Algorithme de construction d'un treillis des concepts formels et de détermination des générateurs minimaux

Sondess Ben Tekaya, Sadok Ben Yahia, Yahia Slimani

Algorithme de construction d'un treillis des concepts formels et de détermination des générateurs minimaux

S. Ben Tekaya, S. Ben Yahia, Y. Slimani

Faculté des Sciences de Tunis, 1060, Tunis, Tunisie
sondess.bentekaya@esti.rnu.tn, {sadok.benyahia, yahya.slimani}@fst.rnu.tn

RÉSUMÉ. Le nombre prohibitif de règles d'association qui peuvent être dérivées même pour des bases de données de taille raisonnable est à l'origine du développement de techniques pour réduire la taille de l'ensemble de ces règles. Dans ce contexte, les résultats obtenus par l'Analyse de Concepts Formels (AFC) a permis de définir un sous-ensemble de règles appelé *base générique*. Cependant, un survol de la littérature montre que tous les algorithmes qui leur sont dédiés ont négligé une composante essentielle : soit la relation d'ordre sous-jacente ou c'est l'extraction des générateurs minimaux qui manque à l'appel. Dans ce papier, nous proposons un algorithme, appelé GENALL, pour construire un treillis de concepts formels, dans lequel chaque concept formel est "décoré" par ses générateurs minimaux. L'objectif de cet algorithme est d'extraire toute la connaissance nécessaire pour extraire les bases génériques des règles d'association, *e.g* ceux de Bastide et *al.* La principale originalité de GENALL est l'emploi d'un processus de raffinement des listes des successeurs immédiats pour déterminer, d'une manière simultanée, l'ensemble des concepts formels, leur ordre partiel sous-jacent et l'ensemble de générateurs minimaux associés à chacun des concepts formels. Les résultats expérimentaux ont prouvé que l'algorithme proposé est particulièrement efficace pour des contextes d'extraction denses comparé à Nourine *et al.* Le temps de réponse de l'algorithme GENALL surpasse celui de l'algorithme de Nourine *et al.* (il est en moyenne 40 fois plus rapide que celui de Nourine *et al.*)

ABSTRACT. The extremely large number of association rules that can be drawn from –even reasonably sized datasets–, bootstrapped the development of more acute techniques or methods to reduce the size of the reported rule sets. In this context, the battery of results provided by the Formal Concept Analysis (FCA) allowed to define "irreducible" nuclei of association rule subset better known as *generic basis*. However, a thorough overview of the literature shows that all the algorithms dedicated neglected an essential component: the relation of order, or the extraction of the minimal generators. In this paper, we introduce the GENALL algorithm to build a formal concept lattice, in which each formal concept is "decorated" by its minimal generators The GENALL algorithm aims to extract generic bases of association rules. The main novelty in this algorithm is the use of refinement process to compute immediate successor lists to simultaneously determine the set of formal concepts, their underlying partial order and the set of minimal generators associated with each formal concept. Carried out experiments showed that the GENALL algorithm is especially efficient for dense extraction con-

Novembre 2005, Numéro spécial CARI'04

Revue ARIMA **pages 171 à 193**

texts compared to that of Nourine et *al.* Response times obtained from algorithm GENALL largely outperform those of Nourine et *al.*

MOTS-CLÉS : Fouille de données, Analyse de Concepts Formels, Treillis de Galois, Règles d'associations génériques, Générateur minimal.

KEYWORDS : Data Mining, Formal Concept Analysis, Galois lattice, Generic association rules, Minimal generator.

1. Introduction

La fouille de données est une discipline qui vise à découvrir des régularités ou corrélations cachées, dans des ensembles de données volumineux, généralement formalisées sous forme de règles d'associations [AGR 94]. La dernière décennie a été marquée par un effort algorithmique "conséquent" pour la réduction du temps de calcul de l'étape d'extraction des motifs intéressants. Le succès obtenu est essentiellement dû à des prouesses de programmation avec la conjonction de la manipulation de structures de données compactes en mémoire centrale. Cependant, il semble évident que cette frénésie a fait perdre de vue l'objectif essentiel de cette étape, *i.e.*, extraire une connaissance fiable, de taille "exploitable" et ayant une plus value pour les utilisateurs finaux. Ainsi, la quasi-totalité de ces algorithmes se sont focalisés sur l'énumération de tous les motifs – présentant une fréquence d'apparition jugée satisfaisante – maximaux ou fermés [GOE 03].

Plusieurs études ont souligné le nombre prohibitif des règles d'association générées même pour des bases de taille raisonnable [ZAK 04, STU 01]. Ce fait a encouragé l'élaboration de techniques et de méthodes pour réduire la taille de l'ensemble des règles générées. Dans ce contexte, la batterie de résultats fournie par l'Analyse de Concepts Formels (ACF) a permis de définir un noyau "irréductible" d'un sous-ensemble de règle d'association mieux connu sous le nom de **bases génériques** [PAS 00b]. Ces bases constituent des ensembles réduits de règles informatives permettant de préserver les règles les plus pertinentes, sans perte d'information. Ainsi, le problème de découverte des règles associatives, considéré sous l'optique de découverte des itemsets fermés, pourrait être reformulé comme suit [BEN 04b] :

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

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

Un aperçu critique de la littérature dédiée a permis de dégager la classification suivante :

1) **Les algorithmes orientés pour la fouille de données** produisent l'ensemble des itemsets fermés[1] et l'ensemble de générateurs minimaux associés [ZAK 02, PEI 02, PAS 00a]. L'ordre partiel entre les itemsets fermés est absent ou n'a pas d'intérêt. L'origine de cet absence ou désintérêt 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. Pour obtenir cet ordre partiel, nous pouvons employer l'al-

1. Ou d'une manière équivalente l'ensemble des intensions des concepts formels.

gorithme proposé par Valtchev *et al.* [VAL 00] pour construire le graphe de couverture.

2) **Les algorithmes orientés pour les concepts formels** produisent l'ensemble des concepts formels et l'ordre sous-jacent [KUZ 02]. L'ensemble des générateurs minimaux associés aux concepts formels est absent. Dans ce cas, nous pouvons appliquer l'algorithme JEN [FLO 03] pour déterminer les générateurs minimaux à partir du graphe de couverture qui est déjà construit.

Dans cet article, notre principale motivation est l'absence d'un algorithme autonome permettant de collecter toute la connaissance requise pour l'extraction des bases génériques des règles d'associations. Ainsi, nous proposons un nouvel algorithme, appelé GENALL, fusionnant les points de vues mentionnés ci-dessus. En effet, visant à dériver les bases génériques des règles d'association, nous proposons de construire le treillis dans lequel chaque concept formel est décoré par l'ensemble des générateurs minimaux qui lui est associé. Comme point de départ, nous choisissons d'améliorer l'algorithme présenté par Nourine *et al.* [NOU 99]. Ce choix peut être justifié par le fait que ce dernier présente la meilleure complexité théorique, bien que ses performances pratiques laissent beaucoup à désirer comparé aux autres algorithmes [KUZ 02, FU 04]. Ainsi, une étude soigneuse de l'algorithme de Nourine *et al.* a prouvé qu'il passe par deux étapes :

1) Une approche originale, basée sur un trie spécial, pour découvrir et stocker les concepts formels,

2) Un accès coûteux répétitif au disque pour calculer l'ordre sous-jacent entre les concepts formels.

Cependant, les mauvais résultats de l'algorithme de Nourine *et al.* ne sont pas une fatalité et nous estimons que ces mauvaises performances peuvent être, en grande partie, améliorée en utilisant l'algorithme GENALL proposé dans cet article.

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 bases génériques de règles d'association. La section 3 discute d'une manière détaillée l'algorithme GENALL. Les résultats expérimentaux effectués sur des ensembles de données benchmarks sont rapporté dans la section 4. La section 5 conclut cet article et présente quelques directions de recherches pour les travaux futurs.

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.

2.1. Notions de base

Contexte formel (ou d'extraction) : Considérons un *contexte d'extraction* $\mathcal{K} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, où $\mathcal{O}$ est un ensemble d'objets, $\mathcal{I}$ est un ensemble d'attributs (ou items) et $\mathcal{R}$ une relation binaire entre les objets et les attributs ($\mathcal{R} \subseteq \mathcal{O} \times \mathcal{I}$). Chaque couple $(o, a) \in \mathcal{R}$ exprime que la transaction $o \in \mathcal{O}$ contient l'attribut $a \in \mathcal{I}$. Dans un contexte d'extraction, les objets sont dénotés par des nombres et les attributs par des lettres.

Nous définissons deux fonctions récapitulant les liens entre les sous-ensembles d'objets et sous-ensembles d'attributs induits par $\mathcal{R}$, qui associe un ensemble d'objets à un ensemble d'attributs et *vice versa*. Ainsi, pour un ensemble $O \subseteq \mathcal{O}$, nous définissons :

$$\phi(O) = \{a \in \mathcal{I} | \forall o \in \mathcal{O}, (o, a) \in \mathcal{R}\}$$

et pour un ensemble $A \subseteq \mathcal{I}$,

$$\psi(A) = \{o \in \mathcal{O} | \forall a \in \mathcal{I}, (o, a) \in \mathcal{R}\}$$

Les deux fonctions ϕ et ψ forment *la connexion de Galois* entre les ensembles des parties $\mathcal{P}(\mathcal{I})$ et $\mathcal{P}(\mathcal{O})$ [BAR 70]. En conséquence, les deux opérateurs composés de ϕ et ψ sont des opérateurs de fermeture, en particulier $\omega = \phi \circ \psi$ est un opérateur de fermeture.

Concept formel : C'est un couple $C_{\mathcal{K}} = (O, A)$, où O est appellé *extension* (ou *extent*), et A est un itemset[2], appelé *intension* (ou *intent*). De plus, les deux ensembles O et A sont liés par la connexion de Galois, *i.e.*, $\phi(O) = A$ et $\psi(A) = O$.

Soit un itemset $A \subseteq \mathcal{I}$, le support de l'itemset A dans le contexte $\mathcal{K}$ est défini par :

$$support(A) = \frac{|\psi(A)|}{|\mathcal{O}|}$$

où la notation $|X|$ signifie la cardinalité de l'ensemble X.

Face : Soit $C_{\mathcal{K}} = (O, A)$ un concept formel et soit $pred_i(C_{\mathcal{K}})$ le $i$ème prédécesseur immédiat de $C_{\mathcal{K}}$ dans un treillis de Galois extrait d'un contexte $\mathcal{K}$. La $i$ème face du concept formel $C_{\mathcal{K}}$ correspond à la différence entre son intension et l'intension de son $i$ème prédécesseur [PFA 02]. Soit p le nombre de prédécesseurs immédiats du concept formel $C_{\mathcal{K}}$. La famille des faces $F_{C_{\mathcal{K}}}$ du concept formel $C_{\mathcal{K}}$ est exprimée par la relation suivante [PFA 02] :

$$F_{C_{\mathcal{K}}} = \{A - Intent(pred_i(C_{\mathcal{K}}))\}, i \in \{1, \ldots, p\}$$

Bloqueur minimal : Soit $G = \{G_1, ..., G_n\}$ une famille de n ensembles. Un *bloqueur* B de la famille G est un ensemble où son intersection avec tout ensemble $G_i \in G$ est

2. Un itemset est un ensemble d'items ou attributs.

non vide [PFA 02]. Un bloqueur B de la famille $G = \{G_1, ..., G_n\}$ est dit minimal si [PFA 02] :

$$\nexists B_1 \subset B \text{ et } \forall G_i \in G, B_1 \cap G_i \neq \emptyset$$

Générateur minimal : soit $C_{\mathcal{K}}$ un concept formel et $F_{C_{\mathcal{K}}}$ sa famille de faces. L'ensemble G des générateurs minimaux associés à l'intension A du concept formel $C_{\mathcal{K}}$, correspond aux bloqueurs minimaux associés à la famille de ses faces $F_{C_{\mathcal{K}}}$ [PFA 02]. D'une manière équivalente, un itemset $g \subseteq \mathcal{I}$ est appelé *générateur minimal* d'un concept formel $C_{\mathcal{K}} = (O, A)$, si et seulement si $\omega(g) = A$ et $\not\exists g_1 \subset g$ tels que $\omega(g_1) = A$ [BAS 00a].

Classe d'équivalence : l'opérateur de fermeture (ω) induit une relation d'équivalence sur l'ensemble des parties de $\mathcal{I}$. *i.e.*, l'ensembles de parties est subdivisé en des sous-ensembles disjoints, appelés aussi *classes d'équivalence*. Dans chaque classe, tous les éléments possèdent la même valeur de support. Les générateurs minimaux d'une classe sont les éléments incomparables (dans la relation d'inclusion) les plus petits, tandis que l'itemset fermé est l'élément le plus large de cette classe.

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 $\mathcal{L}_{\mathcal{C}_{\mathcal{K}}} = (\mathcal{C}_{\mathcal{K}}, \leq)$, appelé *treillis de concepts formels (ou treillis de Galois)*, quand l'ensemble $\mathcal{C}_{\mathcal{K}}$ est considéré avec la relation d'inclusion ensembliste entre les itemsets [BAR 70, GAN 99]. 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* ($Couv_i$).

2.2. Dérivation des règles associatives

Une règle d'association est une implication entre itemsets assortie de mesures statistiques. Elle est de la forme

$$r : X \Rightarrow (Y - X)$$

où X et Y sont des itemsets fréquents (leurs support est supérieur ou égal à un seuil minimal, appelé $minsup$), et $X \subset Y$. Les itemsets X et $(Y - X)$ sont appelés, respectivement, *prémisse* et *conclusion* de la règle r. Les règles d'association valides sont celles dont la mesure de confiance $Conf(r) = \frac{support(Y)}{support(X)}$ est supérieure ou égale à seuil minimal de confiance, appelé $minconf$. Une règle d'association qui a une confiance égale à 1 est appellée *règle d'association exacte*. Dans le cas contraire, elle est dite *règle d'association 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 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 repose essentiellement sur l'utilisation de patrons définis par l'utilisateur (*user-defined templates*) [KEI 96, KLE 94, LIU 99]. 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 97, 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.

Bastide *et al.* [BAS 00a, STU 01]ont caractérisé ce qu'ils ont appelé *"la base générique pour les règles d'associations exactes"* (en adaptant la base d'implication globale de *Guigues et Duquenne* [GUI 86]). La base générique pour les règles d'association exactes est définies comme suit :

Définition 1 *soit $FC_{\mathcal{K}}$ l'ensemble des itemsets fermés fréquents extraits à partir d'un contexte d'extraction $\mathcal{K}$. Pour chaque itemset fermé fréquent I, notons par $\mathcal{G}_I$ l'ensemble de générateurs minimaux de I. La base générique pour les règles d'association exactes GB est définie comme suit :*

$$GB = \{r : g \Rightarrow (I - g) \mid I \in FC_{\mathcal{K}} \text{ et } g \in \mathcal{G}_I \text{ et } g \neq I\ ^{3}\}$$

Bastide *et al.* [BAS 00a, STU 01] ont également caractérisé la base informative pour les règles d'association approximatives, basées sur l'extension de la base d'implications partielles de *Luxenburger* [LUX 91]) qui est définie comme suit :

Définition 2 *La base informative de règles d'associations approximatives est donnée par :*

$$InfB = \{r : g \Rightarrow (I - g) | I \in FC_{\mathcal{K}} \text{ et } \omega(g) \leq I \text{ et } Conf(r) \geq minconf \text{ et } support(I) \geq minsup\}$$

3. La condition $g \neq I$ assure le rejet des règles associatives de la forme $g \Rightarrow \emptyset$.

3. L'algorithme GENALL

Dans cette section, nous présentons un nouvel algorithme, appelé GENALL, pour dériver les bases génériques des règles d'association. Pour ce faire, il recueille en un seul passage de la base toutes les informations exigées qui sont :

1) L'ensemble de concepts formels et leurs générateurs minimaux associés ;

2) L'ordre partiel sous-jacent.

L'algorithme GENALL est inspiré en particulier de la première partie de l'algorithme de Nourine *et al.* [NOU 99], principalement sur la construction du trie[4] des concepts formels. Le "trie" utilisé dans l'algorithme de Nourine présenté dans [NOU 99] est un trie spécial, dont les arcs sont étiquetés par les attributs des concepts formels, et seuls quelques nœuds sont libellés par les extensions des concepts formels. Par exemple, considérons le contexte d'extraction donné par la Table 1. Le trie obtenu est illustré par la Figure 1. Le chemin allant du nœuds de racine vers un nœuds libellé par une extension forme un concept formel, *e.g.*, $(ag, 1234)$ est un concept formel.

$\mathcal{R}$	a	b	c	d	e	f	g	h	i
1	×	×					×		
2	×	×					×	×	
3	×	×	×				×	×	
4	×		×				×	×	×
5	×	×		×		×			
6	×	×	×	×		×			
7	×		×	×	×				
8	×		×	×		×			

Tableau 1. *Le contexte d'extraction $\mathcal{K}$*

Le choix de cet algorithme est motivé par le fait qu'il présente la meilleure complexité théorique [KUZ 02, NOU 99, FU 04]. En plus de sa complexité linéaire, l'originalité de cet algorithme réside dans sa manière de découverte des concepts formels et leur stockage dans un trie spécial. Cependant, cet algorithme présente des inconvénients que nous pouvons récapituler dans les points suivants :

1) Le calcul séquentiel des concepts formels puis de leur ordre

2) Le retour systématique à la base pour calculer le graphe de couverture des concepts formels

4. Le terme Trie est dérivé de "reTrieval". C'est un arbre de recherche, dont les éléments sont stockés d'une manière condensée.

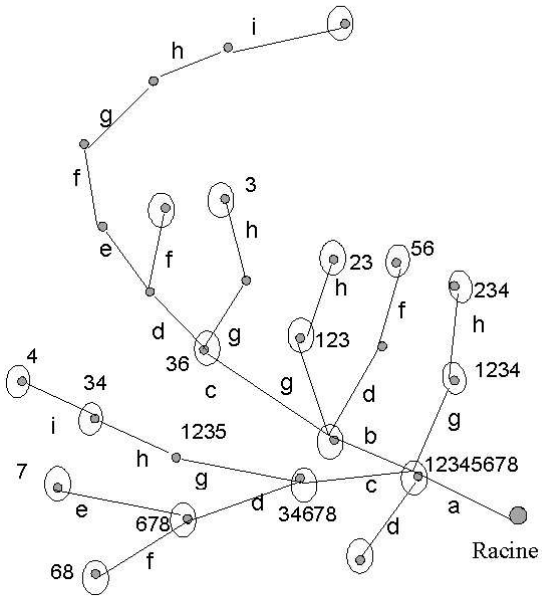

Figure 1. *La structure de Trie associée au contexte d'extraction donné par la Table 1*

3) Sa limitation au calcul de l'ordre partiel

Pour pallier ces inconvénients, nous présentons un nouvel algorithme, qui s'appuie sur la construction du trie pour calculer au fur et à mesure, l'ensemble des concepts formels, leurs générateurs minimaux associés et l'ordre partiel entre ces concepts formels. Les notations utilisées dans l'algorithme GENALL sont donnés dans la Table 2, tandis que le pseudo-code est illustré dans l'*Algorithme* 1.

Dans chaque itération, l'algorithme construit l'ensemble des concepts formels, l'ensemble des générateurs minimaux candidats et un ordre partiel potentiel, qui doit être raffiné plus tard, entre les concepts formels déjà découverts. Chaque transaction est lue une seule fois et un seul accès à la base de données est nécessaire. Chaque itération est composée de deux étapes. La première calcule les concepts formels et produit un ordre potentiel entre les concepts formels. Tandis que, la seconde vise à raffiner la relation l'ordre déjà obtenue et à calculer générateurs minimaux associés. Ces deux étapes sont discutées dans ce qui suit.

Algorithme 1 Construction du treillis décoré par les générateurs minimaux

1: **Algorithme** GENALL
Entrée: Le contexte d'extraction $\mathcal{K}$
Sortie: L'arbre lexicographique de $\mathcal{F}$, ImmSucc, liste-gen
Debut
2: $\mathcal{F} = \mathcal{I}$
{/* Initialisation de la famille des concepts à l'ensemble des attributs */}
3: **Pour** chaque tuple $t \in \mathcal{K}$ **Faire**
4: $L = \emptyset$ {/* Initialisation de la liste de l'ensemble des concepts trouvés dans une itération */}
5: **Pour** chaque concept $C_i \in \mathcal{F}$ **Faire**
6: $C.intent = C_i.intent \cap t.items$
7: **Si** $C.intent \notin \mathcal{F}$ **Alors**
8: {/* Nouveau concept */}
9: $\mathcal{F} = \mathcal{F} \cup C$
10: $C.extent = C_i.extent \cup t.TID$
11: $C.ImmSucc = \{t.items\} \cup \{C_i\} \setminus \{C\}$
12: $L = L \cup C$
13: **Sinon**
14: {/* Concept existant */}
15: $C.extent = C.extent \cup C_i.extent \cup t.TID$
16: **Si** $C \notin L$ **Alors**
17: $L = L \cup C$
18: **Fin Si**
19: $LP = \{t.items\} \cup \{C_i\} \setminus \{C\}$
{/* Mise à jour de $C.ImmSucc$ */}
20: **Pour** chaque $P_i \in LP$ **Faire**
21: **Pour** chaque $Succ \in C.ImmSucc$ **Faire**
22: COMPARE-CONCEPT$(Succ, P_i)$
23: **Fin Pour**
24: **Fin Pour**
25: **Fin Si**
26: **Fin Pour**
{/* Finalisation de l'ensemble des successeurs immédiats et calcul des générateurs minimaux */}
27: **Pour** Chaque concept $C_i \in L$ **Faire**
28: $C_i.ImmSucc =$ FIND-SUCC(C_i, L)
29: **Pour** Chaque $Succ \in C_i.ImmSucc$ **Faire**
30: $face = Succ \setminus C_i$
{/* Calcul de la face du successeur immédiat */}
31: **Pour** Chaque $face_i \in Succ.liste - face$ **Faire**
32: $Succ.liste - face =$ COMPARE-FACE$(face, face_i)$
33: **Fin Pour**
34: **Fin Pour**
35: **Fin Pour**
36: **Fin Pour**
Fin

Structure	Champs	Description
$\mathcal{F}$		La famille des concepts formels C
L		La liste des concepts calculés dans une itération i
C		Le concept formel
	intent	L'intension du concept C
	extent	L'extension du concept C
	ImmSucc	Liste des successeurs immédiats du concept C (les concepts qui le couvrent)
	liste-face	Liste des faces du concept (la différence avec ses fils)
	liste-gen	Liste des générateurs minimaux du concept
T		La transaction de la base
	TID	L'identificateur de la transaction (N° transaction)
	items	Les items qui existent dans la transaction

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

3.1. Génération des concepts formels

Dans cette étape (lignes 4-24), nous commençons par initialiser la liste $\mathcal{L}$ avec l'ensemble vide (ligne 4). Cette liste sera utile pour la mise à jour de l'ensemble des successeurs immédiats de chaque concept formel trouvé dans une itération.

Pour calculer l'ensemble des concepts formels, nous effectuons une intersection entre l'intension de chaque concept formel de la famille $\mathcal{F}$ et chaque transaction de la base de données. Deux cas sont distingués :

1) **L'intension n'existe pas dans la famille $\mathcal{F}$:** Il s'agit dans ce cas d'un nouveau concept formel trouvé, qui doit alors être ajouté à la famille $\mathcal{F}$. Ensuite, l'extension du concept est calculée (ligne 9). Les successeurs immédiats potentiels de ce nouveau concept formel sont initialisés avec les concepts formels utilisés pour le produire (ligne 10). En effet, pour déterminer le successeur immédiat potentiel d'un concept formel, nous devrions distinguer deux cas particuliers :

a) Si le concept formel produit est égal à la transaction, alors seul le concept formel utilisé dans cette intersection peut être un successeur immédiat. En effet, un concept formel ne peut pas être égal à son successeur immédiat ;

b) Sinon, la transaction et le concept formel sont considérés en tant que successeurs immédiats potentiels du concept formel.

Ensuite, ce nouveau concept formel est ajouté à la liste $\mathcal{L}$ (ligne 11). Il est important de mentionner que cette insertion est effectuée en maintenant un ordre croissant des intensions des concepts formels.

2) **L'intension existe déjà dans la famille $\mathcal{F}$:** l'extension du concept formel (ligne 13) doit être mise à jour (lignes 14-15), et nous vérifions si le concept est déjà exis-

tant dans la liste $\mathcal{L}$ (lignes 16-17). Visant à mettre à jour la liste des successeurs immédiats $ImmSucc$ du concept formel, et étant donné que nous maintenons, pour chaque concept formel, une liste $ImmSucc$, nous construisons une liste LP contenant les concepts formels utilisés pour le produire. Cette liste est nécessaire, pour pouvoir faire les comparaisons et pour mettre à jour la liste $ImmSucc$. En effet, pour chaque élément dans LP et pour chaque élément dans la liste $ImmSucc$, nous examinons l'inclusion de ces concepts formels en utilisant la fonction COMPARE-CONCEPT (ligne 20). La fonction COMPARE-CONCEPT est appliquée pour mettre à jour la liste $ImmSucc$ des concepts formels considérés. Cette liste doit être modifiée dans deux cas :

1) L'élément de LP est plus petit (en terme d'inclusion) qu'un élément de $ImmSucc$. Dans ce cas, l'ancien successeur sera remplacé par le nouveau.

2) Les deux concepts sont incomparables : un nouveau successeur sera alors ajouté à la liste $ImmSucc$ du concept formel en considération.

3.2. Raffinement de la liste des successeurs immédiats et la détermination des générateurs minimaux

Notons que les concepts formels trouvés dans une itération, représentent une branche du trellis, *i.e.*, pour chaque concept formel (excepté le plus grand), nous trouvons un autre concept formel calculé dans cette même itération, qui le couvre. Pour cela, nous parcourons la liste $\mathcal{L}$ des concepts formels trouvés dans itération et pour chacun des concepts de L, nous faisons appel à la fonction FIND-SUCC (lignes 25-26). Cette fonction est basée sur le fait qu'un concept formel de cardinalité n (*i.e.*, la longueur de son intension est égale à n) est au moins couvert par un concept formel de cardinalité $(n + 1)$ ou plus. Pour cela, et étant donné que la liste $\mathcal{L}$ est triée selon l'ordre croissant de la cardinalité de l'intension, nous cherchons pour chaque concept formel de la cardinalité n, un concept formel qui le couvre dans la liste de cardinalité $(n + 1)$. En cas de défaut, nous passons à la cardinalité $(n + 2)$, jusqu'à ce que l'on trouve un concept qui le couvre. Par la suite, nous comparons ces concepts formels pour faire la mise à jour de la liste $ImmSucc$.

Une fois la liste des successeurs immédiats ($ImmSucc$) du concept formel courant est établie, nous parcourons cette liste et pour chaque successeur immédiat nous devons calculer l'ensemble de ses générateurs minimaux. Pour obtenir ce résultat, nous calculons les faces associées aux successeurs immédiats (ligne 28), ensuite nous les comparons à la liste des faces correspondante en appelant la fonction COMPARE-FACE (ligne 30). L'ensemble des faces `liste-face` d'un concept formel (successeur immédiat) peut être modifié dans deux cas illustrés dans ce qui suit :

1) $face$ est plus petite (en terme d'inclusion) que l'élément `liste-face` : dans ce cas, nous calculons `difference-face`, et nous remplaçons l'ancienne face par la nouvelle. Si `difference-face` n'existe pas dans une des faces du concept formel, alors nous supprimons chaque générateur contenant cette différence. Cette suppression est né-

cessaire, vu qu'un attribut qui n'existe pas dans les faces ne peut pas exister dans les générateurs.

2) `face` est incomparable avec toutes les faces `liste-face` dans ce cas, nous l'ajoutons à `liste-face`.

Quand la liste des faces est mise à jour, la fonction de l'algorithme JEN CALCUL-BLOQUEURS-MINIMAUX [FLO 03] est appliquée pour déterminer les générateurs minimaux.

Exemple 1 *Considérons le contexte d'extraction illustré par la Table 1 avec l'ensemble d'attributs* $\mathcal{I} = \{a, b, c, d, e, f, g, h, i\}$ *et l'ensemble de transactions du contexte d'extraction, dénotées de 1 à 8.*

Considérons la transaction $4 = \{acghi\}$*, la famille* $\mathcal{F}$ *après le traitement des trois premières itérations est comme suit :*
$\mathcal{F} = \{(abcdefghi, \emptyset), (abg, 123), (abgh, 23), (abcgh, 3)\}$.
L'ensemble des successeurs immédiat de chaque concept est comme suit :
$\{abg\}.ImmSucc = \{\{abgh\}\}$ *;*
$\{abgh\}.ImmSucc = \{\{abcgh\}\}$ *;*
$\{abcgh\}.ImmSucc = \{\{abcdefghi\}\}$.

Première étape : Génération des concepts formels
Chaque concept formel de $\mathcal{F}$ *est manipulé individuellement. Par conséquent, pour le premier concept formel* C_1*,* $C_1.intent = \{abcdefghi\}$*, l'application de l'opération d'intersection avec la quatrième transaction donne un nouveau concept formel avec une intension égale à* $\{acghi\}$*. L'extension de ce nouveau concept formel, dénoté* C_5*, sera calculée plus tard.*
Ainsi, la famille $\mathcal{F}$ *est mise à jour en lui ajoutant* C_5 *:*
$\mathcal{F} = \{(abcdefghi, \emptyset), (abg, 123), (abgh, 23), (abcgh, 3), (acghi, \emptyset)\}$.
L'ensemble des successeurs immédiats de C_5 *est initialisé avec* C_1 *et l'extension de* C_5 *est initialisée avec l'union de* $C_1.extent$ *et l'identificateur de la transaction en question. Par conséquent,* $C_5.extent = \{4\}$ *et* $C_5.ImmSucc = \{\{abcdefghi\}\}$ *et la liste* $\mathcal{L}$ *est initialisée avec* $\{acghi\}$.
Le processus mentionné ci-dessus est répété pour tous les concepts formels existants dans la famille $\mathcal{F}$.
À la fin de cette première étape, nous obtenons la famille des concepts formels mise à jour
$\mathcal{F} = \{(abcdefghi, \emptyset), (abg, 123), (abgh, 23), (abcgh, 3), (acghi, 4), (ag, 1234),$
$(agh, 234), (acgh, 34)\}$*. La liste des successeur immédiats de chaque concept formel est comme suit :*
$\{acghi\}.ImmSucc = \{\{abcdefghi\}\}$;
$\{ag\}.ImmSucc = \{\{abg\}, \{acghi\}\}$;
$\{agh\}.ImmSucc = \{\{abgh\}, \{acghi\}\}$;

$\{acgh\}.ImmSucc = \{\{abcgh\}, \{acghi\}\}$;
La liste $\mathcal{L}$ contient toutes les intensions des concept formels découverts dans l'itération courante : $\mathcal{L} = \{\{ag\}, \{agh\}, \{acgh\}, \{acghi\}\}$.

Seconde étape : Raffinement de la liste des successeurs immédiats et détermination des générateurs minimaux

Pour le premier concept formel de la liste $\mathcal{L}$, nous avons $\{ag\}.ImmSucc = \{\{abg\}, \{acghi\}\}$. *L'intension du concept formel* $\{agh\}$ *couvre* $\{ag\}$ *et nous avons* $\{agh\} \subset \{acghi\}$. *Ainsi, il est nécessaire de remplacer dans* $\{ag\}.ImmSucc$ $\{acghi\}$ *par* $\{agh\}$. *En effet, l'ancien arc liant* $\{ag\}$ *à* $\{acghi\}$ *est un lien transitif. Par conséquent,* $\{ag\}.ImmSucc = \{\{abg\}, \{agh\}\}$.
Pour chaque successeur immédiat de $\{ag\}$, *nous devons calculer* `liste-face` *et* `liste-gen` *qui contient la liste des générateurs minimaux, en utilisant la fonction* COMPARE-FACE. *Par conséquent,*

1) $\{abg\}$.`liste-face`=$\{b\}$ *et* $\{abg\}$.`liste-gen`=$\{b\}$.

2) $\{agh\}$.`liste-face`=$\{h\}$ *et* $\{agh\}$.`liste-gen`=$\{h\}$.

Après le traitement de $\{ag\}.ImmSucc$, *nous devons manipuler la liste de successeurs du concept formel dont l'intention est égale à* $\{agh\}$, *i.e.,*$\{agh\}.ImmSucc$. *Ainsi, nous avons* $\{agh\}.ImmSucc = \{\{abgh\}, \{acghi\}\}$. *Nous commençons par raffiner la liste des successeurs immédiats* $\{agh\}.ImmSucc$. *Cependant, nous constatons que* $\{acgh\} \subset \{acghi\}$, *alors nous devons remplacer dans* $\{agh\}.ImmSucc$, $\{acghi\}$ *par* $\{acgh\}$. *Ainsi,* $\{agh\}.ImmSucc = \{\{abgh\}, \{acgh\}\}$. *Après le calcul de* `liste-face` *de* $\{abgh\}$ *nous obtenons :* $\{abgh\}$.`liste-face`=$\{h\} \cup \{b\}$. *Étant donné que* $\{h\} \cap \{b\} = \emptyset$, *alors* $\{b\}$ *ne peut pas être un bloqueur pour l'ensemble des faces* $\{\{h\}, \{b\}\}$. *C'est pourquoi la liste des générateurs minimaux est remplacée par* $\{bh\}$, *i.e.* $\{abgh\}$.`liste-gen`=$\{bh\}$.

Le processus de raffinement décrit est appliqué sur $\{acgh\}.ImmSucc$. *Ainsi,*
$\{acghi\}.ImmSucc = \{abcdefghi\}$.
$\{abcdefghi\}$.`liste-face`=$\{defi\} \cup \{bdef\}$
$\{abcdefghi\}$.`liste-gen`=$\{d, e, f, bi\}$.

Les processus de génération et de raffinement sont répétés pour toutes les transactions restantes. La Figure 2 décrit le treillis des concepts formels associé au contexte d'extraction $\mathcal{K}$, présenté dans la Table 1. Certains des générateurs minimaux, sont indiqués par des traits pointillés, décorant les nœuds des concepts formels.

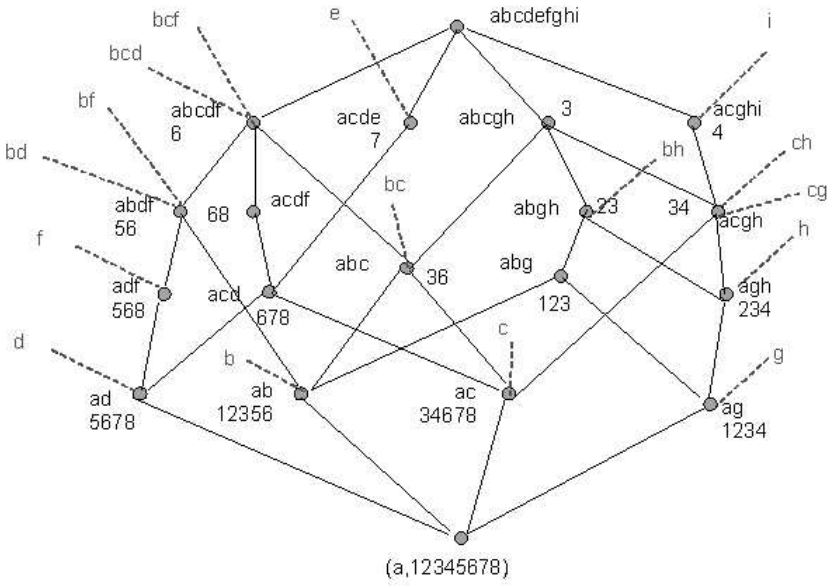

Figure 2. *Treillis des concepts formels, extrait du contexte $\mathcal{K}$, décoré par quelques générateurs minimaux*

4. Etude de la complexité

Dans cette section, nous allons étudier la complexité, dans le pire des cas, de l'algorithme GENALL [BEN 04a]. Soient, $\mathcal{O}$ l'ensemble des transactions d'une base, $\mathcal{I}$ l'ensemble des attributs de la base et $\mathcal{F}$ l'ensemble des concepts trouvés. L'étude de la boucle **Pour** (ligne 5) donne :

– L'instruction 6 s'effectue en $O(|\mathcal{I}|)$ étant donné que dans le pire des cas, nous allons effectuer une intersection avec tous les attributs de la base.

– L'instruction 7 se fait en $O(|\mathcal{I}|)$ puisqu'on peut avoir au maximum $|\mathcal{I}|$ branches.

Par ailleurs, la complexité du bloc d'instructions 9-12 (partie **alors**) étant inférieure à celle du bloc d'instructions 15-25, nous ne comptabiliserons que cette dernière :

– L'instruction 15 peut se faire dans le pire des cas en $O(|\mathcal{O}|)$.

– La recherche dans la liste L (ligne 16) est réalisée en $O(\frac{|\mathcal{F}|}{2})$ dans le pire des cas. En effet, une itération peut donner autant de nouveaux concepts qu'il y a dans $|\mathcal{F}|$.

– La boucle **Pour** (ligne 20) peut se répéter au maximum deux fois.

– Celle de la ligne 21 se répète $|ImmSucc|$ fois.

– La fonction COMPARE_CONCEPT se fait en $O(|\mathcal{I}|)$.

De ce fait, la complexité des instructions 5-25 est en $O(|\mathcal{F}|*2|\mathcal{I}|+|\mathcal{O}|+\frac{|\mathcal{F}|}{2}+2|\mathcal{I}|.|ImmSucc|)$. L'instruction 29 s'effectue en $O(|ImmSucc|)$, alors que la complexité de l'instruction 32 est en $O(|\mathcal{I}| + |liste\_gen|)$. Par conséquent, la complexité du bloc comprenant les instructions 27 à 36 est en $O(\frac{|\mathcal{F}|}{2} * |ImmSucc|^2.(|\mathcal{I}| + |liste\_gen|))$.
Puisque notre algorithme se répète $|\mathcal{K}|$ fois (boucle **Pour** de la ligne 3), la complexité totale de notre algorithme est alors en :

$$O(|\mathcal{K}|.|\mathcal{F}|.[\frac{1}{2}.|ImmSucc|^2.(|\mathcal{I}| + |liste\_gen|) + |\mathcal{K}| + \frac{|\mathcal{F}|}{2}])$$

Bien que, l'algorithme GENALL donne plus de résultats (calcul de l'ensemble des concepts, leur ordre et l'ensemble des générateurs minimaux associés à chaque concept) que les algorithmes de Nourine *et al.* [NOU 99] et Le Floc'h et al. [FLO 03], sa complexité reste toujours linéaire en fonction de $|\mathcal{F}|$.

5. Comparaison avec d'autres travaux

L'algorithme présenté dans [NOU 99] est semi-incrémental. En effet, il commence par construire d'une façon incrémentale l'ensemble des concepts, ce qui donne l'arbre des concepts. Ensuite, il utilise cet arbre pour la construction du diagramme. Or dans cette étape, un retour systématique à la base est nécessaire pour le calcul de l'ordre. Chacune des étapes se fait en $O((|\mathcal{I}| + |\mathcal{O}|).|\mathcal{O}|.|\mathcal{F}_{\mathcal{O}}|)$. Par contre dans GENALL, le calcul des concepts et l'ordre sous-jacent se fait en une seule étape, avec en plus le calcul de l'ensemble des générateurs minimaux associés à chaque concept. En ce qui concerne l'algorithme présenté dans [FLO 03], le calcul des générateurs minimaux suppose que l'ordre est déjà établi. Cette hypothèse est due au fait que le calcul des concepts et de leur treillis constitue une étape lourde et coûteuse. Dans notre approche, les générateurs minimaux sont déterminés au fur et à mesure de la construction du treillis. Malgré ce surcoût de calcul, la complexité de notre algorithme n'a pas augmenté.

6. Résultats expérimentaux

Nous avons implémenté les algorithmes GENALL et celui de Nourine *et al.* en language C sur une plate-forme Linux avec une distribution de Mandrake 8.1 pour évaluer leurs performances. Les deux algorithmes utilisent la même structure de données. Les tests expérimentaux ont été conduits sur une machine Pentium IV ayant une fréquence d'horloge de 2Ghz et 512 Mo de mémoire centrale. Pour examiner l'efficacité pratique de

notre algorithme, nous avons mené une sérié d'expérimentations sur des bases de données réelles et synthétiques, dont les caractéristiques sont détaillées dans ce qui suit.

6.1. Les bases de test

Les algorithmes ont été examinés sur deux types de bases de données : bases synthétiques et des bases de données denses, qui appartiennent au domaine des bases de données statistiques. Toutes ces bases de données de test sont librement disponibles sur Internet[5]. La base *Mushroom* contient les caractéristiques de diverses espèces de champignons. La base *Chess* est dérivée des étapes du jeu d'échec. Typiquement, ces bases de données réelles sont très denses. D'une façon générale, les bases de données synthétiques sont éparses comparées aux bases réelles.

La Table 3 présente les divers paramètres des bases de données synthétiques. La Table 4 montre les caractéristiques des bases réelles[6] et des bases de données synthétiques, utilisées dans le cadre de l'évaluation des performances des algorithmes GENALL et de Nourine *et al.*

|D|	Nombre de transactions
|T|	Taille moyenne des transactions
|I|	Taille moyenne des itemsets maximaux potentiellement fréquents

Tableau 3. *Paramètres des bases synthétiques*

Base	Nbre. Items	Taille moy. transactions	Nbre. transactions	Taille
Chess	76	37	3196	334Ko
Mushroom	120	23	8124	557 Ko
C20D10K	386	20	10000	155 Ko
T10I4D1K	775	10	1000	38,10 Ko
T10I4D100K	1000	10	100000	3830 Ko
Kosarak	41270	8	990002	30500 Ko

Tableau 4. *Caractéristiques des bases de test*

5. *http ://fimi.cs.helsinki.fi/data*
6. Indiquées en caractères gras.

6.2. Expérimentations

Des travaux antérieurs ont démontré que le comportement des algorithmes d'extraction des règles d'associations varie fortement selon les caractéristiques des jeux de données utilisés [PAS 98]. Le type de ces jeux de données dépend du coefficient de corrélations entre les items [PAL 04]. Les données faiblement corrélées (éparses), telles que les données synthétiques, constituent des cas faciles pour l'extraction des concepts formels, vu leur petit nombre par rapport au nombre de transactions. Pour de telles données, tous les algorithmes donnent des temps de réponse acceptables. En revanche, les données corrélées (denses) constituent des cas bien plus difficiles, du fait de l'importante proportion de concepts formels. De plus, ces données représentent une part importante des jeux de données réelles et les temps de réponse obtenus varient fortement selon l'algorithme utilisé [BAS 00b].

Dans la suite, nous allons évaluer les performances des algorithmes de Nourine *et al.* et GENALL sur des bases de données éparses et denses.

6.2.1. Cas des ensembles de données éparses

Pour ce type de bases de données, la liste des successeurs immédiats associée à un concept formel donné change fréquemment. Ainsi, des comparaisons sur toute la liste seront effectuées. Vu que la liste des successeurs est longue, dans les bases éparses par rapport à celle des bases denses, cette étape de mise à jour de la liste des successeurs consomme plus de temps comparativement à l'algorithme de Nourine *et al.* qui n'effectue pas cette comparaison. En effet, ce dernier ne calcule que la liste des prédécesseurs immédiats de chaque concept formel.

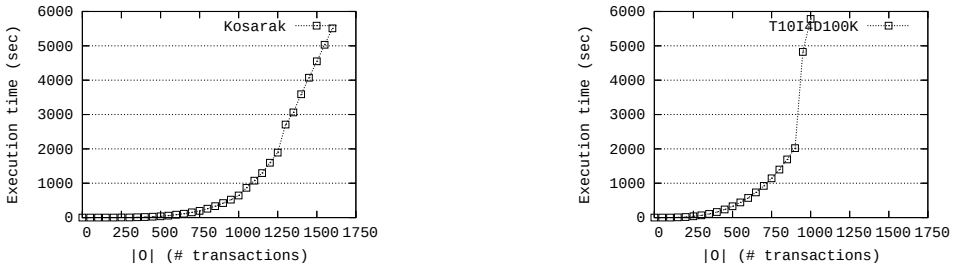

Figure 3. *Temps d'exécution pour le calcul des concepts et de l'ordre simultanément, de l'algorithme* GENALL *(cas des bases éparses)*

A partir des courbes représentées par la figure 3, nous pouvons dégager ce qui suit :

– Plus le nombre d'items par transaction est petit, plus l'algorithme GENALL, pour le calcul des concepts formels et de leur ordre, est plus rapide.

– Bien que les bases *Kosarak* et *T10I4D100K* ont presque les mêmes caractéristiques, nous remarquons que, pour le même nombre de transactions, l'algorithme GENALL est plus rapide dans le cas de *Kosarak* que pour l'autre base. Nous pouvons expliquer cette singularité par le taux de corrélation des items dans la base de *Kosarak* (L'analyse de corrélation entre les items mesure la relation entre deux items, le coefficient de corrélation indique que le changement d'un item en résulte le changement d'autres items).

6.2.2. Cas des bases de données denses

Dans le cas de bases denses, le nombre de concepts est beaucoup plus grand que celui des bases éparses. Ceci va ralentir le temps d'exécution par rapport à celui des bases éparses. Du point de vue exécution, nous avons remarqué que la liste des successeurs immédiats ne change pas fréquemment comme c'est le cas pour les données faiblement corrélées.

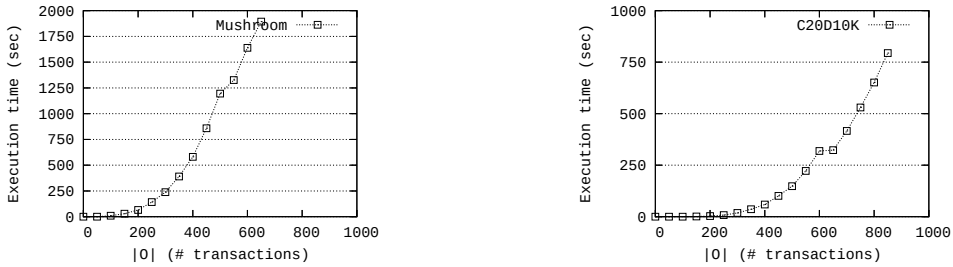

Figure 4. *Temps d'exécution pour le calcul des concepts et l'ordre simultanément, de l'algorithme* GENALL *(cas des bases denses)*

A partir de la figure 4, nous pouvons noter que :

– Plus les items sont corrélés, plus le temps d'exécution est important.

– Bien que les caractéristiques des bases *Mushroom* et *C20D10K* soient similaires, le temps d'exécution, pour un nombre de transactions bien déterminé, est largement supérieur pour la base *Mushroom* que pour la base *C20D10K*. Nous pouvons expliquer ce résultat par le fait que les items dans la base *C20D10K* sont plus corrélés que ceux de la base *Mushroom*, en plus de la différence de taille des transactions.

6.3. Comparaison de l'algorithme GENALL vs Nourine*et al.*

Dans cette section, nous allons comparer les performances de l'algorithme GENALL vs celui de Nourine *et al.*, sur le plan de la génération des concepts et le calcul de leur ordre.

Selon la figure 6, nous constatons que l'algorithme GENALL est très performant comparativement à l'algorithme de Nourine *et al.* pour les bases denses. En effet, le rapport de temps est très grand (la moyenne est de 40 et peut atteindre parfois 83).

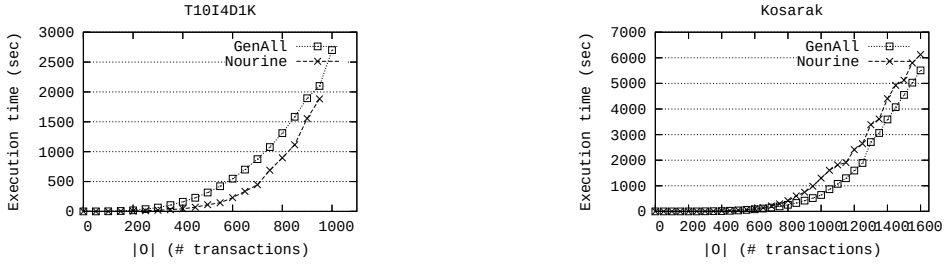

Figure 5. *Performances de l'algorithme* GENALL *vs celui de Nourine* et al. *(cas d'ensemble de données éparses)*

Cependant, selon la Figure 5, l'algorithme Nourine *et al.* est plus rapide que GENALL quand il est appliqué sur des données éparses. Néanmoins, le rapport de temps est proche de 1. Cependant, nous remarquons une singularité dans le comportement de la base éparse *Kosarak*, par rapport aux autres bases éparses. En effet, le temps d'exécution de l'algorithme GENALL sur la base *Kosarak* est plus petit que celui de Nourine *et al.* [NOU 99].

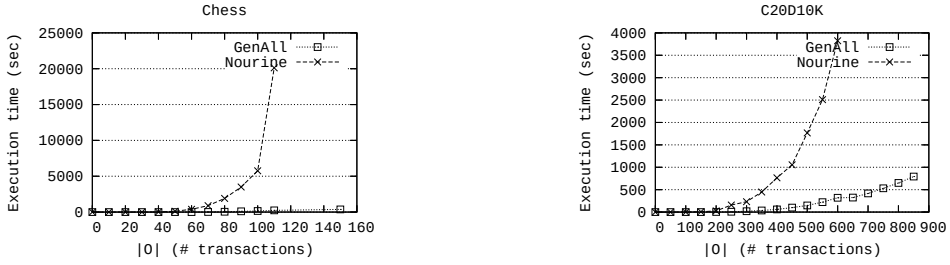

Figure 6. *Performances de l'algorithme* GENALL *vs celui de Nourine* et al. *(cas des données denses)*

7. Conclusion

Nous avons proposé un nouvel algorithme pour la construction **simultanée** de l'ensemble des concepts formels, leur ordre sous-jacent et la détermination de l'ensemble de générateurs minimaux qui sont associés à chaque concept formel. GENALL algo-

rithme [BEN 04a] exécute un balayage unique sur la base de données. En utilisant les résultats de l'algorithme, la dérivation des bases génériques pour les règles d'associations peut être effectuée d'une façon triviale. Dans un proche avenir, nous projetons d'aborder deux pistes. Premièrement, nous essayons d'améliorer les performances de GENALL en utilisant des structures de données avancées pour réduire le coût d'opérations de recherche, *e.g.*, la structure CFI présenté dans [GRA 03]. Deuxièmement, nous proposons d'examiner les avantages potentiels d'une version parallèle de l'algorithme GENALL sur une machine de MIMD (IBM SP2 avec 32 processeurs).

8. Bibliographie

[AGR 94] AGRAWAL R., SKIRANT R., « Fast algorithms for Mining Association Rules », *Proceedings of the 20th International Conference on Very Large Databases, Santiago, Chile*, 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 International Conference DOOD'2000, Lecture Notes in Computer Sciences, 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.

[BEN 04a] BENTEKAYA S., BENYAHIA S., SLIMANI Y., « Algorithme de construction d'un treillis des concepts formels et de détermination des générateurs minimaux », *Proceedings of the 7th African Conference on Research in Computer Science, Hammamet, Tunisie*, 22–25 Novembre 2004, p. 247-254.

[BEN 04b] BENYAHIA S., NGUIFO E. M., « Approches d'extraction de règles d'association basées sur la correspondance de Galois », *Ingénierie des Systèmes d'Information (ISI), Hermès-Lavoisier*, vol. 3–4, n° 9, 2004, p. 23-55.

[BRI 97] 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.

[FLO 03] FLOC'H A. L., FISETTE C., MISSAOUI R., VALTCHEV P., GODIN R., « JEN : un algorithme efficace de construction de générateurs pour l'identification des règles d'association », *Numéro spécial de la revue des Nouvelles Technologies de l'Information, Vol. 1 No. 1, Editions Cépaduès*, 2003, p. 135–146.

[FU 04] 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.

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

[GOE 03] GOETHALS B., ZAKI M. J., « FIMI'03 : Workshop on frequent itemset mining implementations », *FIMI Repository : http ://fimi.cs.helsinki.fi/*, November 2003.

[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.

[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.

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

[HUE 01] HUELLERMEIR E., « Implication-based fuzzy association rules », *Proceedings of the fifth Conference on Principles of Data Mining and Knowledge Discovery (PKDD'2001), 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.

[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.

[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. C., BENYAHIA S., « Textmining : Discovering explicit formal concepts from unstructutred data », *Proceedings of the XIXème Congrés Informatique des Organisations et Systèmes d'Information et de Décision, INFORSID'01, Suisse*, Mai 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 international Conference on research and development in Knowledge Discovery and Data mining (PAKDD'99), Lecture Notes in Computer Sciences, Springer-verlag*, vol. 1574, April 1999, p. 380–389.

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

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

[PAL 04] PALMERINI P., ORLANDO S., PEREGO R., « Statistical Properties of Transactional Databases », *ACM Symposium on Applied Computing, DM-track*, 2004.

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

[PAS 00a] 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.

[PAS 00b] PASQUIER N., « Datamining : Algorithmes d'extraction et de réduction des règles d'association dans les bases de données », Thèse de doctorat, Janvier 2000, Ecole Doctorale Sciences pour l'Ingénieur de Clermont Ferrand, Université Clermont Ferrand II.

[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.

[STU 01] STUMME G., TAOUIL R., BASTIDE Y., PASQUIER N., LAKHAL L., « Intelligent structuring and reducing of association rules with formal concept analysis », *Proceedings of KI'2001 Conference, Vienna, Austria, Lecture Notes in Artificial Intelligence 2174, Springer-Verlag*, September 2001, p. 335–350.

[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 00] VALTCHEV P., MISSAOUI R., LEBRUN P., « A Fast Algorithm for Building the Hasse Diagram of a Galois Lattice », *Proceedings of the Colloque LaCIM 2000,Montréal (CA)*, septembre 2000, p. 293-306.

[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.

[ZAK 04] ZAKI M., « Mining Non-Redundant Association Rules », *Data Mining and Knowledge Discovery*, , n° 9, 2004, p. 223–248.