois autres algorithmes montre qu'il est le meilleur pour les données fortement corrélées et bien performant dans le cas des données faiblement corrélées.

6. Bibliographie

Agrawal R., Srikant R., Fast algorithms for mining association rules in larges databases. *In Proceeding of the 20th international conference on Very Large Dada Bases (VLDB'94)*, pages 478-499. Morgan Kaufmann, September 1994.

Bastide Y., Taouil R., Pasquier N., Stumme G., Lakhil L. « PASCAL : un algorithme d'extraction de motifs fréquents », *Technique et Science Informatiques*, vol. 21, n° 1, 2002, p. 65-95.

1. Par manque d'espace mémoire, la courbe de génération des règles d'association informatives exactes et approximatives n'a pu être complétée avec AClose pour des supports faibles (<1.5%).

- Ganter B., Wille R., *Formal Concept Analysis, Logical Foundations*. Springer-Verlag, 1999.
- Godin R., Missaoui R., *An incremental concept formation approach for learning from databases*. In *theoretical Computer Science*, volume 133, pages 387-419, 1994.
- Pasquier N., *Data mining : algorithmes d'extraction et de réduction des règles d'association dans les bases de données*. Thèse de doctorat, Université de Clermont-Ferrant II, Janvier 2000.
- Pfaltz J.L., Taylor C.M., *Scientific Discovery through Iterative Transformations of Concept Lattices*. Workshop on Discrete Applied Mathematics in conjunction with *the 2nd SIAM International Conference on Data Mining*, pages 65-74, Arlington, VA, Apr. 2002.
- Zaki M.J., Hsiao C.-J., *ChARM: An Efficient Algorithm for Closed Association Rule Mining*. Technical Report 99-10, Rensselaer Polytechnic Institute, Troy, New York, 1999.