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
rokia.missaoui@uqo.ca
*** Département d'informatique et recherche opérationnelle, Université de Montréal

Résumé. L'article décrit un algorithme, appelé *JEN*, d'identification efficace des générateurs à partir du treillis de concepts (Galois) pour l'extraction des règles d'association. Cette dernière est immédiate dans notre approche : les règles exactes sont obtenues à partir des concepts individuels en exploitant leurs générateurs et *l'itemset* fermé correspondant, tandis que les règles approximatives sont identifiées en consultant les générateurs d'un concept et *l'itemset* fermé des prédécesseurs immédiats de ce concept. 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.

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 1994) en mettant davantage l'accent sur les travaux opérant dans le contexte de l'analyse formelle des concepts (Ganter 1999). Plusieurs études (Godin 1994), (Zaki 2002), (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 1994), (Pasquier 2000), (Pfaltz 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 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)

Étant donné un contexte *K = (O, A, R)* dans lequel *R* est une relation binaire entre un ensemble d'objets *O* et un ensemble d'attributs *A*, l'analyse formelle de concepts (Ganter 1999) produit un treillis de concepts constitué d'un ensemble de concepts partiellement ordonnés par inclusion ensembliste. Un concept est défini par son extension (*extent*) et sa compréhension (*intent*) lesquelles sont obtenues en utilisant la fermeture de la connexion de Galois d'une relation binaire. L'extension énumère les objets du concept alors que la compréhension spécifie les attributs partagés par ces objets. Cette dernière correspond à la notion d'*itemset* fermé dans l'analyse du panier du consommateur (Pasquier 2000).

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	x	x	x	x	x	x	x	x
2	x	x	x		x	x		
3					x	x	x	x
4							x	
5					x	x		x
6	x	x	x	x				
7		x	x	x				
8				x				

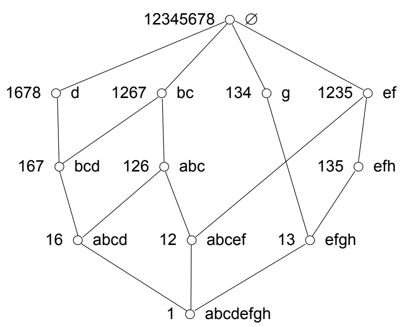

FIG. 1 - *Gauche: contexte K. Droite: Diagramme de Hasse du treillis de Galois induit par le contexte K.*

Le treillis de concepts est représenté par un diagramme de Hasse qui permet de visualiser les concepts et la relation d'ordre. Par exemple, (167, *bcd*) est un concept ayant deux successeurs (parents) immédiats et un seul prédécesseur immédiat.

2.2 Règles d'association

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 (totale) dont la validité exige que Y_2 soit présent chaque fois que Y_1 l'est, et une règle approximative (partielle) 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 (c-à-d ayant un support dépassant un seuil prédéfini) 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.

2.2.1 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.

Définition 1 (Générateur)

Le générateur Z de l'*itemset* fermé X est un sous-ensemble minimal de X tel que sa fermeture (notée par ") soit égale à X. L'ensemble G_Z des générateurs de X est :

$G_Z = \{Z \subseteq A \mid Z'' = X$ et il n'existe aucun $Y \subset Z$ tel que $Y'' = X \}$

Définition 2 (Règle d'association informative exacte)

Une règle d'association informative exacte (Guigues 1986), (Godin 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 3 (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. À titre d'exemple, l'ensemble des générateurs du concept (126, *abc*) correspond à au singleton $\{a\}$. La règle d'association informative exacte $a \rightarrow bc$ de support égal à 3/8 va être générée. Comme le concept courant admet deux prédécesseurs (enfants) immédiats : (16, abcd) et (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 4 (Face)

Soit N = (X, Y) un concept formé de l'*extent* X et de l'*intent* Y et $\text{Successeur}_i(N)$ le $i^{ème}$ successeur (parent) immédiat de N dans le treillis de Galois. La $i^{ème}$ face du concept N correspond à la différence entre son *intent* et l'*intent* de son $i^{ème}$ 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 (Pfaltz 2002):

$F_N = \{ Y - \mathit{Intent}(\text{Successeur}_i(N)), i \in \{1..p\} \}$

Définition 5 (Bloqueur)

Soit $G = \{ G_1, G_2, \ldots G_n \}$ 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 6 (Bloqueur minimal)

Un bloqueur B d'une famille d'ensemble $G = \{ G_1, G_2, \ldots G_n \}$ 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

Dans ce qui suit, nous décrivons brièvement quelques travaux de recherche qui se sont basés sur la notion de générateur pour la construction des règles d'association.

3.1.1 Les différentes approches

L'approche proposée par Godin et Missaoui (Godin 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. Tous les candidats inclus dans l'*intent* d'un des succcesseurs immédiats du concept courant sont écartés. 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* à la manière de l'algorithme *Apriori* (Agrawal 1994). 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*[1] (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 que la procédure *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 $F_N = \{F_1, F_2, \ldots, F_j, F_{j+1}, \ldots, F_p\}$ 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 de F_{j+1} avec le générateur courant g 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 de F_{j+1} 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*

[1] Algorithme non publié.

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 F_{j+1} au générateur courant *g*. Ces nouveaux générateurs ne seront pas rajoutés directement à l'ensemble JEN_{j+1}, car ceux-ci peuvent être non minimaux. 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 : l'*intent* (**intent**), l'*extent* (**extent**), le support (**support**), la liste de prédécesseurs immédiats (**predecesseurs**), la liste de successeur immédiats (**successeurs**) ainsi que l'ensemble de générateurs (**generateurs**).

La procédure suivante permet de déterminer, pour un nœud donné, JEN_{j+1} (ensemble des bloqueurs minimaux de la famille des j+1 premières faces) à partir de JEN_j et de la face associée au j+1ème 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} ← ∅
JEN_{j+1} ← ∅
pour tout generateur ∈ JEN_j faire
        intersection ← generateur ∩ F_{j+1}
        si intersection = ∅ alors
                pour tout attribut ∈ F_{j+1} faire
                        generateur' ← generateur ∪ { 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 extrêmes.

L'algorithme 2 se base sur la procédure précédente pour calculer l'ensemble des générateurs associés à l'*intent* d'un concept spécifique, alors que l'algorithme 3 effectue ce traitement pour l'ensemble des concepts du treillis.

```
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 – *calculGenerateurNoeud*

```
Entrée : L: ensemble des nœuds du treillis
Sortie : L : ensemble des nœuds du treillis avec spécification de leurs générateurs
pour tout noeud ∈ L faire
    nœud.generateurs ← calculGenerateurNoeud (nœud)
fin pour
renvoyer L
```

ALGORITHME 3 – *JEN*

Exemple

Soit le concept *c = (12, abcef)*. L'ensemble des faces de ce nœud va d'abord être calculé. Comme *c* admet comme successeurs immédiats *(126, abc)* et *(1235, ef)*, l'ensemble des faces est : *ensembleFaces = {ef, abc}*. Ensuite, l'ensemble des générateurs courants de ce nœud, JEN_1 est initialisé avec l'ensemble des attributs de la première face : JEN_1 = *{e, f}* et la première face est retirée de *ensembleFaces*. 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}. Enfin, la face *abc* va être intersectée avec le dernier bloqueur *f* de l'ensemble JEN_j. Le même raisonnement est appliqué et JEN'_{j+1} contient finalement les bloqueurs *ae, be, ce, af, bf et cf*. 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 l'ensemble des générateurs du concept *c*.

3.1.4 Comparaison avec l'approche de Pfaltz

Dans l'approche de Pfaltz, la détermination des générateurs se fait d'une manière incrémentale comme suit : l'ajout d'un nouveau successeur (*newc*) à un concept existant du treillis (*covc*) 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 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 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 informatives exactes (RIE) nécessitent deux balayages: celui de chaque concept et pour chaque concept celui de ses générateurs. Les bases pour les règles informatives 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 une confiance donnée.

4 Résultats expérimentaux

La programmation des algorithmes a été effectuée en Java dans la plate-forme Galicia qui est un environnement de développement d'outils de construction, visualisation et exploitation des treillis de concepts. Les tests de performance des algorithmes ont été réalisés sur un pentium III (1Ghz) possédant 512 Mo de mémoire. Les différents programmes *AClose*, *Godin 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 bases sont énoncées ci-après.

Bases de données	**Nombre d'*items***	**Nombre d'objets**	**Taille moyenne des objets**
mushroom	120	8124	23
10K10K	10000	10000	40

TAB. 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 2002). 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.

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* (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 répétitifs de jointure et le test de minimalité dans la construction des générateurs.

Au même titre que la procédure *AgenLocal*, l'algorithme *AClose* souffre de la lourdeur du 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 dans 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 2002). 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*.

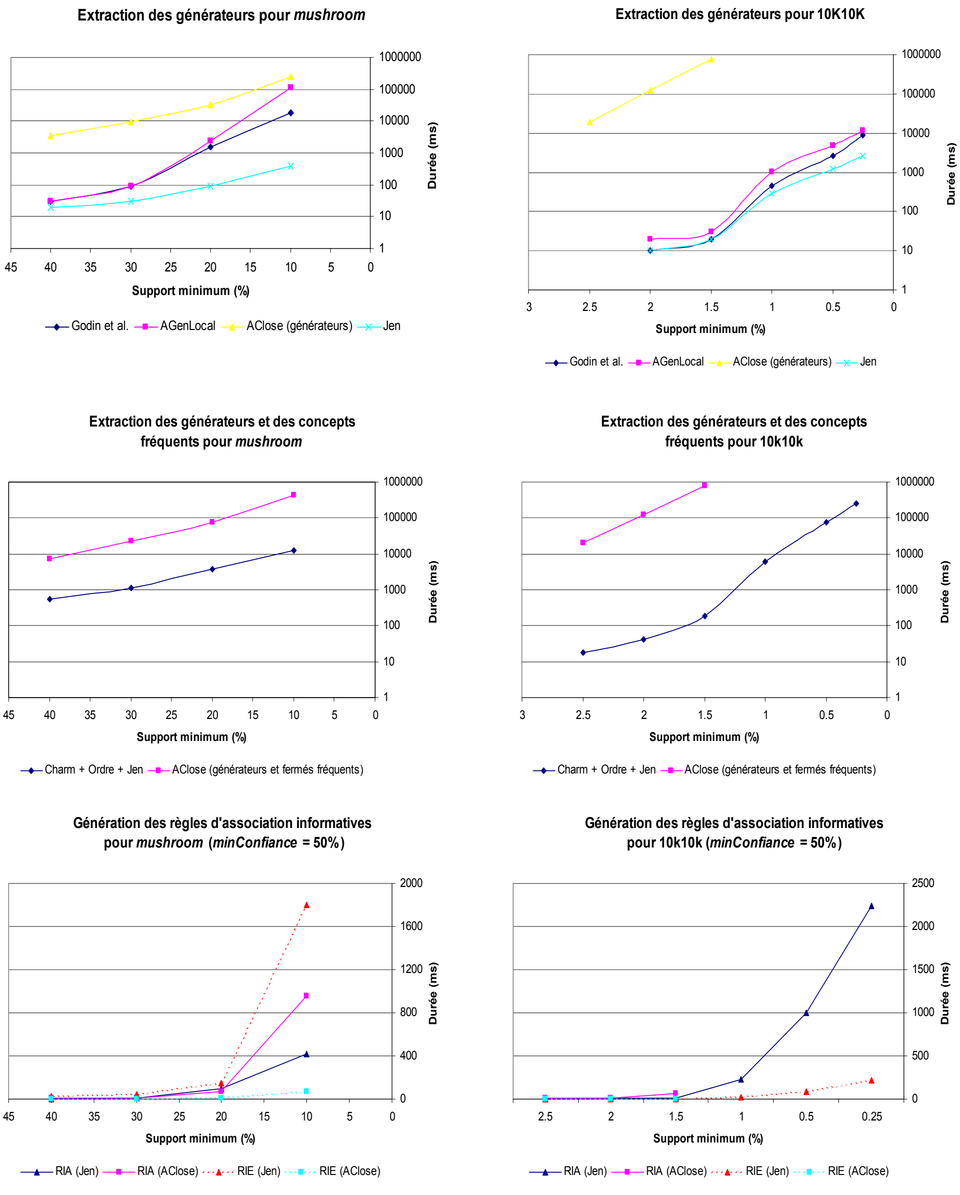

FIG. 2 - *Résultats expérimentaux. Gauche : données corrélées. Droite : données faiblement corrélées*

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 (colonne de droite). En effet, notre algorithme est respectivement 3 et 4 fois plus rapide que *Godin et al* et *AGenLocal* pour les supports faibles. Pour les supports élevés (supérieurs à 1.5%), les performances de 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 graphique[2]), et optimalité des performances de *AClose* dans la génération des règles exactes.

5 Conclusion

Comme étape préliminaire à la génération des règles d'association, nous proposons un nouvel algorithme *JEN* de construction des générateurs à partir des *itemsets* fermés identifiés dans le treillis des concepts. Une analyse empirique montre qu'il est beaucoup plus performant que les algorithmes existants d'extraction des générateurs (Godin et al, AgenLocal, AClose), notamment pour les données corrélées. En outre, du fait que cet algorithme est incorporé dans un environnement de construction de treillis de concepts, la génération des règles d'association informatives ne nécessite aucun traitement supplémentaire, alors que les algorithmes AClose, Close et Pascal doivent tous calculer les sur-ensembles immédiats de chaque fermé. En effet, les règles exactes sont obtenues à partir des concepts individuels en exploitant leurs générateurs et *l'itemset* fermé correspondant, tandis que les règles approximatives sont identifiées en consultant les générateurs d'un concept et *l'itemset* fermé des prédécesseurs immédiats de ce concept.

---

[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%).

Une de nos recherches courantes consiste à définir un cadre général pour une mise à jour incrémentale efficace à la fois du treillis de concepts, des générateurs et des bases de règles lorsqu'un ensemble d'objets est rajouté à la collection initiale.

Remerciements

Nous tenons à remercier les évaluateurs anonymes pour leurs judicieuses remarques et remercier le CRSNG (Conseil de Recherches en Sciences Naturelles et Génie du Canada) et VRQ (Valorisation Recherche Québec) pour les subventions accordées.

Références

Agrawal R., Srikant R. (1994), 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., Lakhal L. (2002), « PASCAL : un algorithme d'extraction de motifs fréquents », Technique et Science Informatiques, vol. 21, n° 1, 2002, p. 65-95.

Ganter B., Wille R. (1999), Formal Concept Analysis, Logical Foundations. Springer-Verlag, 1999.

Godin R., Missaoui R. (1994), An incremental concept formation approach for learning from databases. In theorical Computer Science, Vol. 133, pp 387-419, 1994.

Guigues, J.L, Duquenne V. (1986), Familles minimales d'implications informatives résultant d'un tableau de données binaires. Mathématiques et Sciences Humaines, Vol. 95, pp 5-18, 1986.

Pasquier N. (2000), 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 JL., Taylor CM. (2002), 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, 2002.

Zaki M.J., Hsiao C.-J. (2002), ChARM : An Efficient Algorithm for Closed Association Rule Mining. Proceedings of the 2nd SIAM International Conference on Data Mining, 2002.