Approches d'extraction de règles d'association basées sur la correspondance de Galois

Sadok Ben Yahia — **Engelbert Mephu Nguifo**

Centre de Recherche en Informatique de Lens - IUT de Lens
Rue de l'Université SP 16, F-62307 Lens cedex
{benyahia,mephu}@cril.univ-artois.fr

RÉSUMÉ. *Les données stockées peuvent cacher un certain nombre de connaissances, de dépendances ou de corrélations, qui sont implicites et utiles, et qui n'attendent qu'à être explorées. Dans ce contexte, un certain nombre d'algorithmes, s'inspirant de l'algorithme* APRIORI, *basés sur l'extraction des itemsets fréquents ont été présentés. La particularité de ces algorithmes est qu'ils génèrent un nombre exorbitant de règles rendant leur exploitation quasiment impossible par des experts. Dans ce papier, nous proposons de faire un état de l'art sur les algorithmes d'extraction des règles associatives basés sur l'extraction des itemsets fermés. Ces algorithmes sont vus comme une alternative prometteuse pour réduire la taille des règles associatives découvertes. Nous proposons ainsi une catégorisation de ces différents algorithmes sur la base de critères et de dimensions que nous avons définis à cet effet.*

ABSTRACT. *As a side effect of unprecedented amount of digitization of data, classical retrieval tools found themselves unable to go further beyond the tip of the Iceberg. Data Mining, with a clear promise to furnish adequate tools to do so, is the discovery of hidden information found in databases. In this paper, we are interested in presenting a thorough survey of Galois connection semantics-based approaches for mining association rules. We provide critical classification and comparison of theses algorithms based on criteria that we introduce to try to find an answer to the question: Does this novel approach avoid knowledge user-overwhelming?*

MOTS-CLÉS : *fouille de données, analyse de concepts formels, règles d'associations.*

KEYWORDS: *data mining, formal concept analysis, association rules.*

1. Introduction

La puissance de calcul des ordinateurs d'aujourd'hui, ainsi que la diminution des coûts de stockage, laissent prédire que nous disposerons de plus en plus de moyens physiques très performants pour faire face à l'accroissement du volume de données. Cependant malgré ces ressources, les outils classiques de gestion des données s'avèrent incapables de détecter des connaissances à partir d'ensembles de données. Le domaine de la fouille de données (ou Datamining) est apparu avec la promesse de fournir les outils et/ou techniques pour découvrir les connaissances, utiles et préalablement inconnues dans ces gisements de données [ADR 97].

La technique la plus connue concerne la découverte de règles associatives. Le problème de découverte des règles associatives, initialement introduit par Agrawal pour l'analyse du panier de la ménagère [AGR 93], peut être décomposé en deux sous-problèmes :

1) Trouver tous les « patrons » ou itemsets fréquents, qui apparaissent dans la base de données avec une fréquence supérieure ou égale à un seuil défini par l'utilisateur, appelé *minsup*.

2) Générer l'ensemble des règles associatives, à partir de ces patrons fréquents, ayant une mesure de confiance supérieure ou égale à un seuil défini par l'utilisateur, appelé *minconf*.

La quasi-totalité des travaux liés à ce problème se sont intéressés au premier sous-problème. En effet, l'espace de recherche, pour trouver les patrons fréquents, est de taille exponentielle en fonction du nombre des items. Ainsi, de longs efforts ont été faits, suite à l'apparition de l'algorithme APRIORI [AGR 94], pour trouver les bonnes heuristiques pour l'élagage de l'espace de recherche. De manière générale, nous pouvons dire que les algorithmes de découverte des règles associatives doivent faire face aux défis suivants :

– trouver des heuristiques pour élaguer l'espace de recherche. Ces heuristiques peuvent être de nature syntaxique (e.g. ne s'intéresser qu'à un sous-ensemble des itemsets) et/ou basées sur des métriques statistiques (e.g. élaguer tous les itemsets dont le support est inférieur à minsup) ;

– trouver des techniques pour réduire les coûts d'entrée/sortie, puisque les bases à analyser sont originellement sur disque ;

– trouver des solutions techniques ou algorithmiques (plus précisément des structures de données adéquates) pour minimiser le coût de l'étape de calcul du support des itemsets.

L' algorithme APRIORI a introduit une approche appelée « *tester-et-générer* ». L'idée de base est de parcourir l'espace de recherche en largeur d'abord, ne retenir que les fréquents et d'en générer d'autres éléments dans le niveau suivant par autojointure. Les supports des itemsets candidats sont calculés et les candidats non fréquents sont élagués. Cet élagage, de nature statistique, est basé essentiellement sur la propriété de monotonicité du support, à savoir que le support est monotone et sa valeur décroît in-

versement par rapport à l'extension des itemsets. Cependant, cette approche souffre de la gestion du nombre de candidats qu'elle pourrait générer, surtout pour des contextes fortement corrélés et/ou des valeurs de support relativement faibles. Ce constat est renforcé par des accès disque répétitifs pour le calcul du support des candidats.

Des travaux récents ont proposé une série d'algorithmes pour améliorer les performances du processus d'extraction des itemsets fréquents. Ces algorithmes se sont focalisés essentiellement sur la réduction des entrées/sorties et la minimisation du coût de l'étape de calcul du support. Pour plus de détails sur ces algorithmes, prière de se référer essentiellement à [BAS 00b, BAS 02, BOU 03, BRI 97b, BYK 01, CAL 01, GEE 01].

L'inconvénient majeur de ces algorithmes concerne la génération d'un très grand nombre de règles associatives, et ce même pour des contextes d'extraction de taille raisonnable.

En revanche, les approches basées sur la découverte d'itemsets « fermés », issues de la théorie des concepts formels [GAN 99], proposent de ne générer qu'un sous-ensemble compact et générique de règles associatives. Ce sous-ensemble a une taille largement inférieure à la taille de l'ensemble de toutes les règles. Ce sous-ensemble générique de règles a l'avantage d'être complet du point de vue de la connaissance (pas de perte), tout en étant réduit du point de vue de la taille [BAS 00a, STU 01, BEN 03]. Par exemple, considérons le contexte d'extraction contenant seulement les deux tuples suivants : $\{i_1, i_2, \ldots, i_{50}\}$ et $\{i_1, i_2, \ldots, i_{100}\}$. Alors, un algorithme basé sur l'extraction des itemsets devra calculer les supports de $2^{100} - 1 \approx 10^{30}$ itemsets pour minsup = 1 et minconf = $0,5$[1]. Cependant, un algorithme basé sur les itemsets fermés ne va calculer que le support de deux itemsets fermés et générer qu'une seule règle générique.

Dans ce papier, nous nous proposons de présenter un survol de l'état de l'art des algorithmes basés sur la découverte des itemsets fermés. Ainsi, le reste du papier est organisé comme suit. La section 2 introduit les fondements mathématiques de la théorie des concepts formels et sa connexion avec la dérivation de règles associatives non redondantes. La section 3 passe en revue les algorithmes basés sur la découverte des itemsets fermés. La section 4 introduit une classification critique et une comparaison de ces algorithmes sur la base de critères que nous avons définis à cet effet. La section 5 conclut ce papier et présente un certain nombre de perspectives de recherche.

2. Fondements mathématiques

Dans ce qui suit, nous présentons quelques résultats clefs provenant de la théorie des concepts formels et leur application dans le contexte de dérivation des règles associatives.

1. C'est un coût intrinsèque indépendamment de la technique d'implémentation utilisée.

2.1. *Notions de base*

Dans la suite du papier, nous utilisons le cadre théorique présenté dans [GAN 99]. Dans cette partie, nous rappelons brièvement les notions de base de ce cadre théorique.

Contexte de fouille : un contexte de fouille est un triplet $\mathcal{D} = (\mathcal{O}, \mathcal{A}, \mathcal{R})$ décrivant un ensemble fini $\mathcal{O}$ d'objets, un ensemble fini $\mathcal{A}$ d'items et une relation (d'incidence) binaire $\mathcal{R}$ (*i.e.,* $\mathcal{R} \subseteq \mathcal{O} \times \mathcal{A}$). Chaque couple $(o, a) \in \mathcal{R}$, désigne le fait que l'objet $o \in \mathcal{O}$, possède l'item $a \in \mathcal{A}$. Dans un contexte (e.g. figure 1 en Haut), les objets sont désignés par des numéros et les attributs par des caractères.

On définit deux fonctions, f et g[2], résumant les liens qui pourraient exister entre des sous-ensembles d'objets et des sous-ensembles d'attributs induits par $\mathcal{R}$. Ainsi,

– $f : \mathcal{P}(O) \rightarrow \mathcal{P}(A)$, $f(X) = X' = \{a \in A | \forall o \in X, (o, a) \in \mathcal{R}\}$,
– $g : \mathcal{P}(A) \rightarrow \mathcal{P}(O)$, $g(Y) = Y' = \{o \in O | \forall a \in Y, (o, a) \in \mathcal{R}\}$.

Les opérateurs f et g, tels que définis, constituent une **correspondance de Galois**. Par exemple, dans la figure 1, $f(123) = AC$ et $g(ABC) = 35$[3]. De plus, les opérateurs composites $g \circ f(X)$ et $f \circ g(Y)$ (notés par $''$) sont des opérateurs de *fermeture* définis respectivement sur $\mathcal{P}(O)$ et $\mathcal{P}(A)$, puisque les propriétés suivantes sont vérifiées : $\forall A, A_1 \subseteq \mathcal{A}$ (resp. $\forall O, O_1 \subseteq \mathcal{O}$).

– Isotonie : $A \subseteq A_1 \Rightarrow A'' \subseteq A_1''$;
– Extensivité : $A \subseteq A''$;
– Idempotence :$(A'')'' = A''$.

Dans ce qui suit, nous introduisons la notion d'itemset fermé fréquent[4], puisque on peut être amené à ne s'intéresser qu'à des itemsets qui ont une fréquence assez importante.

Itemset fermé frequent : un itemset $A \subseteq \mathcal{A}$ est dit *fermé* si $A = A''$, et il est considéré *fréquent* par rapport au seuil *minsup* si $support(A) = \frac{|A'|}{|\mathcal{O}|} \geq$ *minsup*.

Concept formel : un concept formel est une paire $c = (O, A)$, où O et A sont reliés avec l'opérateur de la correspondance de Galois, *i.e.* $O' = A$ et $A' = O$. L'ensemble O est appelé *extension* et A est appelé *intension*.

Générateur minimal (ou itemset clé [STU 00]) : un itemset g est dit *générateur minimal* d'un itemset fermé A, ssi $g'' = A$ et $\nexists g_1 \subset g$ tel que $(g_1)'' = A$ [BAS 00a]. L'opérateur de fermeture ($''$) induit une relation d'équivalence sur l'ensemble de parties de $\mathcal{A}$, *i.e.* l'ensemble de parties est partionné en des sous-ensembles disjoints, appelés aussi *classes*. Dans chaque classe, tous les éléments possèdent la même valeur de support. Les générateurs minimaux d'une classe sont les éléments incomparables

---

2. Dans la suite, f et g sont désignés par $'$.
3. On utilise une notation omettant les séparateurs pour représenter les ensembles, e.g. 127 est équivalente à $\{1, 2, 7\}$, et *AB* à $\{A, B\}$.
4. Un itemset désigne un ensemble d'items.

les plus petits, tandis que l'itemset fermé est l'élément le plus large de cette classe. La figure 1 (Milieu) représente un échantillon des classes d'équivalence induites sur le contexte d'extraction $\mathcal{K}$.

Treillis de concepts formels (de Galois) : étant donné un contexte d'extraction $\mathcal{K}$, l'ensemble de concepts formels $\mathcal{C}_\mathcal{K}$ est un treillis complet $\mathcal{L}_c = (\mathcal{C}, \leq)$, appelé *treillis de Concepts formels (de Galois)*, quand l'ensemble $\mathcal{C}_\mathcal{K}$ est considéré avec la relation d'inclusion entre les itemsets [BAR 70, GAN 99]. La relation d'ordre partiel entre des concepts formels est définie comme suit : $\forall$ c_1 = (O_1,A_1), c_2 = (O_2,A_2)$\in \mathcal{C}_\mathcal{K}$, $c_1 \leq c_2$ ssi $A_2 \subseteq A_1$ ($\Leftrightarrow O_1 \subseteq O_2$).

Les opérateurs *sup* et *inf* permettent d'obtenir, respectivement, la plus petite borne supérieure et la plus grande borne inférieure. Les opérateurs *sup* et *inf* sont définis comme suit : $\forall\, O_1, O_2 \subseteq \mathcal{O}$ and $A_1, A_2 \subseteq \mathcal{A}$:

– $(O_1, A_1) \vee (O_2, A_2) = ((O_1 \cup O_2)'', A_1 \cap A_2)$,
– $(O_1, A_1) \wedge (O_2, A_2) = (O_1 \cap O_2, (A_1 \cup A_2)'')$.

La relation d'ordre partiel est utilisée pour générer le graphe du treillis, appelé *Diagramme de Hasse*, comme suit : il existe un arc (c_1, c_2), si $c_1 \preceq c_2$ où $\preceq$ est la réduction transitive de $\leq$, *i.e.* $\forall c_3 \in \mathcal{C}_\mathcal{K}$, $c_1 \leq c_3 \leq c_2 \Rightarrow c_1 = c_3$ ou $c_2 = c_3$. Dans ce graphe, chaque élément $c \in \mathcal{C}_\mathcal{K}$ est connecté aussi bien à un ensemble de ses successeurs immédiats, appelé *couverture supérieure* ($Couv^s$), et à un ensemble de ses prédécesseurs immédiats, appelé *couverture inférieure*($Couv_i$).

Treillis d'Iceberg de Galois : quand on considère seulement l'ensemble des itemsets fermés fréquents ordonnés par la relation d'inclusion ensembliste, la structure obtenue $(\hat{\mathcal{L}}, \subseteq)$ préserve seulement l'opérateur *sup*. Cette structure forme un semi-treillis supérieur et elle est désignée par « *Treillis d'Iceberg de Galois* » [BAS 00b, STU 00, VAL 02].

Exemple 1 *Considérons le contexte d'extraction $\mathcal{K}$ donné par la figure 1 (Haut). Le treillis d'Iceberg de Galois, pour minsup = 2, est donné par la figure 1 (Bas). Chaque nœud dans l'Iceberg est représenté sous la forme d'un couple (itemset fermé ; support) et est décoré par la liste de ses générateurs minimaux associés.*

Dans ce qui suit, on présente le cadre général pour la dérivation des règles associatives, puis on établit sa connexion avec la théorie des concepts formels.

2.2. *Dérivation des règles associatives*

La génération des règles associatives est réalisée à partir d'un ensemble F d'itemsets fréquents dans un contexte d'extraction $\mathcal{K}$, pour le seuil minimal de support *minsup*. Une règle associative R est une relation entre itemsets de la forme $R : X \Rightarrow (Y - X)$, dans laquelle X et Y sont des itemsets fréquents, tels que $X \subset Y$. Les itemsets X et $(Y - X)$ sont appelés, respectivement, *prémisse* et *conclusion* de

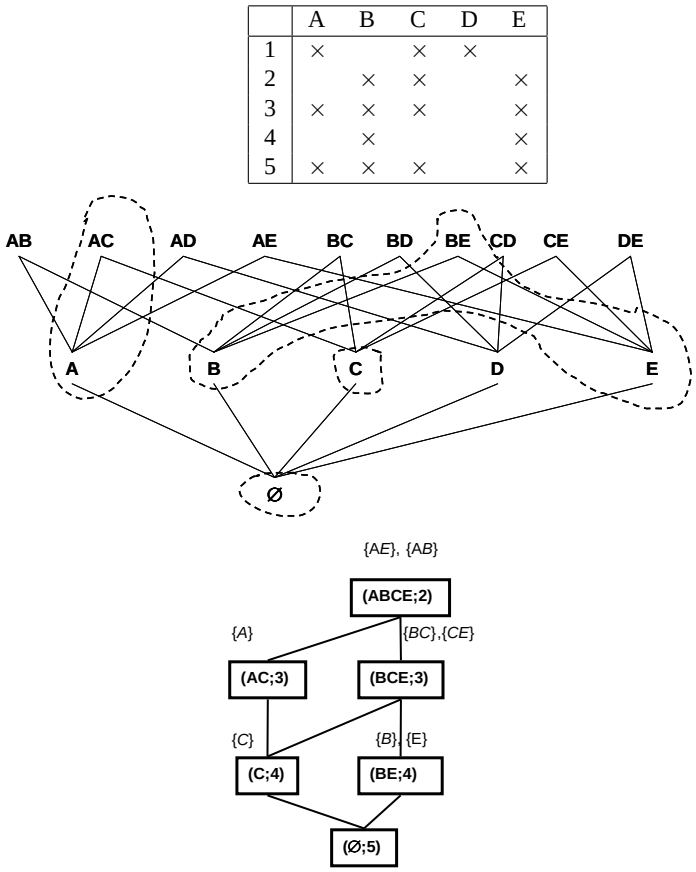

Figure 1. ***Haut :*** *contexte d'extraction $\mathcal{K}$.* ***Milieu :*** *échantillon des classes de la relation d'équivalence induite.* ***Bas :*** *le treillis d'Iceberg de Galois associé pour minsup = 2*

la règle r. Les règles associatives valides sont celles dont la mesure de confiance $Conf(r) = \frac{support(Y)}{support(X)}$ [5], est supérieure ou égale à un seuil minimal de confiance. Si $Conf(R) = 1$ alors R est appelée *règle associative exacte*, sinon elle est appelée *approximative*.

Cependant, le nombre d'itemsets fréquents extraits et leur taille moyenne sont élevés dans la plupart des jeux de données. Le nombre de règles associatives générées varie en général de plusieurs dizaines de milliers à plusieurs millions [STU 01, ZAK 00]. Le problème de la pertinence et de l'utilité des règles extraites demeure un problème

5. Le nombre de transactions de $\mathcal{K}$ contenant Y.

majeur pour l'extraction des règles associatives. Il est lié au nombre de règles extraites, qui est en général très important, et à la présence d'une forte proportion de règles redondantes, *i.e.* règles convoyant la même information, parmi celles-ci.

Néanmoins, ce problème a encouragé le développement d'outils de classification des règles selon leurs propriétés, de sélection de sous-ensembles de règles selon des critères définis par l'utilisateur, et de visualisation de ces règles sous une forme intelligible. Dans la littérature, deux approches ont été explorées pour la sélection de règles.

1) *Sélection avec perte d'information :* elle peut reposer sur des patrons définis par l'utilisateur (*user-defined templates*) [KEI 96, KLE 94, LIU 99], des opérateurs booléens ou *SQL-Like* [MEO 96, NG 98, SRI 97]. Le nombre de règles peut être aussi réduit en les élaguant avec une information additionnelle, telle qu'une taxonomie [LAT 01, HAN 95, HIP 98, SKI 95] ou une métrique d'intérêt additionnelle [BRI 97a, HUE 01] (e.g. corrélation de Pearson ou le χ^2-test). Pour un survol détaillé des mesures d'intérêt proposées dans la littérature de la fouille de donnée, prière de se référer à [TAN 02].

2) *Sélection sans perte d'information :* cette sélection est basée sur l'extraction d'un sous-ensemble générique de règles associatives, appelé *base*, à partir duquel toutes les autres règles associatives (redondantes) peuvent être générées. Ainsi, considérons la notion de règle redondante qui peut être définie comme suit [BEN 04] :

Définition 2 *Soit $\mathcal{AR}_\mathcal{K}$ l'ensemble de règles associatives qui peut être généré à partir d'un contexte d'extraction $\mathcal{K}$. Une règle R : X$\stackrel{c}{\Rightarrow}$Y* [6] *$\in \mathcal{AR}_\mathcal{K}$ est considérée comme redondante (ou dérivable) par rapport à (de) R_1 : $X_1 \stackrel{c}{\Rightarrow} Y_1$ si R vérifie les conditions suivantes :*

a) supp(R)=supp(R_1) et conf(R)=conf(R_1)=c ;

b) ($X_1 \subset$ X et Y $\subset Y_1$) ou (X$_1$ = X et Y $\subset Y_1$).

Bastide *et al.*, en adaptant la base d'implications globale de *Duquenne et Guigues* [GUI 86], ont présenté une nouvelle base appelée « *base générique de règles associatives exactes* ». Cette dernière est définie comme suit :

Définition 3 *Soit FC l'ensemble des itemsets fermés fréquents extrait d'un contexte d'extraction. Pour chaque itemset fermé fréquent c $\in$FC, nous désignons par $\mathcal{G}_c$ l'ensemble de ses générateurs minimaux. La base générique de règles associatives exactes est alors comme suit : GB =*$\{R : g \Rightarrow (c - g) \mid c \in FC$ *et g* $\in \mathcal{G}_c$ *et g* $\neq$ *c* $\}$[7].

Intuitivement, une règle générique exacte de la forme R : $X \Rightarrow (Y - X)$, est une implication entre deux itemsets X et Y tel que $X'' = Y''$. En effet, à partir de l'égalité des fermetures, on peut déduire X et Y appartiennent à la même classe d'équivalence. Alors, nous avons support(X)= support(Y), et par conséquent conf(R) = 1. Bastide *et al.*, en adaptant la base d'implications partielles de *Luxenburger* [LUX 91], ont aussi

6. Quand conf(R : X$\stackrel{c}{\Rightarrow}$Y) =1, alors c est omis, *i.e.* R est écrite comme R : X$\Rightarrow$Y.
7. La condition $g \neq c$ permet de ne pas retenir les règles de la forme $g \Rightarrow \emptyset$.

caractérisé une base générique de règles associatives approximatives, appelée *Base informative* et qui est définie comme suit :

Définition 4 *La base informative de règles associatives approximatives IB est donnée par : IB = $\{R|$ R : LHS $\Rightarrow$ RHS, RHS $\in$ FC et (LHS)$'' \leq$ RHS et confidence(R) $\geq$ minconf et support(RHS) $\geq$ minsup$\}$.*

Notons qu'il a été prouvé que les bases génériques de règles associatives sont sans perte d'information [BAS 00a]. Ainsi, étant donné un treillis d'Iceberg de Galois, dans lequel chaque itemset fermé est décoré par sa liste de générateurs minimaux, la dérivation de bases génériques de règles peut se faire d'une manière directe. En effet, les règles approximatives génériques sont des implications « inter-nœuds » assorties d'une mesure de confiance. Cette implication met en jeu deux classes d'équivalence comparables, *i.e.* d'un itemset fermé vers un autre itemset fermé le couvrant dans la structure partiellement ordonnée. Par exemple dans le contexte $\mathcal{K}$, la règle approximative générique C$\stackrel{.5}{\Rightarrow}$ABE est générée à partir des deux classes d'équivalence, dont leurs sommets respectifs sont les itemsets fermés « C » et « ABCE ». Par contre, les règles génériques exactes sont des implications « intra-nœud », avec une confiance égale à 1, extraites de chaque nœud dans la structure partiellement ordonnée. Par exemple dans le contexte $\mathcal{K}$, à partir de l'itemset fermé « ABCE », deux règles génériques exactes sont obtenues : AB$\Rightarrow$CE et AE$\Rightarrow$BC.

La figure 2 illustre, respectivement, les bases génériques exactes et approximatives qu'on peut extraire à partir du contexte $\mathcal{K}$.

Règle #	"$\Rightarrow$"
R_1	E$\Rightarrow$B
R_2	B$\Rightarrow$E
R_3	A$\Rightarrow$C
R_4	BC$\Rightarrow$E
R_5	CE$\Rightarrow$B
R_6	AB$\Rightarrow$CE
R_7	AE$\Rightarrow$BC

Règle #	"$\Rightarrow$"	Règle #	"$\Rightarrow$"
R_8	C$\stackrel{.75}{\Rightarrow}$A	R_{13}	B$\stackrel{.75}{\Rightarrow}$CE
R_9	C$\stackrel{.75}{\Rightarrow}$BE	R_{14}	E$\stackrel{.5}{\Rightarrow}$ABC
R_{10}	C$\stackrel{.5}{\Rightarrow}$ABE	R_{15}	B$\stackrel{.5}{\Rightarrow}$ACE
R_{11}	A$\stackrel{.66}{\Rightarrow}$BCE	R_{16}	BC$\stackrel{.66}{\Rightarrow}$AE
R_{12}	E$\stackrel{.75}{\Rightarrow}$BC	R_{17}	CE$\stackrel{.66}{\Rightarrow}$AB

Figure 2. ***Haut :*** *règles génériques exactes.* ***Bas :*** *règles génériques approximatives*

Une fois les règles génériques extraites, toutes les autres règles (redondantes) peuvent être dérivées facilement. Prière de se référer aux travaux de [BEN 04, LUO 01,

KRY 02] pour une discussion sur les mécanismes de dérivation et de raisonnement à partir des règles génériques. Par exemple, en utilisant les axiomes de décomposition et d'augmentation introduits dans [BEN 04], l'ensemble des 33 règles associatives qu'on peut dériver à partir des bases génériques est illustré par la figure 3.

AB⇒C	AB⇒E	AE⇒B	AE⇒C	ABC⇒E	ABE⇒C	ACE⇒B
AC$\overset{.66}{\Rightarrow}$BE	BE$\overset{.75}{\Rightarrow}$C	BE$\overset{.5}{\Rightarrow}$AC	BCE$\overset{.66}{\Rightarrow}$A	C$\overset{.75}{\Rightarrow}$B	C$\overset{.75}{\Rightarrow}$E	C$\overset{.5}{\Rightarrow}$AB
C$\overset{.5}{\Rightarrow}$AE	A$\overset{.66}{\Rightarrow}$B	A$\overset{.66}{\Rightarrow}$BC	A$\overset{.66}{\Rightarrow}$BE	A$\overset{.66}{\Rightarrow}$CE	E$\overset{.75}{\Rightarrow}$C	B$\overset{.75}{\Rightarrow}$C
E$\overset{.5}{\Rightarrow}$A	E$\overset{.5}{\Rightarrow}$AB	E$\overset{.5}{\Rightarrow}$AC	B$\overset{.5}{\Rightarrow}$A	B$\overset{.5}{\Rightarrow}$AC	B$\overset{.5}{\Rightarrow}$AE	BC$\overset{.66}{\Rightarrow}$A
CE$\overset{.66}{\Rightarrow}$	AC$\overset{.66}{\Rightarrow}$B	AC$\overset{.66}{\Rightarrow}$E	BE$\overset{.5}{\Rightarrow}$A	BE$\overset{.5}{\Rightarrow}$C		

Figure 3. *L'ensemble des règles associatives dérivé à partir des bases génériques données par la figure 2*

3. Les algorithmes

Dans cette section, nous allons présenter les lignes directrices des algorithmes les plus importants parus dans la littérature, en nous focalisant essentiellement sur l'étape de découverte des itemsets fermés. Un premier survol de ces algorithmes permet de les classer selon la technique adoptée pour l'exploration de l'espace de recherche, à savoir « Tester-et-générer » et « Diviser-pour-régner ».

1) **La technique « Tester-et-générer » :** les algorithmes parcourent l'espace de recherche par niveau. A chaque niveau k, un ensemble de candidats de taille k est généré. Cet ensemble de candidats est, généralement, élagué par la conjonction d'une métrique statistique (e.g. le support) et des heuristiques basées essentiellement sur les propriétés structurelles des itemsets fermés.

2) **La technique « Diviser-pour-régner » :** les algorithmes essaient de diviser le contexte d'extraction en des sous-contextes et d'appliquer le processus de découverte des itemsets fermés récursivement sur ces sous-contextes. Ce processus de découverte repose sur un élagage du contexte basé essentiellement sur l'imposition d'une métrique statistique et d'heuristiques introduites.

3.1. *Les algorithmes de type « Tester-et-générer »*

Dans cette sous-section, nous allons passer en revue les algorithmes les plus connus dans la littérature opérant selon la technique « Tester-et-générer ». Avant d'aborder la description des algorithmes, nous allons présenter une structure générale ou générique d'un algorithme entrant dans cette catégorie.

Notons que la structure générique, présentée par l'algorithme 1, indique globalement les différentes étapes que suit un algorithme pour l'extraction des itemsets fermés. Les différents algorithmes que nous allons présenter instancient cette structure

Algorithme 1 Algorithme générique

Entrée: $\mathcal{K}$:Contexte d'Extraction, minsup
Sortie: Ensemble des itemsets fermés fréquents
1: Initialiser l'ensemble de candidats de taille 1
2: **tant que** ensemble de candidats non vide **faire**
3: **Étape d'élagage (ou de test)**
1) Calculer le support des candidats
2) Élaguer l'ensemble de candidats par rapport à minsup
3) (Éventuellement) calculer les fermetures des candidats retenus
4: **Étape de construction**
1) Construire l'ensemble de candidats à utiliser lors de l'itération suivante
2) Élaguer cet ensemble en utilisant les propriétés structurelles des itemsets fermés et/ou des générateurs minimaux.
5: **fin tant que**
6: **retourner** Ensemble des itemsets fermés fréquents

en modifiant parfois l'ordre chronologique des actions effectuées (e.g. l'algorithme A-CLOSE effectue l'action de calcul des fermetures après la boucle principale Tant que).

La deuxième action de l'étape de construction est une action spécifique aux algorithmes d'extraction des itemsets fermés comparativement aux algorithmes d'extraction des itemsets. En effet, comme un algorithme d'extraction des itemsets fermés doit supporter le coût supplémentaire du calcul des fermetures des itemsets, alors il est impératif de trouver un moyen additionnel, outre la métrique statistique, pour élaguer cet ensemble de candidats. Ainsi, l'heuristique appliquée à ce niveau dépendrait de l'information collectée. Cette information peut être soit l'ensemble des générateurs minimaux seul, soit accompagné de l'ensemble des itemsets fermés.

3.1.1. *L'algorithme* CLOSE

Pasquier *et al.* [PAS 98, PAS 99a] ont proposé cet algorithme pour la découverte des règles associatives. Il est basé sur l'élagage de l'espace de recherche des itemsets fermés. Ainsi étant donné un contexte d'extraction $\mathcal{K}$, CLOSE génère toutes les règles associatives en trois étapes successives :

1) découvrir des itemsets fermés fréquents ;

2) dériver tous les itemsets fréquents à partir de ceux fermés obtenus durant la première étape ;

3) pour chaque itemset fréquent i, générer toutes les règles dérivables et ayant une confiance au moins égale à minconf.

FFC_k	Ensemble des k-itemsets fréquents candidats.
FC_k	Ensemble des k-itemset fermés fréquents.
	Chaque élément de ces ensembles possède trois champs :
	i) **gen** : le générateur ; ii) **supp** : le support
	iii) **ferm** : la fermeture.

Tableau 1. *Notations et paramètres utilisés dans* CLOSE

Le pseudo-code de CLOSE est donné par l'algorithme 2. Les différentes notations et paramètres utilisés dans cet algorithme sont résumés dans le tableau 1. Les items sont triés dans un ordre lexicographique. A chaque itération, l'algorithme construit un ensemble d'itemsets fermés candidats (FFC). Cet ensemble est élagué par rapport à minsup, obtenant ainsi un ensemble d'itemsets fermés fréquents. Finalement, en utilisant cet ensemble, il construit l'ensemble des générateurs qui seront utilisés dans la prochaine itération. L'algorithme s'arrête quand la liste des générateurs est vide. Ainsi, chaque itération est composée de deux étapes :

1) **Étape d'élagage :** durant cette étape, la fonction GEN-FERMETURE est appliquée à chaque générateur de FFC_k, déterminant ainsi son support et sa fermeture. Notons ici une particularité de l'algorithme CLOSE qui élague par rapport à minsup après avoir calculé les fermetures des générateurs dont certains peuvent être non fréquents.

2) **Étape de construction :** dans cette étape, on commence par éliminer les générateurs non fréquents. Ensuite, la fonction GEN-GENERATEUR prend comme l'ensemble des itemsets fermés fréquents FC_k et calcule l'ensemble FFC_{k+1} contenant tous les $(k+1)$-itemsets, qui seront utilisés dans l'itération suivante. A ce niveau, comme l'algorithme CLOSE dispose de l'ensemble des itemsets fermés fréquents obtenus au niveau k, alors l'ensemble FFC_{k+1} est élagué comme suit. Pour tout $c \in FFC_{k+1}$, si c est inclus dans la fermeture d'un des sous-ensembles, *i.e.* les éléments de FC_k dont la jointure a permis d'obtenir c. Dans ce cas, c est éliminé FFC_{k+1}.

L'algorithme s'arrête quand il n'y a plus de générateurs à traiter, *i.e.* FFC_k est vide.

Exemple 5 *L'ensemble des générateurs FFC_1 est initialisé par l'ensemble des 1-itemsets du contexte $\mathcal{K}$ (étape 2). L'appel de la fonction* GEN-FERMETURE *permet de calculer pour chaque générateur g sa fermeture et son support (étape 6). L'ensemble FC_1 est généré en élaguant l'ensemble FFC_1 par rapport à minsup (étapes 7-10). Ensuite, l'ensemble des 2-générateurs FFC_2 est obtenu en appliquant la fonction* GEN-GENERATEUR *à l'ensemble FC_1. Comme l'illustre la figure 4, l'appel de la fonction* GEN-GENERATEUR *avec FC_1 produit l'ensemble des 2-itemsets suivant :{AB, AE, BC, CE}. Notons que la jointure des 1-itemsets A et C de FC_1 ne produit pas un nouveau 2-générateur AC, puisque la fermeture du 1-itemset A est AC et celle du 1-itemset C est égale à C. Pour la même raison, le 2-générateur BE a été omis. L'appel de la fonction* GEN-FERMETURE *avec FFC_2 comme argument, permet de calculer les fer-*

Algorithme 2 L'algorithme CLOSE.

Entrée: $\mathcal{K}$ *:Contexte d'Extraction*, minsup
Sortie: *FC=*$\cup_k FC_k$ *: Ensemble des itemsets fermés fréquents*

{/* **Initialisation***/}
1: FFC_1={1-itemsets}
2: **pour** (k=1 ; FFC_k.gen $\neq \emptyset$;$k++$) **faire**
3: FFC_k.ferm=$\emptyset$
4: FFC_k.supp=$\emptyset$
5: FFC_k=GEN-FERMETURE(FFC_k)
{/* **Étape de construction***/}
6: **pour tout** c $\in$ FCC_k **faire**
7: **si** c.supp $\geq$ minsup **alors**
8: FC_k = $FC_k \cup$ c
9: **fin si**
10: FFC_{k+1}=GEN-GENERATEUR(FC_k)
11: **fin pour**
12: **fin pour**
13: **retourner** FC=$\cup_k FC_k$

metures respectives des 2-générateurs de FFC_2. Après l'étape d'élagage, les deux ensembles FFC_2 et FC_2 sont identiques. L'ensemble des 3-générateurs FFC_3, construit en appelant la fonction GEN-GENERATEUR *avec FC_2, est vide. L'algorithme s'arrête ainsi donnant en sortie l'ensemble des itemsets fermés fréquents, ainsi que leurs supports respectifs.*

Une fois les itemsets fermés fréquents découverts, les itemsets fréquents avec leurs supports respectifs peuvent être déterminés en mémoire centrale. En effet, le support d'un itemset fréquent est égal à celui de l'itemset fermé fréquent le plus petit le contenant [PAS 98].

Partageant la même technique que l'algorithme APRIORI, l'algorithme CLOSE effectue n passages sur le contexte d'extraction, même si ce nombre peut diminuer d'une façon importante surtout pour des contextes denses. Par exemple sur le contexte d'extraction $\mathcal{K}$, l'algorithme CLOSE nécessite seulement deux passages pour extraire les itemsets fermés fréquents, tandis que l'algorithme APRIORI nécessiterait quatre passages. Cependant dans chaque itération, là où l'algorithme APRIORI n'effectue que des calculs des supports, l'algorithme CLOSE effectue beaucoup plus de calculs, surtout pour la détermination des fermetures des itemsets candidats.

Ainsi dans le but de réduire le temps de calcul des itemsets, les auteurs ont introduit une nouvelle structure de donnée permettant d'accélérer le temps de localisation des générateurs associés à un objet. Dans cette structure et pour localiser un générateur, il faut parcourir un chemin en partant du nœud. Ainsi, la fermeture d'un générateur se trouverait dans un nœud feuille du chemin le représentant.

FFC_1

Gen	Supp	Ferm
A	**3**	AC
B	**4**	BE
C	**4**	C
D	**1**	ACD
E	**4**	BE

FC_1

Gen	Supp	Ferm
A	3	AC
B	4	BE
C	4	C
E	4	BE

FFC_2

Gen	Supp	Ferm
AB	2	ABCE
AE	2	ABCE
BC	3	BCE
CE	3	BCE

FC_2

Gen	Supp	Ferm
AB	2	ABCE
AE	2	ABCE
BC	3	BCE
CE	3	BCE

Output :

itemset fermé fréquent	Support
AC	3
BE	4
C	4
ABCE	2
BCE	3

Figure 4. *Trace d'exécution de* CLOSE *sur le contexte d'extraction $\mathcal{K}$ pour minsup =2*

La performance de l'algorithme CLOSE a été évaluée par rapport à l'algorithme APRIORI, en menant diverses expérimentations sur une machine biprocesseur IBM PowerPC 43P240. La plate-forme utilisée est *AIX* 4.1.5, avec une fréquence d'horloge de 166 MHz, 1 Giga octets de mémoire centrale et 9 Giga octets de mémoire secondaire. Deux types de contextes ont été utilisés durant ces expérimentations : bases synthétiques (typiquement des contextes épars) et la base *Census* (contexte dense). Les résultats de ces expérimentations ont montré que l'algorithme APRIORI donnait de meilleurs résultats sur les bases synthétiques. Tandis que sur le contexte dense *Census*, l'algorithme CLOSE est plus performant qu'APRIORI, surtout pour des valeurs de supports faibles.

A noter que pour l'algorithme CLOSE, aucune technique de gestion du buffer n'a été proposée. En effet, les auteurs supposent que l'ensemble FC_k des itemsets fermés fréquents et l'ensemble FC_{k+1} des itemsets candidats, utilisés dans la phase de construction d'un passage k, peuvent tenir en mémoire centrale. Cependant, on trouve

qu'une telle quantité d'information est difficilement tenable en mémoire centrale, surtout pour des contextes denses et des valeurs de supports très faibles.

3.1.2. *L'algorithme* A-CLOSE

L'algorithme A-CLOSE [PAS 99b] est une variation de l'algorithme CLOSE. Il a pour objectif de réduire l'espace de recherche en élaguant le treillis des itemets fermés au lieu de celui des itemsets. Ainsi, il procède en deux étapes :

1) déterminer les générateurs minimaux, *i.e.* les plus petits éléments incomparables des classes de la relation d'équivalence induite ;

2) pour chaque classe, déterminer l'élément maximal résidant au sommet de la hiérarchie.

Pour l'algorithme A-CLOSE, les deux étapes sont implémentées par la fonction AC-GENERATEUR. Ainsi, l'algorithme A-CLOSE instancie les deux étapes de l'algorithme générique comme suit :

1) **Étape d'élagage :** durant cette étape, la fonction AC-GENERATEUR est appliquée à chaque générateur de G_k, déterminant ainsi son support. Les k-générateurs infréquents sont éliminés.

2) **Étape de construction :** la fonction AC-GENERATEUR prend en entrée l'ensemble G_k des k-générateurs et calcule l'ensemble G_{k+1} contenant tous $(k+1)$-générateurs, qui seront utilisés dans l'itération suivante. A ce niveau, comme l'algorithme A-CLOSE dispose seulement de l'ensemble des générateurs minimaux obtenus au niveau k, l'ensemble G_{k+1} est élagué comme suit. Si un élément de G_{k+1} est aussi fréquent qu'un des ses k-sous-ensembles de G_k, alors il est éliminé de G_{k+1}.

La boucle principale s'arrête quand il n'y a plus de générateurs à traiter, *i.e.* G_k est vide. Une fois les générateurs minimaux déterminés, l'algorithme A-CLOSE effectue une autre passe sur le contexte pour le calcul des fermetures des générateurs minimaux retenus. Le pseudo-code de A-CLOSE est donné par l'algorithme 3, où l'ensemble *G* possède la même structure que l'ensemble *FC* présenté dans l'algorithme CLOSE.

Exemple 6 *La figure 5 illustre le processus d'exécution de l'algorithme* A-CLOSE *sur le contexte d'extraction $\mathcal{K}$, pour minsup = 2. Au début, l'algorithme détermine l'ensemble G_1 des 1-itemsets, ainsi que leurs supports (étapes 1-2). Les 1-générateurs non fréquents sont éliminés de G_1 (étape 5). Ensuite, l'ensemble des 2-générateurs de G_2 est obtenu, en appliquant la fonction* AC-GENERATEUR *à G_1 (étape 8). Les 2-générateurs AC et BE sont éliminés de G_2, puisque support(AC) = support(A) et support(BE) = support(B). L'appel de la fonction* AC-GENERATEUR *avec l'ensemble G_2 en entrée produit un seul 3-générateur ABE dans G_3. Ce 3-générateur ABE est éliminé de G_3, puisque support(BE) = support(ABE). Ainsi, la boucle principale prend fin, puisque G_3 est vide. Finalement, les fermetures associées des générateurs minimaux retenus sont calculées (étapes 11-14).*

Algorithme 3 L'algorithme A-CLOSE

Entrée: $\mathcal{K}$ *:contexte d'extraction*, minsup
Sortie: FC=$\cup_k$FC$_k$:Ensemble des itemsets fermés fréquents
1: G$_1$= {1-itemsets}
2: G$_1$= CALCUL-SUPPORT(G$_1$)
3: {/*CALCUL-SUPPORT calcule de support de chaque générateur g de $\in$ G$_1$*/}
4: G$_1$= SUPPRIMER-INFREQUENT(G$_1$,minsup)
5: {/*supprimer les 1-générateurs non fréquents par rapport à minsup */}
6: **pour** (k=1 ; G$_k$.gen $\neq \emptyset$;$k++$) **faire**
7: G$_{k+1}$=AC-GENERATEUR(G$_k$)
8: {/*Construction de (k+1)-générateurs*/}
9: **fin pour**
10: **pour tout** p $\in$ G **faire**
11: FC=CALCULER-FERMETURE(G)
12: {/*Calculer les fermetures des générateurs*/}
13: **fin pour**
14: **retourner** FC=$\cup_k$FC$_k$

Les performances de l'algorithme A-CLOSE ont été comparées à celles de CLOSE et APRIORI, respectivement. Les tests ont été conduits dans le même environnement que celui utilisé pour l'évaluation de l'algorithme CLOSE. Comme c'était prévisible, les résultats de ces expérimentations ont montré que l'algorithme APRIORI est légèrement meilleur que A-CLOSE sur les contextes éparses. En effet, dans le cas de contextes éparses, l'espace de recherche des itemsets fermés fréquents a tendance à se « calquer » sur celui des itemsets fréquents. Ainsi, les ensembles de candidats générés par les deux algorithmes sont presque identiques. Tandis que pour les contextes corrélés, l'algorithme A-CLOSE présente largement de meilleurs performances que l'algorithme APRIORI. Comparativement à l'algorithme CLOSE, les performances de l'algorithme A-CLOSE sont légèrement inférieures [PAS 00].

Comme cela a été mentionné pour l'algorithme CLOSE, un schéma de gestion de buffer est passé sous silence dans le cas de l'algorithme A-CLOSE.

3.1.3. *L'algorithme* CHARM

L'algorithme CHARM a été proposé par Zaki *et al.* [ZAK 02]. L'originalité de CHARM réside dans le fait qu'il privilégie une exploration en profondeur d'abord de l'espace de recherche. L'idée clef est d'exploiter la maximalité d'un itemset fermé, *i.e.* un itemset fermé couplé avec l'ensemble des objets le vérifiant n'est pas inclus dans aucun autre itemset fermé. Ainsi, l'algorithme CHARM explore simultanément l'espace de recherche des itemsets, ainsi que celui des identificateurs des transactions dans une structure appelée *IT-tree* (Itemset-Tidset tree). Une autre particularité à mentionner au crédit de CHARM, est qu'il utilise une représentation verticale, appelée

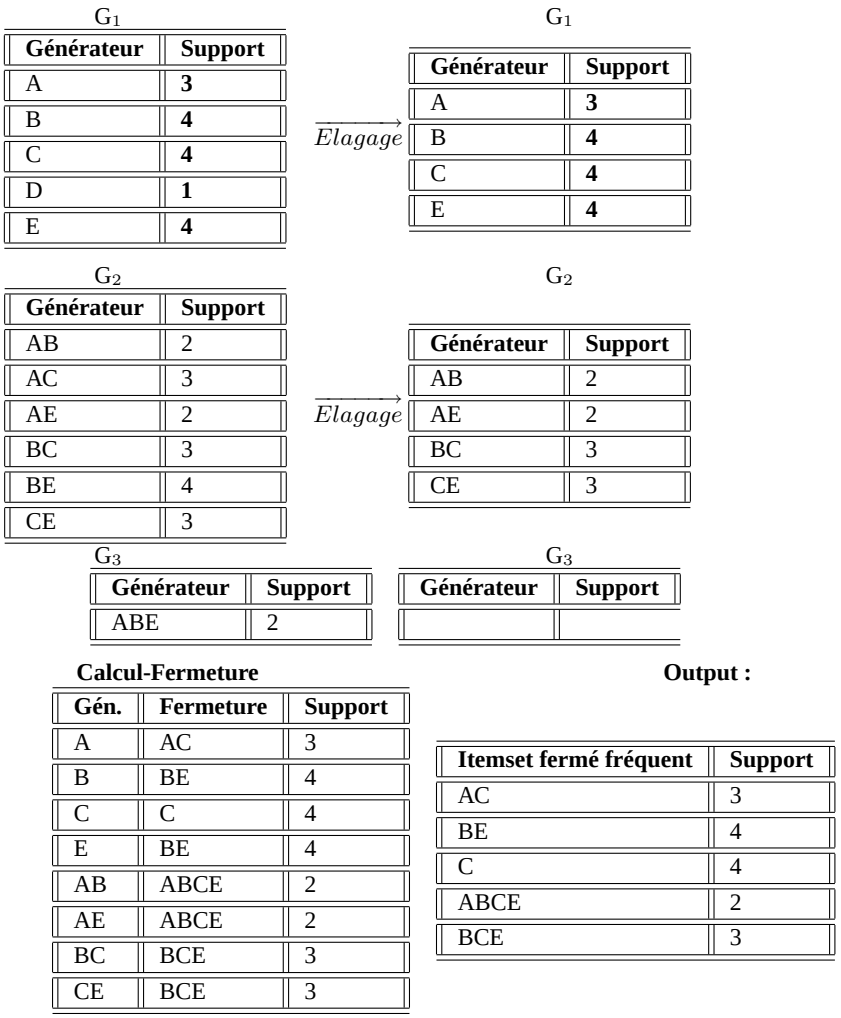

Figure 5. *Trace d'exécution de* A-CLOSE *sur contexte d'extraction $\mathcal{K}$ pour minsup=2*

diffset, pour accélérer le calcul des supports. Le pseudo-code de l'algorithme CHARM est donné par l'algorithme 4.

CHARM commence par initialiser la classe de prefixes [P] des nœuds à examiner par les 1-itemsets fréquents et leurs tidsets associés. Les deux étapes génériques sont instanciées comme suit :

– **Étape d'élagage :** cette étape est implémentée *via* la procédure CHARM-PROPRIÉTÉ. Cette procédure peut modifier la classe courante [P] en supprimant des paires-IT ou en insérant de nouvelles dans [P_i]. Une paire-IT est d'abord élaguée com-

parativement à *minsup*. Ensuite, on vérifie si elle est maximale ou non. Pour ce faire, il suffit de vérifier que son Tidset est inclus dans celui de la paire l'ayant généré. Une fois toutes les IT-paires traitées, la nouvelle classe [P_i] est récursivement explorée en profondeur d'abord, en appelant la procédure CHARM-ÉTEND.

– **Étape de construction :** cette étape est implémentée *via* la procédure CHARM-ÉTEND. Elle combine les IT-paires qui apparaissent dans la classe des préfixes [P]. Pour chaque IT-paire $X_i \times (X_i)'$, il la combine avec d'autres paires-IT $X_j \times (X_j)'$ la suivant dans l'ordre lexicographique. Chaque X_i va générer une nouvelle classe de préfixes [P_i], qui serait initialement vide. Les deux paires-IT combinées vont produire une nouvelle paire $X \times Y$, où $X = X_i \cup X_j$ et $Y = (X_i)' \cap (X_j)'$.

Algorithme 4 L'algorithme CHARM

Entrée: $\mathcal{K}$ *: Contexte d'extraction*, minsup
Sortie: $\mathcal{FC}$ *: Ensemble des itemsets fermés fréquents*
1: [P] ={$X_i \times (X_i)' : X_i \in \mathcal{I} \wedge$ support(X_i) $\geq$ minsup }
2: CHARM-ÉTEND([P],$\mathcal{FC}$=$\emptyset$)
3: **retourner** $\mathcal{FC}$

Algorithme 5 La procédure CHARM-ÉTEND

Entrée: [P],$\mathcal{FC}$
Sortie: $\mathcal{FC}$
1: **pour tout** $X_i \times (X_i)' \in$ [P] **faire**
2: [P_i]=$\emptyset$ and **X**=X_i
3: **pour tout** $X_j \times (X_j)' \in$ [P] $\wedge$ $X_j \geq X_i$ **faire**
4: **X**= **X** $\cup$ X_i
5: **Y**= $(X_i)' \cap (X_j)'$
6: CHARM-PROPRIÉTÉ([P],[P_i])
7: **si** [P_i] $\neq \emptyset$ **alors**
8: CHARM-ÉTEND([P_i],$\mathcal{FC}$)
9: SUPPRIMER ([P_i])
10: $\mathcal{FC}$= $\mathcal{FC}$ $\cup$ **X**
11: **fin si**
12: **fin pour**
13: **fin pour**
14: **retourner** $\mathcal{FC}$

Exemple 7 *La figure 6 montre la trace d'exécution du processus de découverte des itemsets fermés pour minsup = 2.* CHARM *commence par initialiser la classe racine par* [ ] = {*A* $\times$ *135, B* $\times$ *2345, C* $\times$ *1235, E* $\times$ *2345*}*. Le 1-itemset D est élagué, puisqu'il est non fréquent (étape 2). Au début, le nœud A* $\times$ *135 (X = A) est traité, et sera combiné avec les autres éléments. Les 1-itemsets A et B sont combinés, et puisque* $(A)'$ *diffère de* $(B)'$ *alors AB est inséré dans [A] (étape 14 de la procédure* CHARM-PROPRIÉTÉ*). Ensuite, A et C sont combinés. Comme on a* $(A)' \subset (C)'$*, alors*

Algorithme 6 La procédure CHARM-PROPRIÉTÉ.

Entrée: [P],[P_i]
Sortie: [P]

si support(**X**) $\geq$ minsup **alors**
  si $(X_i)' = (X_j)'$ **alors**
    Supprimer X_j de [P]
    Remplacer tout X_i par **X**
  sinon
    si $(X_i)' \subset (X_j)'$ **alors**
      Remplacer tout X_i par **X**
    sinon
      si $(X_i)' \supset (X_j)'$ **alors**
        Remplacer X_j de [P]
        ajouter **X** $\times$ **Y** à [P_i]
      sinon
        si $(X_i)' \neq (X_j)'$ **alors**
          ajouter **X** $\times$ **Y** à [P_i]
        fin si
      fin si
    fin si
  fin si
fin si
retourner [P]

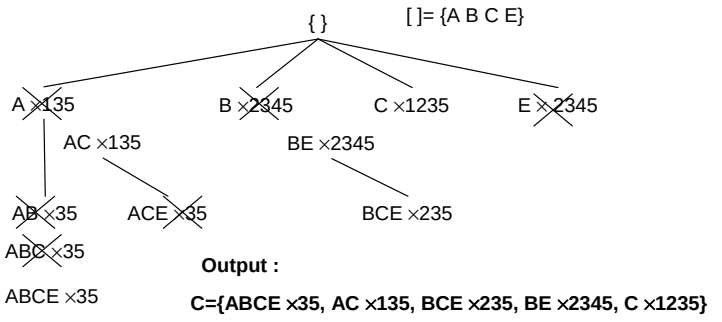

Figure 6. *Trace d'exécution de* CHARM *sur le contexte d'extraction $\mathcal{K}$ pour minsup=2*

toutes les occurrences de A sont remplacées par AC (étape 8 de la procédure CHARM-PROPRIÉTÉ*). En combinant AC et E, on trouve que $(AC)'$ est différent de $(E)'$. Dans ce cas, ACE est ajouté à [AC] (étape 14 de la procédure* CHARM-PROPRIÉTÉ*). Ensuite, un appel récursif à la procédure* CHARM-ÉTEND *ayant la classe [AC] comme entrée. Le même processus continue jusqu'à l'insertion de [ABCE] dans l'ensemble*

des itemsets fermés fréquents. Ensuite, c'est la branche de B qui est explorée. Ainsi, B et C sont combinés. Comme $(B)'$ est différent de $(C)'$, alors BC est inséré dans la nouvelle classe de [B] (étape 14 de la procédure CHARM-PROPRIÉTÉ*). B et E sont combinés et on se trouve dans le cas où $(B)'$ est égal à $(E)'$. Ainsi, E est enlevé de [ ] et toutes les occurrences de B sont remplacées par BE (étapes 4-5 de la procédure* CHARM-PROPRIÉTÉ*). Un appel récursif de* CHARM-ÉTEND*, ayant la classe [BC] comme entrée, est effectué. Comme il y a un seul élément, BCE est inséré dans la liste des itemsets fermés fréquents (étape 11 de la procédure* CHARM-ÉTEND*). Pour la branche C et comme elle ne peut pas être étendue, alors C est inséré dans la liste des itemsets fermés fréquents. La liste des itemsets fermés fréquents consiste en l'ensemble des IT-paires non barrées dans la figure 6.*

Les performances de l'algorithme CHARM ont été évaluées comparativement aux algorithmes CLOSE, PASCAL [BAS 00b] et CLOSET. Ces expérimentations ont été faites sur un PC Pentium avec une fréquence d'horloge de 400 MHz et disposant de 256 Méga octets de mémoire centrale. La plate-forme utilisée est RedHat Linux 6.0. Les résultats des expérimentations ont montré que l'algorithme CHARM présente de meilleures performances que les algorithmes CLOSE et PASCAL sur les contextes épars et denses.

L'inconvénient majeur de l'algorithme CHARM est qu'il nécessite un espace considérable de stockage. En effet, le fait de stocker les itemsets et leurs tidsets ne fait qu'accroître la quantité de mémoire utilisée. Notons aussi qu'aucune technique de gestion du buffer n'a été présentée pour cet algorithme. Même si pour pallier cet inconvénient, les auteurs ont utilisé une représentation astucieuse pour un stockage incrémental des Tids, on trouve qu'il est difficile de stocker une telle quantité d'information en mémoire centrale, et ceci surtout pour de larges contextes épars.

3.1.4. *L'algorithme* TITANIC

L'algorithme TITANIC a été proposé par Stumme *et al.* [STU 00, STU 02] pour la découverte des itemsets fermés fréquents. L'idée clé est de minimiser l'étape de calcul du support des itemsets. Ainsi, l'algorithme traverse l'espace de recherche par niveau en focalisant sur la détermination des générateurs minimaux (ou itemsets clés) des différentes classes de la relation d'équivalence induite par l'opérateur de fermeture.

Ainsi, l'algorithme TITANIC instancie les deux étapes de l'algorithme générique comme suit :

1) **Étape d'élagage :** durant cette étape, la fonction CALCUL-SUPPORT est appliquée à chaque générateur de FFC_k, déterminant ainsi son support. Sachant que ce calcul nécessite un accès au contexte d'extraction. Notons qu'à chaque itemset est assigné un support estimatif, qui est égal à la valeur minimale des supports des deux itemsets joints pour l'obtenir [8].

8. Pour le cas particulier des 1-itemsets, leur support estimatif est égal à $\mid \mathcal{K} \mid$.

FFC_k	Ensemble des k-générateurs candidats.
FC_k	Ensemble des k-générateurs fréquents minimaux.
	Chaque élément de ces ensembles est composé de quatre champs :
	i)**item** : le générateur ;
	ii)**S-E** : le support estimatif ;
	iii)**S-R** : le support réel ;
	iv)**clé** : (Oui/Non).

Tableau 2. *Notations utilisées dans l'algorithme* TITANIC.

Les k-générateurs non fréquents sont éliminés en appelant la fonction ÉLAGUER-INFREQUENT-NONCLÉ. Les k-générateurs non clés, dont la valeur de leur support estimatif est égale à la valeur de support réel, sont aussi éliminés. A ce niveau, l'algorithme TITANIC se propose de calculer les fermetures des générateurs minimaux qui ont été retenus durant l'itération précédente, *i.e.* ceux appartenant à FC_{k-1}.

2) **Étape de construction :** la fonction GENERER-CANDIDAT prend en entrée l'ensemble k-générateurs des FC_k et calcule l'ensemble FFC_{k+1} contenant tous les $(k+1)$-générateurs, qui seront utilisés dans l'itération suivante. A ce niveau, comme l'algorithme TITANIC dispose seulement de l'ensemble des générateurs obtenus au niveau k, alors l'ensemble FFC_{k+1} est élagué comme suit. Partant de la constatation que l'ensemble des générateurs minimaux forme un « ordre idéal », alors tout élément de FFC_{k+1} doit voir ses k-sous-ensembles appartenir à FC_k, sinon il est éliminé.

La boucle principale s'arrête quand il n'y a plus de générateurs à traiter, *i.e.* FFC_k est vide.

Les notations et paramètres utilisés par l'algorithme TITANIC sont donnés par le tableau 2, tandis que son pseudo-code est donné par l'algorithme 7.

Exemple 8 *La figure 7 illustre la trace d'exécution de l'algorithme* TITANIC *sur le contexte d'extraction $\mathcal{K}$ pour minsup = 2. Durant la phase d'initialisation, les 1-générateurs de l'ensemble FC_1 sont considérés comme des itemsets clés (étapes 2-4). Cet ensemble est élagué en appelant la fonction* ÉLAGUER-INFREQUENT-NONCLÉ *pour obtenir l'ensemble FC_1 (étape 9). L'ensemble des 2-générateurs candidats FCC_2 est obtenu en appelant la fonction* GENERER-CANDIDAT*, qui effectue une autojointure sur l'ensemble FC_1. La fonction* CALCUL-SUPPORT *détermine le support de chaque candidat de FCC_2 (étape 6). Les 2-générateurs non fréquents et non clés sont éliminés. Par exemple, les 2-générateurs fréquents AC et BE sont éliminés, puisque leur support estimatif est égal à la valeur de leur support réel. Durant cette itération, les fermetures des éléments de l'ensemble FC_1 sont calculés. La fonction* GENERER-CANDIDAT *est appliquée à l'ensemble FC_2 pour obtenir l'ensemble FFC_3 composé de 3-générateurs candidats. L'unique élément de cet ensemble est élagué en appelant la fonction* ÉLAGUER-INFREQUENT-NONCLÉ*. Avant de terminer, puisque FC_3 est vide, l'algorithme calcule les fermetures des 2-générateurs retenus dans FC_2.*

Algorithme 7 L'algorithme TITANIC

Entrée: $\mathcal{K}$ *: Contexte d'extraction*, minsup
Sortie: *FC=*$\cup_k FC_k$ *: Ensemble des itemsets fermés fréquents*

1: $FFC_0=\{\emptyset\}$
2: $\emptyset$.support=$|\mathcal{K}|$
3: $FFC_1=\{$1-itemsets$\}$
4: **pour** (k=1 ; FFC_k.gen $\neq \emptyset$;$k++$) **faire**
5: FFC_k= CALCUL-SUPPORT(FFC_k)
6: FC_{k-1}= CALCUL-FERMETURE(FFC_{k-1})
{/* $\forall$ i $\in FC_k$ (i)$''$= i $\cup\{x \in \mathcal{A}$ -{i} | support(i)= support(i $\cup$ {x}) } */}
7: FC_k=ÉLAGUER-INFREQUENT-NONCLÉ(FFC_k)
{/* $\forall$ i $\in FC_k$ si support (i) $<$ minsup ou $\exists$ j $\in$ i | support(i)= support (i -{j}) */}
8: FFC_{k+1}=GENERER-CANDIDAT(FFC_k)
9: **fin pour**
10: **retourner** FC=$\cup_k FC_k$

Les auteurs avancent que, particulièrement pour des contextes denses, le nombre de générateurs minimaux, qui est au moins égal au nombre d'itemsets fermés, est relativement bas comparativement au nombre d'itemsets fréquents. Aussi, la taille du générateur minimal le plus large, en termes d'items le composant, est généralement beaucoup plus petite que l'itemset fréquent le plus large. Par conséquent, le nombre d'accès au contexte d'extraction tend à la baisse.

Les performances de l'algorithme TITANIC ont été évaluées comparativement à l'algorithme NEXT-CLOSURE [GAN 91]. Les expérimentations ont été menées sur un PC Pentium III avec une fréquence d'horloge de 600 MHz et disposant de 512 Méga octets de mémoire centrale. Les expérimentations ont porté sur deux types de contextes : un contexte épars (INTERNET) et un contexte dense (MUSHROOM), qui sont disponibles à l'archive UCI KDD [9]. Les résultats des expérimentations ont montré que l'algorithme NEXT-CLOSURE présente légèrement de meilleurs performances que l'algorithme TITANIC sur les contextes denses et pour un nombre réduit d'items. Cependant, l'algorithme TITANIC commence à prendre le dessus au fur et à mesure que le nombre d'items augmente. Pour les contextes épars, l'algorithme TITANIC donne de meilleurs résultats. Les auteurs argumentent ce résultat par le fait que l'algorithme TITANIC est « inspiré » de l'algorithme APRIORI, dont il est connu qu'il est performant sur des contextes faiblement corrélés.

9. http ://kdd.ics.uci.edu

FFC_0

Itemset	S-E.	clé
$\emptyset$	**5**	Oui

FFC_1

Item	S-E	S-R	clé
A	**5**	**3**	Oui
B	**5**	**4**	Oui
C	**5**	**4**	Oui
D	**5**	**1**	Oui
E	**5**	**4**	Oui

FC_1

Item	S-E	S-R	clé
A	**5**	**3**	Oui
B	**5**	**4**	Oui
C	**5**	**4**	Oui
E	**5**	**4**	Oui

FC_0 Calcul fermeture

clé	Ferm.
$\emptyset$	$\emptyset$

FFC_2

Item	S-E	S-R	clé
AB	**3**	**2**	Oui
AC	**3**	**3**	Non
AE	**3**	**2**	Oui
BC	**4**	**3**	Oui
BE	**4**	**4**	Non
CE	**4**	**3**	Oui

FC_2

Item	S-E	S-R	clé
AB	**3**	**2**	Oui
AE	**3**	**2**	Oui
BC	**4**	**3**	Oui
CE	**4**	**3**	Oui

FC_1 Calcul fermeture

clé	Ferm.
A	AC
B	BE
C	C
E	BE

FFC_3

Item	S-E	S-R	clé
ABE	**3**	**3**	Non

$FC_3=\emptyset$

FC_2 Calcul fermeture

clé	Ferm.
AB	ABCE
AE	ABCE
BC	BCE
CE	BCE

Output :

clé	Itemset fermé fréquent
A	AC
B	BE
C	C
E	BE
AB	ABCE
AE	ABCE
BC	BCE
CE	BCE

Figure 7. *Trace d'exécution de* TITANIC *sur le contexte d'extraction $\mathcal{K}$ pour min-sup=2*

3.2. *Les algorithmes de type « Diviser-pour-régner »*

Dans ce qui suit, nous allons aborder le(s) algorithme(s) respectant la technique « Diviser-pour-régner ». En effet, l'idée motivant cette technique est d'éviter le goulot

d'étranglement de l'approche « tester-et-générer », à savoir la génération d'un nombre prohibitif de candidats. Dans la littérature, on retrouve essentiellement un seul algorithme implémentant cette technique, à savoir l'algorithme CLOSET [PEI 02]. Des améliorations ou variantes de cet algorithme ont été proposées, tout en se gardant de respecter l'idée motrice de l'algorithme originel [WAN 03, GRA 03].

L'algorithme CLOSET propose une approche originale et efficace, appelée « Diviser-pour-régner », pour la découverte des itemsets fermés fréquents. L'algorithme CLOSET proposait d'adopter une structure de données avancée, où la base de données peut être compressée pour arriver à achever le processus de fouille de données (e.g. la structure FP-tree [HAN 00]). L'idée motivant cette structure de donnée compacte, basée sur la notion de trie[10] [KRU 99], est que lorsque plusieurs transactions se partagent un item, alors elles peuvent être fusionnées en prenant soin d'enregistrer le nombre d'occurrences des items. L'algorithme CLOSET effectue le processus d'extraction des itemsets fermés en deux étapes successives :

1) Les items des transactions sont ordonnés par support décroissant. Ensuite, l'arbre FP-Tree est construit. Cette structure de donnée est construite comme suit. Premièrement, le nœud racine est créé et étiqueté par « *root* ». Pour chaque transaction du contexte, les items sont traités et une branche est créée pour chaque transaction. Dans chaque nœud du FP-tree, il y a un compteur qui garde trace du nombre d'apparitions de l'item dans le contexte d'extraction. Spécifiquement dans le cas où une transaction présente un préfixe commun avec une branche du FP-tree, alors le compteur de chaque nœud appartenant à ce préfixe est incrémenté de 1 et une sous-branche va être créée contenant le reste des items de la transaction. Par exemple, la figure 8 illustre le contexte d'extraction $\mathcal{K}$ réordonné et la structure FP-tree associée pour minsup= 2 ;

2) Au lieu d'une exploration en largeur des itemsets fermés candidats, il effectue une partition de l'espace de recherche pour effectuer ensuite une exploration en profondeur d'abord. Ainsi, il commence par considérer les 1-itemsets fréquents, et examine seulement leur sous-contexte conditionnel. Un sous-contexte conditionnel ne contient que les items qui co-occurrent avec le 1-itemset en question. Le FP-Tree conditionnel associé est construit et le processus se poursuit récursivement.

Il est à noter que l'algorithme CLOSET, dont le pseudo-code est donné par l'algorithme 8, est essentiellement basé sur les propriétés suivantes :

1) l'itemset fermé, disons p, qu'on extrait d'un sous-contexte conditionnel est constitué par la concaténation des 1-itemsets qui sont aussi fréquents que p.

2) il n'y a nul besoin de développer un sous-contexte conditionnel d'un itemset, disons p, qui est inclus dans un itemset fermé déjà découvert, disons c, tel que support(p)=support(c).

Exemple 9 *La figure 9 présente le processus de découverte des itemsets fermés par* CLOSET *pour minsup = 2. L'algorithme* CLOSET *commence par extraire le sous-*

10. de reTRIEval.

	B	C	E	A
1		×		×
2	×	×	×	
3	×	×	×	×
4	×		×	
5	×	×	×	×

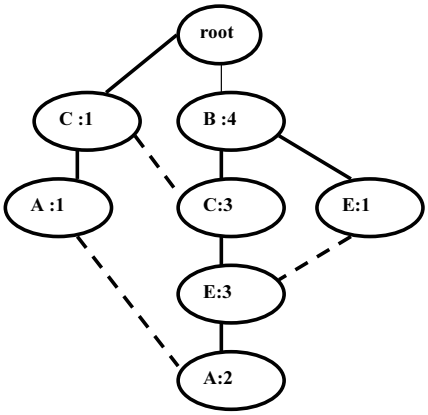

Figure 8. ***Haut :*** *contexte d'extraction trié.* ***Bas :*** *le FP-tree associé pour minsup =2*

Algorithme 8 L'algorithme CLOSET

Entrée: $\mathcal{K}$ *:contexte d'extraction* ,minsup
Sortie: *FC*=$\cup_k FC_k$ *:Ensemble d'itemsets fermés*

1: FC=$\emptyset$
2: {/*Trier le contexte d'extraction $\mathcal{K}$ avec une valeur du support décroissante */}
3: $\hat{\mathcal{K}}$= TRI($\mathcal{K}$)
4: $Freq_1$= { 1-itemset frequents}
5: **pour tout** i $\in$ $Freq_1$ (dans un ordre croissant) **faire**
6: FC_i=MINE-FERME(i,$\hat{\mathcal{K}}$)
{/*MINE-FERME construit le sous-contexte conditionnel de **i** et effectue un parcours en profondeur d'abord par des appels récursifs pour l'extraction des itemsets fermés fréquents*/}
7: FC=FC $\cup FC_i$
8: **fin pour**
9: **retourner** FC

contexte conditionnel de l'item ***A*** *(i.e. celui ne contenant que les transactions dans lesquelles l'item A est présent) (étape 7). A partir de ce sous-contexte, l'itemset fermé fréquent* ***CA*** *est découvert ayant comme support 3. Le même processus va s'appliquer récursivement en profondeur pour créer deux autres sous-contextes condition-*

*nels pour les 2-itemsets **BA** et **EA**, respectivement. Ceci permet de découvrir l'itemset fermé fréquent **BCEA** avec comme support égal à 2. Une fois le traitement de l'itemset **A** terminé, l'algorithme applique le même processus à l'1-itemset fréquent suivant **E**. Lors de la construction du sous-contexte conditionnel de **E**, on omet le 1-itemset déjà traité **A**. Cette omission est légitime, puisque tous les itemsets fermés fréquents contenant **A** ont été déjà extraits. A partir du sous-contexte conditionnel de **E**, l'itemset fermé fréquent **BE** est découvert ayant comme support 4. Le même processus va s'appliquer récursivement en profondeur pour créer un autre sous-contexte conditionnel pour le 2-itemset **CE**. Ceci permet de découvrir l'itemset fermé fréquent **BCE** avec comme support égal à 3. Le processus de découverte continue ainsi jusqu'à épuisement de la liste des 1-itemsets fréquents. Notons que c'est grâce à la seconde propriété que l'algorithme* CLOSET *ne génère pas l'itemset fermé inexistant (BC ;3) (cf. le sous-contexte conditionnel de C dans la figure 9).*

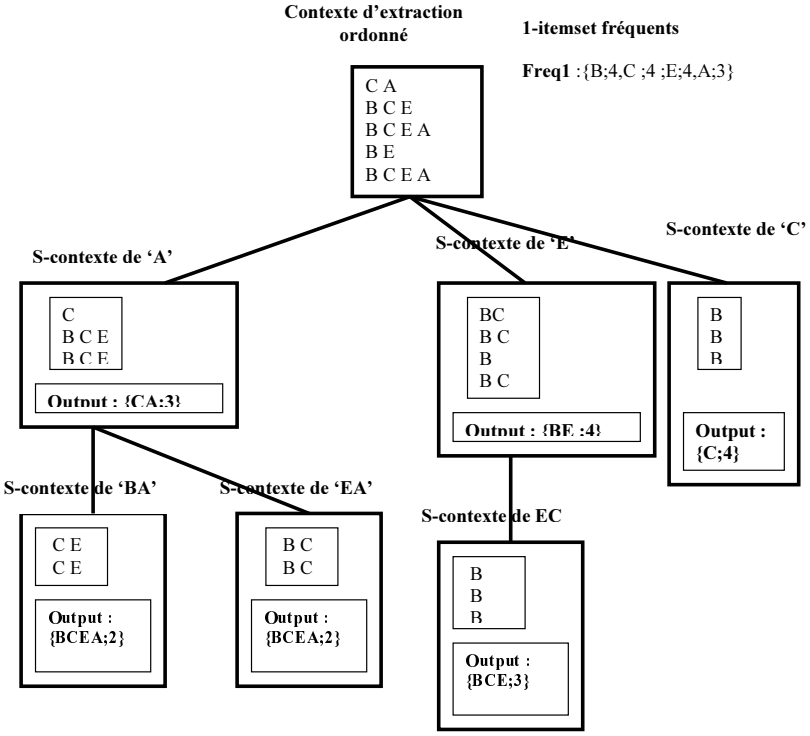

Figure 9. *Trace d'exécution de* CLOSET *sur le contexte d'extraction $\mathcal{K}$ pour minsup = 2*

Cependant, en plus du coût non négligeable de l'étape du tri du contexte d'extraction, la structure FP-Tree proposée est malheureusement non adaptée pour un processus de fouille interactif, dans lequel l'utilisateur peut être amené à varier la valeur

du support. Dans ce cas, la structure doit être reconstruite entièrement (cf. le travail présenté dans [CHE 03] pour pallier cette insuffisance en proposant une structure alternative appelée CATS).

Les performances de l'algorithme CLOSET ont été évaluées comparativement aux algorithmes A-CLOSE et CHARM. Les expérimentations ont été menées sur un Pentium PC, fonctionnant sur la plate-forme windows NT, avec une fréquence d'horloge de 233 MHz, 128 Méga octets de mémoire centrale. Les contextes de test utilisés lors des expérimentations sont de deux types : contexte épars (les bases synthétiques) et contextes denses (CONNECT-4 et PUMBS)[11]. Les résultats de ces expérimentations ont montré que l'algorithme CLOSET est plus performant que les algorithmes A-CLOSE et CHARM sur les contextes épars. Aussi pour les contextes denses, l'algorithme CLOSET est beaucoup plus performant et efficace que A-CLOSE et CHARM, qui souffre du coût non négligeable des opérations d'intersection effectuées sur de grandes listes de TID.

Les auteurs de l'algorithme CLOSET ne présentent pas une technique de gestion du buffer. Cependant, il est quasiment irréaliste de supposer que toute l'information nécessaire pour le déroulement du processus de découverte des itemsets fermés pourrait tenir en mémoire centrale. En effet, pour les bases faiblement corrélées, l'arbre représentant la trace d'exécution a tendance à augmenter en profondeur, tandis que pour les bases denses, cet arbre a tendance à augmenter en largeur avec peu de profondeur. Ce constat ne fait que contredire les affirmations des auteurs de l'algorithme CLOSET. En effet, ces derniers affirment que leur méthode ne fait que des tests seulement sans génération de candidats (« *restricted testing only* »). Cependant, l'augmentation de l'arbre en profondeur indique implicitement que l'algorithme CLOSET génère un nombre important de candidats, pour lesquels il devrait construire des sous-contextes.

4. Discussion

Il est bien connu que l'étape la plus défiante et la plus consommatrice en temps d'exécution est celle de la découverte des itemsets (fermés) fréquents. Comme effet de bord, cette étape peut générer un nombre impressionnant d'itemsets fréquents et par conséquent de règles associatives, même pour un contexte de taille raisonnable. Les algorithmes basés sur l'extraction des itemsets fermés fréquents sont une nouvelle alternative avec la promesse claire de réduire considérablement la taille des listes des règles associatives. Ainsi, le problème de découverte des règles associatives, considéré sous l'optique de découverte des itemsets fermés, pourrait être reformulé comme suit :

1) découvrir deux « systèmes de fermeture » distincts, *i.e.* ensembles d'ensembles fermés sous l'opérateur d'intersection, à savoir les deux « ordres idéaux » : l'ensemble des itemsets fermés et l'ensemble des générateurs minimaux. Aussi, la relation d'ordre sous-jacente devrait être déterminée ;

11. disponibles à l'adresse suivante : http ://www.ics.uci.edu/∼mlearn/MLRepository.html.

	CLOSE	A-CLOSE	CHARM	TITANIC	CLOSET
Architecture	Séquentiel	Séquentiel	Séquentiel	Séquentiel	Séquentiel
Stratégie de parallélisme	aucune	aucune	aucune	aucune	aucune
Technique	Teste-et...	Teste-et...	Teste-et...	Teste-et...	Diviser-pour...
Stratégie de transactions	exhaustive	exhaustive	exhaustive	exhaustive	exhaustive
Stockage des itemsets candidats	arbre H.	arbre H.	diffset	trie	arbre H.
Type de source de données	horizontal	horizontal	vertical	horizontal	horizontal

Tableau 3. *Classification des algorithmes*

2) à partir de toute l'information collectée durant la première étape, dériver des bases génériques de règles associatives.

A la lumière de cette reformulation et de ces objectifs, particulièrement celui de réduire la liste des règles associatives, on présente dans ce qui suit une liste de dimensions ou caractéristiques permettant de mettre en exergue les différences majeures qui pourraient exister entre les algorithmes passés en revue.

1) **Architecture :** cette dimension s'intéresse à la manière de conception de l'algorithme : une fonction séquentielle destinée à tourner sur machine monoprocesseur, ou plutôt conçue pour s'exécuter sur une architecture parallèle ou distribuée, e.g. CLOSET ;

2) **Stratégie de Parallélisme :** des algorithmes parallèles (non existants encore) peuvent être considérés par la stratégie de parallélisme adoptée : tâche, donnée ou même hybride ;

3) **Technique :** jusqu'à maintenant tous les algorithmes commencent leur exploration par un noyau composé des 1-itemsets. Pour trouver l'information manquante, *i.e.* le reste des générateurs minimaux et leurs fermetures, deux techniques principales ont été avancées : « tester-et-générer » et « diviser-pour-régner » ;

4) **Stratégie de transactions :** la valeur « exhaustive » de cette dimension indique que toutes les transactions seront balayées au moins une fois. En revanche, la valeur « échantillon » indiquerait que seul un échantillon de la base sera considéré du moins pour un premier balayage ;

5) **Stockage des itemsets candidats :** au fur et à mesure que les itemsets candidats sont générés, différentes structures de données peuvent être utilisées pour garder trace de ces générateurs. Les structures les plus privilégiées semblent être les arbres de hachage et les structures trie ;

6) **Type de source de données :** elle peut prendre comme valeur : format horizontal (étendu), format de données vertical, relationnel, données texte brut, données multimédias ;

Le tableau 3 résume la catégorisation des algorithmes par rapport aux dimensions données précédemment. Un premier round d'observation de ce tableau met en exergue le fait que la majorité de ces algorithmes se focalisent sur l'optimisation d'une exploration en largeur d'abord dans un processus de découverte des deux « ordres idéaux ». Comme point de départ, ils ont considéré en premier l'ensemble des éléments *inf irréductibles*[12] [DAV 02]. Cet ensemble est augmenté par d'autres éléments issus d'autojointures soumis à des opérations d'élagages. Cependant, on remarque qu'aucun de ces algorithmes n'est arrivé à tenir les promesses avancées par cette nouvelle alternative d'algorithmes en matière de réduction de la redondance des règles associatives. L'origine de cet échec réside dans le fait qu'aucun de ces algorithmes n'a pu supporter le coût assez élevé de la construction de la relation d'ordre. Cette relation d'ordre est une condition, *sine qua non*, pour l'obtention de la base générique des règles associatives approximatives.

Il est important de remarquer qu'un schéma de gestion du buffer est passé sous silence dans tous les algorithmes passés en revue. En effet, ce schéma est nécessaire puisque l'étape de génération de candidats peut générer un résultat qui peut ne pas tenir en mémoire centrale.

Une remarque d'ordre général concerne l'absence des algorithmes connus dans la communauté de la théorie des concepts formels[13], pour la découverte des concepts formels et leur utilisation dans le domaine de la fouille de données. Cependant, on pense qu'une conjonction des algorithmes présentés par [NOU 99] et [PFA 02][14], peuvent être appliqués avec succès pour l'extraction des bases génériques de règles associatives.

5. Conclusion

Dans ce papier, nous avons présenté un survol de l'état de l'art des algorithmes basés sur la découverte des itemsets fermés. Le constat essentiel de cet état de l'art, à part des performances quelconques sur des contextes épars, est que ces algorithmes se sont focalisés sur l'extraction des itemsets fermés en négligeant la composante relation d'ordre sous-jacente.

Nous pouvons citer plusieurs perspectives d'approches futures.

Mettre en ligne un répertoire ou une plate-forme d'algorithmes basés sur l'extraction des itemsets fermés : la prolifération des implémentations en « stand-alone » des algorithmes peut mener à des résultats conflictuels (e.g. les résultats conflictuels concernant l'évaluation des performances présentés dans les manuscrits présentant

12. Ce sont les éléments dont l'ensemble de leurs prédécesseurs est un singleton.
13. Un survol est donné dans [GUE 90], actualisé par une étude comparative expérimentale dans [KUZ 02] et [FU 04a].
14. Cette conjonction est nécessaire, puisque le premier donne en sortie l'ensemble des itemsets fermés, et la relation d'ordre sous-jacente, tandis que le second permet de retrouver les générateurs minimaux associés grâce à la propriété introduite de *bloqueur minimal* [PFA 02].

respectivement les algorithmes CHARM et CLOSET). En effet, différentes implémenations d'un même algorithme peuvent se comporter d'une manière complètement différentes sur différentes bases de test. Ainsi, un espace algorithmique unifié pourrait fournir les bonnes caractérisations de performances de ces algorithmes, e.g., savoir pourquoi et sous quelles conditions un algorithme peut présenter de meilleures performances qu'un autre ?

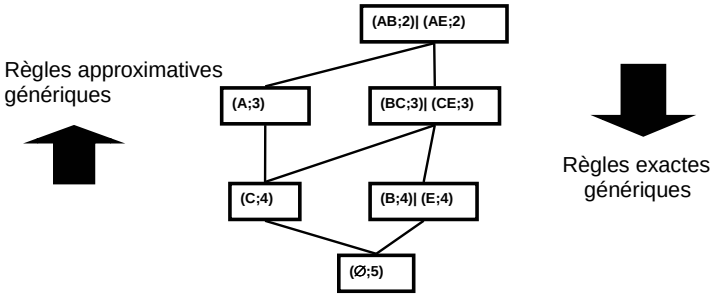

Figure 10. *Le treillis des générateurs minimaux associé au contexte d'extraction $\mathcal{K}$*

Parallélisation : comme le montre la deuxième ligne du tableau 3, force est de constater l'absence de tentatives pour la parallélisation des algorithmes basés sur l'extraction des itemsets fermés. En effet, la majorité des algorithmes parallèles pour la découverte sont basés sur la version séquentielle de l'algorithme APRIORI. On pense aussi qu'il serait intéressant de combiner les deux paradigmes : le parallélisme de tâche et celui des données. Deux travaux récents s'intéressent respectivement au partitionnement de l'espace de recherche [FU 04b] et au partitionnement de données à partir d'approche algorithmique incrémentale [KEN 04].

De nouvelles approches pour l'extraction d'itemsets fermés : il est connu que les algorithmes basés sur l'extraction des itemsets fermés sont moins performants que ceux basés sur l'extraction des itemsets fréquents pour les contextes faiblement corrélés. En effet, la détermination des fermetures des itemsets dans ce type de contextes pèse lourd sur les performances des algorithmes du premier type. Ainsi, deux pistes peuvent être approchées pour contourner cet handicap :

1) Estimer la densité du contexte par un moyen non coûteux[15] et d'adapter l'algorithme selon la nature du contexte. Même une approche « hybride » pourrait être préconisée. Par exemple, dans l'algorithme CLOSET et selon la densité du sous-contexte, le type d'algorithme approprié serait utilisé.

2) Il serait intéressant de proposer de nouvelles approches qui éviteraient le calcul de fermetures. Ainsi, on pourrait obtenir une structure partiellement ordonnée appelée

15. Un indicateur simple, proposé dans [PAS 00], est d'estimer la proportion d'1-itemsets fermés. Si cette proportion est importante, alors le contexte est plutôt dense.

« *Treillis de générateurs minimaux* ». Par exemple, le treillis des générateurs minimaux associé au contexte d'extraction $\mathcal{K}$ est donné par la figure 10. Dans une telle structure, chaque nœud représente les générateurs minimaux. Pour un nœud donné, l'itemset fermé associé pourrait être obtenu en effectuant l'union de tous les générateurs minimaux arrivant à ce nœud. Les bases de règles génériques pourraient être facilement obtenues d'une telle structure. Ainsi, il suffit de remonter dans la structure pour dériver la base de règles approximatives, tandis que la base de règles exactes est obtenue en descendant dans la hiérarchie.

6. Bibliographie

[ADR 97] ADRIAANS P., ZANTINGE D., *Data mining*, Addison-Wesley, 1997.

[AGR 93] AGRAWAL R., IMIELINSKI T., SWAMI A., « Mining Association Rules between sets of items in large Databases », *Proceedings of the ACM SIGMOD Intl. Conference on Management of Data*, Washington, USA, June 1993, p. 207-216.

[AGR 94] AGRAWAL R., SKIRANT R., « Fast algorithms for Mining Association Rules », *Proceedings of the 20th International Conference on Very Large Databases*, June 1994, p. 478–499.

[BAR 70] BARBUT M., MONJARDET B., *Ordre et classification. Algèbre et Combinatoire*, Hachette, Tome II, 1970.

[BAS 00a] BASTIDE Y., PASQUIER N., TAOUIL R., LAKHAL L., STUMME G., « Mining minimal non-redundant association rules using frequent closed itemsets », *Proceedings of the Intl. Conference DOOD'2000, LNCS, Springer-verlag*, July 2000, p. 972-986.

[BAS 00b] BASTIDE Y., TAOUIL R., PASQUIER N., STUMME G., LAKHAL L., « Mining frequent patterns with counting inference », *SIGKDD Explorations*, vol. 2, n° 2, 2000, p. 66-75.

[BAS 02] BASTIDE Y., TAOUIL R., PASQUIER N., STUMME G., LAKHAL L., « PASCAL : un algorithme d'extraction des motifs fréquents », *Technique et Science Informatiques*, vol. 21, n° 1, 2002, p. 65-95.

[BEN 03] BENYAHIA S., CHERIF C. L., MINEAU G., JAOUA A., « Découverte des règles associatives non redondantes : application aux corpus textuels », *Revue d'Intelligence Artificielle (special issue of Intl. Conference of Journées francophones d'Extraction et Gestion des Connaissances(EGC'2003)*, Lyon, France, vol. 17, n° 1-2-3, 2003, p. 131-143.

[BEN 04] BENYAHIA S., NGUIFO E. M., « Revisiting Generic bases of association rules », *Proceedings of 6th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2004) (Springer-Verlag) to appear*, Zaragoza, Spain, 2004.

[BOU 03] BOULICAUT J., BYKOWSKI A., RIGOTTI C., « Freesets : A condensed representation of boolean data for the approximation of frequency queries », *Data Mining and Knowledge Discovery*, vol. 7, n° 1, 2003, p. 5-22.

[BRI 97a] BRIN S., MOTAWNI R., SILVERSTEIN C., « Beyond Market baskets : Generalizing association rules to correlation », *Proceedings of the SIGMOD*, Tucson, Arizona,USA, May 1997, p. 265-276.

[BRI 97b] BRIN S., MOTAWNI R., ULLMAN J. D., « Dynamic itemset counting and implication rules for market basket data », *Proceedings of the ACM SIGMOD*, Tucson, Arizona USA, May 1997, p. 255-264.

[BYK 01] BYKOWSKI A., RIGOTTI C., « A condensed representation to find frequent patterns », *In Proceedings of the Twentieth ACM SIGACTSIGMOD SIGART Symposium on Principles of Database Systems, ACM Press*, 2001, p. 267-273.

[CAL 01] CALDERS T., GOETHALS B., « Mining all non derivable frequent itemsets », *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery, LNCS, Springer*, vol. 2431, 2001, p. 74-85.

[CHE 03] CHEUNG W., ZAIANE O., « Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint », *Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS 2003)*, Hong Kong, China, July 2003.

[DAV 02] DAVEY B., PRIESTLEY H., *Introduction to Lattices and Order*, Cambridge University Press, 2002.

[FU 04a] FU H., NGUIFO E. M., « Etude et conception d'algorithmes de génération de concepts formels », *Revue Ingénierie des Systèmes d'Information*, vol. 9, n° 3-4, 2004, p. 109–132, Hermès-Lavoisier.

[FU 04b] FU H., NGUIFO E. M., « A parallel algorithm to generate formal concepts for large data », EKLUND P., Ed., *Second Intl. Conf. on Formal Concept Analysis*, Springer-verlag, february 2004, p. 394–401.

[GAN 91] GANTER B., REUTER K., « Finding All Closed Sets », *Order*, vol. 3, n° 8, 1991, p. 283-290.

[GAN 99] GANTER B., WILLE R., *Formal Concept Analysis*, Springer-Verlag, 1999.

[GEE 01] GEERTS F., GOETHALS B., DEN BUSSCHE J. V., « A tight upper bound on the number of candidate patterns », CERCONE N., LIN T., WU X., Eds., *Proceedings of the 2001 IEEE International Conference on Data Mining*, IEEE Computer Society, 2001, p. 155-162.

[GRA 03] GRAHNE G., ZHU J., « Efficiently Using Prefix-trees in Mining Frequent Itemsets », GOETHALS B., ZAKI M. J., Eds., *Proceedings of Workshop on Frequent Itemset Mining Implementations (FIMI '03)*, Florida, USA, IEEE, November 2003.

[GUE 90] GUENOCHE A., « Construction du treillis de Galois d'une relation binaire », *Mathématiques et Sciences Humaines*, vol. 2, n° 109, 1990, p. 41-53.

[GUI 86] GUIGUES J., DUQUENNE V., « Familles minimales d'implications informatives résultant d'un tableau de données binaires », *Mathématiques et Sciences Humaines*, , n° 95, 1986, p. 5-18.

[HAN 95] HAN J., FU Y., « Discovery of multiple-level association rules from large databases », *Proceedings of the VLDB Conference*, 1995, p. 420-431.

[HAN 00] HAN J., PEI J., YIN Y., « Mining frequent patterns without candidate generation », *Proceedings of the ACM-SIGMOD Intl. Conference on Management of Data (SIGMOD'00)*, Dallas, Texas, May 2000, p. 1-12.

[HIP 98] HIPP J., MYKA A., WIRTH R., GUENTZER U., « A new algorithm for faster mining of generalized association rules », *LNAI 1510, Springer-verlag*, 1998.

[HUE 01] HUELLERMEIR E., « Implication-based fuzzy association rules », *Proceedings of the PKDD'2001 : Principles of Data Mining and Knowledge Discovery, LNAI 2168, Springer-verlag*, Freiburg, Germany, September 2001, p. 241-252.

[KEI 96] KEIM D., KRIEGEL H., « Visualization techniques for mining large databases : A comparison », *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, n° 8, 1996, p. 923-938.

[KEN 04] KENGUE J.-F. D., « Une étude de la parallélisation de la construction de treillis de Galois par des méthodes de type diviser-pour-régner », Mémoire de dea, July 2004, Université de Yaoundé I, Cameroun, Département Informatique - Faculté des sciences.

[KLE 94] KLEMETTINEN M., MANNILA H., RONKAINEN P., TOIVONEN H., VERKAMO A. I., « Finding interesting rules from large sets of discovered association rules », *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94),ACM Press*, November 1994, p. 401-407.

[KRU 99] KRUSE R. L., RYBA A. J., *Data structures and program design in c++*, 1999.

[KRY 02] KRYSZKIEWICZ M., « Concise Representations of Association Rules », HAND D. J., ADAMS N., BOLTON R., Eds., *Proceedings of Pattern Detection and Discovery, ESF Exploratory Workshop, LNCS*, vol. 2447 de *LNCS*, London, UK, September 2002, p. 92-109.

[KUZ 02] KUZNETSOV S., OBEDKOV S., « Comparing Performance of Algorithms for Generating Concept Lattices », *JETAI*, vol. 14, n° 1, 2002, p. 189-216.

[LAT 01] LATIRI C., BENYAHIA S., « Textmining : Generating association rules from textual data », *Proceedings of the XIXème Congrés INFORSID'2001*, Geneva, Switzerland, 29-30 May 2001, p. 27-39.

[LIU 99] LIU B., HSU W., ANG K. W., CHE S., « Visually aided exploration of interesting association rules », *Proceedings of the 3rd Intl. Conf. on Research and Development in Knowledge Discovery and Data mining (PAKDD'99), LNCS,Vol. 1574, Springer-verlag*, April 1999, p. 380-389.

[LUO 01] LUONG V. P., « Raisonnement sur les règles d'association », *Proceedings 17ème Journées Bases de Données Avancées BDA'2001, Cépaduès Edition*, Agadir, Maroc, November 2001, p. 299-310.

[LUX 91] LUXENBURGER M., « Implication partielles dans un contexte », *Mathématiques et Sciences Humaines*, vol. 29, n° 113, 1991, p. 35-55.

[MEO 96] MEO R., PSAILA G., CERI S., « A new SQL-Like operator for mining association rules », *Proceedings of the VLDB Conference*, 1996, p. 122-133.

[NG 98] NG R. T., LAKSHMANAN V. S., HAN J., PANG A., « Exploratory mining and pruning optimizations of constrained association rules », *Proceedings of the SIGMOD Conference*, 1998, p. 13-24.

[NOU 99] NOURINE L., RAYNAUD O., « A fast algorithm for building lattices », *Information Processing Letters*, , n° 71, 1999, p. 199-214.

[PAS 98] PASQUIER N., BASTIDE Y., TOUIL R., LAKHAL L., « Pruning closed itemset lattices for association rules », BOUZEGHOUB M., Ed., *Proceedings of 14th Intl. Conference Bases de Données Avancées*, Hammamet, Tunisia, 26-30 October 1998, p. 177-196.

[PAS 99a] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Efficient Mining of Association Rules Using Closed Itemset Lattices », *Information Systems Journal*, vol. 24, n° 1, 1999, p. 25-46.

[PAS 99b] PASQUIER N., BASTIDE Y., TOUIL R., LAKHAL L., « Discovering Frequent Closed Itemsets », *Proceedings of 7th International Conference on Database Theory (ICDT'99), LNCS, Vol. 1540, Springer Verlag*, January 1999, p. 398-416.

[PAS 00] PASQUIER N., « Data Mining : algorithmes d'extraction et de réduction des règles d'association dans les bases de données », Doctorat d'Université, Université de Clermont-Ferrand II, France, 2000.

[PEI 02] PEI J., HAN J., MAO R., NISHIO S., TANG S., YANG D., « CLOSET : An efficient algorithm for mining frequent closed itemsets », *Proceedings of the ACM SIGMOD DMKD'00*, Dallas,TX, 2002, p. 21-30.

[PFA 02] PFALTZ J. L., TAYLOR C. M., « Scientific Discovery through Iterative Transformation of Concept Lattices », *Proceedings of Workshop on Discrete Applied Mathematics in conjunction with the 2nd SIAM International Conference on Data Mining*, Arlington, 2002, p. 65-74.

[SKI 95] SKIRANT R., AGRAWAL R., « Mining generalized association rules », Research report, 1995, Almaden Research Center, IBM Research Division, San Jose, USA.

[SRI 97] SRIKANT R., VU Q., AGRAWAL R., « Mining Association Rules with Item Constraints », *Proc. of the 3rd Intl. Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, California*, August 1997, p. 67-73.

[STU 00] STUMME G., TAOUIL R., BASTIDE Y., PASQUIER N., LAKHAL L., « Fast Computation of Concept Lattices Using Data Mining Techniques », BOUZEGHOUB M., KLUSCH M., NUTT W., SATTLER U., Eds., *Proceedings of 7th Intl. Workshop on Knowledge Representation Meets Databases (KRDB'00)*, Berlin, Germany, 2000, p. 129-139.

[STU 01] STUMME G., TAOUIL R., BASTIDE Y., PASQUIER N., LAKHAL L., « Intelligent structuring and reducing of association rules with formal concept analysis », *Proc. KI'2001 conference, LNAI 2174, Springer-verlag*, September 2001, p. 335-350.

[STU 02] STUMME G., TAOUIL R., BASTIDE Y., PASQUIER N., LAKHAL L., « Computing Iceberg Concept Lattices with TITANIC », *J. on Knowledge and Data Engineering (KDE)*, vol. 2, n° 42, 2002, p. 189-222.

[TAN 02] TAN P., KUMAR V., SRIVASTAVA J., « Selecting the right interestingness measure for association patterns », *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ICDM'02), ACM Press*, 2002, p. 32-41.

[VAL 02] VALTCHEV P., MISSAOUI R., GODIN R., MERIDJI M., « Generating frequent itemsets incrementally : two novel approaches based on Galois lattice theory », *J. Expt. Theoretical Artificial Intelligence*, vol. 14, n° 1, 2002, p. 115-142.

[WAN 03] WANG J., HAN J., PEI J., « CLOSET+ : Searching for the Best Strategies for Mining Frequent Closed Itemsets », *Proceedings ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)*,Washington, USA, IEEE, August 2003.

[ZAK 00] ZAKI M. J., « Generating Non-Redundant Association Rules », *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, August 2000, p. 34-43.

[ZAK 02] ZAKI M. J., HSIAO C. J., « CHARM : An Efficient Algorithm for Closed Itemset Mining », *Proceedings of the 2nd SIAM International Conference on Data Mining*, Arlington, April 2002, p. 34-43.