Pascal : un algorithme d'extraction des motifs fréquents

Yves Bastide, Rafik Taouil, Nicolas Pasquier, Gerd Stumme, Lotfi Lakhal

PASCAL : un algorithme d'extraction des motifs fréquents

Yves Bastide [*] — **Rafik Taouil** [**] — **Nicolas Pasquier** [***] — **Gerd Stumme** [****] — **Lotfi Lakhal** [*****]

[*] *IRISA-INRIA - 35042 Rennes Cedex*
yves.bastide@irisa.fr

[**] *INRIA Lorraine - 54506 Vandœuvre-lès-Nancy Cedex*
rafik.taouil@loria.fr

[***] *I3S - université de Nice-Sophia Antipolis - 06903 Sophia Antipolis Cedex*
Nicolas.Pasquier@unice.fr

[****] *AIFB - Universität Karlsruhe (TH) - 76128 Karlsruhe - Allemagne*
stumme@aifb.uni-karlsruhe.de

[*****] *LIM - université de la Méditerranée - 13288 Marseille Cedex 09*
lakhal@lim.univ-mrs.fr

RÉSUMÉ. *Nous proposons dans cet article l'algorithme* PASCAL *qui introduit une nouvelle optimisation de l'algorithme de référence Apriori. Cette optimisation est fondée sur le* comptage des motifs par inférence, *qui utilise le concept de* motifs clés. *Le support des motifs fréquents non clés peut être inféré du support des motifs clés sans accès à la base de données. Expérimentalement, la comparaison de* PASCAL *avec Apriori, Close et Max-Miner montre son efficacité. Les motifs clés permettent aussi de définir les* règles d'association informatives, *potentiellement plus utiles que l'ensemble complet des règles d'association et beaucoup moins nombreuses.*

ABSTRACT. *In this paper, we propose the algorithm* PASCAL *which introduces a novel optimization of the well-known algorithm Apriori. This optimization is based on* pattern counting inference *that relies on the concept of* key patterns. *The support of frequent non-key patterns can be inferred from frequent key patterns without accessing the database. Experiments comparing* PASCAL *to Apriori, Close and Max-Miner, show that* PASCAL *is among the most efficient algorithms for mining frequent patterns. Key patterns also let us define* informative association rules, *potentially more useful to the user than the full set of association rules, and far less numerous.*

MOTS-CLÉS : *data mining, extraction de connaissances dans les bases de données, motifs fréquents, règles d'association.*

KEYWORDS: *data mining, knowledge discovery in databases, frequent patterns, association rules.*

1. Introduction

L'extraction de connaissances dans les bases de données (ECBD) est « le processus non trivial d'extraction d'informations valides, implicites, potentiellement utiles et ultimement compréhensibles à partir de grandes bases de données » [HAN 00a]. De nombreuses applications dans divers domaines ont bénéficié des techniques de l'ECBD et de nombreux travaux ont été menés sur ce sujet. Nous nous intéressons dans cet article au problème de la découverte des motifs fréquents, c'est-à-dire des groupements d'items apparaissant ensemble avec une fréquence significative, qui constitue l'étape principale de l'extraction de règles d'association [AGR 94], mais également d'un certain nombre d'autres problèmes de l'ECBD [HAN 00b] : l'extraction de série chronologiques [AGR 95], d'épisodes [MAN 97b], de corrélations [BRI 97a, SIL 98], de motifs multidimensionnels [KAM 97, LEN 97], de motifs maximaux [BAY 98, ZAK 97, LIN 98] et de motifs fermés [PAS 99c, PAS 99b, BOU 00a, PEI 00, TAO 00]. Ce problème est exponentiel dans la taille de la relation de base de données et cette relation doit être parcourue à plusieurs reprises durant le processus. Il est donc nécessaire de définir des algorithmes efficaces pour réaliser cette extraction.

Dans cet article, nous proposons également une solution au problème de la pertinence des règles produites, à travers la notion de règle d'association informative. Leur détermination classique conduit en effet à un nombre de règles beaucoup trop grand pour être utilisables aisément. La définition de règles d'association informatives, qui sont non redondantes et ne perdent pas d'informations, évite cet écueil [BAS 00b]. Une règle d'association est informative si sa prémisse est minimale et sa conclusion maximale ; elle est informative et réduite si elle ne peut pas se déduire facilement d'une autre règle informative.

1.1. *Travaux antérieurs*

Trois grandes approches ont été proposées pour l'extraction des motifs fréquents. La première consiste à parcourir itérativement par niveaux [1] l'ensemble des motifs. Durant chaque itération ou niveau, un ensemble de motifs candidats est créé en joignant les motifs fréquents découverts durant l'itération précédente ; les supports de ces motifs sont calculés et les motifs non fréquents sont supprimés. L'algorithme de référence basé sur cette approche est l'algorithme Apriori [AGR 94], proposé concurremment à l'algorithme OCD [MAN 94]. Un certain nombre de modifications de cet algorithme ont été proposées [BRI 97b, GAR 98, PAR 95, SAV 95, TOI 96] afin d'en améliorer divers aspects. Toutefois, tous ces algorithmes doivent déterminer le support de *tous* les motifs fréquents et de certains motifs non fréquents.

La seconde approche est basée sur l'extraction des motifs fréquents maximaux [2] dont tous les sur-ensembles sont non fréquents et tous les sous-ensembles sont fré-

1. Cette approche est appelée approche « levelwise » [MAN 97a].
2. « Maximaux » signifie « maximaux selon la relation d'inclusion ».

quents. Les algorithmes utilisant cette approche combinent un parcours par niveaux en largeur du bas vers le haut et un parcours en largeur du haut vers le bas de l'ensemble des motifs. Lorsque les motifs fréquents maximaux sont découverts, tous les motifs fréquents sont dérivés de ces derniers et un ultime balayage de la base de données est réalisé afin de calculer leur support. L'algorithme le plus efficace basé sur cette approche est l'algorithme Max-Miner [BAY 98]. Les résultats expérimentaux démontrent que cette approche est particulièrement efficace pour l'extraction des motifs fréquents maximaux. Mais dans le cas de l'extraction de tous les motifs fréquents, les performances se dégradent considérablement du fait du coût du dernier balayage : il nécessite approximativement un test d'inclusion entre chaque motif fréquent et chaque objet de la base de données. Comme dans le premier cas, les algorithmes basés sur cette méthode doivent calculer les supports de tous les motifs fréquents depuis la base de données.

La troisième approche, représentée par l'algorithme Close [PAS 99c], est basée sur le cadre théorique introduit dans [PAS 98] qui utilise la fermeture de la connexion de Galois [GAN 99, DUQ 99]. Ici, les motifs fermés fréquents (et leurs supports) sont extraits de la base de données en réalisant un parcours par niveaux. Un motif fermé est un ensemble maximal commun à un ensemble d'objets de la base de données. Tout motif non fermé est inclus dans le même ensemble d'objets et possède donc le même support que sa fermeture (le plus petit motif fermé qui le contient). Tous les motifs fréquents et leur support peuvent donc être déduits des motifs fermés fréquents avec leur support, sans accéder à la base de données. En conséquence, les motifs fréquents ne sont pas tous considérés durant la phase la plus coûteuse de l'algorithme, le comptage des supports ; et l'espace de recherche est considérablement réduit, particulièrement dans le cas de données corrélées. Les expérimentations ont démontré que cette approche est bien plus efficace que les deux précédentes sur ce type de données.

Le lecteur intéressé par une description complète des algorithmes d'extraction des motifs fréquents peut se référer à la thèse de N. Pasquier [PAS 00a]. Une typologie de structures de données pour l'implantation de ces algorithmes, avec des évaluations expérimentales, se trouve dans la thèse de Y. Bastide [BAS 00a].

1.2. *Contribution*

Nous proposons une nouvelle méthode, appelée *comptage par inférence*, destinée à réduire le nombre de calculs de supports des motifs lors de l'extraction des motifs fréquents. Cette méthode repose sur le concept de *motifs clés*. Un motif clé est un motif minimal d'une *classe d'équivalence* regroupant tous les motifs contenus dans exactement les mêmes objets de la base de données[3]. Tous les motifs d'une classe d'équivalence possèdent le même support, et le support des motifs non clés d'une

3. Une notion similaire de classe d'équivalence a été proposée récemment par R. Bayardo et R. Agrawal [BAY 99b] afin de caractériser les règles d'association d'antécédent maximal, qu'ils appellent « A-maximal rules ».

classe d'équivalence peut donc être déterminé en utilisant le support des motifs clés de cette classe. Avec le comptage par inférence, seuls les supports des motifs clés fréquents (et de certains non fréquents) sont calculés depuis la base de données.

Les concepts de motifs clés et de motifs fermés ont été introduits initialement dans [PAS 98] sous les noms d'« itemsets » générateurs et fermés. Ils sont à la base des algorithmes Close [PAS 98, PAS 99c] et A-Close [PAS 99b]. J.-F. Boulicaut et son équipe ont étendu le concept de motifs fermés avec les motifs δ-fermés [BOU 00a], où $\delta \in [0,1]$, puis le concept de motifs clés par celui de motifs δ-clés (ou ensembles δ-libres) [BOU 00b]. Ils ont ainsi montré concouramment à nos travaux [BAS 00d] que les motifs clés fréquents associés à leur frontière négative forment une représentation condensée des motifs fréquents. L'utilisation des motifs clés a aussi permis une caractérisation originale de la couverture des dépendances fonctionnelles et l'élaboration d'un algorithme efficace pour son extraction [NOV 01].

Le comptage par inférence a été implanté dans l'algorithme PASCAL, qui est une optimisation de l'algorithme simple et efficace Apriori. Dans PASCAL comme dans Apriori, les motifs fréquents sont extraits par niveaux : durant chaque itération, les motifs candidats de taille k sont créés en joignant les motifs fréquents de taille $k-1$, leur support est déterminé et les motifs non fréquents sont supprimés. Utilisant le comptage par inférence, si un motif candidat de taille k est un motif non clé, alors son support est égal au plus petit des supports de ses sous-ensembles de taille $k-1$. Ceci permet de réduire lors de chaque balayage de la base de données le nombre de motifs considérés et, encore plus important, de réduire le nombre total de balayages réalisés. La validité de cette optimisation est liée à la propriété suivante des motifs clés qui est compatible avec l'élagage d'Apriori : tous les sous-ensembles d'un motif clé sont des motifs clés (et réciproquement tous les sur-ensembles d'un motif non clé sont des motifs non clés). En comparaison des autres modifications d'Apriori proposées, l'inférence de comptage a un impact minimal sur la compréhensibilité et la simplicité de l'implantation de l'algorithme. La différence importante vient de ce que cette optimisation permet de calculer les supports d'autant de motifs que possible sans accéder à la base de données, en utilisant les informations acquises durant les itérations précédentes.

Pour déterminer les règles d'association informatives, les classes d'équivalence définies pour PASCAL sont utilisées : la partie gauche d'une règle est motif clé d'une classe d'équivalence, et la partie droite vient du motif maximal de cette classe ou d'une autre qui lui est « supérieure ».

Dans de précédents travaux [TAO 00, PAS 99a, PAS 00b], nous nous sommes intéressés à adapter les bases des règles d'implications exactes (*via* la base de Duquenne-Guigues [GUI 86]) et approximatives (par la base de Luxenburger [LUX 91]), ceci dans une approche de l'analyse de données par les treillis [DUQ 99]. Ces bases réduisent significativement le nombre de règles d'association ; mais les règles résultantes sont plus difficiles à interpréter que celles définies ici.

Nous avons réalisé des expérimentations afin de comparer l'algorithme PASCAL aux algorithmes Apriori, Close et Max-Miner, chacun représentatif d'une approche d'extraction des motifs fréquents, sur divers types de données. Les résultats démontrent que PASCAL améliore nettement l'efficacité de l'extraction des motifs fréquents dans le cas de données corrélées et que le comptage par inférence n'entraîne pas de perte de temps significative dans le cas de données faiblement corrélées. D'autre part, des expérimentations sur les règles d'associations montrent l'importance de la réduction du nombre de règles engendrées.

1.3. *Organisation de l'article*

Nous rappelons dans la section suivante le problème de l'extraction des motifs fréquents. Les définitions des motifs clés et du comptage par inférence, puis l'algorithme PASCAL qui en découle, sont présentés dans la section 3. La section 4 présente les règles d'association et leur simplification. Finalement, les résultats expérimentaux comparant l'efficacité des algorithmes Apriori, Max-Miner, Close et PASCAL sont présentés dans la section 5, de même que l'évaluation des règles d'association simplifiées. La section 6 conclut l'article.

2. Motifs fréquents

Qu'est-ce qu'un motif fréquent ? C'est un ensemble d'éléments présents dans un nombre « suffisamment grand » de lignes d'une base de données. Le cadre mathématique qui suit est classique, et formalise celui défini par R. Agrawal, T. Imielinski et A. Swami en 1993 [AGR 93] et raffiné par Agrawal et R. Srikant l'année suivante [AGR 94].

Définition 1. Base de données ou contexte formel.

Soit $\mathcal{O}$ un ensemble fini d'objets, $\mathcal{P}$ un ensemble fini d'éléments ou items, et $\mathcal{R}$ une relation binaire entre ces deux ensembles. On appelle *base de données* ou *contexte formel* [GAN 99] le triplet $\mathcal{D} = (\mathcal{O}, \mathcal{P}, \mathcal{R})$. On supposera que l'ensemble $\mathcal{P}$ est (totalement) ordonné, par exemple par l'ordre lexicographique.

Définition 2. Motif.

Un *motif* est un sous-ensemble de $\mathcal{P}$. On dit qu'un motif P est *inclus* dans l'objet o (ou que o *contient* P) si P et o sont en relation : $\forall p \in P, (o, p) \in \mathcal{R}$. Un motif de taille k est noté k-motif. Les motifs sont aussi appelés ensembles d'items (« itemsets » dans la littérature anglo-saxonne). Par abus de langage, nous parlerons indifféremment de « motif de $\mathcal{P}$ » et de « motif de $\mathcal{D}$ ».

Définition 3. Image d'un motif, image d'un ensemble d'objets.

Soit f la fonction qui fait correspondre à un motif l'ensemble des objets qui le contiennent : $f(P) = \{ o \in \mathcal{O} \mid o \text{ contient } P \}$. Soit g sa fonction duale, définie pour un ensemble d'objet : $g(O) = \{ p \in \mathcal{P} \mid \forall o \in O, (o, p) \in \mathcal{R} \}$.

Les fonctions f et g définissent la connexion de Galois d'une relation binaire. Les opérateurs associés $h = g \circ f$ et $h' = f \circ g$ sont les opérateurs de fermeture de Galois [BIR 67].

Définition 4. Support.

Le *support* d'un motif est la proportion des objets de la base de données qui le contiennent :

$$\sup(P) = \frac{\text{card}(f(P))}{\text{card}(\mathcal{O})}.$$

Cette définition est relative à la taille de la base de données ; le support est parfois défini de manière absolue. Puisque f est antitone [4], cette fonction est décroissante par rapport à la taille du motif : le support d'un ensemble est toujours inférieur ou égal au support de ses sous-ensembles.

Définition 5. Motif fréquent.

Soit un seuil *minsup* $\in [0,1]$, appelé le support minimum. Un motif est dit *fréquent* si $\sup(P) \geq$ *minsup*.

Le motif vide est évidemment fréquent quel que soit le support minimal, et fort peu intéressant.

Exemple 1. La base de données qui illustrera nos exemples est donnée sur le tableau 1. Le diagramme de Hasse du treillis des parties de $\mathcal{P} = \{a,b,c,d,e\}$, c'est-à-dire l'ensemble de tous les motifs possibles, est représenté sur la figure 1. Dans cette base de données :

– $\{b,c,d\}$ est un 3-motif (le fait qu'il n'apparaisse pas dans la base de données n'est pas important) ;

– $\{a,e\}$ est un 2-motif de support $2/6$. Si *minsup* $\leq 2/6$, c'est un motif fréquent.

3. L'algorithme PASCAL

Le principe que nous proposons ici est de classer les motifs selon un aspect particulier (les objets qui les contiennent) et d'utiliser les premiers motifs trouvés dans une classe pour inférer le support de tous les autres sans accéder à la base de données.

Nous allons voir dans cette section quelques définitions et résultats mathématiques justifiant l'algorithme PASCAL [BAS 00c, BAS 00d], puis en dériver celui-ci.

4. Les deux termes « antitone » et « anti-monotone » sont utilisés pour nommer cette propriété fondamentale : si $X \subseteq Y$, alors $f(X) \supseteq f(Y)$.

Id	Motif
1	$\{a,c,d\}$
2	$\{b,c,e\}$
3	$\{a,b,c,e\}$
4	$\{b,e\}$
5	$\{a,b,c,e\}$
6	$\{b,c,e\}$

Tableau 1. *Base de données*

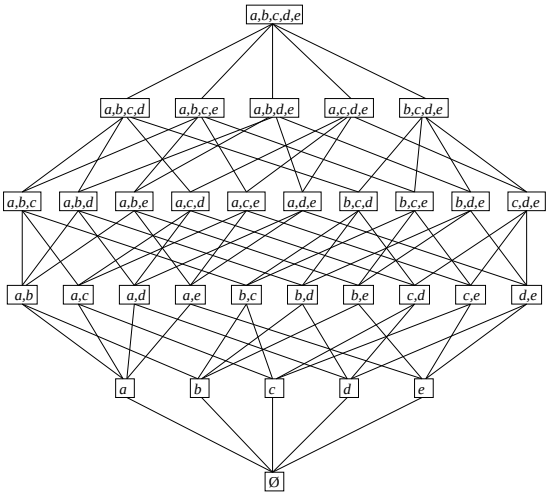

Figure 1. *Treillis des parties associé à $\mathcal{D}$*

3.1. *Aspects théoriques*

3.1.1. *Classes d'équivalences entre motifs*

L'idée de base dans PASCAL est le regroupement de motifs considérés comme équivalents. Il s'agit d'un principe très puissant, qui est utilisé ici pour regrouper dans des classes d'équivalences les motifs correspondant aux mêmes objets.

Définition 6. Soient $P, Q \subseteq \mathcal{P}$ deux motifs. La *relation d'équivalence* θ est définie comme suit :

$$P \theta Q \Longleftrightarrow f(P) = f(Q).$$

La *classe d'équivalence* de P est donnée par :

$$[P] = \{ Q \subseteq \mathcal{P} \mid P \theta Q \}.$$

Par exemple, $[\{b,c\}] = [\{b,c,e\}]$ si tout objet contenant b et c contient aussi e.

Si P et Q sont des motifs appartenant à la même classe d'équivalence, on voit aisément qu'ils ont le même support ; de même, deux motifs de support égal et dont l'un est inclus dans l'autre font partie de la même classe d'équivalence :

Proposition 1. *Soient deux motifs P et Q. On a :*

1) $P \theta Q \implies \sup(P) = \sup(Q)$;
2) $P \subseteq Q$ et $\sup(P) = \sup(Q) \implies P \theta Q$.

Démonstration. La première partie de cette proposition découle directement de la définition de la relation d'équivalence θ. Pour démontrer la seconde, on remarque que $f(P) \supseteq f(Q)$. Puisque $\sup(P) = \sup(Q)$ est équivalent à $\text{card}(f(P)) = \text{card}(f(Q))$, on a $f(P) = f(Q)$. □

Si la relation θ était connue d'avance, on pourrait compter le support d'un seul motif pour chaque classe d'équivalence. On ne connaît évidemment pas cette relation ; mais elle peut être construite au fur et à mesure. D'une manière générale, on calculera le support d'au moins un motif d'une classe d'équivalence. Si on a déterminé le support d'un motif P et qu'on trouve plus tard un motif $Q \in [P]$, il est inutile d'accéder à la base de données pour calculer le support de Q.

Dans l'algorithme de recherche des motifs fréquents par niveaux, les premiers motifs d'une classe d'équivalence rencontrés sont minimaux au sens de l'inclusion. Nous les appellerons *motifs clés*.

Définition 7. Un motif P est dit *motif clé* si $P \in \min[P]$. Un *motif clé candidat* est un motif dont tous les sous-ensembles (stricts) sont des motifs clés fréquents.

Sur notre exemple, les trois motifs $\{b,c\}$, $\{c,e\}$ et $\{b,c,e\}$ constituent une classe d'équivalence ; $\{b,c\}$ et $\{c,e\}$ sont donc des motifs clés.

On remarque qu'un motif clé candidat est également motif candidat.

L'algorithme étend la technique d'élagage de candidats de Apriori-Gen [AGR 94] à la fois sur les candidats et sur les candidats clés. Le théorème suivant le justifie :

Théorème 2. *L'ensemble des motifs clés fréquents forme un idéal d'ordre dans le treillis* $(2^{\mathcal{P}}, \subseteq)$.

Démonstration. Nous allons démontrer d'abord que l'ensemble des motifs clés (fréquents ou non-fréquents) forme un idéal d'ordre. Soient $P, Q \subseteq \mathcal{P}$; si P n'est pas un motif clé et si $P \subseteq Q$, alors Q n'est pas un motif clé.

P n'étant pas clé par hypothèse, il existe $P' \in \min[P]$ avec $P' \subset P$. Sachant que $f(P) = f(P')$, montrons que $f(Q) = f(Q \setminus (P \setminus P'))$. Nous savons que $f(P') = f(P) \subseteq f(P \setminus P')$ (antitonie de f). $f(Q) = f((Q \setminus (P \setminus P')) \cup (P \setminus P')) = f(Q \setminus (P \setminus P')) \cap f(P \setminus P')$. Donc, $f(Q) \supseteq f(Q \setminus (P \setminus P')) \cap f(P')$. Cette expression se récrit $f(Q) \supseteq f((Q \setminus (P \setminus P')) \cup P')$. Puisque $P' \subseteq Q \setminus (P \setminus P')$, nous trouvons $f(Q) \supseteq f(Q \setminus (P \setminus P'))$. L'inclusion opposée est immédiate, et notre égalité est vérifiée : Q n'est donc pas minimal dans sa classe d'équivalence.

Pour finir de démontrer le théorème en tenant compte du support, il suffit d'utiliser le fait que tout sous-ensemble d'un motif fréquent est fréquent. □

3.1.2. *Comptage des motifs par inférence*

Lors de la génération des motifs candidats, s'il apparaît un motif non clé, un motif clé appartenant à la même classe d'équivalence a déjà été traité lors d'une itération

précédente ; s'il est fréquent, nous connaissons donc déjà son support. La proposition suivante permet de le déterminer :

Proposition 3. *Comptage par inférence.*

P est un motif non clé si et seulement si

$$\sup(P) = \min_{p \in P}(\sup(P \setminus \{p\})).$$

Démonstration. Si l'égalité est vraie, P est évidemment non clé ; il faut donc montrer que tout motif non clé vérifie cette égalité. « $\leq$ » vient de la décroissance du support. « $\geq$ » : si P n'est pas un motif clé, alors il existe $q \in P$ tel que $P\theta(P \setminus \{q\})$. Donc, $\sup(P) = \sup(P \setminus \{q\}) \geq \min_{p \in P}(\sup(P \setminus \{p\}))$. □

Lors de la lecture de la base de données, on n'a donc à compter que le support des motifs clés candidats. C'est ce que fait l'algorithme PASCAL : il fonctionne comme Apriori, mais ne compte pas le support des motifs dont on sait qu'ils ne sont pas clés.

À chaque niveau, on réduit ainsi la phase (coûteuse) de comptage. De plus, à partir d'un certain niveau, tous les candidats peuvent être des candidats non clés : l'algorithme n'a donc plus à accéder à la base de données pour déterminer les motifs fréquents et leur support. Dans le pire des cas, c'est-à-dire si les données sont faiblement corrélées, tous les candidats sont clés, et PASCAL se comporte comme Apriori. Contrairement à Close, il n'y a pas ici d'opération coûteuse dans une boucle critique.

3.2. *Présentation de l'algorithme*

La preuve de l'algorithme s'obtient facilement à l'aide des théorèmes énoncés dans la section précédente. Les notations utilisées sont présentées dans le tableau 2 et le pseudo-code dans les algorithmes 1 et 2.

k	Numéro de l'itération actuelle
C_k	Candidats de taille k
F_k	Motifs fréquents de taille k
p.sup	Support de p
p.*key*	Vrai si p est clé (ou clé potentielle)
c.pred_sup	Minimum des supports des $(k-1)$-sous-ensembles du candidat c

Tableau 2. PASCAL *: notations*

L'algorithme principal traite d'abord l'ensemble vide. Par définition, son support est 1 et il est clé (lignes 1 et 2). Les 1-motifs fréquents sont ensuite déterminés par une première passe sur la base de données, et marqués comme motifs clés à moins d'être dans la classe d'équivalence de l'ensemble vide (lignes 3 à 6). La boucle principale est similaire à celle de Apriori (lignes 7 à 25). Pascal-Gen est d'abord appelé pour créer

```
1: ∅.sup ← 1 ; ∅.key ← true
2: F0 ← {∅}
3: F1 ← {1-motifs fréquents}
4: pour tout P ∈ F1 faire
5:    P.pred_sup ← 1 ; P.key ← (P.sup ≠ 1)
6: fin pour
7: pour (k ← 2 ; Fk−1 ≠ ∅ ; k ← k+1) faire
8:    Ck ← Pascal-Gen(Fk−1)
9:    si ∃C ∈ Ck | C.key alors
10:      pour tout o ∈ D faire
11:         Co ← subset(Ck, o)
12:         pour tout C ∈ Co | C.key faire
13:            C.sup ← C.sup+1
14:         fin pour
15:      fin pour
16:   fin si
17:   pour tout C ∈ Ck faire
18:      si C.sup ≥ minsup alors
19:         si C.key et C.sup = c.pred_sup alors
20:            C.key ← false
21:         fin si
22:         Fk ← Fk ∪ {C}
23:      fin si
24:   fin pour
25: fin pour
26: renvoyer ⋃k>0 Fk
```

Algorithme 1. PASCAL

les motifs candidats. Le support des motifs clés est ensuite déterminé par un accès à la base de données (lignes 9 à 16). L'étape suivante consiste à sélectionner les motifs fréquents (lignes 17 à 24). Parmi eux, les candidats qui s'avèrent non clés voient leur statut mis à jour (lignes 19 à 21).

Pascal-Gen fonctionne pratiquement comme Apriori-Gen. À l'itération k, les candidats sont engendrés à partir des motifs fréquents de l'étape $k-1$. En plus de la jointure et de l'élagage effectués par Apriori-Gen, on propage le fait qu'un candidat soit clé ou non en utilisant la proposition 3 (lignes 10 à 12), et on détermine le support des motifs non clé (lignes 15 à 17).

Exemple 2. L'exécution de PASCAL sur la base de données $\mathcal{D}$ est présentée sur la figure 2. La figure 3 indique les classes d'équivalences et les motifs clés sur le treillis des parties de $\mathcal{D}$.

La première passe de PASCAL donne les motifs candidats de taille 1. Tous sont des motifs clés puisqu'aucun n'a un support de 100 %. Les candidats de la deuxième itération sont donc tous présumés clés. Après lecture de la base de donnée, il s'avère que $\{a,c\}$ et $\{b,e\}$ ne sont pas des motifs clés : le support de $\{a,c\}$ est le même

```
1: insert into C
   select F.item_1, ... , F.item_{k-1}, F'.item_{k-1}
   from F_{k-1} F, F_{k-1} F'
   where F.item_1 = F'.item_1, ... , F.item_{k-2} = F'.item_{k-2}, F.item_{k-1} < F'.item_{k-1} ;
2: pour tout C ∈ C faire
3:    C.key ← true ; C.pred_sup ← ∞
4:    pour tout (k−1)-sous-ensemble S de C faire
5:       si S ∉ F_{k-1} alors
6:          C ← C \ {C}
7:          exit for
8:       sinon
9:          C.pred_sup ← min(C.pred_sup, S.sup)
10:         si ¬S.key alors
11:            C.key ← false
12:         fin si
13:      fin si
14:   fin pour
15:   si ¬C.key alors
16:      C.sup ← C.pred_sup
17:   fin si
18: fin pour
19: renvoyer C
```

Algorithme 2. *Pascal-Gen*(F_{k-1})

que celui de $\{a\}$, le support de $\{b,e\}$ est identique à celui de $\{b\}$ (et de $\{e\}$). Par conséquent, aucun candidat de C_3 n'est clé, car chacun d'eux est sur-ensemble de $\{a,c\}$ ou de $\{b,e\}$, et il est inutile de parcourir la base de données pour connaître les supports. De même à la dernière étape : le support de $\{a,b,c,e\}$ est égal à celui de $\{a,b,c\}$ (ou $\{a,b,e\}$, ou $\{a,c,e\}$).

PASCAL recherche le support de 11 motifs dans la base de données en deux passes. Sur cet exemple, Apriori rechercherait 16 supports en 4 passes, et Close, 9 supports en deux passes. La différence entre Close et PASCAL vient de ce que Close détermine que $\{a,c\}$ et $\{b,e\}$ ne sont pas candidats générateurs, puisque $h(\{a\}) = \{a,c\}$ et $h(\{b\}) = h(\{e\}) = \{b,e\}$. PASCAL ne sait pas qu'ils ne sont pas clés avant d'avoir calculé leur support.

4. Simplification des règles d'association

Une règle d'association est une règle d'implication probabiliste, disant par exemple « 71,1 % des clients qui achètent tels produits achètent aussi tels autres ». Ou « 99,07 % des habitants blancs du Kansas sont américains de naissance et sont nés aux États-Unis ».

Balayage de $\mathcal{D}$ $\longrightarrow$ Fréquents

F_1

Motif	*key*	sup
$\{a\}$	oui	3/6
$\{b\}$	oui	5/6
$\{c\}$	oui	5/6
$\{e\}$	oui	5/6

Génération des candidats $\longrightarrow$

C_2

Motif	pred_sup	*key*	sup
$\{a,b\}$	3/6	oui	?
$\{a,c\}$	3/6	oui	?
$\{a,e\}$	3/6	oui	?
$\{b,c\}$	5/6	oui	?
$\{b,e\}$	5/6	oui	?
$\{c,e\}$	5/6	oui	?

Balayage de $\mathcal{D}$ $\longrightarrow$ Fréquents

F_2

Motif	*key*	sup
$\{a,b\}$	oui	2/6
$\{a,c\}$	non	3/6
$\{a,e\}$	oui	2/6
$\{b,c\}$	oui	4/6
$\{b,e\}$	non	5/6
$\{c,e\}$	oui	4/6

Génération des candidats $\longrightarrow$

C_3

Motif	pred_sup	*key*	sup
$\{a,b,c\}$	2/6	non	2/6
$\{a,b,e\}$	2/6	non	2/6
$\{a,c,e\}$	2/6	non	2/6
$\{b,c,e\}$	4/6	non	4/6

$\longrightarrow$ Fréquents

F_3

Motif	*key*	sup
$\{a,b,c\}$	non	2/6
$\{a,b,e\}$	non	2/6
$\{a,c,e\}$	non	2/6
$\{b,c,e\}$	non	4/6

Génération des candidats $\longrightarrow$

C_4

Motif	pred_sup	*key*	sup
$\{a,b,c,e\}$	2/6	non	2/6

$\longrightarrow$ Fréquents

F_4

Motif	*key*	sup
$\{a,b,c,e\}$	non	2/6

Figure 2. *Exemple d'exécution de* PASCAL

4.1. *Définitions*

Soit une base de données $(\mathcal{O},\mathcal{P},\mathcal{R})$. Soient A et B deux sous-ensembles non vides de $\mathcal{P}$.

Définition 8. Règle d'association.

$A \xrightarrow{v} B$ est une règle d'association entre A et B de précision v.

A est appelé antécédent, prémisse ou partie gauche de la règle ; B est appelé conséquent, conclusion ou partie droite.

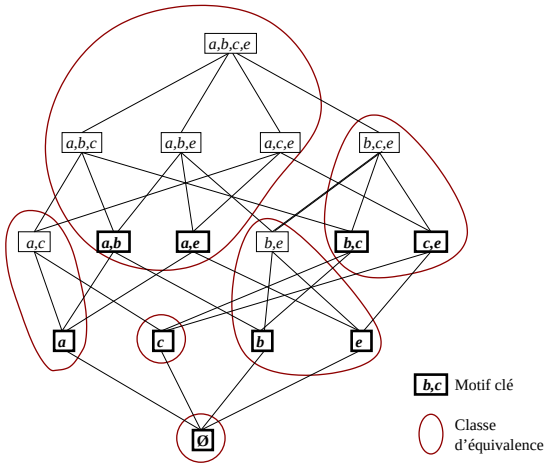

Figure 3. *Classes d'équivalences de* $\mathcal{D}$

La précision d'une règle d'association est sa confiance, définie comme suit :

$$v = \text{conf}(A \rightarrow B) = \frac{\sup(A \cup B)}{\sup(A)}.$$

De plus, le support d'une règle d'association est défini comme le support de l'union de ses deux parties :

$$\sup(A \rightarrow B) = \sup(A \cup B).$$

Comme le montrent ces deux définitions, on n'indique généralement pas la précision au-dessus du symbole d'implication.

Définition 9. Règle d'association valide.

Une règle d'association est dite « valide » si son support et sa confiance sont supérieurs ou égaux à deux seuils, *minsup* et *minconf* respectivement.

Définition 10. Règle d'association exacte.

Une règle d'association est dite « exacte » si sa précision est 100 % ; autrement dit, si les supports de A et $A \cup B$ sont égaux.

Définition 11. Règle d'association approximative.

Une règle d'association est dite « approximative » si sa précision est inférieure à 100 %.

4.2. *Algorithme classique de détermination des règles d'association*

Cette section présente l'algorithme de génération des règles d'association déterminé par Agrawal et Srikant et généralement utilisé [AGR 94]. Soient les seuils *minsup* et *minconf* définis par l'utilisateur ; comme indiqué auparavant, une règle d'association est valide si son support est supérieur ou égal à *minsup*, et sa confiance supérieure ou égale à *minconf*. Si la règle est écrite sous la forme $P_1 \rightarrow P_2 \setminus P_1$, avec $P_1 \subset P_2$, le support de la règle est $\sup(P_2)$. On voit donc que P_2 doit être un motif fréquent.

Par ailleurs, la validité d'une règle $P_1 \rightarrow P_2 \setminus P_1$ entraîne celle des règles $P_1' \rightarrow P_2 \setminus P_1'$, où $P_1 \subset P_1' \subset P_2$ (de par la décroissance du support). Par exemple, si $\{a,b\} \rightarrow \{c,d\}$, alors $\{a,b,c\} \rightarrow \{d\}$ et $\{a,b,d\} \rightarrow \{c\}$. Pour découvrir les règles, il suffit de retourner cette observation : après avoir déterminé que les confiances de $\{a,b,c\} \rightarrow \{d\}$ et $\{a,b,d\} \rightarrow \{c\}$ sont suffisantes, l'algorithme va combiner ces règles et tester $\{a,b\} \rightarrow \{c,d\}$. Comme dans la découverte des motifs fréquents, une méthode par niveaux est utilisée.

Exemple 3. Le tableau 3 rappelle les motifs fréquents de la base de données $\mathcal{D}$ pour $minsup = 2/6$; en prenant $minconf = 2/5$, nous trouvons 50 règles d'association dont 14 sont exactes (pour des contraintes de place, ces règles n'ont pas été indiquées ici).

F_1	
Motif	sup.
$\{a\}$	3/6
$\{b\}$	5/6
$\{c\}$	5/6
$\{e\}$	5/6

F_2	
Motif	sup.
$\{a,b\}$	2/6
$\{a,c\}$	3/6
$\{a,e\}$	2/6
$\{b,c\}$	4/6
$\{b,e\}$	5/6
$\{c,e\}$	4/6

F_3	
Motif	sup.
$\{a,b,c\}$	2/6
$\{a,b,e\}$	2/6
$\{a,c,e\}$	2/6
$\{b,c,e\}$	4/6

F_4	
Motif	sup.
$\{a,b,c,d,e\}$	2/6

Tableau 3. *Motifs fréquents de $\mathcal{D}$ pour $minsup = 2/6$*

4.3. *Critique des règles d'associations*

Le nombre de règles d'associations est généralement énorme. De plus, il existe souvent des règles de même support et même confiance qui s'avèrent redondantes. Dans la base de données MUSHROOMS [5], par exemple, les règles de support 28,850 % et de confiance 51,165 % [6] sont les suivantes :

1) Pied lisse sous l'anneau, un anneau → comestible, inodore, lamelles libres.

2) Pied lisse sous l'anneau, un anneau → comestible, inodore, voile blanc.

5. MUSHROOMS est une des base de données utilisées pour les expérimentations.
6. Le problème est particulièrement frappant pour ces valeurs, mais il existe partout.

3) Pied lisse sous l'anneau, voile partiel, un anneau → comestible, inodore, lamelles libres.

4) Pied lisse sous l'anneau, un anneau → comestible, inodore, lamelles libres, voile partiel.

5) Pied lisse sous l'anneau, un anneau → comestible, inodore, lamelles libres, voile blanc.

6) Pied lisse sous l'anneau, voile partiel, un anneau → comestible, inodore, voile blanc.

7) Pied lisse sous l'anneau, un anneau → comestible, inodore, voile partiel, voile blanc.

8) Pied lisse sous l'anneau, voile partiel, un anneau → comestible, inodore, lamelles libres, voile blanc.

9) Pied lisse sous l'anneau, un anneau → comestible, inodore, lamelles libres, voile partiel, voile blanc.

De toutes ces règles de mêmes support, confiance et items en jeu, seule la dernière est nécessaire. Une règle dont la partie droite est plus petite est inutile, puisqu'elle apporte moins d'informations. Une règle dont la partie gauche est plus grande est également inutile (elle laisse croire que plus d'items sont nécessaires qu'en réalité).

4.4. *Règles d'association informatives*

Parmi plusieurs règles d'association de mêmes support et confiance, la seule qu'il faille produire est donc celle dont l'antécédent est le plus petit et le conséquent le plus grand. C'est ce type de règles que nous appellerons règles d'association informatives. Nous allons maintenant voir séparément comment construire les règles qui vérifient ces deux contraintes.

Minimalité de l'antécédent

Soient deux règles $r : P_1 \rightarrow (P_2 \setminus P_1)$ et $r' : P_1' \rightarrow (P_2 \setminus P_1')$, avec $\sup(r) = \sup(r')$, $\mathrm{conf}(r) = \mathrm{conf}(r')$ et $P_1' \subset P_1$. Les égalités du support et de la confiance permettent d'écrire

$$\sup(P_1) = \sup(P_1').$$

Nous avons vu dans la section 3 (proposition 1) que les conditions $P_1' \subset P_1$ et $\sup(P_1) = \sup(P_1')$ impliquent que P_1 et P_1' sont dans la même classe d'équivalence : $P_1' \in [P_1]$. Dans ces conditions, si P_1' est minimal, c'est un motif clé de $[P_1]$.

Maximalité du conséquent

Soient deux règles $r : P_1 \rightarrow (P_2 \setminus P_1)$ et $r' : P_1 \rightarrow (P_2' \setminus P_1)$, avec $\sup(r) = \sup(r')$, $\mathrm{conf}(r) = \mathrm{conf}(r')$ et $P_2 \subset P_2'$. Nous avons donc

$$\sup(P_2) = \sup(P_2').$$

Ainsi, P_2 et P_2' font partie de la même classe d'équivalence ; et P_2' est maximal s'il est l'élément maximal de cette classe. Une classe d'équivalence n'a en effet qu'un seul élément maximal, comme le montre la proposition 4.

Proposition 4.

1) Une classe d'équivalence contient tous les motifs de même fermeture.

2) Une classe d'équivalence a un unique plus grand élément, qui est le fermé.

Démonstration.

1) Soient P et P' deux motifs. $f(P) = f(P') \implies f \circ g(P) = f \circ g(P')$. Réciproquement, $h(P) = h(P') \implies f \circ h(P) = f \circ h(P')$; et, puisque $f \circ h = f \circ g \circ f = f$, $f(P) = f(P')$.

2) Puisque les éléments d'une classe d'équivalence contiennent les motifs de même fermeture, on peut utiliser la propriété d'extensivité de la fermeture : $\forall P$, $P \subseteq h(P)$; $h(P)$ est donc le plus grand élément de $[P]$. □

Nous venons donc de montrer comment construire les règles avec un antécédent minimal et un conséquent maximal : elles sont de la forme

$$r : P \rightarrow P' \setminus P \quad \text{avec} \quad \begin{cases} P \text{ est clé} \\ P' = \max[P'] \\ P \subset P'. \end{cases}$$

Dans cette définition, P et P' sont deux motifs fréquents, P étant une clé et P' le maximum d'une classe d'équivalence « supérieure ou égale » à celle de P sur le treillis des motifs fréquents. Si la classe d'équivalence de P' est la même que celle de P, la règle d'association est exacte puisque $\sup(P) = \sup(P')$. Sinon la règle est approximative.

Exemple 4. Une illustration graphique est sans doute de rigueur. La figure 4 rappelle l'ensemble des motifs fréquents et des classes d'équivalences sur la base de données exemple $\mathcal{D}$ avec *minsup* $= 2/6$, et montre les règles informatives valides pour *minconf* $= 2/5$. Ces règles sont reprises dans les tableaux 4 et 5. Les règles informatives exactes sont celles qui restent dans une classe d'équivalence, comme $\{a\} \rightarrow \{c\}$. Les règles informatives approximatives sont inter-classes : par exemple, $\{c\} \rightarrow \{a\}$. Nous voyons que le nombre de règles exactes est passé de 14 à 7, et le nombre de règles approximatives de 36 à 10.

Réduction par transitivité

Les règles d'associations informatives telles qu'elles ont été définies jusqu'à présent sont correctes, on le voit aisément, et répondent au « cahier des charges » que nous nous étions fixé. Pourtant, nous allons encore réduire leur nombre ; mais cette fois en éliminant des règles de support et confiance différents, et en montrant qu'elles

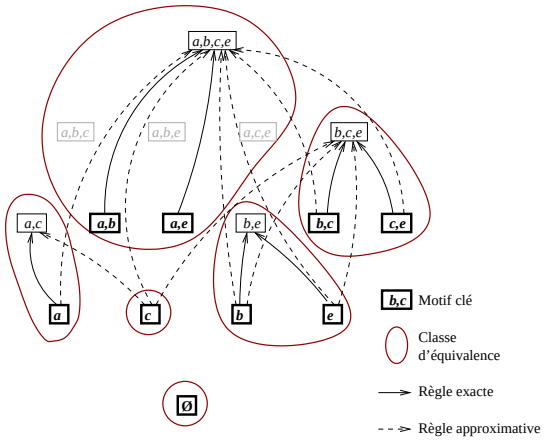

Figure 4. *Règles informatives dans $\mathcal{D}$*

Clé P	max$[P]$	support	confiance	règle
$\{a\}$	$\{a,c\}$	3/6	100 %	$\{a\} \rightarrow \{c\}$
$\{b\}$	$\{b,e\}$	5/6	100 %	$\{b\} \rightarrow \{e\}$
$\{c\}$	$\{c\}$			
$\{e\}$	$\{b,e\}$	5/6	100 %	$\{e\} \rightarrow \{b\}$
$\{a,b\}$	$\{a,b,c,e\}$	2/6	100 %	$\{a,b\} \rightarrow \{c,e\}$
$\{a,e\}$	$\{a,b,c,e\}$	2/6	100 %	$\{a,e\} \rightarrow \{b,c\}$
$\{b,c\}$	$\{b,c,e\}$	4/6	100 %	$\{b,c\} \rightarrow \{e\}$
$\{c,e\}$	$\{b,c,e\}$	4/6	100 %	$\{c,e\} \rightarrow \{b\}$

Tableau 4. *Règles informatives exactes dans $\mathcal{D}$*

Clé P	max$[P']$	support	confiance	règle
$\{a\}$	$\{a,b,c,e\}$	2/6	2/3	$\{a\} \rightarrow \{b,c,e\}$
$\{b\}$	$\{b,c,e\}$	4/6	4/5	$\{b\} \rightarrow \{c,e\}$
»	$\{a,b,c,e\}$	2/6	2/5	$\{b\} \rightarrow \{c,e\}$
$\{c\}$	$\{a,c\}$	3/6	3/5	$\{c\} \rightarrow \{a\}$
»	$\{b,c,e\}$	4/6	4/5	$\{c\} \rightarrow \{b,e\}$
»	$\{a,b,c,e\}$	2/6	2/5	$\{c\} \rightarrow \{a,b,e\}$
$\{e\}$	$\{b,c,e\}$	4/6	4/5	$\{e\} \rightarrow \{b,c\}$
»	$\{a,b,c,e\}$	2/6	2/5	$\{e\} \rightarrow \{a,b,c\}$
$\{b,c\}$	$\{a,b,c,e\}$	2/6	2/4	$\{b,c\} \rightarrow \{a,e\}$
$\{c,e\}$	$\{a,b,c,e\}$	2/6	2/4	$\{c,e\} \rightarrow \{a,b\}$

Tableau 5. *Règles informatives approximatives dans $\mathcal{D}$*

sont retrouvables. Dans l'exemple qui vient d'être vu, il y avait les règles $\{b\} \rightarrow \{c,e\}$ (de support $4/6$ et de confiance $4/5$) et $\{b\} \rightarrow \{a,c,e\}$ (de support $2/6$ et de confiance $2/5$). Seule la première sera engendrée, au moins initialement. Pourquoi ? Parce que moins il existe de règles, mieux l'utilisateur peut les visualiser et les utiliser ; et sauf rares exceptions, les règles les plus intéressantes sont celles de support et confiance supérieures. Mathématiquement, comme nous allons le démontrer, nous ne perdons pas d'informations (la deuxième règle se déduit de la première).

Les règles exactes produites restent les mêmes – puisque le but est de garder des supports et confiances maximaux. Par contre, les règles approximatives informatives réduites seront de la forme :

$$r : P_1 \rightarrow P_2 \setminus P_1 \quad \text{avec} \quad \begin{cases} P_1 \text{ est clé} \\ P_2 = \max[P_2] \\ P_1 \subset P_2 \\ \nexists r' : P_1 \rightarrow P_2' \setminus P_1 \text{ informative réduite } \mid P_2' \subset P_2. \end{cases}$$

Autrement dit, la conclusion de la règle provient d'une classe d'équivalence « directement supérieure » (sur le diagramme de Hasse) à sa prémisse [7]. Le théorème suivant prouve que ces règles informatives forment une famille génératrice de l'ensemble des règles d'association, ou, en langage simple, que toutes les règles d'association (avec leurs supports et confiances) peuvent en être déduites.

Lemme 5. *Transitivité de la confiance [LUX 91].*

Soient trois motifs fermés $P_1, P_2, P_3 \in \mathcal{P}$ tels que $P_1 \subseteq P_2$ et $P_2 \subseteq P_3$. On a alors :

$$\text{conf}(P_1 \rightarrow P_2) \times \text{conf}(P_2 \rightarrow P_3) = \text{conf}(P_1 \rightarrow P_3).$$

Théorème 6. *L'ensemble des règles informatives approximatives réduites par transitivité forme une famille génératrice : toutes les règles approximatives valides dans le contexte peuvent en être déduites avec leurs support et confiance.*

Démonstration. Les règles d'association informatives forment clairement une famille génératrice. Les règles d'association informatives réduites sont obtenues en supprimant les transitivités, qui peuvent être retrouvées grâce au lemme 5. □

Le support des règles d'associations retrouvées ainsi est simplement le minimum des deux supports. Par exemple, si l'on prend la règle d'association $\{c\} \rightarrow \{b,e\}$ (de support $4/6$ et de confiance $4/5$) et soit la règle $\{b,c\} \rightarrow \{a,e\}$, soit la règle $\{c,e\} \rightarrow \{a,b\}$ (toutes deux de support $2/6$ et de confiance $2/4$), on en déduit $\{c\} \rightarrow \{a,b,e\}$ (support $2/6$, confiance $2/5$).

La figure 5 et le tableau 6 montrent les règles informatives approximatives réduites dans le même cas que précédemment : il y en a sept.

7. Plus exactement dans le treillis des maximaux des classes d'équivalence.

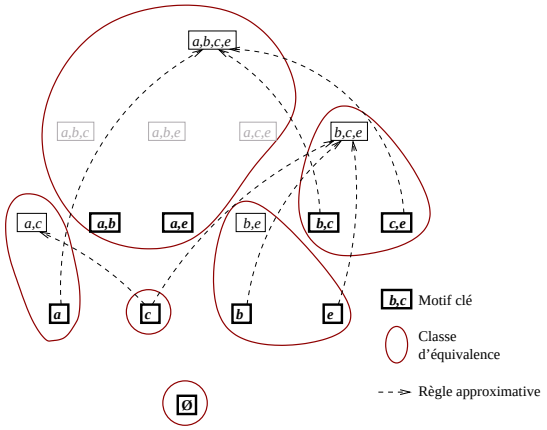

Figure 5. *Règles informatives approximatives réduites dans $\mathcal{D}$*

Clé P	max$[P']$	support	confiance	règle
$\{a\}$	$\{a,b,c,e\}$	$2/6$	$2/3$	$\{a\} \rightarrow \{b,c,e\}$
$\{b\}$	$\{b,c,e\}$	$4/6$	$4/5$	$\{b\} \rightarrow \{c,e\}$
$\{c\}$	$\{a,c\}$	$3/6$	$3/5$	$\{c\} \rightarrow \{a\}$
»	$\{b,c,e\}$	$4/6$	$4/5$	$\{c\} \rightarrow \{b,e\}$
$\{e\}$	$\{b,c,e\}$	$4/6$	$4/5$	$\{e\} \rightarrow \{b,c\}$
$\{b,c\}$	$\{a,b,c,e\}$	$2/6$	$2/4$	$\{b,c\} \rightarrow \{a,e\}$
$\{c,e\}$	$\{a,b,c,e\}$	$2/6$	$2/4$	$\{c,e\} \rightarrow \{a,b\}$

Tableau 6. *Règles informatives approximatives réduites dans $\mathcal{D}$*

4.5. *Génération des règles d'association informatives*

La génération des règles d'association informatives utilise les motifs fréquents déterminés auparavant. Les deux algorithmes qui suivent renvoient l'un les règles informatives exactes, l'autre les règles informatives approximatives réduites, à partir des motifs clés et des motifs maximaux des classes d'équivalence.

4.5.1. *Génération des règles informatives exactes*

L'algorithme 3 calcule l'ensemble R_E des règles informatives exactes. Nous ne présenterons pas ici de méthode pour retrouver toutes les règles exactes ; un tel algorithme comprendrait deux types d'opérations pour une règle de R_E ($\{a,b\} \rightarrow \{c,e\}$ par exemple) :

– ôter des items du conséquent (ce qui donne $\{a,b\} \rightarrow \{c\}$ et $\{a,b\} \rightarrow \{e\}$) ;

– déplacer des items du conséquent dans l'antécédent (d'où $\{a,b,c\} \rightarrow \{e\}$ et $\{a,b,e\} \rightarrow \{c\}$).

Entrée : K : ensemble des motifs clés fréquents
Sortie : R_E : règles informatives exactes
1: $\mathsf{R}_E \leftarrow \emptyset$
2: **pour tout** $P \in \mathsf{K} \mid P \neq \max[P]$ **faire**
3: $\quad \mathsf{R}_E \leftarrow \mathsf{R}_E \cup \{ r : P \rightarrow \max[P] \setminus P \}$
4: **fin pour**
5: **renvoyer** R_E

Algorithme 3. *Génération des règles informatives exactes*

4.5.2. *Génération des règles informatives approximatives réduites*

Le pseudo-code de l'algorithme 4 montre comment calculer l'ensemble R_A des règles informatives approximatives réduites, à partir des motifs clés fréquents. Cet algorithme montre une possibilité pour ne traiter que les motifs maximaux successeurs immédiats d'un $\max[P]$: prendre tous ses motifs successeurs, les trier par taille et ôter à mesure les successeurs de celui que l'on traite.

Entrée : K : ensemble des motifs clés fréquents
$\quad$ *minconf* : confiance minimale
Sortie : R_A : ensemble des règles informatives approximatives réduites
1: $\mathsf{R}_A \leftarrow \emptyset$
2: **pour tout** $P_1 \in \mathsf{K}$ **faire**
3: $\quad \mathsf{C} \leftarrow \{ \max[P'] \mid P' \in \mathsf{K} \wedge \max[P_1] \subset \max[P'] \}$
4: $\quad$ **pour tout** motif $P_2 \in \mathsf{C}$ triés par cardinalité croissante **faire**
5: $\quad\quad r : P_1 \rightarrow P_2 \setminus P_1$
6: $\quad\quad$ Ôter de C les motifs incluant P_2
7: $\quad\quad$ **si** $\mathrm{conf}(r) \geq$ *minconf* **alors**
8: $\quad\quad\quad \mathsf{R}_A \leftarrow \mathsf{R}_A \cup \{r\}$
9: $\quad\quad$ **fin si**
10: $\quad$ **fin pour**
11: **fin pour**
12: **renvoyer** R_A

Algorithme 4. *Génération des règles informatives approximatives réduites*

5. Évaluations expérimentales

Nous avons évalué les performances de PASCAL par rapport aux trois algorithmes Apriori, Max-Miner et Close, chacun représentatif d'une approche d'extraction différente. L'implantation de Max-Miner pour l'extraction des motifs fréquents maximaux a été fournie pas Roberto Bayardo, et nous avons implanté le calcul des supports de

tous les motifs fréquents (dernier balayage)[8]. PASCAL, Apriori, Close et cette phase finale pour Max-Miner utilisent les même structures de données et techniques d'implantation. Les optimisations telles que la gestion particulière de la deuxième itération et le réordonnancement des items n'ont pas été utilisées.
REMARQUE. — Les complexités théoriques des différents algorithmes n'ont pas été comparées, pour plusieurs raisons. D'une part, ce sont les mêmes au pire des cas. Le lecteur intéressé par la complexité d'Apriori (et donc de Pascal) peut se reporter à une étude de J. L. Han et A. W. Plank [HAN 96]. Mais surtout cette complexité est totalement dominée par le coût des accès aux disques, pour lequel il n'existe pas, à notre connaissance, de modélisation réaliste.

Les caractéristiques des jeux de données utilisés pour les expérimentations sont présentées dans le tableau 7. Ce sont : les jeux de données synthétiques T20I6D100K, T25I10D10K et T25I20D100K[9], construits selon les propriétés des données de ventes, les jeux de données de recensement C20D10K et C73D10K, extraits du fichier « PUMS sample file »[10], et le jeu de données MUSHROOMS[11] décrivant les caractéristiques de champignons [BAY 99a]. Pour toutes les expérimentations, nous avons tenté de choisir des seuils minimaux de supports significatifs.

Nom	Nombre d'objets	Taille moyenne des objets	Nombre d'items
T20I6D100K	100 000	20	1 000
T25I10D10K	10 000	25	1 000
T25I20D100K	100 000	25	10 000
C20D10K	10 000	20	386
C73D10K	10 000	73	2 178
MUSHROOMS	8 416	23	128

Tableau 7. *Jeux de données*

5.1. *Extraction des motifs fréquents*

Des travaux antérieurs ont démontré que le comportement des algorithmes d'extraction des motifs fréquents varie fortement selon les caractéristiques des jeux de données utilisés. Les données faiblement corrélées, telles que les données synthétiques, constituent des cas faciles pour l'extraction car peu de motifs sont fréquents. Pour de telles données, tous les algorithmes donnent des temps de réponse acceptables, comme

8. Dans les tableaux présentés, nous distinguons les temps d'exécution de Max-Miner et ceux du calcul des supports de tous les motifs fréquents. Sur les figures, ces deux opérations sont réunies (« Max-Miner$^+$ »).
9. `http://www.almaden.ibm.com/cs/quest/syndata.html`
10. `ftp://ftp2.cc.ukans.edu/pub/ippbr/census/pums/pums90ks.zip`
11. `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/mushroom/agaricus-lepiota.data`

nous pouvons l'observer dans la section 5.1.1 où sont présentés les résultats obtenus pour T20I6D100K, T25I10D10K et T25I20D100K. Au contraire, les données corrélées constituent des cas bien plus difficiles du fait de l'importante proportion de motifs fréquents. De plus, ces données représentent une part importante des jeux de données réels et les temps de réponse obtenus varient fortement selon l'algorithme utilisé. Les résultats expérimentaux obtenus pour C20D10K, C73D10K et MUSHROOMS, qui sont constitués de données corrélées, sont présentés dans la section 5.1.2.

5.1.1. *Données faiblement corrélées*

Les jeux de données synthétiques T20I6D100K, T25I10D10K et T25I20D100K sont construits selon les propriétés des données de ventes qui sont typiquement faiblement corrélées. Dans ces derniers, le nombre de motifs fréquents est faible comparé au nombre total de motifs et, dans la plupart des cas, presque tous les motifs fréquents sont également des motifs clés.

Les temps de réponse pour le jeu de données T20I6D100K sont présentés numériquement dans le tableau 8 et graphiquement dans la figure 6. Pour ce jeu de données, tous les motifs fréquents sont des motifs clés et Apriori et PASCAL se comportent de manière identique. Les temps de réponse obtenus avec Apriori, PASCAL et Max-Miner sont très proches. L'algorithme Close donne des temps de réponse supérieurs dus au nombre d'opérations d'intersection nécessaires pour calculer les fermetures des motifs candidats.

Support	Fréquents	PASCAL	Apriori	Close	Max-Miner	
1,00	1 534	13,14	13,51	25,91	2,60	5,03
0,75	4 710	20,41	20,67	35,29	4,44	11,06
0,50	26 950	44,00	44,38	67,82	6,87	35,37
0,25	155 673	117,97	117,79	182,95	15,64	109,14

Tableau 8. *Temps de réponse pour T20I6D100K*

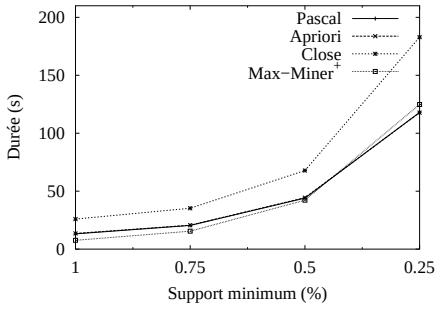

Figure 6. *Résultats expérimentaux pour T20I6D100K*

Support	Fréquents	PASCAL	Apriori	Close	Max-Miner	
1,00	583	5,15	5,76	11,15	1,24	1,3
0,75	1 155	9,73	11,13	35,67	1,99	1,77
0,50	1 279 254	968,64	935,14	2 151,34	24,94	879,85

Tableau 9. *Temps de réponse pour T25I20D100K*

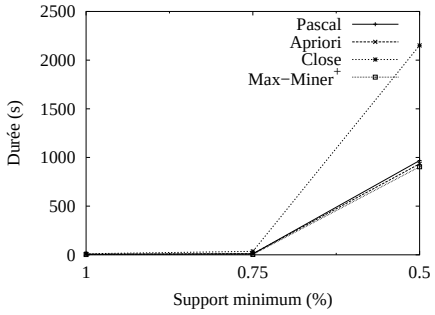

Figure 7. *Résultats expérimentaux pour T25I20D10K*

Support	Fréquents	PASCAL	Apriori	Close	Max-Miner	
1,00	3 300	3,24	3,62	6,67	0,63	1,05
0,75	17 583	5,17	6,95	9,38	1,09	3,83
0,50	331 280	17,82	41,06	26,43	2,76	35,53
0,25	2 270 573	70,37	187,92	86,08	6,99	154,89

Tableau 10. *Temps de réponse pour T25I10D10K*

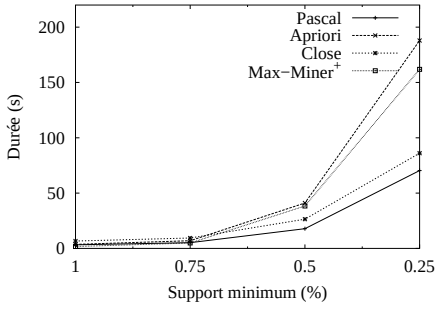

Figure 8. *Résultats expérimentaux pour T25I10D10K*

Les résultats obtenus pour T25I20D100K sont présentés numériquement dans le tableau 9 et graphiquement dans la figure 7. Pour ce jeu de données, presque tous les motifs fréquents sont des motifs clés et les résultats sont similaires à ceux obtenus pour T20I6D100K : PASCAL et Apriori donnent des temps de réponse identiques et légèrement supérieurs à ceux de Max-Miner. Les temps de réponse de Close sont les plus élevés.

Les temps d'exécution pour T25I10D10K sont présentés dans le tableau 10 et la figure 8. Dans ce jeu de données, la proportion de motifs fréquents qui ne sont pas clés est bien plus importante que pour T25I20D100K. Pour les seuils *minsup* de 1,00 % et 0,75 %, Max-Miner donne de meilleurs temps de réponses que Apriori et PASCAL qui donnent eux-mêmes des temps inférieurs à ceux de Close. Pour les seuils de 0,50 % et 0,25 %, PASCAL devient le plus efficace et est légèrement meilleurs que Close, alors que tous deux donnent des temps inférieurs à Apriori et Max-Miner. Lorsque la proportion de motifs fréquents qui ne sont pas clés est significative, le mécanisme utilisé par PASCAL (resp. Close) pour considérer seulement les motifs clés (resp. fermés) permet de réduire considérablement le nombre de calculs de supports réalisés.

5.1.2. *Données corrélées*

Les temps de réponse obtenus pour les jeux de données de recensement C20D10K et C73D10K sont présentés numériquement dans les tables 11 et 12, et graphiquement dans les figures 9 et 10. Les résultats pour le jeu MUSHROOMS sont présentés dans la table 13 et la figure 11. Dans ces trois jeux de données, constitués de données corrélées, la proportion de motifs qui sont fréquents est importante mais peu de motifs fréquents sont aussi des motifs clés. En conséquence, utilisant le comptage par inférence, PASCAL doit réaliser moins de calculs de supports que Apriori et Max-Miner. La même observation se vérifie pour Close, qui utilise le mécanisme de fermeture pour réduire le nombre de calculs de supports, et PASCAL et Close sont tous deux bien plus rapides que Apriori et Max-Miner. Les différences entre les temps de réponse de PASCAL et Close et ceux de Apriori et Max-Miner se mesurent en dizaines de minutes pour C20D10K et MUSHROOMS et en heures pour C73D10K. De plus, le comptage par inférence et le mécanisme de fermeture permettent de réduire le nombre de balayages du jeu de données réalisés car les support de tous les candidats de certaines itérations sont déduits de ceux des motifs clés ou fermés des itérations précédentes. Pour C73D10K avec $minsup = 60$ % par exemple, PASCAL et Close réalisent 13 balayages alors que les motifs fréquents les plus grands contiennent 19 items – soit 19 balayages pour Apriori. Pour ce jeu de données et ce seuil, il n'a pas été possible de calculer les supports de tous les motifs fréquents (ultime étape de Max-Miner) car la gestion de la mémoire n'était pas implantée et cette étape nécessitait plus d'espace mémoire que disponible.

5.2. *Extraction des règles d'association*

Les gains de taille apportés par l'usage des règles informatives définies dans la section 4 ont été testés sur différentes bases de données : T10I4D100K, C20D10K,

Support	Fréquents	PASCAL	Apriori	Close	Max-Miner	
20,0	20 239	9,44	57,15	14,36	0,17	77,40
15,0	36 359	12,31	85,35	18,99	0,26	113,22
10,0	89 883	19,29	164,81	29,58	0,34	201,33
7,5	153 163	23,53	232,40	36,02	0,35	268,80
5,0	352 611	33,06	395,32	50,46	0,48	428,65
2,5	1,160 363	55,33	754,64	78,63	0,81	775,56

Tableau 11. *Temps de réponse pour C20D10K*

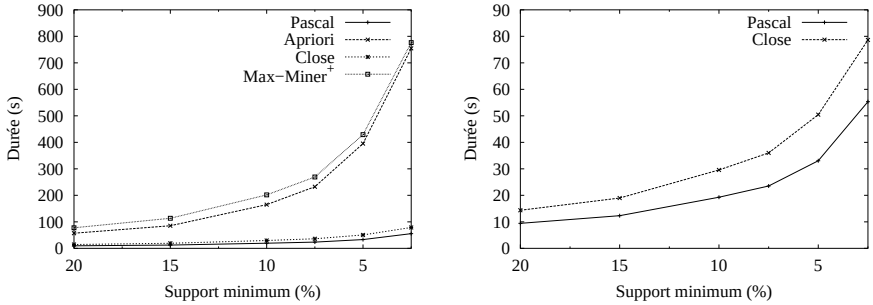

Figure 9. *Résultats expérimentaux pour C20D10K*

Support	Fréquents	PASCAL	Apriori	Close	Max-Miner	
80	109 159	177,49	3 661,27	241,91	0,87	3 717,99
75	235 271	392,80	7 653,58	549,27	1,06	7 730,36
70	572 087	786,49	17 465,10	1 112,42	2,28	17 618,40
60	4 355 543	3 972,10	109 204,00	5 604,91	7,72	

Tableau 12. *Temps de réponse pour C73D10K*

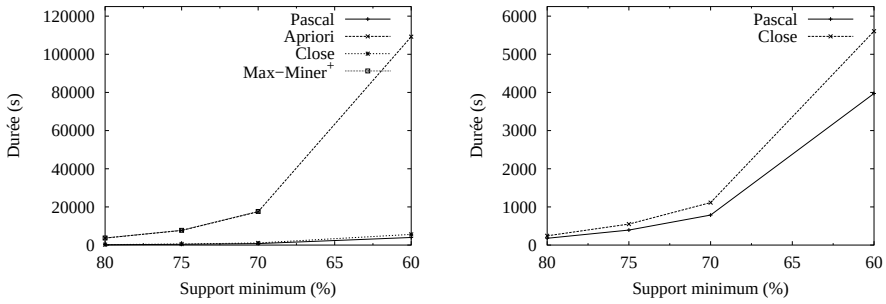

Figure 10. *Résultats expérimentaux pour C73D10K*

Support	Fréquents	PASCAL	Apriori	Close	Max-Miner	
20,0	53 337	6,48	115,82	9,63	0,31	134,31
15,0	99 079	9,81	190,94	14,57	0,50	218,93
10,0	600 817	23,12	724,35	29,83	0,89	745,72
7,5	936 247	32,08	1 023,24	41,05	1,25	1 035,48
5,0	4 140 453	97,12	2 763,42	98,81	1,99	2 752,05

Tableau 13. *Temps de réponse pour* MUSHROOMS

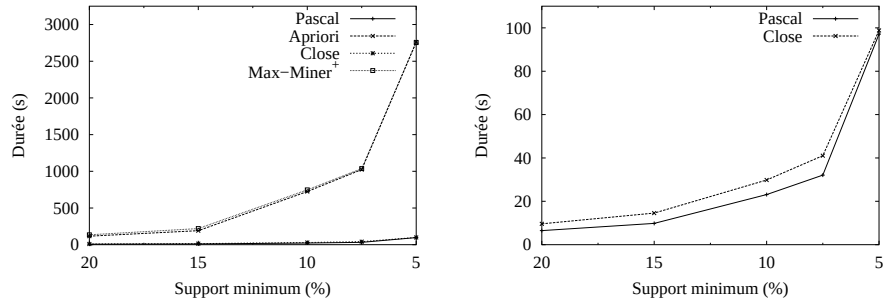

Figure 11. *Résultats expérimentaux pour* MUSHROOMS

C73D10K, et MUSHROOMS, déjà décrits [12]. Les temps d'exécution sont négligeables par rapport au temps de recherche des motifs fréquents, et ne sont donc pas indiqués.

5.2.1. *Règles exactes*

Le nombre total de règles exactes et le nombre de règles exactes informatives sont indiqués pour chaque base de données dans la table 14. Pour ce support minimum, T10I4D100K ne contient aucune règle exacte : tous les motifs fréquents sont maximaux dans leur classe d'équivalence. Les trois autres contextes contiennent un nombre important de règles pour un seuil *minsup* relativement élevé, de 2 000 à 52 000. L'utilisation des règles informatives réduit ce nombre de règles de 12 à 50 fois, sans pour autant perdre d'information.

5.2.2. *Règles approximatives*

Le tableau 15 présente le nombre de règles d'association extraites des mêmes quatre bases avec différents seuils minimaux de confiance. Le nombre total de règles approximatives valides est énorme : il varie de 20 000 pour la base synthétique à plus de 2 000 000 pour C73D10K. Ce tableau illustre bien la réduction apportée par les règles informatives réduites, qui offre des gains d'un facteur de 40 à 500.

12. T10I4D100K est une base de données synthétique de 100 000 objets contenant 10 items en moyenne.

Jeu	*minsup*	Règles exactes	Règles exactes informatives
T10I4D100K	0,5 %	0	0
C20D10K	50 %	2 277	457
C73D10K	90 %	52 035	1 369
MUSHROOMS	30 %	7 476	543

Tableau 14. *Règles d'associations exactes et règles exactes informatives*

Jeu (*minsup*)	*minconf*	Règles approximatives	Règles informatives réduites
T10I4D100K	70 %	20 419	4 004
(0,5 %)	30 %	22 952	4 519
MUSHROOMS	70 %	37 671	1 221
(30 %)	30 %	71 412	1 578
C20D10K	70 %	89 601	1 957
(50 %)	30 %	116 791	1 957
C73D10K	90 %	2 053 896	5 718
(90 %)	80 %	2 053 936	5 718

Tableau 15. *Règles d'associations approximatives et règles informatives réduites*

6. Conclusion

Nous proposons un nouvel algorithme, PASCAL, pour l'extraction efficace des motifs fréquents dans les grandes bases de données ainsi qu'une méthode de simplification des règles d'association. Basé sur une nouvelle optimisation, simple et efficace, de l'algorithme Apriori, il est donc facile à intégrer dans les implantation existante. Cette optimisation, l'inférence de comptage, s'appuie sur les notions de classes d'équivalence des motifs et de motifs clés. Elle permet de calculer depuis la base de données les supports de certains motifs seulement – les motifs clés – contrairement aux algorithmes basés sur l'extraction par niveaux des motifs fréquents ou l'extraction des motifs fréquents maximaux. Nous avons conduit des expérimentations afin de comparer les performances de PASCAL avec celles de Apriori, Max-Miner et Close, chacun représentatif d'une approche d'extraction des motifs fréquents et de leur support. Les résultats démontrent que PASCAL donne des temps de réponses équivalents à ceux de Apriori et Max-Miner, et inférieurs à ceux de Close, dans le cas de données faiblement corrélées, et qu'il est le plus efficace parmi les quatre algorithmes dans le cas de données corrélées. D'autre part, les règles d'association déterminées à l'aide de PASCAL sont les plus informatives parce qu'elles ont un antécédent minimum et un conséquent maximum. Les évaluations expérimentales montrent de plus une réduction importante du nombre de règles produites.

Nous pensons qu'une perspective intéressante de travaux ultérieurs concerne l'intégration du comptage par inférence dans les systèmes de gestion de bases de données. L'intégration des techniques de data mining dans les bases de données relationnelles et objets est un thème de recherche important [SAR 98]. L'implantation de l'algorithme PASCAL en SQL ou OQL peut en outre bénéficier des possibilités d'indexation, de gestion des requêtes, de parallélisation du processus (par exemple dans une optique SMP) et de gestion de l'espace proposés par les SGBD.

Le concept de motifs clés a aussi une application plus générale pour le calcul des systèmes de fermeture et des treillis (algorithme Titanic [STU 00]).

7. Bibliographie

[AGR 93] AGRAWAL R., IMIELINSKI T., SWAMI A., « Mining Association Rules between sets of Items in Large Databases », *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, mai 1993, p. 207-216.

[AGR 94] AGRAWAL R., SRIKANT R., « Fast Algorithms for Mining Association Rules in Large Databases », *Proc. of the 20th Int'l Conf. on Very Large Data Bases (VLDB)*, juin 1994, p. 478-499, version étendue : IBM Research Report RJ 9839.

[AGR 95] AGRAWAL R., SRIKANT R., « Mining Sequential Patterns », *Proc. of the 11th Int'l Conf. on Data Engineering (ICDE)*, mars 1995, p. 3-14.

[BAS 00a] BASTIDE Y., « *Data mining* : algorithmes par niveaux, techniques d'implantation et applications », thèse d'université, université Clermont-Ferrand II, décembre 2000.

[BAS 00b] BASTIDE Y., PASQUIER N., TAOUIL R., STUMME G., LAKHAL L., « Mining minimal non-redundant rules using frequent closed itemsets », *Proc. of the 1st Int'l Conf. on Computational Logic (6th Int'l Conf. on Database Systems – DOOD)*, n° 1861 Lectures Notes in Artificial Intelligence, Springer, juillet 2000, p. 972-986.

[BAS 00c] BASTIDE Y., TAOUIL R., PASQUIER N., STUMME G., LAKHAL L., « Levelwise search of frequent patterns with counting inference », *Actes des 16es journées « Bases de données avancées »*, Blois, octobre 2000, p. 307-322.

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

[BAY 98] BAYARDO R. J., « Efficiently Mining Long Patterns from Databases », *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, juin 1998, p. 85-93.

[BAY 99a] BAY S. D., « The UCI KDD Archive [`http://kdd.ics.uci.edu`] », 1999, Irvine, CA: University of California, Department of Information and Computer Science.

[BAY 99b] BAYARDO R., AGRAWAL R., « Mining the most interesting rules », *Proc. of the 5th Int'l Conf. on Knowledge Discovery and Data Mining (KDD)*, août 1999, p. 145-154.

[BIR 67] BIRKHOFF G., « Lattices Theory », *Coll. Pub. XXV*, vol. 25, American Mathematical Society, 1967, 3e édition.

[BOU 00a] BOULICAUT J.-F., BYKOWSKI A., « Frequent Closures as a Concise Representation for Binary Data Mining », *Proc. of the 4th PAKDD Conf.*, avril 2000, p. 62-73.

[BOU 00b] BOULICAUT J.-F., BYKOWSKI A., RIGOTTI C., « Approximation of frequency queries by mean of free-sets », *Proc. of the 4th PKDD Conf.*, septembre 2000, p. 75-85.

[BRI 97a] BRIN S., MOTWANI R., SILVERSTEIN C., « Beyond Market Baskets: Generalizing Association Rules to Correlation », *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, mai 1997, p. 265-276.

[BRI 97b] BRIN S., MOTWANI R., ULLMAN J. D., TSUR S., « Dynamic Itemset Counting and Implication Rules for Market Basket Data », *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, mai 1997, p. 255-264.

[DUQ 99] DUQUENNE V., « Latticial Structures in Data Analysis », *TCS*, vol. 217, n° 2, 1999, p. 407-436.

[GAN 99] GANTER B., WILLE R., *Formal Concept Analysis: Mathematical Foundations*, Springer, 1999.

[GAR 98] GARDARIN G., PUCHERAL P., WU F., « Bitmap based algorithms for mining association rules », *Actes des 14es journées « Bases de données avancées »*, octobre 1998, p. 157-175.

[GUI 86] GUIGUES J.-L., DUQUENNE V., « Famille minimale d'implication informatives résultant d'un tableau de données binaires », *Math. Sci. Hum.*, vol. 24, n° 95, 1986, p. 5-18.

[HAN 96] HAN J. L., PLANK A. W., « Background for Association Rules and Cost Estimate of Selected Mining Algorithms », *Proc. of the 5th Int'l CIKM*, novembre 1996, p. 73-80.

[HAN 00a] HAN J., KAMBER M., *Data Mining : Concepts and Techniques*, Morgan Kaufmann, septembre 2000.

[HAN 00b] HAN J., PEI J., YIN Y., « Mining frequent patterns without candidate generation », *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, mai 2000, p. 1-12.

[KAM 97] KAMBER M., HAN J., CHIANG Y., « Metarule-guided mining of multi-dimensional association rules using data cubes », *Proc. of the 3rd KDD Int'l Conf.*, août 1997.

[LEN 97] LENT B., SWAMI A., WIDOM J., « Clustering association rules », *Proc. of the 13th Int'l Conf. on Data Engineering (ICDE)*, mars 1997, p. 220-231.

[LIN 98] LIN D., KEDEM Z. M., « Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set », *Proc. of the 6th Int'l Conf.on Extending Database Technology (EDBT)*, mars 1998, p. 105-119.

[LUX 91] LUXENBURGER M., « Implications partielles dans un contexte », *Math. Inf. Sci. Hum.*, vol. 29, n° 113, 1991, p. 35-55.

[MAN 94] MANNILA H., TOIVONEN H., VERKAMO A. I., « Efficient Algorithms for Discovering Association Rules », *Proc. AAAI Workshop on Knowledge Discovery in Databases (KDD)*, juillet 1994, p. 181-192.

[MAN 97a] MANNILA H., TOIVONEN H., « Levelwise Search and Borders of Theories in Knowledge Discovery », *Data Mining and Knowledge Discovery*, vol. 1, n° 3, 1997, p. 241-258, voir également le rapport C-1997-8.

[MAN 97b] MANNILA H., TOIVONEN H., VERKAMO A. I., « Discovery of frequent episodes in event sequences », *Data Mining and Knowledge Discovery*, vol. 1, n° 3, 1997, p. 259-289.

[NOV 01] NOVELLI N., CICCHETTI R., « FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies », *Proc. of the 8th Int'l ICDT Conf.*, janvier 2001, p. 189-203.

[PAR 95] PARK J. S., CHEN M. S., YU P. S., « An Efficient Hash Based Algorithm for Mining Association Rules », *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, mai 1995, p. 175-186.

[PAS 98] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Pruning Closed Itemset Lattices for Association Rules », *Actes des 14es journées « Bases de données avancées »*, octobre 1998, p. 177-196.

[PAS 99a] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Closed Set Based Discovery of Small Covers for Association Rules », *Actes des 15es journées « Bases de données avancées »*, octobre 1999, p. 361-381, version étendue parue dans le journal NIS.

[PAS 99b] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Discovering Frequent Closed Itemsets for Association Rules », *Proc. of the 7th Int'l Conf. on Database Theory (ICDT)*, n° 1540 Lectures Notes in Computer Sciences, Springer, janvier 1999, p. 398-416.

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

[PAS 00a] PASQUIER N., « Data Mining : algorithmes d'extraction et de réduction des règles d'association dans les bases de données », thèse d'université, université Blaise Pascal, Clermont-Ferrand, janvier 2000.

[PAS 00b] PASQUIER N., BASTIDE Y., TAOUIL R., LAKHAL L., « Closed set based discovery of small covers for association rules », *Networking and Information Systems Journal*, vol. 3, n° 2, 2000, p. 344-377.

[PEI 00] PEI J., HAN J., MAO R., « Closet: An efficient algorithm for mining frequent closed itemsets », *Proc. Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)*, mai 2000, p. 21-30.

[SAR 98] SARAWAGI S., THOMAS S., AGRAWAL R., « Integrating association rule mining with databases: alternatives and implications », *Proc. Int'l Conf. on Management of Data (SIGMOD)*, 1998.

[SAV 95] SAVASERE A., OMIECINSKI E., NAVATHE S., « An Efficient Algorithm for Mining Association Rules in Large Databases », *Proc. of the 21th Int'l Conf. on Very Large Data Bases (VLDB)*, septembre 1995, p. 432-444.

[SIL 98] SILVERSTEIN C., BRIN S., MOTWANI R., « Beyond Market Baskets: Generalizing Association Rules to Dependence Rules », *Data Mining and Knowledge Discovery*, vol. 2, n° 1, 1998.

[STU 00] STUMME G., TAOUIL R., BASTIDE Y., PASQUIER N., LAKHAL L., « Fast Computation of Concept lattices Using Data Mining Techniques », *Proc. of the 7th Int'l Workshop on Knowledge Representation meets Databases (KRDB)*, août 2000, p. 129-139.

[TAO 00] TAOUIL R., PASQUIER N., BASTIDE Y., LAKHAL L., « Mining basis for association rules using closed sets », *Proc. of the 16th Int'l Conf. on Data Engineering (ICDE)*, n° 1861 Lecture Notes in Computer Science, Springer, février–mars 2000, page 307.

[TOI 96] TOIVONEN H., « Sampling Large Databases for Association Rules », *Proc. of the 22nd Int'l Conf. on Very Large Data Bases (VLDB)*, septembre 1996, p. 134-145.

[ZAK 97] ZAKI M. J., PARTHASARATHY S., OGIHARA M., LI W., « New Algorithms for Fast Discovery of Association Rules », *Proc. of the 3rd Int'l Conf. on Knowledge Discovery in Databases (KDD)*, août 1997, p. 283-286.

Article reçu le 12 octobre 2000.
Version révisée le 8 mars 2001.
Rédacteur responsable : BRUNO MARRE

Yves Bastide *est docteur d'université en informatique. Ses activités de recherche portent sur les algorithmes et structures de données pour la fouille de données, et sur l'application de celle-ci à des données complexes.*

Rafik Taouil *est actuellement post-doctorant à l'INRIA Lorraine. Il étudie notamment l'algorithmique du treillis des fermés et ses différentes applications à la fouille de données et aux bases de données.*

Nicolas Pasquier *est docteur d'université en informatique. Il est actuellement maître de conférences au laboratoire d'informatique, signaux et systèmes de Sophia-Antipolis de l'université de Nice Sophia-Antipolis. Ses activités de recherche concernent l'extraction de connaissances dans les bases de données.*

Gerd Stumme *est docteur d'université en mathématique, actuellement maître de conférences dans l'équipe « Gestion des connaissances » à l'institut d'informatique appliquée de l'université de Karlsruhe (Allemagne). Ses activités de recherche portent sur les méthodes pour l'acquisition et la découverte des connaissances basées sur l'Analyse de concepts formels, ainsi que sur les ontologies et le Web sémantique.*

Lotfi Lakhal *est professeur à l'université de la Méditerrané-IUT d'Aix-en-Provence. Ses activités de recherche concernent l'algorithmique des treillis, le* data mining *et le* data warehousing.