UNIVERSITÉ DU QUÉBEC À MONTRÉAL

EXPANSION DE LA REPRÉSENTATION SUCCINCTE DES GÉNÉRATEURS MINIMAUX

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE DE GESTION

PAR

HAFIDA ABBAS

MARS 2013

REMERCIEMENTS

Je tiens à remercier Monsieur Petko Valtchev professeur à l'Université du Québec à Montréal pour son encadrement tout au long de ce travail, ainsi que pour ses directives et ses remarques les plus constructives, qu'il trouve dans cet ouvrage un témoignage de ma profonde reconnaissance.

Mes remerciements les plus sincères vont au professeur Martin Cloutier, directeur du département informatique de gestion, pour son soutien durant mes études au 2e cycle, et sans qui cette très belle expérience à l'université du Québec à Montréal n'aurait pas été possible.

Je remercie également Monsieur Hafedh Mili professeur à l'université du Québec à Montréal, pour ses remarques et suggestions faites sur le projet alors qu'il était encore en chantier, et qui ont contribué grandement à l'amélioration de ce travail.

Je remercie aussi tous les étudiants du laboratoire GDAC pour leur aide et leur bonne humeur.

Enfin, je tiens à remercier tout particulièrement mon conjoint pour son soutien, son attention et son aide avec les enfants à la maison. Merci encore à tous.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

Tableau Page

LISTE DES ALGORITHMES

RÉSUMÉ

L'évolution rapide des techniques de génération et de stockage de données a permis à de nombreux organismes la création de bases de données volumineuses, pour stocker l'information nécessaire à leurs activités. Ces bases de données qui deviennent de plus en plus importantes sont réellement peu exploitées, alors qu'elles cachent des connaissances potentiellement utiles pour l'organisation. L'extraction de ces informations enfouies dans ces masses de données est traitée par la fouille de données ("Data Mining").

Ce projet de mémoire traite plus particulièrement le problème d'extraction des informations sous forme de règles d'associations. Le problème de la pertinence et de l'utilité des règles extraites est un problème majeur de l'extraction des règles d'associations. Ce problème est lié au nombre important de règles extraites et à la présence d'une forte proportion de règles redondantes. Nombreuses techniques de réduction de la famille de règles ont été publiées. Dans ce contexte, les résultats obtenus par l'analyse formelle des concepts (AFC) ont permis de définir un sous-ensemble de l'ensemble des règles d'associations valides appelés bases informatives. La génération de ces bases informatives se fait par une extraction efficace des itemsets fermés fréquents et leurs générateurs minimaux associés. Les générateurs minimaux composent les prémisses minimales de ces règles alors que leurs fermetures composent les conclusions maximales de ces règles. Cependant un survol de la littérature montre que les générateurs minimaux composant l'antécédent et la conséquence de ces bases, contiennent encore de la redondance. Une représentation réduite de ces générateurs minimaux est utile pour révéler la relation d'équivalence parmi les générateurs minimaux. Une étude a été menée dernièrement dans ce sens dans laquelle l'algorithme DSFS_MINER a été proposé et validé, permettant l'extraction d'une représentation succincte sans perte d'informations des générateurs minimaux.

Notre contribution dans ce projet réside d'une part, dans l'étude et l'expérimentation d'approches de représentations succinctes des générateurs minimaux, et d'autre part, dans la proposition d'un algorithme d'expansion permettant la dérivation de tous les générateurs minimaux afin de constituer la famille entière des générateurs minimaux du contexte d'extraction.

Mots clés

Data Mining, Règles d'associations, Analyse formelle des concepts, Générateurs minimaux, Itemset fermés, Générateur minimal, représentation succincte des générateurs minimaux.

CHAPITRE I

INTRODUCTION

1.1 Fouille de données

De nos jours, les organisations, collectent et stockent d'énormes volumes de données nécessaires à leur activité. Ces méga-bases de données qui ne cessent d'augmenter jour après jour constituent la mémoire de l'entreprise, pour lesquelles d'importants moyens sont mobilisés afin de les recueillir et de les conserver dans les entrepôts (base de données, documents, etc..,). Par ailleurs, les décideurs se trouvent très souvent confrontés à des choix pour lesquelles il manque d'informations et de connaissances. Les données sont disponibles mais sont en réalité peu exploitées dans le processus décisionnel à cause de l'absence d'outils adaptés à l'extraction des connaissances cachées dans ces données et qui sont utiles face au marché et à la concurrence. Data mining (Fouille de données), est apparu pour répondre à ces nouvelles exigences et à ce besoin pressant exprimés par les décideurs des organisations dans le but d'extraire des connaissances décisionnelles à partir de ces méga-bases de données. Le Data mining est en effet un élément essentiel du processus d'extraction des connaissances dans les bases de données (KDD).Ce processus est défini par (Fayyad et al., 96) comme étant un processus semi-automatique et itératif, constitué de plusieurs étapes, allant de la sélection et préparation de données jusqu'à l'interprétation des résultats, en passant par la phase de recherche de connaissances qu'est le Data mining. Les différentes étapes de ce processus sont présentées dans la figure 1.1. L'étape du Data mining dans le processus du KDD correspond donc à l'ensemble de méthodes et techniques, qui appliquées sur des données préparées, permettent d'obtenir des informations et des connaissances exploitables qui ne sont pas accessibles par les méthodes classiques (SQL, méthodes

statistiques). Ces informations sont des règles, des concepts, des régularités, des anomalies et des modèles qui sont intéressants, compréhensibles et ont une grande valeur du point de vue de l'utilisateur. La nature du modèle à extraire dépend du problème de fouille traité.

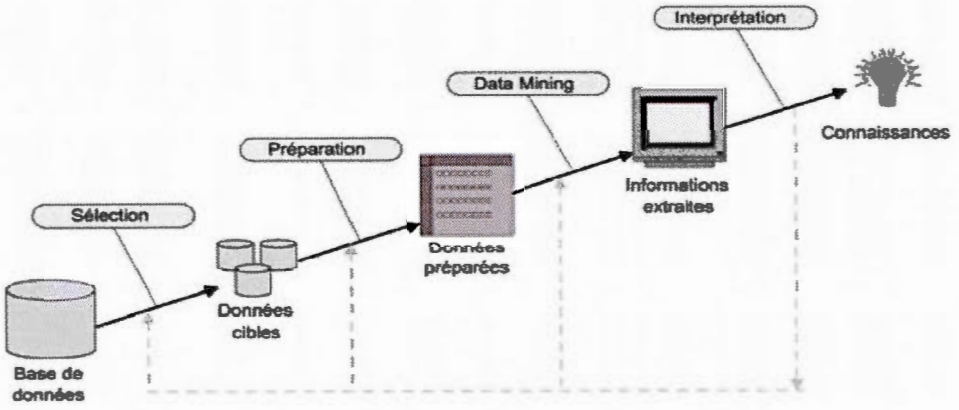

Figure 1.1 Étapes du processus d'extraction de connaissances dans les bases de données.

Selon (Pasquier, 2000) il existerait trois catégories principales de problèmes de fouille : Classification, regroupement et découverte des règles d'associations.

Ce projet de mémoire traite plus particulièrement le problème d'extraction des informations sous forme de règles d'association. L'extraction des règles d'association est une méthode dont le but est de découvrir des relations entre deux ou plusieurs variables stockées dans de très importantes bases de données et ayant un intérêt pour l'utilisateur. Les relations identifiées peuvent être utiles pour de nombreux organismes commerciaux, scientifiques, industriels et gestion de l'information, afin d'a=méliorer leurs résultats dans leurs activités. Par exemple, une règle découverte dans les données de ventes dans un supermarché pourrait indiquer qu'un client achetant du riz et du poisson simultanément, serait susceptible d'acheter du vin. Une telle information peut être utilisée comme base pour prendre des décisions marketing telles que par exemple des promotions ou des emplacements bien choisis pour les produits associés. Les informations provenant de l'analyse de données d'organismes médicaux peuvent offrir un support supplémentaire pour les recherches épidémiologiques en

santé (Pasquier, 2000). D'une manière plus générale, ces informations permettent d'apporter une aide pour améliorer l'efficacité de l'organisme dans ses activités. Une règle d'association R extraite à partir d'une base de donnée de transaction de vente serait par exemple : (**R : Céréales→ Lait (30%, 45%)**. Cette règle R indique que les transactions contenant les céréales ont tendance à contenir du lait. L'ensemble formé par l'article céréales est appelé antécédent ou prémisse de R, l'ensemble formé par l'article lait est appelé la conséquence ou conclusion de R. 30% représente le support de la règle R et indique la proportion de transactions qui contiennent Céréales et Lait. 45% représente la confiance de la règle et indique la proportion de transactions contenant Lait parmi celles contenant céréales. Ces mesures d'utilité (le support) et de précision (confiance) sont utilisées dans le problème de l'extraction des règles d'association afin de filtrer, parmi le grand nombre de règles générées à partir des méga-base de données, les règles d'association significatives. Ces mesures sont en effet, associées à chaque règle d'association et les règles générées sont celles dont le support et la confiance sont supérieurs ou égaux à des seuils minimaux définis par l'utilisateur en fonction de ses objectifs et du type de données traitées.

1.2 Définition du problème d'extraction des règles d'association

Introduit initialement par Agrawal et Srikant pour l'analyse des bases de données de transactions de vente, le problème de la découverte des règles d'association a suscité beaucoup d'intérêt dans la communauté scientifique, et plusieurs travaux ont été menés dans ce contexte incluant un nombre impressionnant de publications, pour développer des outils permettant l'amélioration des performances de l'algorithme de base Apriori (Agrawal et al., 1994).

1.2.1 Définitions

Considérons un ensemble $I = \{i_1, i_2, ..., i_m\}$ constitué de m articles ou m propriétés ou m *items*. Soit $ß = \{t_1, t_2, ..., t_n\}$ une base de données de n transactions et dans laquelle chaque transaction t_i est constituée d'un sous-ensemble $I_1 \subseteq I$ et identifiée par un identifiant unique appelé TID (identifiant de la transaction).

- **Définition 1.1** (*Itemset*)

Un itemset I_1 est un ensemble d'attributs ou de propriétés contenus dans I. Le plus petit itemset correspond à l'ensemble vide de taille 0. Le plus grand itemset est constitué de tous les attributs ou *items* de l'ensemble I (taille m). Un itemset de taille k est appelé un *k-itemset.*

- **Définition 1.2** (Support d'un itemset)

Le support d'un itemset I_1, noté $Sup(I_1)$, est défini par le pourcentage de transactions de ß contenant I_1.

Support $(I_1) = |\{t \in ß / I_1 \subseteq t\}| / |\{t \in ß\}|$

- **Définition 1.3** (*Itemset fréquent*)

Un itemset I_1 est dit fréquent si son support est supérieur ou égal au seuil minimal de support (*minSupport*) défini par l'utilisateur : Sup $(I_1) \geq$ *minSupport.*

- **Définition 1.4** (Règle d'association)

Une règle d'association est une règle d'implication de la forme $I_1 \rightarrow I_2$ entre deux itemsets $I_1, I_2 \subseteq I$ telle qu'$I_1 \cap I_2 = \emptyset$.

- **Définition 1.5** (Support d'une règle d'association)

Le support d'une règle d'association $r : I_1 \rightarrow I_2$ est défini par le pourcentage de transactions de ß contenant l'union des itemsets la constituant. Sup (r) = Sup $(I_1 \cup I_2)$.

- **Définition 1.6** (Confiance d'une règle d'association)

La confiance d'une règle d'association $r : I_1 \rightarrow I_2$ est la probabilité conditionnelle qu'une transaction de ß contienne I_2 sachant qu'elle contienne I_1.

Conf (r) = Sup $(I_1 \cup I_2)$ / Sup (I_1)

- **Définition 1.7** (Règle d'association forte ou valide)

Une règle d'association *r* est dite forte ou valide si son support et sa confiance sont supérieurs à des seuils minimaux de support (*minSupport*) et de confiance (*minConfiance*) définis par l'utilisateur : Sup (*r*) $\geq$ *minSupport* et Conf (*r*) $\geq$ *minConfiance*.

1.2.2 Objectif de l'extraction des règles d'associations

Soient *minSupport* et *minConfiance*, les seuils minimaux de support et de confiance définis par l'utilisateur. L'objectif de la découverte des règles d'association à partir d'une base de données ß consiste à déterminer, les règles d'associations fortes. Cette découverte se fait en deux étapes :

1) Déterminer l'ensemble des itemsets fréquents de la base de données ß (les itemsets qui ont un support $\geq$ *minSupport*).

2) Pour chaque itemset fréquent I_1, générer toutes les règles d'association r dont la confiance est $\geq$ *minConfiance*, avec r : $I_j \rightarrow I_1 \setminus I_j$ tels que : $I_j \subset I_1$ et $I_j \neq \emptyset$.

1.3 Étapes d'extraction des règles d'association

L'extraction des règles d'associations est un processus itératif et interactif (Fayyad et al., 1996 ; Pasquier, 1998) composé de quatre étapes dont celle de la découverte des règles d'associations (extraction des items fréquents et génération des règles d'association) (Pasquier, 2000).

1.3.1 Sélection et préparation des données

Cette phase consiste à la sélection d'un sous ensemble de données de la base qui permettront d'extraire les informations intéressant l'utilisateur et la transformation de ces données en contexte d'extraction. Cette transformation est nécessaire afin de pouvoir appliquer les

algorithmes d'extraction des règles d'association sur n'importe quel type de base de données du moment que les données peuvent provenir de divers types de base de données.

- **Définition 1.8** (Contexte d'extraction)

Un contexte d'extraction est un triplet $C = (O, I, R)$ où O et I sont respectivement des ensembles finis d'objets et d'items/attributs/propriétés et $R \subseteq O \times I$ est une relation binaire entre les objets et les items. Un couple $(o, i) \in R$ indique que l'objet o est en relation avec l'item i.

Tableau 1.1 Contexte d'extraction C des règles d'association D

	A	B	C	D	E
1	x		x	x	
2		x	x		x
3	x	x	x		x
4		x			x
5	x	x	x		x
6		x	x		x

Un contexte d'extraction peut être représenté sous la forme d'un tableau à deux dimensions où les lignes correspondent aux objets et les colonnes aux items. Les cases du tableau sont remplis suivant la présence/absence de l'item/propriété, autrement dit, si le ième objet o est en relation avec le jème item i, alors la case d'intersection de la ième ligne avec la jème colonne contiendrait un "x" sinon elle serait vide. La table 1.1 (Pasquier et al., 2000) donne

un exemple de contexte d'extraction constitué de six objets, identifié chacun par son OID et de cinq items.

1.3.2 Extraction des itemsets fréquents[1]

Soient C = (*O*, *I*, *R*) un contexte d'extraction, et *minSupport* le seuil minimal de support défini par l'utilisateur. L'ensemble *IF* des itemsets fréquents est défini par la relation suivante : $IF = \{I_1, \subseteq I \mid I_1 \neq \emptyset \ \wedge \text{Sup}\ (I_1) \geq minSupport\}$

La phase de découverte des itemsets fréquents consiste à parcourir l'ensemble des itemsets potentiels et à en extraire ceux qui ont un support supérieur ou égal à *minSupport*. Dans le cas d'un ensemble d'items de taille m, le nombre d'itemsets potentiellement fréquents est de l'ordre de $2^{|m|}$. Ainsi la découverte des itemsets fréquents s'effectue dans un espace de recherche exponentiel au nombre d'items. Plusieurs balayages du contexte sont aussi nécessaires durant cette phase afin d'évaluer le support des itemsets candidats qui permettront de déterminer lesquels sont fréquents, et c'est pourquoi cette phase est considérée comme la phase la plus couteuse en temps d'exécution et en espace de recherche, dans l'extraction des règles d'associations.

Vu le coût élevé de cette approche, les chercheurs ont focalisé leurs efforts pour développer des méthodes efficaces d'exploration de cet espace de recherche exponentiel. Quatre grandes approches ont été définies dans la littérature, elles sont présentées en détail dans le prochain chapitre de ce mémoire

1.3.3 Génération des règles d'associations

Soit *IF*, l'ensemble des itemsets fréquents extraits du contexte *C* pour un seuil minimal de support *minSupport*. La génération des règles d'associations pour un seul minimal de confiance *minConfiance* est exponentielle dans le nombre d'itemsets fréquents extraits. À partir d'un itemset fréquent *f* de taille supérieure ou égale à un, $2^{|f|}$ -2 règles non triviales

---

[1] **Itemsets** fréquents sont aussi référés dans le document par **motifs** fréquents

peuvent être générées (Bastide et al., 2000). L'ensemble *RA* des règles d'associations fortes dans *C* est défini par la relation suivante :

$RA = \{r: I_2 \rightarrow (I_1\text{-}I_2) \mid I_1, I_2 \in \text{IF} \wedge \text{support}(I_1)/\text{support}(I_2) \geq minConfiance\}$.

La phase de génération des règles d'association est réalisée d'une façon directe, sans accéder au contexte d'extraction. Par conséquent le coût de cette phase en temps d'exécution est faible par rapport au temps d'exécution de la phase d'extraction des itemsets fréquents. Le principe général de génération des règles d'association est le suivant : pour chaque itemset fréquent I_1 de *IF*, tous les sous ensembles I_2 sont déterminés et la valeur du rapport qui détermine la confiance est calculé (Support(I_1)/Support(I_2)). Si cette valeur est supérieure ou égale à *minConfiance*, alors la règle $r : I_2 \rightarrow (I_1\text{-}I_2)$ est générée. L'algorithme de génération des règles d'association proposé par (Agrawal et al., 94) suit le même principe et utilise les propriétés des itemsets fréquents pour réduire le nombre de tests à effectuer.

Toutefois, comme la phase de génération des règles d'association est exponentielle au nombre d'itemsets fréquents obtenu dans l'étape précédente et que ce nombre est souvent élevé, le nombre de règles générées devient excessif et inexploitable par l'utilisateur. Des travaux de recherche opérant sur l'analyse des concepts formels ont conduit à définir un sous ensemble de l'ensemble des règles d'associations appelé bases informatives (Bastide et al., 2000). Les bases informatives sont présentées en détail dans le prochain chapitre.

1.3.4 Visualisation et interprétation des résultats

Cette phase est effectuée par l'utilisateur qui visualise, analyse et interprète les règles d'association extraites dans la phase précédente afin de déduire des connaissances utiles pour l'amélioration de son activité. Cependant le nombre excessif de règles d'association produites, rend leur interprétation par l'utilisateur complexe. Des méthodes objectives telles que les mesures statistiques et déviation ainsi que des méthodes subjectives telles que les templates ont été incorporées dans cette phase afin de réduire le nombre de règles (Pasquier, 2000). Les méthodes objectives sont cependant complexes et nécessitent un post-traitement

important, et l'utilisation des méthodes subjectives dépendent beaucoup des connaissances de l'utilisateur dans le domaine et un grand nombre de règles inattendues pourrait être oublié.

1.4 Contribution

Les limitations de l'approche classique d'extraction des règles d'association résident principalement dans la phase d'extraction des itemsets fréquents. En effet, cette étape est considérée comme la plus coûteuse en terme de temps d'exécution vu le nombre de balayages du contexte réalisé et le nombre d'itemsets candidats considéré. Ce dernier augmente proportionnellement avec le nombre d'items du contexte d'extraction, ce qui rend le nombre d'itemsets fréquents, critique. Comme la génération des règles d'association est exponentielle au nombre d'itemsets fréquents extraits, l'ensemble des règles d'association fortes générées devient élevé et inexploitable par l'utilisateur. Plusieurs travaux de recherche ont été menés dans le domaine afin de développer des techniques pour réduire la taille de ces règles. Dans ce contexte, les résultats obtenus par l'analyse formelle des concepts (AFC) a permis de définir un sous ensemble de l'ensemble des règles d'association valides appelés bases informatives. La génération de ces bases informatives se fait par une extraction efficace des itemsets fermés fréquents et leurs générateurs minimaux associés. Les générateurs minimaux composent les prémisses minimales de ces bases informatives alors que leurs fermetures composent les conclusions maximales de ces règles. Cependant l'analyse de (Dong et al., 2005) a montré que même ces bases comportent de la redondance au niveau des générateurs minimaux qui composent l'antécédent et la conséquence des bases génériques. Une représentation réduite de ces générateurs minimaux est utile pour révéler la relation d'équivalence parmi les générateurs minimaux. Une étude a été menée dans ce sens par (Hamrouni et al., 2007) dans laquelle l'algorithme *DSFS_MINER* a été proposé et validé par les auteurs qui permet l'extraction d'une représentation succincte et sans perte d'informations des générateurs minimaux. Notre contribution dans ce projet réside dans l'étude et l'expérimentation d'approches de représentations succinctes des générateurs minimaux ainsi que la proposition d'un algorithme d'expansion permettant la dérivation de tous les générateurs minimaux afin de constituer la famille entière des générateurs minimaux du contexte d'extraction.

1.5 Organisation du document

Ce mémoire est organisé en cinq chapitres. Le premier chapitre, présente longuement la technique d'*extraction des règles d'association,* en précisant notamment les deux phases indispensables par lesquelles elle passe, à savoir la découverte des itemsets fréquents et la génération des règles d'association. Le second chapitre traite l'état de l'art de l'approche de découverte des règles d'association par l'extraction des itemsets fréquents. Les trois grandes familles d'algorithmes recensés dans la littérature pour l'extraction des itemsets fréquents sont présentées dans ce chapitre, accompagnées des descriptions des principaux algorithmes, cités dans la littérature (Agrawal et al., 1994) comme des algorithmes piliers pour la phase d'extraction des itemsets fréquents, il y est notamment présenté l'algorithme *Apriori* et l'algorithme *Eclat*. Puis avant de passer au troisième chapitre, il présente l'algorithme *GEN-Règles* permettant la génération des règles d'association à partir des itemsets fréquents, et énumère la liste non exhaustive des méthodes proposées dans la littérature pour éliminer la redondance dans les règles générées avec cette approche classique. Le chapitre suivant consiste à définir l'approche basée sur l'analyse formelle des concepts (AFC) qui s'est avérée être un cadre théorique très intéressant pour l'extraction des règles d'association. Il fournit quelques bases théoriques sur l'AFC puis traite l'état de l'art en effectuant un survol des principaux algorithmes proposés dans la littérature pour l'extraction des itemsets fermés et décrit en détail l'*algorithme Close*, que nous avons utilisé pour valider notre algorithme *DSFS_Expander*. Les bases génériques et informatives des règles d'associations basées sur l'analyse formelle des concepts sont également présentées dans ce troisième chapitre. Le quatrième chapitre est consacré à la représentation succincte des générateurs minimaux et aux travaux menés dans ce sens pour aboutir à une représentation réduite sans perte d'information de l'ensemble de tous les générateurs minimaux pouvant être extraits du contexte d'extraction. Dans ce contexte, les différentes approches des systèmes succincts des générateurs minimaux, proposées dans la littérature, y sont citées avec leurs avantages et leurs inconvénients, et sont accompagnés de la description détaillée de l'algorithme *DSFS_Miner* proposé dans ce cadre par (Hamrouni et al., 2007). Le dernier chapitre est réservé pour décrire notre contribution dans le cadre de ce présent mémoire, et qui consiste à définir un processus d'expansion permettant de dériver à partir de la représentation succincte

des générateurs minimaux, définie dans le chapitre quatre, l'ensemble de tous les générateurs minimaux redondants du contexte, afin de constituer la famille entière des générateurs minimaux pouvant être générés du contexte. Le principe du processus d'expansion y est décrit, accompagné de l'algorithme *DSFS_Expander* que nous proposons pour supporter le processus d'expansion. Les résultats des expérimentations que nous avons effectuées dans ce cadre, sont également présentés dans ce chapitre afin de valider notre approche d'expansion.

CHAPITRE II

APPROCHES D'EXTRACTION DES RÈGLES D'ASSOCIATION

2.1 Introduction

Le problème de découverte des règles d'association à partir des données, consiste à déterminer l'ensemble des règles dont le support et la confiance sont au moins égaux à des seuils minimaux de support *minSupport* et de confiance *minConfiance* fixés par l'utilisateur. Ce problème est décomposé dans la littérature en deux sous-problèmes :

1) Déterminer à partir des données, l'ensemble des itemsets fréquents(les itemsets possédant un support supérieur ou égal au seuil minimal de support *minSupport*).

2) Pour chaque itemset fréquent I_1 extrait, générer toutes les règles d'association r, possédant une confiance supérieure ou égale à $\geq$ *minConfiance*, avec $r : I_j \rightarrow I_1 \setminus Ij$ tels que : $I_j \subset I_1$ et $I_j \neq \emptyset$.

Vu le coût élevé de la phase d'extraction des itemsets fréquents, et vu son importance pour la génération des règles d'association, beaucoup de chercheurs ont focalisé leurs efforts à trouver des techniques et développer des algorithmes efficaces afin de réduire ce coût.

Nous entamons ce chapitre par un rappel des notions de base sur les ensembles ordonnées, sur la correspondance de Galois et sur les concepts ensuite nous faisons un état de l'art sur ce qui a été proposé dans la littérature concernant d'une part, les différentes approches

d'extraction des itemsets fréquents utilisées pour la dérivation des règles d'associations et d'autre part les différentes méthodes de génération de règles d'association.

2.2 Notions de base

2.2.1 Ensembles ordonnés

Nous présentons quelques définitions sur les ensembles ordonnés afin que nous puissions introduire par la suite le treillis des parties associé à l'ensemble fini d'items I qui est utilisé pour l'extraction des itemsets fréquents. Ces définitions ont été extraites des ouvrages de références (Barbut et al., 1970 ; Birkhoff et al., 1967 ; Davey et al., 1994)

- **Définition 2.1** (Ensemble ordonné)

Soit E un ensemble. Un ordre partiel sur l'ensemble E est une relation binaire (notée $\leq$) sur les éléments de E telle que pour tout $x, y, z \in E$, nous avons les propriétés suivantes :

1) Réflexivité : $x \leq x$

2) Anti -Symétrie : $x \leq y$ et $y \leq x \implies x = y$

3) Transitivité : $x \leq y$ et $y \leq z \implies x \leq z$

Un ensemble doté d'une telle relation binaire $\leq$, noté $(E, \leq)$, est appelé ensemble ordonné.

- **Définition 2.2** (Relation de couverture)

Soit $(E, \leq)$ un ensemble ordonné, et $x, y \in E$ deux éléments de E. La relation de couverture entre les éléments de E notée $\prec_E$, est définie pas : $x \prec_E y$ si et seulement si $x \leq y$ et il n'existe pas d'élément $z \in$ E tel que $x \leq y \leq z$ pour $z \neq x$ et $z \neq y$. Si $x \prec_E y$, nous dirons qu'y couvre x ou bien y est un successeur immédiat de x.

- **Définition 2.3** (Majorant / Minorant)

Soit $(E, \leq_E)$ un ensemble ordonné, et S un sous-ensemble de E. Les éléments majorants (successeurs) et minorants (prédécesseurs) de S sont définis par :

Majorant $(S) = \{x \in E \mid \forall\ y \in S, y \leq_E x\}$

Minorant $(S) = \{x \in E \mid \forall\ y \in S, x \leq_E y\}$

- **Définition 2.4** (Join / Meet)

Soit $(E, \leq_E)$ un ensemble ordonné et S un sous-ensemble de E. le plus petit majorant de S s'il existe est le plus petit élément de l'ensemble des majorants de S. Cet élément noté $\wedge_S$ est appelé *Meet(S)* ou *infimum* de S. Dualement, le plus grand des minorants de S s'il existe est le plus grand élément de l'ensemble des minorants de S. Cet élément noté $\vee_S$ est appelé *Join (S)* ou *supremum* de S.

2.2.1 Treillis des itemsets

- **Définition 2.5** (Treillis)

Soit $(E, \leq_E)$ un ensemble ordonné non vide, est un *treillis* si pour tout couple $(x, y) \in E$, l'ensemble $\{x, y\}$ possède un plus petit majorant noté *Join* $(\{x, y\})$ et un plus grand minorant noté *Meet* $(\{x, y\})$. L'ensemble ordonné $(E, \leq_E)$ est un treillis complet si pour tout sous-ensemble $S \subseteq E$, les éléments *Join (S)* et *Meet(S)* existent. Un exemple du treillis d'itemsets représentant l'ensemble des parties de {A, B, C, D, E} est présenté dans la figure 2.1.

- **Définition 2.6** (Diagramme de Hasse)

La représentation graphique d'un treillis $(E, \leq_E)$ s'exprime à l'aide d'un diagramme appelé Diagramme de Hasse, dans lequel les nœuds correspondent aux éléments de E et les arrêtes reliant les nœuds correspondent aux relations de couverture (successeurs et prédécesseurs immédiats) entre ces nœuds.

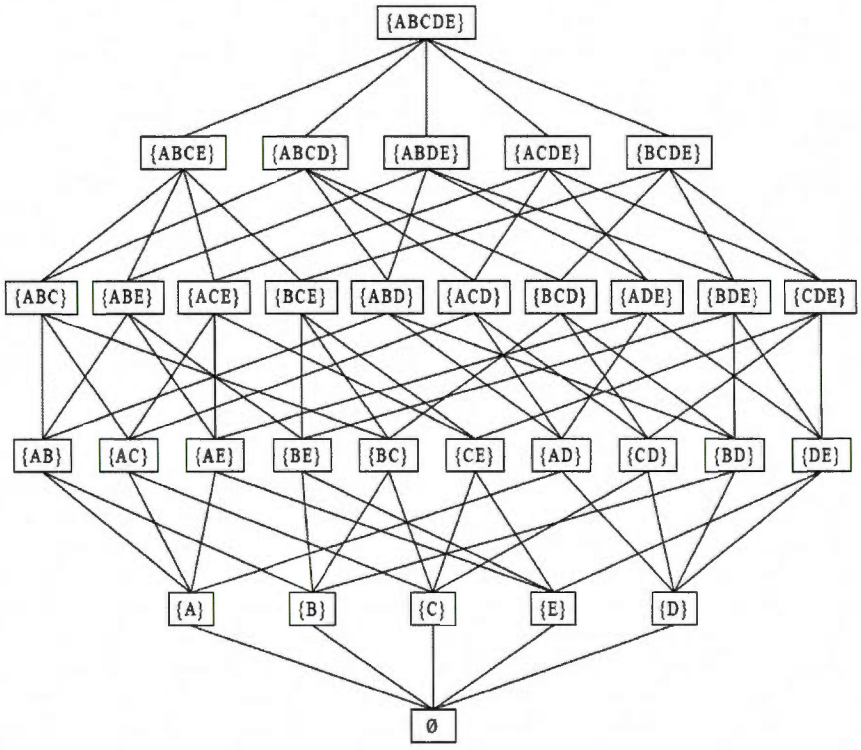

Figure 2.1 Treillis représentant les parties de {A. B, C, D, E} (Pasquier, 2000)

2.3 Algorithmes d'extraction d'itemsets fréquents

De nombreux algorithmes ont été proposés dans la littérature pour traiter le problème d'extraction des itemsets fréquents (Park et al., 1995 ; Houtsma et al., 1995 ; Mueller, 1995 ; Toivonen, 1996 ; Bin et al., 1997 ; Lin et al., 1998 ; Lin, 1 998), considéré comme problème majeur dans l'extraction des règles d'association, nous en détaillons dans cette section, seulement deux qui nous semblent les plus importants

Dans un contexte d'extraction $C = (O, I, R)$ où O est l'ensemble fini de n objets, I ensemble fini de m items et R la relation binaire associant les objets du contexte aux items, les 2^m itemsets potentiellement fréquents sont les sous-ensembles de I. Ces itemsets constituent

l'ensemble des parties de I noté $\mathcal{P}(I)$. L'ensemble ordonné $(\mathcal{P}(I), \subseteq)$ forme un treillis complet, appelé treillis des parties de I. La découverte des itemsets fréquents consiste à extraire des itemsets du treillis des parties de I dont le support est supérieur ou égal au seuil minimal de support *minSupport* fixé par l'utilisateur.

Quatre grandes familles d'algorithmes ont été recensées dans la revue de littérature, et sont présentées dans la présente section.

2.3.1 Famille des algorithmes par niveaux

Les algorithmes adoptant cette approche, procèdent d'une façon itérative en parcourant le treillis des itemsets, par niveaux (par taille d'itemset). Le parcours se fait en largeur du treillis, du bas vers le haut, en déterminant lors de chaque itération, tous les itemsets fréquents d'un niveau, c'est-à-dire d'une taille donnée. Autrement dit, tous les itemsets fréquents de taille 1 sont trouvés en premier, ensuite pour chaque itération k, tous les itemsets fréquents de taille k sont identifiés. Ces itérations cessent lorsque le plus long itemset fréquent est trouvé. Ces algorithmes réalisent donc μ itérations afin de déterminer les itemsets fréquents dans l'ordre croissant de leurs tailles, μ étant la taille du plus long itemset fréquent. Pour chaque itération k, un seul et même balayage du contexte est réalisé afin de calculer le support des itemsets. Ainsi le nombre de balayages réalisés est équivaut à la taille du plus long itemset fréquent μ.

Cette approche est très simple et efficace sur des données faiblement corrélées et éparses (Szathmary, 2008). Elle est basée sur deux propriétés principales des itemsets fréquents :

- **Propriété 2.1**

Tous les sous-ensembles d'un itemset fréquent sont fréquents.

- **Propriété 2.2**

Tous les sur-ensembles d'un itemset non fréquent, sont non fréquents.

Considérons le contexte d'extraction C (O, I, R) représenté par le tableau 2.1, composé d'un ensemble O de cinq objets, d'un ensemble I = {A, B, C, D, E} de cinq attributs et d'une relation R associant les objets de O aux attributs de I. Soit *minSupport* =3 le seuil minimal de support fixé par l'utilisateur.

Tableau 2.1 Contexte d'extraction C (utilisé par les algorithmes d'extraction *IF*) (Pasquier, 2000)

	A	B	C	D	E
1	x	x		x	x
2	x		x		
3	x	x	x		x
4		x	x		x
5	x	x	x		x

Les itemsets du treillis peuvent être regroupés en deux ensembles disjoints : les itemsets fréquents et les itemsets rares (non fréquents). Entre les deux ensembles, une bordure peut être tracée divisant l'espace du treillis des itemsets en deux (un espace positif et un autre négatif). Les itemsets fréquents se trouvent du coté positif de la bordure (Mannila et al., 1997). Les itemsets ayant la même taille se trouvent tous au même niveau dans le treillis. Le plus petit itemset formé par l'ensemble vide se trouve à l'extrémité inférieure du treillis, alors que le plus grand itemset constitué de tous les attributs du contexte est localisé à l'extrémité supérieure du treillis. Le support de chacun des itemsets est indiqué à droite de l'itemset en question. La recherche des itemsets fréquents consiste donc en la génération de tous les itemsets se trouvant dans la partie positive de la bordure.

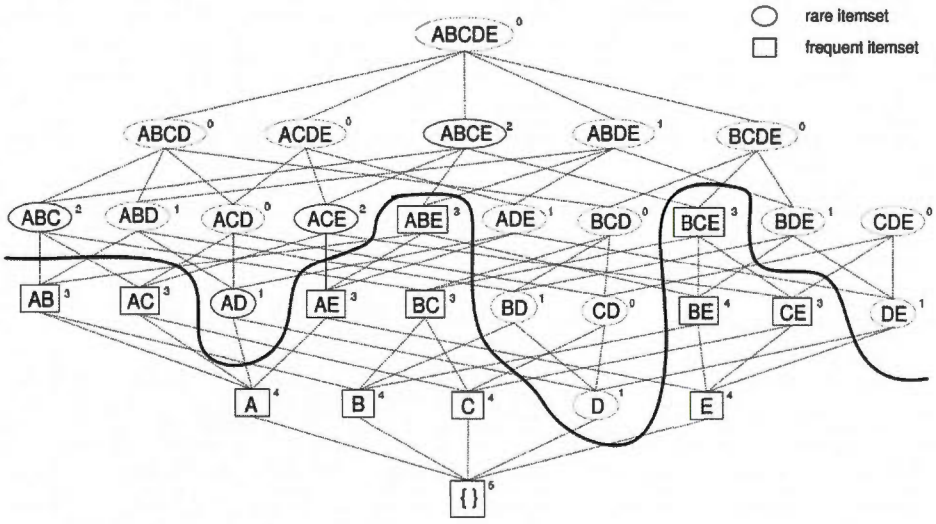

Figure 2.2 Treillis des itemsets fréquents du contexte C (Szathmary, 2008)

2.3.1.1 Exploration par niveaux de la partie positive de la bordure

Les algorithmes à niveaux effectuent une recherche d'itemsets fréquents dans le treillis des itemsets d'une façon itérative. C'est-à-dire, à chaque niveau i, uniquement les i-itemsets fréquents sont utilisés pour générer leurs sur-ensembles (i+1)-itemsets. Ces sur-ensembles sont appelés candidats, et parmi eux, uniquement les itemsets potentiellement fréquents sont conservés. Pour stocker les itemsets, deux sortes de tables sont utilisées. La table F_i pour les i-itemsets fréquents et la table C_i pour les i-itemsets potentiellement fréquents. Un (i+1)-itemset est dit potentiellement fréquent si tous sous ensembles de taille i sont fréquents, sinon il est considéré non fréquent conformément à la propriété 2.2 des itemsets fréquents, et peut ainsi être élagué. Le support de ces (i+1)- itemsets candidats potentiellement fréquents est calculé par balayage de la base de données et les itemsets qui s'avèrent non fréquents sont élaguées. Les (i+1)-itemsets fréquents sont ainsi utilisés pour générer les (i+2)-itemsets candidats, etc. Le processus continue jusqu'à ce qu'aucun candidat ne puisse être généré. La génération des (i+1)-itemsets candidats à partir des i-itemsets fréquents, s'effectue en deux étapes : la première étape concerne l'opération de jointure réalisée sur la table F_i avec elle-même. L'union de p $\cup$ q des itemsets p et q appartenant à F_i est insérée dans C_{i+1} s'ils partagent leurs premiers i-1 items. La seconde étape est l'élagage qui consiste à supprimer les

itemsets candidats de C_{i+1} si au moins un de leurs sous-ensembles de taille i n'appartient pas à F_i.

Les itemsets candidats potentiellement fréquents qui s'avèrent être non fréquents suite au calcul de support, forment l'ensemble des *itemsets rares minimaux* (Szathmary, 2008). Un *itemset rare minimal* est un itemset rare (non fréquent) dont tous les sous ensembles sont fréquents (nécessairement, tous ses sur-ensembles sont non fréquents). L'ensemble des itemsets rares minimaux est appelé *Bordure négative* (Mannila et al., 1 997). Par le biais des propriétés 2.1 et 2.2, les algorithmes à niveaux garantissent qu'une fois qu'un itemset rare minimal est trouvé, aucun de ses sur-ensembles n'est généré. Par conséquent le support des sur-ensembles des itemsets rares minimaux n'est jamais calculé. Et c'est de cette façon que les algorithmes à niveaux réduisent l'espace de recherche dans le treillis des itemsets (figure 2.2). La génération d'itemsets candidats et le processus de calcul de support nécessitent cependant un test de sous-ensembles. Autrement dit, lors de la génération d'un (i+1)-itemset candidat, ses sous ensembles immédiats de taille i doivent être identifiés dans F_i, et lors du calcul de son support, ce dernier est comparé à chacun des objets de la base de données. Son support ne peut être incrémenté lors d'une comparaison avec un des objets de la base de données, que si chacun de ses sous ensembles est contenu dans l'objet en question. Comme ces opérations sur les sous-ensembles sont exécutées plusieurs fois, leur implémentation doit être effectuée efficacement pour garantir une bonne performance. Deux structures de données sont fréquemment utilisées pour les tests sur les sous-ensembles : arbre de hachage (Hash Tree) comme le suggère (Agrawal et al., 1996), ou arbre préfixé (Prefix-Tree) (Aho et al., 1985 ; Pasquier et al., 1999).

Une panoplie d'algorithmes d'extraction des itemsets fréquents par niveaux a été retrouvée dans la revue de littérature, le plus connu est sans doute l'algorithme Apriori proposé par (Agrawal et al., 1994) et qui est présenté dans la sous section suivante. Plusieurs variations de cet algorithmes ont été développés par la suite mais la plupart d'entre eux se sont concentrés

Algorithme 2.1 *Apriori*

Entrées C : contexte d'extraction, *minSupport* : support minimal défini par l'usager
Sortie F : ensemble formé par tous les i-*itemsets* fréquents
1) $F_1 \leftarrow$ {1-*itemset* fréquent}//initialisation de l'ensemble des items fréquents
2) $F \leftarrow F_1$
3) **pour** ($i \leftarrow 2$; $F_{i-1} \neq \emptyset$; i++) **faire**
4) $C_i \leftarrow$ *AprioriCandidats* (F_{i-1})
5) // Mise à jour des supports des i-*itemsets* candidats par un balayage du contexte C
6) **pour chaque** transaction $t \in C$ **faire**
7) **pour chaque** *itemset* candidat $c \in C_i$ **faire**
8) **si** $c \subseteq t$ **alors**
9) $sup\ (c) \leftarrow sup\ (c) + 1$
10) **fin si**
11) **fin pour**
12) **fin pour**
13) // Élagage des i-*itemsets* non fréquents
14) $F_i \leftarrow \emptyset$
15) **pour chaque** *itemset* candidat $c \in C_i$ **faire**
16) **si** ($sup\ (c) / |O|^2$) $\geq$ *minSupport* **alors**
17) $F_i \leftarrow F_i \cup \{c\}$
18) **fin si**
19) **fin pour**
20) $F \leftarrow F \cup F_i$
21) **fin pour**
22) **retourner** F

[2] $|O|$ désigne la cardinalité de l'ensemble O

sur l'extraction d'un sous ensembles spécial d'itemset comme les itemsets fermés et les générateurs minimaux qui font l'objet du chapitre suivant.

2.3.1.2 Extraction des itemsets fréquent avec l'algorithme Apriori

L'algorithme démarre avec la liste de tous les items fréquents du contexte respectant le seuil de support *minSupport* (Ligne 1). Pour chaque itération i, l'ensemble des i-itemsets candidats C_i est déterminé à partir des (i-1)-itemsets fréquents (ligne 4).

Algorithme 1.2 *AprioriCandidats*

Entrée F_{i-1} : ensemble des *itemsets* fréquents de taille i-1
Sortie C_i : ensemble des i-*itemsets* candidats
1) // Jointure des candidats possédant un préfixe commun de taille i-2
2) $C_i \leftarrow \emptyset$
3) **pour** chaque paire d'*itemsets* $I_1, I_2 \in F_{i-1}$ ne différant que par leur dernier *item* **faire**
4) // Union disjonctive de leur *itemset*
5) $c_i \leftarrow \emptyset$
6) $c_i \leftarrow I_1 \cup I_2$
7) $C_i \leftarrow C_i \cup \{c_i\}$
8) **fin pour**
9) // Élagage des candidats non fréquents
10) **pour** chaque *itemset* candidat $c \in C_i$ **faire**
11) **pour** chaque sous-ensemble s de taille i-1 tel que $s \subset c$ **faire**
12) **si** $s \notin F_{i-1}$ **faire**
13) supprimer c de C_i
14) **fin si**
15) **fin pour**
16) **fin pour**
17) retourner C_i

Le support de ces i-itemsets candidats est calculé en effectuant un seul et unique balayage du contexte (lignes 5-12), et les itemsets candidats ayant un support au moins égal au seuil minimal de support sont désignés fréquents et sont donc insérés dans l'ensemble F_i (lignes 13-19) qui va servir à constituer les (i+1)-itemsets candidats. Les i-itemsets candidats (ligne 4) sont obtenus par l'algorithme *AprioriCandidats*.

AprioriCandidats effectue toutes les jointures des (i-1)-itemsets fréquents possédant un préfixe identique de taille i-2 (ligne 1-8). Parmi les i-itemsets obtenus par jointure (C_i), tous ceux ne comportant pas l'un des (i-1)-itemsets fréquents (lignes 9-16) sont élagués.

- **Exemple 2.1 :** L'application de l'algorithme *Apriori* au contexte C défini par le tableau 2.1, pour un seuil minimal *minSupport*=3 est représenté ci-dessous par la figure 2.3. Quatre itérations sont exécutées par l'algorithme qui calcule quatre ensembles de candidat et d'itemsets fréquents et réalise quatre balayages du contexte C.

Balayage de C →

C_1	
Itemset	Support
{A}	4
{B}	4
{C}	4
{D}	1
{E}	4

Suppression des itemsets non fréquents →

F_1	
Itemset	Support
{A}	4
{B}	4
{C}	4
{E}	4

C_2
{AB}
{AC}
{AE}
{BC}
{BE}
{CE}

Balayage de C →

C_2	
Itemset	Support
{AB}	3
{AC}	3
{AE}	3
{BC}	3
{BE}	4
{CE}	3

Suppression des itemsets non fréquents →

F_2	
Itemset	Support
{AB}	3
{AC}	3
{AE}	3
{BC}	3
{BE}	4
{CE}	3

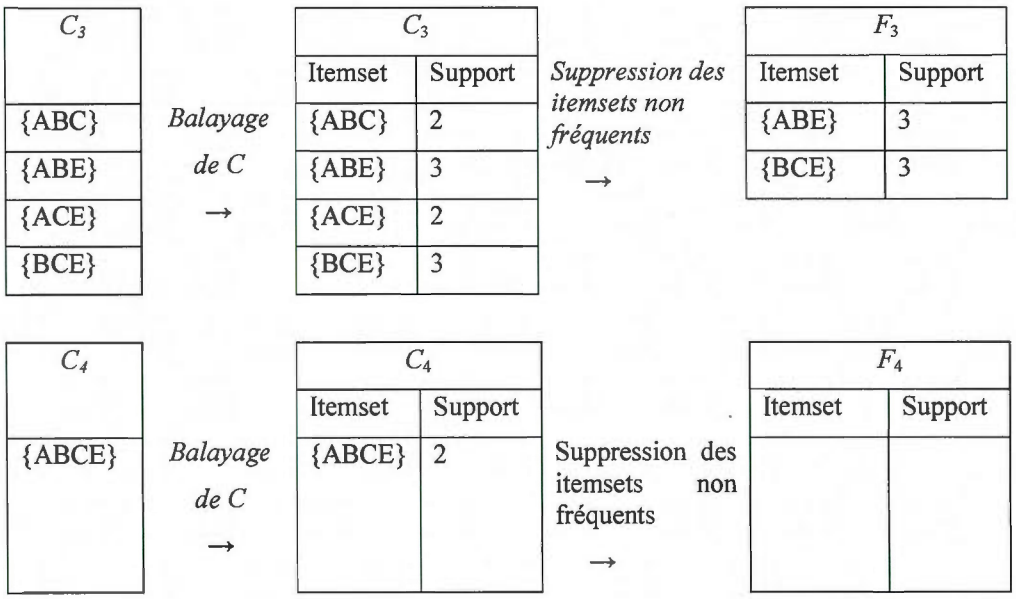

Figure 2.3 Exemple d'application de l'algorithme *Close*

2.3.2 Famille des algorithmes verticaux

Le premier algorithme proposant une approche en profondeur pour la recherche d'itemsets fréquents est l'algorithme *Eclat*. *Charm* est la modification d'*Eclat* mais pour l'exploration des itemsets fermés uniquement, qui fait l'objet du chapitre suivant. Ces deux algorithmes utilisent une représentation verticale de la base de données. Les expérimentations dans la littérature, ont montré que cette représentation offrait des améliorations significatives de performances comparativement à la représentation horizontale de la base de données, utilisée par les algorithmes à niveaux (Szathmary, 2008).

2.3.2.1 Concepts de base

Les concepts définis dans cette partie de la section sont reliés à la terminologie utilisée dans les travaux menés par (Zaki et al., 1997; Zaki, 2000; Zaki et al., 2002, Zaki et al., 2003).

Soient I un ensemble d'items, D une base de donnée transactionnelle, où chaque transaction a un identifiant unique (tid) et contient un ensemble d'items, et T l'ensemble de tous les tids. Un ensemble d'items est appelé itemset et un ensemble de tid est appelé tidset. Un itemset $\{A, B, C\}$ est représenté par ABC et un tidset $\{1, 2, 3\}$ est représenté par 123. À un itemset I_1 correspond le tidset $t(I_1)$ représentant l'ensemble des tids (transactions) contenant I_1 comme sous-ensemble et à un tidset O_1 correspond l'itemset $i(O_1)$ représentant l'ensemble d'items communs à tous les tids (transactions) contenus dans O_1. $t(I_1) = \cap_{i \in I1} t(i)$ et $i(O_1) = \cap_{o \in O1} i(o)$. Exemple : Soit le contexte d'extraction C défini par le tableau 2.1.$t(AC)=t(A) \cap t(C)= 1235 \cap 2345 = 235$. $i(13)= i(1) \cap i(3)=ABDE \cap ABCE = ABE$. Le support d'un itemset I_1 est **défini par Support $(I_1)= |t(I_1)|$.**

- **Propriété 2.3**

Soient X et Y deux itemsets. Si $X \subseteq Y \Rightarrow t(X) \supseteq t(Y)$. (Zaki et al., 1997)

2.3.2.2 Arbre de recherche IT et classes d'équivalences basées préfixe

Soit I un ensemble d'items ordonné suivant l'ordre total lexicographique, une fonction de préfixe notée p définie par $p(X, k)= X[1 : k]$, k étant la taille du préfixe de X, et une relation d'équivalence basée sur la notion de préfixe θ_k définie sur les itemsets comme suit : $\forall X, Y \subseteq I, X \equiv_{\theta k} Y \Leftrightarrow p(X, k) = p(Y, k)$. Ainsi deux itemsets sont dans la même k-classe s'ils partagent le même préfixe de taille k.

Les algorithmes verticaux effectuent une recherche d'itemset fréquents (fermés fréquents) dans un espace de recherche appelé arbre IT (arbre itemset-tidset) comme le montre la figure 2.4. Alors que la majorité des précédentes méthodes effectuent une recherche dans l'espace d'itemsets, les algorithmes verticaux comme *Eclat* explorent simultanément l'espace des itemsets et l'espace des transactions. Chaque nœud dans l'arbre-IT, appelé nœud-IT, représenté par la paire itemset-tidset, $X \times t(X)$, est en réalité une classe d'équivalence par préfixe. Tous les enfants du nœud X (représenté par $X \times t(X)$) appartiennent à sa classe d'équivalence par préfixe puisqu'ils partagent tous le même préfixe X. la classe d'équivalence par préfixe est représentée par $[P] = \{l_1, l_2, \ldots l_n\}$, où P désigne le nœud parent

(le préfixe) et chacun des l_i désigne l'item, représentant le nœud Pl_i x t(Pl_i). Un exemple simple est celui de la racine de l'arbre-IT défini par la figure 2.4 : [] = {*A*, *B*, *C.D*, *E*}. L'enfant de la racine le plus à gauche consiste en [*A*] les itemsets contenant A comme préfixe. Chaque membre de la classe représente un seul enfant du nœud parent. Une classe représente des items dont le préfixe peut être étendu pour obtenir avec, un nouveau nœud fréquent. De toute évidence, aucun sous arbre d'un préfixe rare ne doit être examiné. La force de cette approche est qu'elle divise l'espace de recherche original en sous-problèmes indépendants. Quand tous les enfants directs d'un nœud *X* (représenté par *X* x t(*X*)) sont connus, il peut être traité comme un problème entièrement nouveau. Les itemsets sous le nœud *X* peuvent être énumérés simplement en les préfixant de *X*, ...etc. Par ailleurs, à partir de la propriété 2.3 qui stipule que si *X* est sous ensemble de *Y*, la cardinalité de la liste des transactions de *Y* est plus petite que celle de *X*, il est conséquent d'affirmer donc que les cardinalités des listes de transactions intermédiaires diminuent au fur et à mesure que s'effectue la descente dans l'arbre-IT. Ceci entraine une intersection et un calcul de support très rapide.

2.3.2.3 Représentation verticale

Il est nécessaire d'accéder au contexte d'extraction pour déterminer de support d'un ensemble d'itemsets. Les algorithmes d'extraction d'itemsets utilisent des tables binaires. Un contexte d'extraction peut être facilement représenté par une matrice binaire bidimensionnelle. Deux représentations sont utilisées pour l'implémentation d'une telle matrice. La représentation horizontale utilisée par les algorithmes à niveaux et la représentation verticale utilisée par les algorithmes verticaux et qui consiste en l'ensemble d'items avec leurs listes de tids (transactions). La figure 2.3 illustre les deux représentations de la base de données (la représentation horizontale et la représentation verticale).

Pour calculer le support d'un itemset *X* dans une représentation horizontale, un balayage du contexte est nécessaire pour vérifier pour chaque transaction T si $X \subseteq T$. Une structure de donnée telle qu'un arbre peut être utilisé à cet effet. Dans une représentation verticale par contre, le calcul de support peut être fait par une simple opération d'intersection. Selon

(Zaki et al., 2000) le support d'un quelconque k-itemset peut être déterminé par l'intersection de deux quelconques de ses sous ensembles de taille k-1. La cardinalité de la liste de tids obtenue suite à cette opération d'intersection permet de statuer sur la fréquence ou la rareté du k-itemset en question. Ceci revient à dire que pour calculer le support d'un itemset de niveau donné (taille donnée) dans l'arbre-IT, seuls ses deux premiers sous-ensembles selon l'ordre lexicographique, obtenus au niveau précédent, sont utilisés dans l'opération d'intersection.

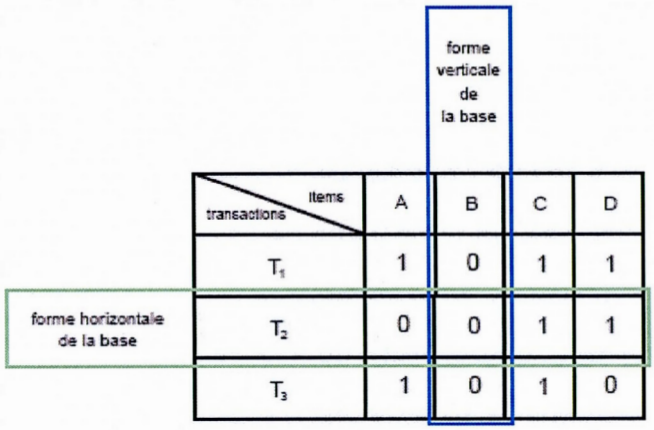

Figure 2.4 Formes Horizontale et verticale de la base de données

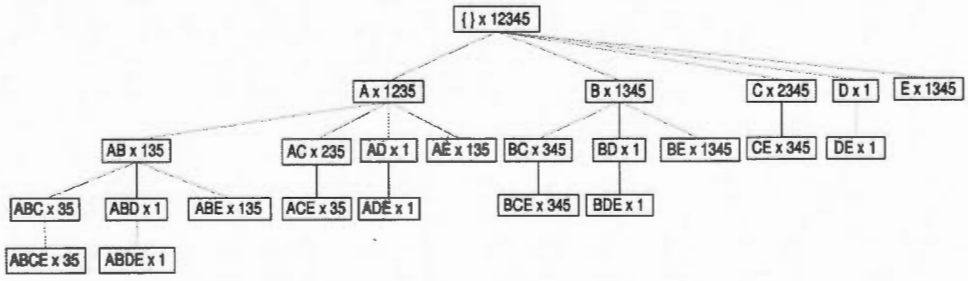

Figure 2.5 Arbre de recherche Itemset-Tidset du contexte d'extraction C défini par le tableau 2.1

2.3.3 Famille des algorithmes hybrides

Ces algorithmes sont des combinaisons d'algorithmes à niveaux et d'algorithme verticaux. L'algorithme *Eclat-Z* et *Charm-MFI* font parties de cette famille d'algorithmes. *Eclat-Z* est composé de deux parties : d'abord les itemsets fréquents sont découverts par *Eclat* ensuite ils sont post-traités d'une façon séquentielle ou à niveaux, c'est-à-dire dans l'ordre ascendant de leurs tailles.

2.3.4 Autres algorithmes

En plus des trois types d'algorithmes présentés dans les sections précédentes, il existe un autre type d'algorithme comme *FP-Growth*. *FP-Growth* a été proposé par (Han et al., 2 000). Il est innovant par rapport aux autres algorithmes de recherche de règles associatives presque tous basés sur Apriori. L'algorithme utilise pour le stockage des motifs, une structure de données compacte appelée, Frequent-Pattern tree et qui apporte une solution au problème de la fouille de motifs fréquents dans une grande base de données transactionnelle. Stocker l'ensemble des items fréquents de la base de transactions dans une structure compacte, supprime la nécessité de devoir balayer de façon répétée la base de transactions. De plus, trier les éléments dans la structure compacte, accélère la recherche des motifs.

Un *FP-tree* est composé d'une racine nulle et d'un ensemble de nœuds préfixés par l'item représenté. Un nœud est composé par : le nom de l'item, le nombre d'occurrence de transaction où figure la portion de chemin jusqu'à ce nœud, un lien inter-nœud vers les autres occurrences du même item figurant dans d'autres séquences de transactions. Une table d'en-tête pointe sur la première occurrence de chaque item. L'avantage de cette représentation des données est qu'il suffit de suivre les liens inter-nœuds pour connaître toutes les associations fréquentes où figure l'item fréquent.

Plusieurs variations de *FP-Growth* ont été proposés par la suite dans la littérature, entre autre l'algorithme *Closet* (Pei et al., 2000) et *Closet*$^{+}$ (Wang et al., 2003) conçus pour l'extraction des itemsets fermés fréquents uniquement

2.4 Génération des règles d'association

La génération des règles d'association est réalisée à partir de l'ensemble des itemsets fréquents F extrait du contexte d'extraction avec un seuil minimal de support *minSupport*. Une règle d'association r est une règle entre itemsets de la forme r : $I_2 \rightarrow (I_1\text{-}I_2)$ dans laquelle, I_1et I_2 sont des itemsets fréquents appartenant à F et $I_2 \subset I_1$. I_2 est appelé antécédent ou prémisse de la règle r, et $(I_1\text{-}I_2)$ est appelé conséquence ou conclusion de la règle r. Les règles d'association valides ou fortes sont celles qui ont une confiance (Support (I_1)/Support (I_2)), au moins égale au seuil minimal de confiance *minConfiance* fixé par l'utilisateur.

Le principe général de génération de règles d'association est donc simple. Pour chaque itemset fréquent I_1 dans F de taille supérieure ou égale deux, tous ses sous ensembles I_2 dans F sont déterminés et le rapport Support (I_1)/Support (I_2) est calculé. Si la valeur de ce rapport est supérieure ou égale au seuil minimal de support *minConfiance* défini par l'utilisateur, alors la règle $I_2 \rightarrow (I_1\text{-}I_2)$ est générée. Un algorithme efficace de génération de règles a été proposé dans la littérature par (Agrawal et al., 1994), il s'agit de l'algorithme *GEN-Règles* qui est décrit dans la section suivante.

2.4.1 Génération des règles d'association avec *GEN-Règles*

L'algorithme *GEN-Règles* est l'un des premiers algorithmes proposés dans la littérature pour la génération des règles d'associations, il se base sur la propriété 2.4 pour réduire le nombre d'opérations à effectuer sur le support afin de déterminer les règles fortes à générer.

- **Propriété 2.4**

Étant donné un itemset I_1, le support d'un sous ensemble I_2 de I_1 est supérieur ou égal au support de I_1.

Soient I_1, I_2, I_3 trois itemsets fréquents appartenant à F tels que $I_3 \subset I_2 \subset I_1$. D'après la propriété 2.4, support(I_3) ≥ support(I_2) ≥ support(I_1) et par conséquent la confiance de la règle $r_1 : I_3 \rightarrow (I_1 - I_3)$ est inférieure ou égale à la confiance de la règle $r : I_2 \rightarrow (I_1 - I_2)$. Donc si la règle r n'est pas forte alors la règle r_1 ne l'est pas non plus. Réciproquement si la règle r_1 est

valide ou forte, r aussi est valide. Cette propriété permet de diminuer le nombre de règles d'association à tester par l'algorithme.

Algorithme 2.3 *GEN-Règles*

Entrées F : ensemble des *itemsets* fréquents, *minConfiance* : confiance minimale définie par l'usager

Sortie RA : ensemble des règles d'association fortes dans C

1) $RA \leftarrow \varnothing$

2) **pour chaque** *itemset* fréquent $I_2 \in F$ et $|I_2| > 1$ **faire**

3) **pour chaque** sous-ensemble non vide $I_i \subset I_2$ **faire**

4) $sup \leftarrow sup\ (I_2)$

5) $conf \leftarrow sup\ (I_2) / sup\ (I_i)$

6) **si** $conf \geq minConfiance$ **alors**

7) $RA \leftarrow RA \cup \{r : I_i \rightarrow I_2 \setminus I_i\ sup, conf\}$

8) **fin si**

9) **fin pour**

10) **fin pour**

11) retourner RA

L'algorithme *GEN-Règles* suit le principe général, il consiste donc à parcourir l'ensemble F des itemsets fréquents du contexte et à identifier pour chaque itemset fréquent I_2 de taille strictement supérieure à 1 (ligne 2), tous les sous ensembles $I_i \subset I_2$ tels que $I_i \neq \varnothing$ (ligne 3). Le support (ligne 4) et la confiance (ligne 5) de la règle candidate $r : I_i \rightarrow (I_2 - I_i)$ sont ensuite calculés. Finalement seules les règles dont la confiance est supérieure ou égale au seuil minimal de confiance *minConfiance* sont conservées (lignes 6-8).

2.4.2 Complexité de l'approche classique de génération de règles

Le problème d'extraction des règles d'association est un problème exponentiel dans la taille des itemsets fréquents. En effet $2^{|f|} - 2$ règles d'association, non triviales peuvent être générées à partir d'un seul itemset fréquent f de taille supérieure à 1. Toutefois cette extraction se fait sans accès à la base de données et les temps d'exécution de cette étape sont faibles comparés au temps d'extraction des itemsets fréquents. Cependant, pour un ensemble d'itemsets fréquents extraits, F le nombre de règles d'association fortes pouvant être générées à partir de F est de l'ordre de $\sum_{f \in F} 2^{|f|} - 2$(Bastide et al., 2000). Ce nombre varie en général de quelques dizaines à des millions de règles d'association parce que le nombre d'itemsets fréquents extraits et leurs tailles sont élevés dans la plupart des contextes d'extraction. Ce nombre important de règles d'association extraites constitue un problème majeur pour la pertinence et l'utilité du résultat, accentué par la présence d'un grand nombre de règles redondantes du point de vue de l'information convoyée (Pasquier, 2000).

Beaucoup de travaux de recherche ont été menés dans ce contexte pour trouver des méthodes qui permettent de réduire l'ensemble des règles d'association générées et améliorer la pertinence du résultat. Plusieurs approches ont été proposées dans la littérature, permettant de réduire ces règles ou de sélectionner un sous ensemble des règles. Un survol de ces approches, non exhaustif est présenté dans la prochaine section.

2.5 Réduction de l'ensemble des règles d'association

Les approche proposées dans la littérature pour éliminer la redondance au niveau des règles d'extraction générées à partir des itemsets fréquents, sont diverses et appartiennent soit à la catégorie des approches orientées structure de données ou alors à celles orientées utilisateurs.

2.5.1 Réduction de règles par approches orientées structure de données

Les approches orientés structures de données se basent sur les propriétés structurelles des règles d'association pour réduire l'ensemble des règles d'association extraites. Quatre

principales catégories d'approches orientées structures ont été recensées dans la littérature qui sont résumées comme suit :

2.5.1.1 Règles d'association généralisées

Les règles d'association généralisées (Srikant et al., 1995) également appelées règles d'association multi-niveaux (Han et al., 1995) sont définies en utilisant une taxonomie des items du contexte. Cette taxonomie est un graphe dirigé acyclique dont les sommets sont les items et les arcs sont des relations *est-a* entre les items (Pasquier, 2000). S'il existe un arc d'un item i_1 vers l'item i_2 dans la taxonomie alors i_1 est appelé père de i_2 et i_2 fils de i_1, et si cet arc existe dans la fermeture transitive de la taxonomie, alors i_1 est appelé ancêtre de i_2 et i_2 est appelé descendant de i_1.

Étant donné une taxonomie T associée à un contexte d'extraction C= (O, I, R). Les règles d'association généralisées pouvant être extraites du contexte C, sont les règles d'association entre itemsets pouvant appartenir à différents niveaux de T. Par exemple la règle " r : lait → céréales" est appelé sur-règles des deux règles " r_1 : lait écrémé→ céréales" et " r_2 : lait entier→ céréales ", car les items formant les antécédents de r_1 et de r_2 sont des descendants de l'item formant l'antécédent de r. Dans (Han et al., 1995 ; Srikant et al., 1995) les auteurs proposent de générer uniquement la sur-règle r si son support est supérieur à la somme des supports de r_1 et de r_2 et sa confiance est supérieure aux confiances de r_1 et de r_2.

Cette méthode nécessite la création d'une taxonomie des items et ne permet pas de supprimer les règles d'association redondantes. De plus lorsque les données de la base sont denses ou étroitement corrélées, cette méthode ne permet pas de réduire significativement le nombre de règles extraites (Bastide et al., 2000).

2.5.1.2 Utilisation d'autres mesures statistiques

L'utilisation des mesures statistiques autre que la confiance pour déterminer la précision des règles d'association a fait l'objet de nombreuses études (Brin et al., 1997 ; Piatetsky-shapiro, 1991 ; Silverstein et al., 1998). Deux mesures des plus intéressantes ont été retenues, il s'agit

de la mesure de conviction définie par (Brin et al., 1997) et la mesure du $\chi 2$ (Brin et al., 1997; Silverstein et al., 1998).

Soit P(I_m) la probabilité d'occurrence d'un itemset $I_m \subseteq I$ dans un objet o $\in O$ du contexte d'extraction $C=(O, I, R)$.

La mesure de conviction permet de mesurer la déviation de la dépendance entre la présence de la prémisse et l'absence de la conclusion d'une règle dans les objets. Cette mesure est basée sur l'hypothèse qu'une règle $r : I_1 \rightarrow I_2$ peut être exprimée en logique par $\neg(I_1 \wedge \neg I_2)$. La conviction de la règle est définie par (Brin et al., 1997) :

$$\text{Conviction } (r) = \text{P } (I_1) \wedge \text{P } (\neg I_2) \,/\, \text{P } (I_1 \wedge \neg I_2)$$

P $(I_1 \wedge \neg I_2)$ est la probabilité réelle de la présence d'I_1 et l'absence d'I_2 dans un objet. P $(I_1) \wedge$ P $(\neg I_2)$ indique la valeur qu'aurait cette probabilité si la présence d'I_1 et l'absence d'I_2 n'étaient pas en relation. La conviction serait égale à 1 si les deux événements étaient sans relation et serait supérieure à 1 s'ils étaient liés. Par conséquent, la dépendance entre les itemsets est d'autant plus importante que la valeur de la conviction est éloignée de 1 (Pasquier, 2000).

La mesure du $\chi 2$ consiste à spécifier le degré de dépendance entre différents items d'un itemset en comparant la distribution réelle de leur occurrence avec la distribution attendue de leur occurrence sous prémisse d'une distribution normale de l'occurrence de chaque item. L'utilisation du $\chi 2$ pour extraire les itemsets fréquents dont le degré de dépendance des items est supérieur ou égal à un seuil donné a été étudiée dans (Brin et al., 1997, Silverstein et al., 1998). Soit : $R = \{i_1, \neg i_2\}$ x... x $\{i_m, \neg i_m\}$ le produit cartésien des ensembles d'événements correspondant à la présence et l'absence des items $\{i_1, i_2, \ldots i_m\}$ dans un objet. Un élément $r \in R$ correspond à un itemset sous-ensemble de $\{i_1, i_2, \ldots i_m\}$ O(r) est le nombre d'objets du contexte correspondant à l'instance r. E[r] représente la valeur attendue de l'instance r. La mesure $\chi 2$ est définie comme suit :

$$\chi 2 = \sum_{r \in R} ((O(r) - E[r])^2 / (E[r])).$$

La valeur de $\chi 2$ obtenue définit la déviation normalisée par rapport à la valeur attendue. Elle détermine si les m items de l'itemset $\{i_1, i_2, \ldots i_m\}$ sont indépendants deux à deux. Si tous les items de l'itemset sont indépendants la valeur de $\chi 2$ serait nulle.

Toutefois, qu'il s'agisse de la mesure de conviction ou la mesure du $\chi 2$, le calcul de la valeur de ces deux mesures nécessite des temps d'exécution importants qui peut engendrer des problèmes de performance (Bastide et al., 2000).

2.5.1.3 Mesures de déviation

Les mesures de déviations sont des mesures de distances entre les règles d'associations, en fonction de leurs supports et de leurs confiances. Ces mesures peuvent être utilisées pour, identifier les règles fortement semblables et caractérisées par une faible distance entre elles, ensuite classer ou supprimer certaines de ces règles en fonction des mesures de déviation qui leurs sont associées(Bayardo et al., 1999; Toivonen et al., 1995). Cette méthode nécessite une comparaison des règles extraites deux à deux et entraine une perte d'informations.

Une autre utilisation de cette mesure de déviation consiste à identifier les règles d'association qui sont inattendues pour l'utilisateur et qui apportent donc une connaissance nouvelle et importante (Heckerman, 1996 ; Piatetsky-Shapiro et al., 1994 ; Silberschatz et al., 1996). Les connaissances de l'utilisateur sont représentées par des modèles probabilistes auxquels sont comparées les règles d'association extraites. La déviation de la règle correspond à la différence entre la valeur de la règle dans le modèle et sa valeur réelle dans le contexte. Cette méthode nécessite néanmoins la définition par l'utilisateur de ces connaissances, ce qui est très souvent très complexe, et les calculs des mesures de déviations requièrent beaucoup d'opérations ce qui entraine des temps d'exécutions importants(Bastide et al., 2000).

2.5.1.4 Couverture structurelle

Dans (Toivonen et al., 1995) une couverture structurelle d'un ensemble de règles d'association valides extrait *AR*, est construite en supprimant de cet ensemble les règles

redondantes d'un point de vue syntaxique. Pour cela les sous-ensembles de l'ensemble AR sont construits puis traités. Un sous-ensemble de l'ensemble AR est constitué de toutes les règles valides possédant le même itemset conclusion. La sélection des règles à supprimer à l'intérieur du sous-ensemble est basée sur l'hypothèse que pour deux règles $r_1 : A_1 \rightarrow B$ et $r_2 : A_2 \rightarrow B$ telles que $A_2 \subset A_1$ (r_2 sous règle de r_1). En effet, l'ensemble des objets contenant $A_2 \cup B$ est un sur-ensemble de l'ensemble des objets contenant $A_1 \cup B$. La couverture structurelle Δ d'un ensemble AR des règles d'association valides est définie comme suit :

$\Delta = \{A \rightarrow B \in AR \mid$ il n'existe pas $A_1 \in AR$, telle que $A_1 \subset A\}$.

L'ensemble Δ ainsi définie contient les règles d'association de l'ensemble AR qui n'ont aucune sur-règle dans ce même ensemble. Par conséquent la couverture structurelle contient les règles d'association les plus générales au point de vue syntaxique de l'ensemble AR.

Toutefois même si le nombre de règles d'association extraites est réduit, cette méthode ne tient pas compte de la confiance des règles d'association et ceci pose un problème car la confiance d'une règle ne peut être dérivée de la confiance de ses sur-règles ou de ses sous-règles (Pasquier, 2000)

2.5.2 Réduction de règles par approches orientées utilisateur

Les approches orientées utilisateur pour la réduction de l'ensemble des règles d'association requièrent une intervention de l'utilisateur pour définir les critères de sélection des règles qui figureront dans l'ensemble résultat. Trois principales catégories d'approches orientés utilisateur ont été recensés dans la littérature qui sont résumées comme suit :

2.5.2.1 Templates

Les templates sont des expressions booléennes permettant de sélectionner un sous-ensemble de l'ensemble des règles d'association fortes (Klemettinen et al., 1994). Ce sous-ensemble est défini en sélectionnant de l'ensemble des règles valides, uniquement les règles d'association vérifiant les critères spécifiés dans le modèle. Les critères spécifient les

contraintes d'occurrence ou de non occurrence des items de la prémisse et de la conclusion des règles.

Cette méthode de traitement postériori, bien qu'elle facilite la visualisation des règles d'association extraites par la visualisation par groupes, ne permet pas l'élimination toutefois des règles d'association redondantes.

2.5.2.2 Opérateur MINE-RULE

Dans (Meo et al., 1996), les critères de sèlection des règles sont définis par un opérateur nommé *MINE-RULE*. Cet opérateur est une extension du langage de requêtes SQL implémenté dans les SGBD relationnels. L'extraction des règles d'association est réalisée à partir des tuples des relations de la base de données par une requête, instance de *MINE-RULE*. Les règles d'association générées à partir des itemsets fréquents selon les critères définis par la requête.

Toutefois, la définition des requêtes correspondant aux besoins de l'utilisateur par l'opérateur *MINE-RULE* nécessite des connaissances approfondies du langage de requête SQL (Bastide et al., 2000)

2.5.2.3 Contraintes sur les items

Les contraintes sur les items (Bayardo et al., 1999 ; Lakshmanan et al., 1998 ; Srikant at al., 1997) sont des expressions portant sur la prémisse et la conclusion des règles. Elles sont définies par l'utilisateur qui spécifie la forme des règles à extraire. Ces contraintes sont utilisées lors de la phase d'extraction des itemsets fréquents afin de limiter l'espace de recherche aux itemsets candidats permettant de générer les règles vérifiant les contraintes. Toutefois cette approche ne fournit qu'un résultat partiel des règles d'associations et ne permet pas non plus d'éliminer les règles d'association redondantes (Pasquier, 2000).

2.6 Conclusion

Nous avons présenté dans ce chapitre, la méthode classique d'extraction des règles d'association, divisée dans la littérature en deux phases, la phase d'extraction de l'ensemble des itemsets fréquents suivie de la phase de génération des règles d'association. Les résultats obtenus des expérimentations effectuées, par les chercheurs sur les différentes approches proposées, et présentées dans le présent chapitre, aussi bien dans le cadre de l'extraction des itemsets fréquents que dans le cadre de génération des règles d'association, montrent de toute évidence les limitations de l'approche classique. Le problème majeur dans cette approche, réside dans l'étape d'extraction des itemsets fréquents. Le temps d'exécution de cette étape est extrêmement couteux que l'approche adoptée soit à niveau, verticale, hybride ou alors autre. En effet l'efficacité d'Apriori et ses variations diminuent sur des données denses ou corrélées à cause du nombre de candidats à considérer et le nombre de balayages à réaliser, alors que l'efficacité des algorithmes verticaux et hybrides proposées pour l'extraction des itemsets fréquents diminue lors de la transmission des listes de transactions (généralement assez importantes) de chaque item aux autres processeurs, et finalement l'efficacité de *FP-Growth* n'est significative que sur des données éparses ou avec des seuils minimaux de support et confiances élevés.

Comme le nombre de règles d'association générées dans l'étape suivante est exponentiel dans la taille des itemsets fréquents extraits, l'ensemble des règles d'associations valides produites est prohibitif et inexploitable par l'utilisateur. Ce nombre est élevé même sur des contextes d'extractions non significatifs puisque selon (Bastide et al.,, 2000), $2^{|f|} - 2$ règles d'associations valides peuvent être générées à partir d'un seul itemset fréquent *f*, ceci revient à dire que les règles générées à partir des itemsets fréquents est toujours trop important par rapport à l'information convoitée, et selon (Pasquier, 2000) ce nombre élevé de règles est dû à la présence d'une proportion forte de règles d'associations redondantes. Plusieurs travaux ont été menés dans le domaine pour développer des techniques afin de réduire le nombre de règles d'associations extraites, cependant les résultats obtenus par l'analyse formelle des concepts (AFC) qui fait l'objet du prochain chapitre, a permis de définir un sous-ensemble de l'ensemble des règles d'associations valides appelés bases informatives . La génération de ces bases informatives se fait par une extraction efficace des itemsets fermés fréquents et leurs générateurs minimaux associés. Cet ensemble de bases de règles d'association a une

taille largement inférieure par rapport à l'ensemble des règles d'association valides et a l'avantage d'être complet du point de vue de la connaissances tout en étant réduit au point de vu de la taille. Les bases informatives sont longuement décrites dans le prochain chapitre.

CHAPITRE III

ANALYSE FORMELLE DES CONCEPTS ET RÈGLES D'ASSOCIATION

3.1 Introduction

Analyse Formelle de Concepts(AFC) (Wille, 1982) est un formalisme mathématique pour l'analyse de données, la représentation de connaissance et la visualisation des connaissances (Assaghir, 2011). L'idée de base de l'AFC est d'extraire à partir des données, des concepts regroupant des objets et leurs propriétés(ou attributs) et construire une hiérarchie, à partir de ces concepts et de la relation de généralisation/ spécialisation définie entre ces concepts. Chaque concept de la structure est un couple constitué d'une extension représentant un sous ensemble d'objets de la base, et une intention représentant les propriétés communes aux objets du concept. Cette structure hiérarchique de concepts forme un treillis appelé treillis de concepts (Ganter et al., 1999) et offre une représentation compacte de l'information, propice à l'exploration des connaissances. Elle s'est avérée être un cadre théorique intéressant pour la fouille de données spécialement pour l'extraction des règles d'associations.

Dans ce contexte, les approches issues de la théorie des concepts formels, proposent de ne générer qu'un sous-ensemble compact et générique de règles d'associations. Ces approches sont basées sur l'extraction des itemsets fermés fréquents qui constituent, un ensemble générateur, appelé base pour l'ensemble des itemsets fréquents. Cela signifie que les itemsets fréquents et leurs supports peuvent être générés à partir des itemsets fréquents sans accéder à la base de données.

Par ailleurs, avec l'utilisation de ces itemsets fermés accompagné de leurs générateurs minimaux associés, un sous ensemble des règles d'association valides appelé base peut être

défini, à partir duquel les autres règles d'associations (règles d'association redondantes) peuvent être dérivées. L'ensemble de ces bases génériques de règles d'association a une taille largement inférieure à la taille de l'ensemble de toutes les règles. Ce sous-ensemble a l'avantage d'être complet au point de vue de la connaissance (pas de perte), tout en étant réduit du point de vue de la taille (Baside et al., 2000 ; Stumme et al., 2001 ; Benyahia et al., 2003).

Un état de l'art de l'extraction des règles d'association par l'approche d'extraction des itemsets fermés fréquents est présenté dans ce chapitre. Ce dernier est entamé par l'introduction aux fondements mathématiques de la théorie des concepts formels, suivi par la description du principe d'extraction des itemsets fréquents et appuyé avec un des algorithmes adoptant l'approche des fermés, l'algorithme *Close*, qui est utilisé pour valider notre algorithme d'expansion proposé au dernier chapitre de ce présent mémoire. Ce chapitre est clôturé par la présentation des bases génériques des règles d'associations qui constituent des bases de l'ensemble des règles d'association valides et qui ont permis d'améliorer la pertinence et l'utilité de l'ensemble des règles extraites.

3.2 Analyse formelle des concepts

Dans ce qui suit, nous présentons quelques résultats clés provenant de la théorie des concepts formels et leur application dans le contexte de dérivation des règles d'association. Nous utilisons le cadre théorique présenté dans (Ganter et al., 1999). Nous rappelons brièvement les notions de base de ce cadre théorique.

Quelques définitions sont présentées sur, les opérateurs de fermeture, la connexion de Galois, la fermeture de connexion utilisées par la suite pour définir le treillis des itemsets fermés, sur les concepts utilisés pour définir le treillis de Galois et sur les classes d'équivalences induites par la fermeture sur l'ensemble des items du contexte. Ces définitions ont été prises dans les articles de références (Barbut et al., 1970 ; Davey et al., 1992 ; Ganter et al., 1999)

Soit $K = (O, I, R)$ un contexte d'extraction dans lequel O et I représentent respectivement l'ensemble fini d'objets et l'ensemble fini d'items et R la relation binaire qui relie l'ensemble des objets O à l'ensemble d'items I.

Tableau 3.1 [3]Contexte d'extraction K utilisé par les algorithmes d'extraction *IFF* (Gasmi et al., 2007)

	A Œufs	*C* Pains	*D* Sucre	*T* Beurre	*W* Lait
1	x	x		x	x
2		x	x		x
3	x	x		x	x
4	x	x	x		x
5	x	x	x	x	x
6		x	x	x	

3.2.1 Opérateurs de fermeture

- **Définition 3.1** (Opérateur de fermeture)

Soit $(E, \leq)$ un ensemble partiellement ordonné. Une application u de $(E, \leq)$ dans $(E, \leq)$ est appelée opérateur de fermeture si et seulement si elle satisfait les propriétés suivantes pour tous les sous-ensembles $S, S_1, S_2 \subseteq E$

[3] Les items du contexte K seront représentés dans l'exemple d'application de l'algorithme Close, comme suit : "Oeufs" par A, "Pains" par C, "Sucre" par D, "Beurre" par T et " Lait " par W.

1) Isotonie: $S \leq S_1 \Rightarrow u(S) \leq u(S_1)$,

2) Extensivité: $S \Rightarrow u(S)$,

3) Idempotence : $u(u(S)) = u(S)$

Étant donné un opérateur de fermeture u sur un ensemble ordonné $(E, \leq)$, un élément $x \in E$ est un élément fermé si l'image de x par l'opérateur de fermeture est lui-même : $u(x)=x$.

3.2.2 Connexion de Galois

Soit l'application f de l'ensemble des parties de O dans l'ensemble des parties de I, qui associe à un ensemble d'objets $O_1 \subseteq O$, l'ensemble des items $i \in I$ communs à tous les objets $o \in O_1$: $f: \mathcal{P}(O) \rightarrow \mathcal{P}(I)$

$$f(O_1) = O_1' = \{i \in I \mid \forall\, o \in O_1 \text{ nous avons } (o, i) \in R\}$$

Soit l'application g de l'ensemble des parties de I dans l'ensemble des parties de O, qui associe à tout ensemble d'item (itemset) $I_1 \subseteq I$, l'ensemble des objets o $\in O$ contenant tous les items i $\in I_1$ $g : \mathcal{P}(I) \rightarrow \mathcal{P}(O)$

$$g(I_1) = I_1' = \{o \in O \mid \forall\, i \in I_1 \text{ nous avons } (o, i) \in R\}.$$

Le couple d'applications (f, g) est une connexion de Galois entre l'ensemble des parties de O et l'ensemble des parties de I

- **Définition 3.2** (Connexion de Galois)

Le couple (f, g) forme une connexion de Galois entre l'ensemble des parties de O et l'ensemble des parties de I si : $\forall\, O_1 \subseteq O$, $I_1 \subseteq I$, $O_1 \subseteq g(I_1)$ si et seulement si $I_1 \subseteq f(O_1)$.

- **Propriété 3.1** (Connexion de Galois)

Étant donné la connexion de Galois (f, g) entre l'ensemble des parties de O et l'ensemble des parties de I, les propriétés suivantes sont vérifiées quelques soient les ensembles $I_1, I_2, I_3 \subseteq I$ et $O_1, O_2, O_3 \subseteq O$

1) $O_1 \subseteq O_2 \Rightarrow f(O_1) \supseteq f(O_2)$

2) $I_1 \subseteq I_2 \Rightarrow g(I_1) \supseteq g(I_2)$

3) $O_1 \subseteq g(I_1) \Leftrightarrow I_1 \subseteq f(O_1)$

- **Définition 3.3** (Fermeture de la connexion de Galois)

Soient $(\mathcal{P}(I), \subseteq)$ et $(\mathcal{P}(O), \subseteq)$ deux ensembles ordonnées (ensemble des parties de I et ensemble de parties de O doté chacun de la relation $\subseteq$). Les compositions d'applications fog de $(\mathcal{P}(I), \subseteq)$ dans $(\mathcal{P}(I), \subseteq)$, et gof de $(\mathcal{P}(O), \subseteq)$ dans $(\mathcal{P}(O), \subseteq)$ sont des opérateurs de fermeture de la connexion de Galois. $fog(I_1)$ et $gof(O_1)$ sont notées respectivement I_1" et O_1"

- **Propriété 3.2** (Fermeture de la connexion de Galois)

Étant donnée la connexion de Galois (f, g), les propriétés suivantes sont vérifiées à quelque soient les ensembles $I_1, I_2, I_3 \subseteq I$, $O_1, O_2, O_3 \subseteq O$

1) $I_1 \subseteq I_1$" et $O_1 \subseteq O_1$"

2) $(I_1")" = I_1"$ et $(O_1")" = O_1"$

3) $I_1 \subseteq I_2 \Rightarrow I_1" \subseteq I_2"$, et $O_1 \subseteq O_2 \Rightarrow O_1" \subseteq O_2"$

4) $(I_1')" = I_1'$ et $(O_1')" = O_1'$

3.2.3 Treillis des itemsets fermés

Soient (f, g) une connexion de Galois entre l'ensemble des parties de O et l'ensemble des parties de I et $h = fog$, l'opérateur de fermeture de (f, g).

- **Définition 3.4** (Fermeture d'un itemset).

La fermeture d'un itemset I_1, notée $I_1" = h(I_1) = fog(I_1)$

Exemple: l'itemset {œufs, beure, lait, pain} extrait du contexte K (tableau 3.1) avec *minSupport*=0.5, est un itemset fermé fréquent.

- **Définition 3.5** (itemsets fermé/ ensemble des itemsets fermés)

Un itemset $I_1 \subseteq I$ est dit fermé si et seulement si $I_1" = I_1$ ($h(I_1) = I_1$).

Exemple (Itemset fermé) :

À partir de la définition de l'itemset fermé, nous déduisons celle de l'ensemble F des itemsets fermés :

$F = \{ I_1 \subseteq I \mid I_1 \neq \emptyset \wedge I_1" = h(I_1) = I_1 \}$

- **Définition 3.6** (Treillis des itemsets fermés)

Soit F l'ensemble des itemsets fermés d'un contexte $K = (O, I, R)$ selon l'opérateur de fermeture h, de la connexion de Galois (f, g) définie entre l'ensemble $\mathcal{P}(O)$ et l'ensemble $\mathcal{P}(I)$. $\Gamma = (F, \subseteq)$ est un treillis complet, appelé treillis des itemsets fermés.

- **Définition 3.7** (Ensemble des itemsets fermés fréquents)

Soit *minSupport* le seuil minimal de support fixé par l'utilisateur pour l'extraction des règles d'associations à partir du contexte C. L'ensemble FF des itemsets fermés fréquents dans C est le suivant :

$FF = \{I_1 \subseteq I \mid I_1 \neq \emptyset \wedge I_1'' = h(I_1) = I_1 \wedge \text{Support}(I_1) \geq minSupport\}$

- **Définition 3.8** (Ensemble des itemsets fermés fréquents maximaux)

Soit *FF* l'ensemble des itemsets fermés fréquents extraits du contexte *C* avec un seuil minimal de support *minSupport* fixé par l'utilisateur. L'ensemble *FFM* des itemsets fermés. fréquents maximaux dans *C* est le suivant :

$FFM = \{Il \subseteq I \mid I_1 \in FF \wedge \forall I_2 \supset I_1 \text{ Support}(I_2) < minSupport\}$

3.2.4 Concepts formels

- **Définition 3.9** (Concept formel)

Un concept formel est une paire $c = (O_1, I_1)$, où O_1 et I_1 sont reliés avec l'opérateur de la correspondance de Galois, c'est-à-dire $O_1' = I_1$ et $I_1' = O_1$.L'ensemble O_1 est appelé extension et l'ensemble I_1 est appelé intention.

Exemple : la paire ({1, 3, 5}, {œufs, beure, lait, pain}) est un concept formel (contexte *K* du tableau 3.1 et *minSupport*=0.5)

- **Définition 3.10** (Générateur minimal)

Un itemset *g* est dit générateur minimal d'un itemset fermé *f* si et seulement si $g'' = f$ et il n'existe pas de $g_1 \subset g$ tels que $g_1'' = f$.

Exemple : soit l'itemset fermé {œufs, beure, lait, pain}, ses générateurs minimaux sont {œufs, beure} et {lait, beure}

- **Définition 3.11** (Itemset pseudo-fermé)

Un itemset $I_1 \in I$ est un itemset pseudo-fermé s'il n'est pas fermé et s'il contient les fermetures de tous ses sous-ensembles qui sont des itemsets pseudo-fermés. Ainsi l'ensemble des itemsets pseudo-fermés $PF = \{ I_1 \in I \mid I_1 \neq I_1' \wedge \forall I_2 \subset I_1$ tels que $I_2 \in PF, I_2' \subset I_1\}$

Exemple: {œufs, pain} est un itemset pseudo-fermé puisque {œufs, pain} contient {pain} qui est la fermeture de l'itemset pseudo-fermé Ø

- **Définition 3.12** (Classe d'équivalence induite par la fermeture)

L'opérateur de fermeture(") induit une relation d'équivalence sur l'ensemble des parties de I du contexte K (O, I, R), c'est-à-dire l'ensemble de parties est partitionné en sous-ensembles disjoints appelés aussi classes. Dans chaque classe, tous les itemsets ont la même valeur de support. Les générateurs minimaux d'une classe sont les itemsets incomparables les plus petits, tandis que l'itemset fermé est l'itemset le plus large de cette classe.

Exemple : les itemsets {œufs}, {œufs, pain}, {œufs, lait} et {œufs, pain, lait} appartiennent à une même classe d'équivalence puisqu'ils ont la me fermeture.

- **Définition 3.13** (Treillis de concepts ou treillis de Galois)

Étant donné le contexte d'extraction K, l'ensemble des concepts formels C_K est un treillis complet $\Gamma_C = (C, \leq)$, appelé *treillis de concepts formels* (*de Galois*), quand l'ensemble C_K est considéré avec la relation d'inclusion entre les itemsets (Barbut et al., 1970; Ganter et al., 1999). La relation d'ordre partiel entre les itemsets est définie comme suit : $\forall\ c_1 = (O_1, I_1)$, $c_2 = (O_2, I_2) \in C_K$, $c_1 \leq c_1$ si et seulement si $I_2 \subseteq I_1 (\Leftrightarrow O_1 \subseteq O_2)$.

Les opérateurs *Sup* et *Inf* définis au chapitre deux, permettent d'obtenir, respectivement, la plus petite borne supérieure et la plus grande borne inférieure. Les opérateurs *Sup* et *Inf* sont définis comme suit : $\forall\ O_1, O_2 \subseteq O$ et $I_1, I_2 \subseteq I$:

- $(O_1, I_1) \vee (O_2, I_2) = ((O_1 \cup O_2)", I_1 \cap I_2)$,
- $(O_1, I_1) \wedge (O_2, I_2) = (O_1 \cap O_2, (I_1 \cup I_2)")$.

La relation d'ordre partiel est utilisée pour générer le graphe du treillis, appelé *diagramme de Hasse*, comme suit : il existe un arc (c_1, c_2), si $c_1 \leq c_2$ où $\leq$ est la réduction transitive de $\leq$, c'est-à-dire $\forall\ c_3 \in C_K$, $c_1 \leq c_3 \leq c_2 \Rightarrow c_1 = c_3$ ou $c_2 = c_3$. Dans ce graphe, chaque élément $c \in C_K$

est connecté aussi bien à un ensemble de ses successeurs immédiats, appelé couverture supérieure, et à un ensemble de ses prédécesseurs immédiats, appelé couverture inférieure.

- **Définition 3.14** (Treillis d'Iceberg de Galois)

Quand seulement l'ensemble des itemsets fréquents ordonnés par la relation ensembliste d'inclusion est considérée, la structure obtenue ($\Gamma^{\wedge}$, $\subseteq$) préserve seulement l'opérateur Sup. Cette structure forme un semi-treillis supérieure et elle est désignée par «*Treillis d'Iceberg de concepts*» (Bastide et al., 2000; Stumme et al., 2000; Valtchev et al., 2002).

Le treillis d'Iceberg de Galois associé au contexte d'extraction du tableau 3.1 pour un *minSupport*=3 est représenté par la figure 3.1.Chaque nœud du treillis représente le couple (itemset fermé, support) et il est décoré par la liste de ses générateurs minimaux

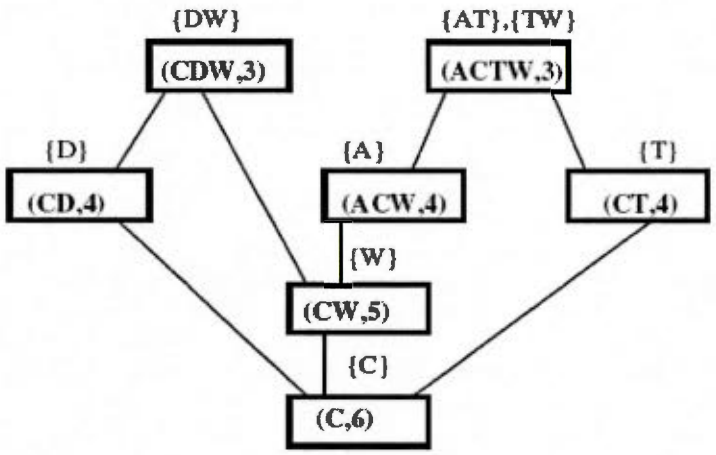

Figure 3.1 Treillis de l'iceberg de Galois associé au contexte d'extraction du tableau 3.1 pour un minSupport=0.5 (Gasmi et al., 2007)

3.3 Génération des règles d'association par extraction d'itemsets fermés fréquents

L'approche de découverte des règles d'association à partir des itemsets fermés fréquents se décompose comme l'approche classique en deux sous problèmes.

3.3.1 Étapes de découverte de règles par d'extraction des itemsets fermés fréquents

Soient un contexte d'extraction $C= (O, I, R)$, *minSupport* et *minConfiance* les seuils minimaux de support et de confiance définis par l'utilisateur. La génération des règles d'association à partir de l'extraction des itemsets fermés fréquents se décompose en trois nouvelles phases :

1) Déterminer l'ensemble des itemsets fermés fréquents dans C, c'est-à-dire ceux dont le support est supérieur au seuil minimal de support *minSupport*

Seule cette phase, nécessite la réalisation de balayages du contexte d'extraction. Comme l'espace de recherche des itemsets fréquents est limité au treillis des itemsets fermés qui est un sous-ordre du treillis des itemsets, le temps d'exécution de cette phase devient nettement inférieur à celui de la première phase de l'approche classique de génération de règles à partir des itemsets fréquents.

2) Énumérer l'ensemble des itemsets fréquents avec leur support à partir des itemsets fermés fréquents. Cette phase consiste à générer tous les sous-ensembles des itemsets fermés fréquents maximaux et dériver leur support à partir des supports des itemsets fermés fréquents.

3) Générer, à partir des itemsets fréquents dérivés (Algorithme *Gen-Règles*), l'ensemble de règles d'association dont la confiance est supérieure ou égale au seuil minimal de confiance *minConfiance* défini par l'utilisateur

Bien que la complexité de l'extraction des itemsets fermés fréquents soit dans le pire des cas exponentielle dans le nombre d'items du contexte (la taille du treillis de concepts coïncide avec la taille de l'ensemble des parties de I qui est 2^I autrement dit, tous les itemsets du contexte sont fermés), des résultats théoriques et expérimentaux ont démontré que dans la plupart des cas réels, la complexité est plus réduite. En effet, s'il existe une borne supérieure n sur la taille des objets des items du contexte, ce qui est souvent le cas dans les bases de données réelles, alors la taille du treillis de concepts est linéaire par rapport au nombre d'objets du contexte (Godin et al., 1986).

3.3.2 Intérêt de l'extraction des itemsets fermés fréquents

La réduction de l'ensemble des itemsets fréquents présente deux avantages :

1) Elle permet d'une part de réduire le temps d'extraction des itemsets fréquents puisque l'espace de recherche est réduit à celui des itemsets fermés uniquement qui est une partie de l'ensemble des itemsets fréquents.

2) D'autre part, bien que l'ensemble des itemsets fermés fréquents soit un sous ensemble des itemsets fréquents, celui-ci conserve toute l'information des itemsets fréquents et celle des règles d'association valides du contexte. Ces caractéristiques se justifient à l'aide des trois propriétés suivantes :

- **Propriété 3.3** (Zaki et al., 1999)

Le support d'un itemset est égal au support de sa fermeture.

- **Propriété 3.4** (Pasquier, 2000)

L'ensemble des itemsets fermés fréquents et leurs supports associés permettent de déduire l'ensemble de tous les itemsets fréquents ainsi que leurs supports

- **Propriété 3.5** (Bastide et al., 2000)

L'extraction des itemsets fermés fréquents et leurs supports, conserve l'information de l'ensemble des règles d'association valides du contexte.

3.3.3 Survol des algorithmes d'extraction des itemsets fermés fréquents

Plusieurs algorithmes ont été proposés dans la littérature pour l'extraction des itemsets fermés fréquents, adoptant chacun l'une ou l'autre des quatre approches recensées et définies au chapitre précédent (approche à niveaux ou séquentielle, approche verticale ou en profondeur, approche hybride, et approche des autres algorithmes). Parmi ces algorithmes, les plus connus :

- Approche à niveau : l'algorithme *Apriori-Close* est une extension d'*Apriori* qui offre la possibilité de découvrir pas uniquement les itemsets fréquents mais aussi les itemsets fermés fréquents. L'algorithme *Close* proposé par (Pasquier, 1998) est un autre algorithme par niveaux d'extraction d'itemsets fermés fréquents, il est détaillé dans la sous-section suivante

- Approche en profondeur ou verticale : L'algorithme *Charm* proposé par (Zaki et al., 1999) est une modification de l'algorithme *Eclat* pour l'extraction des itemsets fermés fréquents.

- Approche hybride : *Charm-MFI* (Zaki et al., 2000) est une extension de *Charm* qui utilise une approche hybride pour l'extraction des itemsets fermés fréquents. *Charm* est utilisé pour la découverte des itemsets fermés fréquents qui sont par la suite traités par ordre ascendant de leurs tailles pour filtrer les maximaux fréquents.

- Autres Algorithmes : parmi les variations de *FP-Growth* nous retrouvons entre autre *Closet* (Pei et al., 2000) et *Closet*$^+$ (Wang et al., 2003) conçus pour l'extraction des itemsets fermés fréquents.

3.3.4 Extraction des itemsets fermés fréquents avec *CLOSE*

L'algorithme *Close*, proposé par (Pasquier, 1998) est un algorithme itératif d'extraction des itemsets fermés fréquents qui parcourt l'ensemble des générateurs des itemsets fermés fréquents par niveaux. Durant chaque itération k, un ensemble FCC_k de k-générateur[4] candidats est considéré, chaque élément de cet ensemble est composé de trois éléments et qui sont le k-générateur candidat, sa fermeture et leur support. À la fin de chaque itération k, l'algorithme stocke un ensemble FF_k contenant les k-générateurs fréquents, leurs fermetures qui sont des itemsets fermés fréquents ainsi que leurs supports.

---

[4] k-générateur désigne le k-itemset candidat (itemset de taille k)

Algorithme 3.1 *Close*

Entrée : contexte B : seuil minimal de support *minSupport*;

Sortie : ensemble FF_k des *k*-groupes fréquents;

1) FCC_1.générateurs ← {1-itemsets};
2) **Pour** (k ← 1; FCC_k.générateurs ≠Ø; k++) **faire**
3) *Gen-Fermeture*(FCC_k)
4) **pour chaque** groupe candidat c ∈ FCC_k **faire**
5) **si** (c.support ≥ *minSupport*) **alors** FF_k ← FF_k ∪ {c};
6) **fin pour**
7) FCC_{k+1} ← *Gen-Générateur*(FF_k);
8) **fin pour**

retourner $\cup_k FF_k$

L'algorithme commence par initialiser l'ensemble FCC_1 des 1-générateurs avec les 1-itemsets (ligne 1) du contexte, c'est-à-dire avec les items de l'ensemble I du contexte d'extraction. Chacune des itérations k suivantes (lignes 2 - 8) consiste en trois phases. La première phase consiste à déterminer les fermetures des k-générateurs dans FCC_k et calculer leurs supports (ligne 3). Cette phase est réalisée par l'algorithme *Gen-Fermeture*. Dans la phase deux, uniquement les éléments de FCC_k qui ont un support au moins égal à *minSupport* (les fréquents) sont insérés dans l'ensemble FF_k des *k*-itemsets fermés fréquents (lignes 4-6). La dernière phase consiste à générer les (k+1)-générateurs candidats à partir des *k*-générateurs fréquents de FF_k (ligne 7). Cette phase est réalisée par l'algorithme *Gen-Générateur*. Ces itérations sont répétées jusqu'à ce qu'aucun nouveau générateur candidat ne peut être généré, c'est-à-dire jusqu'à ce que FCC_k soit vide.

Algorithme 3.2 *Gen-Fermeture*

Entrée : ensembles FCC_k des k-groupes candidats; contexte B;

Sortie : champs fermé et support des candidats de FCC_k mis à jour;

1) $FCC_k.fermés \leftarrow \emptyset$
2) $FCC_k.supports \leftarrow \emptyset$
3) **Pour chaque** objet $o \in B$ **faire**
4) $G_o \leftarrow$ Subset (FCC_k.générateurs, $f(\{o\})$);
5) **pour chaque** générateur $g.générateur \in G_o$ **faire**
6) **si** ($g.fermé = \emptyset$) **alors** $g.fermé \leftarrow f(\{o\})$;
7) **sinon** $g.fermé \leftarrow g.fermé \cap f(\{o\})$;
8) $g.support$++;
9) **fin pour**
10) **fin pour**
11) **retourner** $\cup\{g \in FCC_k \mid g.fermé \neq \emptyset\}$;

Le calcul des fermetures et des supports des k-générateurs candidats est réalisé par l'algorithme *Gen-Fermeture*. Ce dernier reçoit en entrée l'ensemble FCC_k des k-générateurs candidats et met à jour l'ensemble avec la fermeture et le support de chacun des k-générateurs via un seul balayage du contexte d'extraction. La détermination de la fermeture de chacun des k-générateurs de FCC_k est basée sur la propriété que la fermeture d'un itemset $I1$ est égale à l'intersection des images de tous les objets du contexte contenant I_1 (lignes 3-7) dont le décompte fournit le support du k-générateur qui est identique au support de sa fermeture (ligne 8).

Algorithme 3.3 *Gen-Générateur*

Entrée : ensembles FF_k des k-groupes fréquents;

Sortie : ensemble FCC_{k+1} avec les champs générateurs des (k+1)-groupes candidats initialisés

//phase1

1) **insert into** FCC_{k+1}.*générateur*
2) **select** p[1], p[2], … p[k], q [k]
3) **from** FF_k.*générateurs* p, FF_k.*générateurs* q
4) **where** p[1]= q[1], …, p[k-1]= q[k-1], p[k] < q[k];

//phase 2

5) **pour chaque** générateur *g.générateur* ∈ FCC_{k+1} **faire**
6) **pour chaque** *k-sous-ensemble s* de *g.générateur* **faire**
7) **si** (s ∉ FF_k.*générateurs*) **alors supprimer** g de FCC_{k+1};
8) **fin pour**
9) **fin pour**

//phase 3

10) **pour chaque** générateur *g.générateur* ∈ FCC_{k+1} **faire**
11) S_g ← Subset (FF_k.*générateurs, g.générateur*)
12) **pour chaque** s ∈ S_g **faire**
13) **si** (*g.générateur* ⊆ s.*fermé*) **alors supprimer** g de FCC_{k+1};
14) **fin pour**
15) **fin pour**
16) Retourner FCC_{k+1};

Les générateurs candidats sont construits par l'algorithme Gen-Générateur. Ce dernier reçoit en entrée l'ensemble FF_k des k-groupes fréquents, il retourne l'ensemble FCC_{k+1} des (k+1)-groupes candidats contenant les (k+1)-générateurs qui sont utilisés durant l'itération k+1. Cette procédure est constituée de trois phases. Durant la première phase toutes les jointures

des k-générateurs de FF_k possédant un préfixe commun de taille k-1 sont effectués (lignes 2-4). Le résultat de cette jointure est un ensemble de (k+1)-générateurs qui sont insérés dans FCC_{k+1}(ligne 1). La seconde phase consiste à vérifier pour chaque générateur potentiel créé, la présence de tous ses sous-ensembles de taille k dans FF_k (lignes 5 -9) sinon le supprimer de FCC_{k+1} car cela voudrait dire qu'il n'est pas fréquent puisqu'il est sur-ensemble d'itemsets non fréquents. La troisième phase consiste à vérifier pour chaque générateur candidat potentiel créé, s'il est inclus dans la fermeture de l'un de ses sous-ensembles et le supprime s'il vérifie cette condition car selon la définition de générateur minimal (définition 3.10) il ne serait pas générateur minimal (lignes 10-15).

- **Exemple** : L'application de l'algorithme *Close* au contexte K défini par le tableau 3.1, pour un seuil minimal *minSupport*=3 est représenté par la figure 3.2, ci-dessous.

Balayage De K →

FCC_1		
Génér.	*fermés*	*Sup*
{A}	{A, C, W}	4
{C}	{C}	6
{D}	{C, D}	4
{T}	{C, T}	4
{W}	{C, W}	5

Suppression des itemsets non fréquents →

FF_1		
Génér.	*fermés*	*Sup.*
{A}	{A, C, W}	4
{C}	{C}	6
{D}	{C, D}	4
{T}	{C, T}	4
{W}	{C, W}	5

Balayage De K →

FCC_2		
Génér.	*fermés*	*Sup.*
{A, D}	{A, C, D, W}	2
{A, T}	{A, C, T, W}	3
{D, T}	{C, D, T}	2
{D, W}	{C, D, W}	3
{T, W}	{A, C, T, W}	3

Suppression des itemsets non fréquents →

FF_2		
Génér.	*fermés*	*Sup*
{A, T}	{A, C, T, W}	3
{D, W}	{C, D, W}	3
{T, W}	{A, C, T, W}	3

Figure 3.2 Exemple d'application de l'algorithme Close

3.3.5 Dérivation des règles d'association des itemsets fermés fréquents

Cette phase consiste à énumérer l'ensemble des itemsets fréquents avec leurs supports à partir des itemsets fermés fréquents extraits dans la phase précédente puis générer, à partir des itemsets fréquents dérivés, en utilisant par exemple l'algorithme *Gen-Règles*, l'ensemble de règles d'association dont la confiance est supérieure ou égale au seuil minimal de confiance *minConfiance* défini par l'utilisateur.

L'extraction des itemsets fermés fréquents dans la première phase de la découverte des règles d'association a permis de réduire considérablement le temps d'extraction des itemsets fréquents tout en conservant l'information de ces derniers et celle des règles . Cependant, la proportion de règles d'association générées à partir des itemsets fréquents dérivés dans la seconde étape est critique notamment dans le cas de jeux de données éparses (Lefloch, 2003). Ce nombre Prohibitif de règles d'association est dû selon (Pasquier, 2000) à la présence d'une proportion forte de règles d'association redondantes et ceci rend leur exploitation par l'utilisateur quasi-impossible.

Le problème de la réduction du nombre de règles d'association a été étudié principalement dans le cadre des travaux de l'analyse formelle des concepts (Ganter et al., 1999). La solution consiste à définir des représentations compactes des règles d'association (Kryszkiewicz, 2002). Ces représentations constituent des ensembles réduits de règles d'association valides, couplés à un ensemble d'axiomes d'inférence. Idéalement, une représentation de règles d'association doit vérifier les trois critères suivants (Lefloch, 2003) :

- Critère de validité (toutes les règles déduites doivent être fortes),
- Critère de conservation (toutes les règles fortes doivent être déductibles),
- Et le critère informatif (le support et la confiance des règles déduites doivent pouvoir être déterminés).

Plusieurs représentations ont été proposées dans la littérature : la base de Duquenne-Guigues (Guigues et al., 1986), la base propre de Luxenburger (Luxenburger, 1991) et sa réduction transitive (Luxenburger, 1991), la base générique (Godin et al., 1994; Pasquier, 2000), la base informative (Pasquier, 2000) et sa réduction transitive (Pasquier, 2000).Un état de l'art de ces différentes représentations est présenté dans la section suivante de ce présent chapitre.

3.4 Représentation réduite des règles d'association

Le nombre de règles d'association extraites à partir des jeux de données réels est dans la majorité des cas très élevé et inexploitable par l'utilisateur. Ce problème est dû selon (Pasquier, 2000) à la présence d'une forte proportion de règles redondantes. Une règle est dite redondante par rapport à d'autres règles d'association, si l'information qu'elle apporte est par ailleurs présente dans d'autres règles. La solution proposée dans la littérature pour éliminer cette redondance, consiste à générer des bases pour les règles d'association qui sont des sous-ensembles de l'ensemble des règles d'association, ne contenant aucune règle redondante.

Exemple : Afin d'illustrer le problème de règles redondantes, nous présentons sept règles de l'ensemble de cinquante deux règles d'association valides extraites du contexte K définit par le tableau 3.1, avec *minSupport* = 0.5 et *minConfiance* = 0.6 (ces sept règles ont un support et une confiance identique et qui est respectivement 0.5 et 0.75)

1) Beurre → Lait

2) Beurre → Oeufs

3) Beurre → Oeufs, Lait

4) Beurre → Oeufs, Pain, Lait

5) Pain, Beurre → Oeufs

6) Pain, Beurre → Oeufs, Lait

7) Pain, Beurre → Lait

Il est évident que les règles 1 à 3 et 5 à 7 sont redondantes par rapport à la règle 4 (Beurre → Oeufs, Pain, Lait), puisque, du point de vue de l'utilisateur, ces six règles n'apportent aucune information supplémentaire par rapport à la règle 4 qui est la plus générale. Afin d'améliorer

la pertinence et l'utilité des règles extraites, il est souhaitable que seule cette règle 4 soit générée et présentée à l'utilisateur.

3.4.1 Règles d'association exactes et approximatives

Deux types de règles d'association sont distingués dans la revue de littérature : *les règles d'association exactes*, notées $I_{1\rightarrow}$ (I_2 - I_1) dont la confiance est égale à un et les *règles d'association approximatives,* notées $I_{1\rightarrow}$ (I_2 - I_1) dont la confiance est inférieure à un. Les règles d'association exactes sont générées à partir de deux itemsets fréquents I_1 et I_2 tels que $I_1 \subset I_2$ et support(I_1) est identique au support (I_2). Les règles d'association approximatives sont elles aussi générées à partir de deux itemsets fréquents I_1 et I_2 tels que $I_1 \subset I_2$ et le support(I_1) est supérieur au support (I_2).

Les règles d'association exactes et les règles d'association approximatives sont différenciées dans la littérature car elles possèdent des propriétés différentes par rapport aux itemsets fermés fréquents. Ces propriétés permettent d'identifier aussi bien les règles redondantes que les règles les moins significatives c'est-à-dire celles dont la confiance est la plus faible, parmi toutes les règles d'association valides.

3.4.2 Mécanisme d'inférence

Cette sous-section présente les différents mécanismes d'inférences s'appliquant aux règles d'association exactes et/ou approximatives et décrit la notion de représentation introduite dans (Kryszkiewicz, 1998; Kryszkiewicz, 2002).

3.4.2.1 Axiomes d'Armstrong (AA)

Les axiomes d'Armstrong (Armstrong, 1974) sont des règles de dérivation principalement utilisées dans le traitement et la manipulation des dépendances fonctionnelles dans les modèles relationnels. Ils se généralisent aussi aux règles d'association mais ne s'appliquent qu'aux règles exactes (Kryszkiewicz, 1998).

- **Propriété 3.6** (Axiomes d'Armstrong)

Soient, I_1, I_2 et I_3 trois itemsets. Les trois relations suivantes décrivent les propriétés de réflexivité, d'augmentation et de transitivité d'Armstrong

1) si $I_1 \supseteq I_2$, alors Confiance $(I_1 \rightarrow I_2) = 1$
2) si Conf $(I_1 \rightarrow I_2) = 1$ alors Confiance $(I_1 \cup I_3 \rightarrow I_2) = 1$
3) si Conf $(I_1 \rightarrow I_2) = 1$ et Conf $(I_2 \rightarrow I_3) = 1$ alors Confiance $(I_1 \rightarrow I_3) = 1$

3.4.2.2 Propriété de transitivité de la confiance (PTC)

Introduite par (Luxenburger, 1991), la propriété de transitivité de la confiance concerne les règles d'association approximatives.

- **Propriété 3.7** (Transitivité de la confiance)

Soient I_1, I_2, I_3 trois itemsets vérifiant la condition $I_1 \subset I_2 \subset I_3$. Le support et la confiance de la règle $(I_1 \rightarrow I_3 - I_1)$ peuvent être déduits de ceux des règles $(I_2 \rightarrow I_3 - I_2)$ et $(I_1 \rightarrow I_2 - I_1)$ comme suit :

1) support $(I_1 \rightarrow I_3 - I_1)$ = support $(I_2 \rightarrow I_3 - I_2)$
2) confiance $(I_1 \rightarrow I_3 - I_1)$ = confiance $(I_1 \rightarrow I_2 - I_1)$ * confiance $(I_2 \rightarrow I_3 - I_2)$

3.4.2.3 Opérateur de couverture (C)

L'opérateur de couverture permet de dériver, à partir d'une règle d'association valide, un ensemble de règles d'association valides par une simple transformation syntaxique de la règle initiale. La définition de cet opérateur et de ses propriétés (Kryszkiewicz, 1998) sont citées ci-dessous :

- **Définition 3.15** (couverture)

Soit $r : I_1 \rightarrow I_2$ une règle d'association. La couverture C de la règle r s'exprime à l'aide de la relation suivante :

$$C(I_1 \rightarrow I_2) = \{I_1 \cup I_3 \rightarrow I_4 \mid I_3, I_4 \subseteq I_2 \text{ et } I_3 \cap I_4 = \emptyset \text{ et } I_4 \neq \emptyset\}$$

L'antécédent de chacune des règles r_i de la couverture de la règle r contient l'itemset I_1 et un sous-ensemble d'items I_3 inclus dans I_2, alors que la conséquence non vide de r_i contient un itemset disjoint de I_3, sous-ensemble de l'itemset I_2.

- **Propriété 3.8** (couverture)

Soit $r : I_1 \rightarrow I_2$ une règle d'association. Les deux propriétés suivantes sont vérifiées :

1) si $r_i \in C(r)$, alors support $(r_i) \geq$ support(r) et confiance $(r_i) \geq$ confiance (r).

2) si r est une règle d'association valide du contexte et $r_i \in C(r)$, alors la règle d'association r_i est elle aussi valide

- **Propriété 3.9** (couverture)

Soit $r : I_1 \rightarrow I_2$ une règle d'association. La relation suivante est satisfaisante : $|C(I_1 \rightarrow I_2)| = 3^m - 2^m$ où $m = |I_2|$.

- **Propriété 3.10** (couverture)

Soient r: $I_1 \rightarrow I_2$ et $r_i : I_3 \rightarrow I_4$ deux règles d'association. Les trois relations suivantes sont équivalentes :

1) $r_i \in C(r)$

2) $I_3 \cup I_4 \subseteq I_1 \cup I_2$ et $I_3 \supseteq I_1$

3) $I_3 \cup I_4 \subseteq I_1 \cup I_2$ et $I_3 \supseteq I_1 \wedge I_2 \supseteq I_4$

Cette propriété assure qu'une règle déduite par l'opérateur de couverture ne peut être plus longue, en termes de nombre d'items, que celle de la règle auquel s'applique l'opérateur de couverture.

- **Propriété 3.11** (Transitivité de la couverture)

Soient r: $I_1 \rightarrow I_2$, r_i : $I_3 \rightarrow I_4$ et r_j : $I_5 \rightarrow I_6$ trois règles d'association. Si $r \in C(r_i)$ et $r_i \in C(r_j)$ alors $r \in C(r_j)$.

3.4.2.4 Fermeture (F)

La propriété de fermeture suivante (Pasquier, 2000) se base sur la propriété 3.3 qui stipule que le support d'un itemset est égal à celui de sa fermeture.

- **Propriété 3.12** (Fermeture)

Soit r: $I_1 \rightarrow I_2 - I_1$ une règle d'association telle qu'$I_1 \subset I_2$.

1) Support (r)= Support (I_2'')
2) Confiance (r) = support (I_2'') / support (I_1'')
3) Si la règle $I_1'' \rightarrow I_2'' - I_1''$ est une règle d'association valide, alors la règle d'association $I_1 \rightarrow I_2 - I_1$ est elle aussi valide

3.4.3 Représentation des règles d'association et classe de représentation

Soient C un contexte d'extraction, *minSupport* et *minConfiance* les seuils minimaux de support et de confiance fixés par l'utilisateur et RA l'ensemble des règles d'association valides (exactes et approximatives) extraites du contexte C. Les notions de représentation de règles d'association ont été introduites par (Kryszkiewicz, 2002)

- **Définition 3.16** (Représentation de règles d'association)

Soient R un ensemble de règles d'association tel que $R \subseteq RA$ et $\models$ un mécanisme d'inférence. Le couple $(R, \models)$ constitue une représentation de l'ensemble RA des règles d'association valides.

- **Définition 3.17** (Représentation conservatrice)

Soient R un ensemble de règles d'association tel que $R \subseteq RA$ et $\models$ un mécanisme d'inférence. Le couple (R, $\models$) est une représentation conservatrice de l'ensemble RA des règles d'association valides, si le mécanisme d'inférence $\models$ appliqué à l'ensemble des règles R, permet de déduire toutes les règles d'association valides de l'ensemble RA.

- **Définition 3.18** (Représentation valide)

Soient R un ensemble de règles d'association tel que $R \subseteq RA$ et $\models$ un mécanisme d'inférence. Le couple (R, $\models$) est une représentation valide de l'ensemble RA des règles d'association valides, si toutes les règles déduites de l'ensemble R par le mécanisme d'inférence $\models$, sont valides.

- **Définition 3.19** (Représentation informative)

Soient R un ensemble de règles d'association et $\models$ un mécanisme d'inférence. Le couple (R, $\models$) est une représentation informative si le mécanisme d'inférence $\models$ permet d'évaluer le support et la confiance des règles déduites.

Une représentation de règles d'association doit être selon (Kryszkiewicz, 2002) conservatrice, valide et informative. Une représentation conservatrice et valide de règles d'association constitue un ensemble compact de règles dans le sens où seul l'ensemble de, règles d'association valides et uniquement l'ensemble de règles d'association valides est déduit de la représentation.

Les représentations recensées dans la littérature sont décrites dans les sections suivantes.

3.5 Base de Duquennes-Guigues, base de Luxenburger et représentations associées

3.5.1 Base de Duquenne-Guigues

La base de Duquenne-Guigues concerne les règles d'implication globales (Guigues et al., 1986). Les règles de cette base sont vérifiées par l'ensemble des objets du contexte et sont définies à l'aide des ensembles d'attributs fermés et des ensembles pseudo-fermés.

L'adaptation de cette base à l'ensemble de règles d'association a fait l'objet de plusieurs publications (Ganter et al., 1999; Pasquier, 2000; Kryszkiewicz, 2002). La base de Duquenne-Guigues nécessite la prise en considération du support des règles d'implication globales. Elle ne s'applique qu'aux règles d'association exactes et s'exprime à l'aide des itemsets fermés fréquents et les itemsets pseudo-fermés fréquents du contexte d'extraction.

- **Définition 3.20** (Base de Duquenne-Guigues)

Soit C un contexte d'extraction. Soit PFF l'ensemble des itemsets pseudo-fermés extrait du contexte C. La base de Duquenne-Guigues (DG), pour les règles d'association exactes, est définie par la relation suivante :

$DG = \{r: I_1 \rightarrow I_1'' - I_1 \mid I_1 \in PFF\}$

- **Propriété 3.13** (Base de Duquenne-Guigues)

La base de Duquenne-Guigues pour les règles d'association ne contient que les règles d'association exactes valides.

3.5.2 Base de Luxenburger

Plusieurs bases pour les règles d'implication partielles sont proposées dans (Luxenburger, 1991). Les règles de ces bases sont vérifiées par un sous-ensemble d'objets du contexte et sont définies à partir des ensembles fermés d'attributs. L'adaptation de ces bases pour les règles d'association a fait l'objet de plusieurs études dont (Ganter et al., 1999; Pasquier, 2000; Valtchev et al., 2003). Les bases de Luxenburger pour les règles d'association concernent uniquement les règles d'association approximatives et sont construites à partir des itemsets fermés fréquents. Deux de ces bases sont présentées dans cette sous-section, il s'agit de la base propre et sa réduction transitive.

3.5.2.1 Base propre

- **Définition 3.21** (Base propre)

Soient *IFF*, l'ensemble des itemsets fermés fréquents extraits du contexte *C*, et *minConfiance*, le seuil minimal de confiance fixé par l'utilisateur. La base propre, pour les règles d'association approximatives, est définie comme suit :

$BP=\{r: I_1 \rightarrow I_2 - I_1 \mid I_1, I_2 \in IFF \wedge I_1 \subset I_2 \wedge \text{Confiance}(r) > minConfiance\}$

- **Propriété 3.14** (Base propre)

La base propre pour les règles d'association ne contient que les règles d'association approximatives valides.

3.5.2.2 Réduction transitive de la base propre

La réduction transitive de la base propre (Luxenburger, 1991) est un sous-ensemble de la base propre. Toutes les règles approximatives valides de la base propre, ainsi que leur support et leur confiance peuvent être déduits de la réduction de la base propre par l'opérateur de transitivité de la confiance (*PTC*) (Luxenburger, 1991).

- **Définition 3.22** (Réduction transitive de la base propre)

Soient *IFF*, l'ensemble des itemsets fermés fréquents extraits du contexte *C*, et *minConfiance*, le seuil minimal de confiance fixé par l'utilisateur. La réduction transitive de la base propre(*RBP*), pour les règles approximatives, est donnée par la relation suivante :

$RBP=\{r: I_1 \rightarrow I_2 - I_1 \mid I_1, I_2 \in IFF \wedge I_1 \prec I_2 \wedge \text{Confiance (r)} > minConfiance\}$

- **Propriété 3.15** (Réduction transitive de la base propre)

La réduction transitive de la base propre pour les règles d'association ne contient que les règles d'association approximatives valides.

3.5.3 Représentations associées

Ce paragraphe décrit toutes les représentations associées à la base de Duquenne-Guigues et aux bases de Luxenburger pour les règles d'association.

- **Propriété 3.16** (Représentations associées) (Pasquier, 2000)

1) (*DG*, *AA*) où *AA* correspond aux axiomes d'Armstrong (propriété 3.6), est une représentation conservatrice de l'ensemble des règles d'association exactes valides du contexte
2) (*BP*, *F*) et (*RBP*, {*F*, *PTC*}) où *F* et *PTC* correspondent respectivement à l'opérateur de fermeture (propriété 3.12) et à la propriété de transitivité de la confiance (propriété 3.7), sont des représentations conservatrices de l'ensemble des règles d'association approximatives valides du contexte

Cependant ces deux représentations pour les règles exactes et approximatives ne sont en général ni valides, ni informatives (Kryszkiewicz, 2002. Afin d'obtenir une représentation conservatrice, valide et informative (corollaire 3.1), (Kryszkiewicz, 2002) propose de coupler (*BP*, *F*) avec (*DG*, *AA*) (propriété 3.17)

- **Propriété 3.17** (Kryszkiewicz, 2002)

Soit I_1 un itemset, I_2 un itemset fréquent vérifiant $I_1 \subset I_2$. Soit *IFF* l'ensemble des itemsets fermés fréquents extraits du contexte.

1) $I_1 \rightarrow I_2 - I_1$ est déduite de la représentation (*DG*, *AA*) si et seulement si Support(I_1)=Support(I_2)
2) $I_1 \rightarrow I_2 - I_1$ est déduite de la représentation (*DG*, *AA*) si et seulement si Support(I_1'')=Support(I_2'')
3) $I_2 \in IFF \wedge I_1 \rightarrow I_2 - I_1$ est déduite de la représentation (*DG*, *AA*) si et seulement si Support(I_1'')=Support(I_2'')

- **Corollaire 3.1** (Kryszkiewicz, 2002)

($BP \cup DG$, {*F*, *AA*}) et ($RBP \cup DG$, {*F*, *PTC*, *AA*}) sont des représentations conservatrices, valides et informatives de l'ensemble des règles d'association valides du contexte.

3.6 Base générique, informative, réduction transitive et représentations associées

Les représentations suivantes sont caractérisées par des règles d'antécédent minimal et de conséquence maximale. La maximalité de la conséquence est assurée par les itemsets fermés fréquents qui sont par définition maximaux, et la minimalité de l'antécédent de la règle est assurée par le générateur minimal qui est par définition le plus petit de sa classe d'équivalence de fermeture.

3.6.1 Base générique

La base générique contient les règles d'association exactes valides du contexte (Godin et al., 1994; Pasquier 2000). Elle est définie en utilisant les itemsets fermés fréquents et leurs générateurs non fermés associés.

- **Définition 3.23** (Base générique)

Soit *IFF* l'ensemble des itemsets fermés fréquents extraits du contexte, et G_{I2} l'ensemble des générateurs minimaux associé à l'itemset fermé fréquent I_2. La base *BG* est définie comme suit : $BG = \{r: I_1 \rightarrow (I_2 - I_1) \mid I_2 \in IFF \wedge I_1 \in G_{I2} \wedge I1 \neq I2\}$

- **Propriété 3.18** (*BG*) (Pasquier, 2000)

La base générique ne contient que des règles d'association exactes valides

Exemple : Soit le contexte d'extraction du tableau 3.1. Pour *minSupport*= 0.5 et *minConfiance*= 0.6, le tableau 3.2 est une illustration de la base *BG*

Tableau 3.2 Base générique des règles d'association exactes extraites du contexte d'extraction *K* (défini dans le tableau 3.1)

Numéro	Règle	Support	Confiance
R_1	Œufs, beurre →pain, lait	0.5	1
R_2	beurre, lait → Œufs, pain	0.5	1
R_3	sucre, lait → pain	0.5	1
R_4	sucre → pain	0.5	1
R_5	lait → pain	0.83	1
R_6	beurre → pain	0.66	1
R_7	Œufs→pain, lait	0.66	1

3.6.2 Base informative

La base informative ne renferme que les règles d'association approximatives valides (Pasquier, 2000)

- **Définition 3.24** (Base informative)

Soit *minConfiance*, le seuil minimal de confiance fixé par l'utilisateur, *IFF* l'ensemble des itemsets fermés fréquents extraits du contexte et *G* l'ensemble des générateurs minimaux. La base informative *BI* pour les règles d'associations approximatives est définie comme suit :

$$BI = \{r: I_1 \rightarrow (I_2 - I_1) \mid I_2 \in IFF \wedge (I_1 \in G) \wedge (I_1'' \subset I_2) \wedge \text{confiance}(r) \geq minConfiance\}$$

- **Propriété 3.19** (Base informative) (Pasquier, 2000)

La base informative ne contient que des règles d'association approximatives valides.

Exemple : Soit le contexte d'extraction du tableau 3.1. Pour *minSupport*= 0.5 et *minConfiance*= 0.6, le tableau 3.3 est une illustration de la base *BI*

Tableau 3.3 Base informative des règles d'association approximatives extraites du contexte d'extraction K (défini dans le tableau 3.1)

Numéro	Règle	*Support*	*Confiance*
R_8	pain →lait	0.83	0.83
R_9	Beurre → Œufs, pain, lait	0.5	0.75
R_{10}	sucre → pain, lait	0.66	0.75
R_{11}	Œufs → pain, beurre, lait	0.5	0.75
R_{12}	pain → beurre	0.66	0.66
R_{13}	pain → sucre	0.66	0.66
R_{14}	pain →lait, œufs	0.66	0.66
R_{15}	Lait → pain, œufs	0.66	0.8
R_{16}	Lait → pain, œufs, beurre	0.5	0.6
R_{17}	Lait → pain, sucre	0.5	0.6

Néanmoins l'extraction de la base *BI* souffre de la génération d'un nombre important de règles. En effet, selon (Bastide et al., 2000), l'extraction de *BI* à partir d'un jeu de données éparse, n'apporte aucun gain en terme de compacité, et c'est pourquoi (Bastide et al., 2000) ont défini la réduction transitive de la base informative des règles approximatives, notée *RBI*, en se basant sur l'opérateur de transitivité de (Luxenburger, 1991).

3.6.3 Réduction transitive de la base informative

La réduction transitive de la base informative (Pasquier, 2000) est un sous-ensemble de la base informative. Toutes les règles approximatives valides de la base informative, ainsi que leur support et leur confiance, peuvent être déduits de la réduction transitive de la base informative par l'opérateur de transitivité (*PTC*) (Pasquier, 2000)

- **Définition 3.25** (réduction transitive de la base informative)

Soit *minConfiance*, le seuil minimal de confiance fixé par l'utilisateur, *IFF* l'ensemble des itemsets fermés fréquents extraits du contexte et G_{12} l'ensemble des générateurs minimaux

associé à l'itemset fermé I_2. La réduction transitive de la base informative, *RBI* est définie comme suit :

$RBI = \{r: I_1 \rightarrow (I_3 - I_1) \mid I_2, I_3 \in IFF \wedge (I_1 \in G_{I2}) \wedge I_2 < I_3 \wedge$ confiance (R) $\geq minConfiance\}$.

- **Propriété 3.20** (Réduction transitive de la base informative) (Pasquier, 2000)

La réduction transitive de la base informative ne contient que des règles d'association approximatives valides.

Exemple : Soit le contexte d'extraction du tableau 3.1. Pour *minSupport*= 0.5 et *minConfiance*= 0.6, le tableau 3.4 est une illustration de la base *RBI.*

Afin de dériver l'ensemble de toutes les règles redondantes à partir de (*BG*, *BI*), (Kryszkiewicz, 1 998) a proposé d'utiliser l'opérateur de couverture *c*. Il a aussi proposé d'appliquer cet opérateur c avec l'axiome de transitivité de (Luxenburger, 1991) afin de dériver toutes les règles redondantes à partir de (*BG*, *RBI*).

Tableau 3.4 Base *RBI* extraite du contexte *K* définie par le tableau 3.1

Numéro	Règle	*Support*	*Confiance*
R_8	pain →lait	0.83	0.83
R_9	Beurre → Œufs, pain, lait	0.5	0.75
R_{10}	sucre → pain, lait	0.66	0.75
R_{11}	Œufs → pain, beurre, lait	0.5	0.75
R_{12}	pain → beurre	0.66	0.66
R_{13}	pain → sucre	0.66	0.66

3.6.4 Représentations associées

Ce paragraphe décrit les représentations associées à la base générique, à la base informative et à sa réduction transitive.

- **Corollaire 3.2** (Kryszkiewicz, 2002)

Soit RA l'ensemble, des règles d'association valides extraites du contexte.

1) (*BG*, *C*), où C correspond à l'opérateur de couverture, est une représentation conservatrice, valide et informative de l'ensemble des règles d'association exactes valides du contexte.
2) (*BI*, *C*) est une représentation conservatrice de l'ensemble des règles d'association approximatives valides du contexte. Cette représentation n'est en général ni valide, ni informative.
3) ($BI \cup BG$, *C*) est une représentation conservatrice, valide et informative de l'ensemble des règles d'association valides du contexte
4) ($RBI \cup BG$, {*C*, *PTC*}) où *PTC* correspond à la propriété de transitivité de la confiance, est une représentation conservatrice, valide et informative de l'ensemble des règles d'association valides du contexte

3.7 Conclusion

Dans ce chapitre, nous avons présenté les résultats des travaux basés sur l'analyse formelle des concepts, et menés dans le cadre de l'extraction des règles d'association. Les résultats recensés dans la revue de littérature, montrent que l'utilisation des itemsets fermés versus les itemsets fréquents, apporte d'une part une large amélioration dans le temps d'extraction des règles d'association et fournit d'autre part, un noyau irréductible des règles d'association permettant une meilleure exploitation des résultats par l'utilisateur. En effet, il a été montré dans (Bastide et al., 2000) que l'union des deux bases proposées par ces derniers, et formée par la base générique des règles d'association exactes et la réduction transitive de la base informative des règles d'association approximatives fournissait un ensemble générateur non redondant pour toutes les règles d'association valides, leurs supports et leurs confiances. Constituée de règles d'association non redondantes minimales (d'antécédent minimal et de conclusion maximale), elle ne représente aucune perte d'informations et convoie toute l'information que peut convoyer l'ensemble de toutes les règles d'association valides. Cette représentation succincte présente un fort intérêt pour la visualisation des règles extraites car le nombre réduit de règles dans ces bases ainsi que la distinction des règles exactes et des

règles approximatives facilitent la présentation des règles à l'utilisateur. De plus l'absence de règles redondantes dans les bases et la génération de règles non redondantes minimales seulement présentent un intérêt important du point de vue de l'utilisateur (Klemettinen, 1994; Klemettinen, 2004).

Cependant un survol de la littérature montre que même ces bases proposées par (Bastide et al., 2000) comportent encore de la redondance localisée au niveau des générateurs minimaux qui composent l'antécédent de ces bases. Une représentation succincte de ces générateurs minimaux est utile pour révéler la relation d'équivalence parmi les générateurs minimaux qui peut être importante pour les recherches médicales et scientifiques. Un état de l'art de ces travaux menés dans le cadre de la représentation succincte des générateurs minimaux ainsi que notre contribution sur l'expansion de cette représentation succincte sont présentées dans le chapitre suivant.

CHAPITRE IV

REPRÉSENTATION SUCCINCTE DES GÉNÉRATEURS MINIMAUX

4.1 Introduction

Les bases génériques proposées par (Bastides et al., 2000) ont constitué jusqu'à aujourd'hui le noyau compact et irréductible des règles d'associations. Les générateurs minimaux constituent la base de cette représentation réduite puisqu'ils forment la prémisse de ces bases et leurs fermetures constituent la conclusion. Cependant il a été montré dans (Dong et al., 2005) que ces générateurs peuvent contenir de l'information redondante puisque certains générateurs minimaux associés à un itemset fermé peuvent être dérivés d'autres générateurs minimaux appartenant à la même classe d'équivalence induite par l'opérateur de fermeture, via un processus de substitution. Des systèmes succincts de générateurs minimaux ont été alors proposés dans la littérature dans le but de réduire cette redondance localisée au niveau des générateurs minimaux. Un état de l'art de ces systèmes proposés pour la représentation succincte des générateurs minimaux est présenté dans ce chapitre, accompagnés des avantages et inconvénients de chacun de ces systèmes ainsi que la description de l'algorithme *DSFS_Miner* défini par (Hamrouni et al., 2007) pour supporter la représentation succincte des générateurs minimaux proposés par les auteurs.

4.2 Représentation succincte des générateurs minimaux avec perte d'informations

4.2.1 Système succinct des générateurs minimaux initial (*SSMG*)

Il a été montré dans (Dong et al., 2005) que l'ensemble des générateurs minimaux pouvaient contenir de l'information redondante puisque certains générateurs minimaux (notés *MG*s) associés à un itemset fermé f pouvaient être dérivés d'autres générateurs minimaux appartenant à la même classe d'équivalence (classe d'équivalence induite par l'opérateur de fermeture), via un processus de substitution. Ainsi (Dong et al., 2005) ont présenté le système succinct des générateurs minimaux comme une représentation concise de l'ensemble de tous les générateurs minimaux. L'idée principale était de supprimer l'information redondante en choisissant un générateur minimal d'un itemset fermé (le plus petit par rapport à une relation d'ordre totale $\preceq$ définie sur les itemsets) et l'élire comme son générateur minimal représentatif et élaguer tous ceux contenant au moins un générateur minimal non représentatif. Le but est de maintenir dans chaque classe d'équivalence de fermeture seulement les générateurs minimaux qui ne peuvent pas être dérivés d'autres générateurs minimaux appartenant à la même classe d'équivalence que nous désignerons par γ. Afin d'y parvenir, les auteurs ont instauré une relation entre les itemsets qu'ils ont définie comme suit :

Définition 4.1 (relation $\approx_f$)

Soit f un itemset fermé. Soient X et Y deux itemsets. X et Y sont dits *f-équivalents*, noté ($X \approx_f Y$) si :

i) X et Y sont deux générateurs minimaux associés à $f_1 \subset f$

ii) X peut être dérivé de Y en remplaçant un sous-ensemble Z_1 de X ($Z_1 \subset X$) par un sous-ensemble Z_2 de Y ($Z_2 \subset Y$) tel que $Z_1 \approx_f Z_2$.

Exemple 4.1

Pour illustrer la définition 4.1, considérons l'extraction effectuée selon l'ordre alphabétique du tableau 4.2 à partir du contexte d'extraction K du tableau 4.1

La condition (i) de la relation $\approx_f$ est vérifiée par les itemsets *ac* et *af* qui sont générateurs minimaux associées à l'itemset fermé $f_1 = acf \subset f = abcdef$, d'où $ac \approx_{abcdef} af$. La condition (ii)

est vérifiée par contre, par les itemsets *abc* et *abf* qui sont générateur minimaux de l'itemset fermé *f=abcdef.* En effet en remplaçant le sous-ensemble *ac* du générateur minimal *abc* par *af,* nous obtenons *abf.* Ce remplacement est correct puisque $ac \approx_{abcdef} af$

La relation $\approx_f$ a été utilisée par (Dong et al., 2005) pour diviser l'ensemble des générateurs minimaux associé à un itemset fermé donné, en différentes classes d'équivalences que nous désignons par σ pour ne pas confondre avec la classe d'équivalence induite de la fermeture (") désignée par γ. Un seul générateur minimal est ensuite maintenu dans chaque classe d'équivalence σ, il s'agit du générateur minimal représentatif de la classe σ. Le choix du membre représentatif d'une classe σ est selon (Dong et al., 2005), aléatoire pour les plus petits itemsets fermés, alors que pour les autres itemsets fermés, le choix des auteurs se porte sur les générateurs minimaux canoniques, c'est-à-dire ceux ne contenant aucun générateur minimal non représentatif d'un plus petit itemset fermé.

Tableau 4.1 Contexte d'extraction *K* (*minSupport*=1)

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

Tableau 4.2 Ensemble des itemsets fermés et générateurs associés, extraits du contexte K (défini par le tableau 4.1)

	Ordre **alphabétique**		Ordre **ascendant** de support		Ordre **descendant** de support	
	CI	**MGs**	**CI**	**MGs**	**CI**	**MGs**
1	*Ø*	*Ø*	*Ø*	*Ø*	*Ø*	*Ø*
2	*a*	*a*	*a*	*a*	*a*	*a*
3	*b*	*b*	*b*	*b*	*b*	*b*
4	*c*	*c*	*c*	*c*	*c*	*c*
5	*d*	*d*	*d*	*d*	*d*	*d*
6	*be*	*e*	*eb*	*e*	*be*	*e*
7	*f*	*f*	*f*	*f*	*f*	*f*
8	*ab*	*ab*	*ab*	*ab*	*ba*	*ba*
9	*acf*	*ac, af, cf*	*acf*	*ac, af, cf*	*fac*	*fa, fc, ac*
10	*ad*	*ad*	*ad*	*ad*	*da*	*da*
11	*abcdef*	*ae, abc, abd, abf, acd, adf, bcf, bdf, cdf, cef, def*	*eacbdf*	*ea, ecf, edf, acb, acd, abd, abf, adf, cbf, cdf, bdf*	*bdface*	*ae, bdf, bda, bfa, bfc, bac, dfa, dfc, dfe, dac, fce*
12	*bcde*	*bc, bd, ce, de*	*ecbd*	*ec, ed, cb, bd*	*bdce*	*bd, bc, de, ce*
13	*bf*	*bf*	*bf*	*bf*	*bf*	*bf*
14	*cd*	*cd*	*cd*	*cd*	*dc*	*dc*
15	*df*	*df*	*df*	*df*	*df*	*df*
16	*bef*	*ef*	*ebf*	*ef*	*bfe*	*fe*

Cependant, l'étude menée par (Hamrouni et al., 2006) pour analyser le système succinct des générateurs minimaux initial défini par (Dong et al., 2005) a montré que la cardinalité des

générateurs minimaux représentait pour (Dong et al., 2005), le critère principal dans le choix du générateur minimal représentatif ; les plus petits ensembles, étaient par conséquent ceux choisis pour représenter les générateurs minimaux représentatifs.

Une fois les générateurs minimaux représentatifs connus, tous les générateurs minimaux non représentatifs (parce qu'ils ne sont pas les plus petits de leur classe d'équivalence de fermeture) et contenant uniquement les générateurs minimaux représentatifs sont selon (Dong et al., 2005) des générateurs minimaux canoniques. Un ordre total doit donc être défini sur les itemsets pour permettre la détermination des générateurs minimaux représentatifs et par conséquent les générateurs minimaux canoniques. La relation d'ordre totale suivante a été définie pour cela, sur les itemsets:

- **Définition 4.2** (Relation d'ordre totale)

Soit $\preceq_s$ une relation d'ordre totale sur les items, c'est-à-dire $\forall\, i_1, i_2 \in I$, nous avons $i_1 \preceq_s i_2$ ou $i_2 \preceq_s i_1$. Cette relation est aussi utilisée pour comparer des itemsets de cardinalités différentes. Soient X et Y deux itemsets et soient $|X|$ et $|Y|$ leurs cardinalités respectives. Nous avons

4) Si $|X| < |Y|$ alors $X \prec_s Y$

5) $|X| = |Y|$, alors X et Y sont comparés suivant l'ordre lexicographique.

Exemple 4.2

Si nous considérons l'ordre alphabétique sur les items comme étant la relation d'ordre totale $\preceq_s$ appliquée sur les itemsets[5], alors :

- $|d| < |be| \Rightarrow d \preceq_s be$
- $|abd| = |abe| \Rightarrow abd \preceq_s abe$

[5] Dans le reste du document, nous mentionnons le critère d'ordre sur les items (alphabétique, ordre ascendant de support, ordre descendant de support,...etc.). Ce dernier est étendu pour être la relation d'ordre totale à appliquer sur les itemsets.

(Dong et al., 2005) ont présenté donc le système succinct des générateurs minimaux comme étant un ensemble composé de générateurs minimaux représentatifs (les plus petits de sa classe d'équivalence de fermeture, selon la relation d'ordre totale appliquée sur les itemsets) et de générateurs minimaux canoniques (les générateurs minimaux ne contenant aucun générateur minimal non représentatif). Pour le constituer d'une façon formelle, trois grandes catégories de générateurs minimaux ont été introduites, Il s'agit de l'ensemble de générateurs minimaux représentatifs, l'ensemble des générateurs minimaux canoniques et l'ensemble des générateurs minimaux redondants qui sont définis respectivement comme suit :

- **Définition 4.3** (Catégories de générateurs minimaux)

L'ensemble MG_f, de générateurs minimaux associé à l'itemset fermé f, peut être fractionné en trois sous ensembles distincts suivants :

$MGrep_f = \{g \in MG_f \mid$ il n'existe pas $g_1 \in MG_f$ *tel que* $g_1 \lesssim_s g\}$. Contient le plus petit générateur minimal de la classe d'équivalence de fermeture, selon une relation d'ordre totale $\lesssim_s$ appliquée sur les itemsets, constituant ainsi le générateur minimal représentatif.

$MGcan_f = \{g \in MG_f \mid g \notin MGrep_f, \forall\ g_1 \subset g,\ g_1 \in MGrep_{f1}$ avec $f_1 = g_1"\}$ Contient les générateurs minimaux canoniques de f. Un générateur minimal canonique n'est pas le plus petit générateur minimal de sa classe d'équivalence de fermeture, mais tous ses sous-ensembles sont des générateurs minimaux représentatifs de leurs fermetures respectives.

$MGred_f = \{g \in MG_f \mid \exists\ g_1 \subset g,\ g_1 \notin MGrep_{f1}$ avec $f_1 = g_1"\}$ Contient les générateurs minimaux redondants de f

L'ensemble succinct des générateurs minimaux associés à un itemset fermé f serait donc selon (Dong et al., 2005) :

$MGsuc_f = MGrep_f \cup MGcan_f = = \{g \in MG_f \mid \forall\ g_1 \subset g,\ g_1 \in MGrep_{f1}$ avec $f_1 = g_1"\}$ l'ensemble $MGsuc_K$ succinct des générateurs minimaux pouvant être extrait du contexte K est donc bien un idéal d'ordre de $(2^I, \subseteq)$ (Hamrouni et al., 2006).

4.2.2 Limitations du système succinct des générateurs minimaux initial (*SSMG*)

(Dong et al., 2005) ont présenté *SSMG* comme étant le système succinct sans perte d'information de l'ensemble des générateurs minimaux pouvant être extraits du contexte d'extraction donné *K*. La taille du système ainsi défini est selon les auteurs invariable à quelque soit la relation d'ordre totale $\preceq_s$ appliquée sur les itemsets. Or l'étude mené par (Hamrouni et al., 2006) affirme que le *SSMG* ainsi défini ne représente pas un système succinct sans perte d'informations pour plusieurs raisons dont la principale réside dans la relation $\approx_f$ définie par (Dong et al., 2005) pour diviser l'ensemble des générateurs minimaux associé à un itemset fermé en différentes classes d'équivalences σ, qui s'est avérée ne pas être une relation d'équivalence puisque la condition de transitivité n'est pas toujours satisfaite.

En effet selon (Dong et al., 2005), pour déduire les générateurs minimaux redondants de chaque classe d'équivalence de fermeture γ, à partir de son ensemble de générateurs minimaux succincts associé, il suffit de remplacer un ou plusieurs sous-ensembles de ses générateurs minimaux succincts par les générateurs minimaux non représentatifs ayant la même fermeture que les sous-ensembles remplacés. Par exemple, le générateur minimal redondant $\{abd\}$ extrait du contexte *K* (tableau 4.2 avec ordre alphabétique), peut être déduit à partir du générateur minimal succinct $\{abc\}$ en remplaçant son sous-ensemble $\{bc\}$ par $\{bd\}$ puisque $\{bc\}$ et $\{bd\}$ ont la même fermeture $\{bcde\}$.

La constitution de la famille entière des générateurs minimaux à partir du système succinct des générateurs minimaux (*SSMG*) dépend des générateurs minimaux succincts contenus dans le *SSMG* et par conséquent, du choix fait au départ par les auteurs pour désigner les membres représentatifs des différentes classes d'équivalence σ induites de la relation d'équivalence $\approx_f$, et à partir desquelles les autres générateurs minimaux peuvent être dérivés. Cependant choisir ces générateurs minimaux représentatifs et élaguer les éléments redondants contenant des sous-ensembles non représentatifs, peut mener aux résultats suivants :

Tableau 4.3 Ensemble succinct des générateurs minimaux du *SSMG* initial extrait de *K*

	Ordre **alphabétique**			Ordre **ascendant** de support			Ordre **descendant** de support		
	CI	**MGs**	**MGsuc**	**CI**	**MGs**	**MGsuc**	**CI**	**MGs**	**MGsuc**
1	*Ø*	*Ø*	***Ø***	*Ø*	*Ø*	***Ø***	*Ø*	*Ø*	***Ø***
2	*a*	*a*	***a***	*a*	*a*	***a***	*a*	*a*	***a***
3	*b*	*b*	***b***	*b*	*b*	***b***	*b*	*b*	***b***
4	*c*	*c*	***c***	*c*	*c*	***c***	*c*	*c*	***c***
5	*d*	*d*	***d***	*d*	*d*	***d***	*d*	*d*	***d***
6	*be*	*e*	***e***	*eb*	*e*	***e***	*be*	*e*	*e*
7	*f*	*f*	*f*	*f*	*f*	*f*	*f*	*f*	*f*
8	*ab*	*ab*	***ab***	*ab*	*ab*	***ab***	*ba*	*ba*	***ba***
9	*acf*	*ac, af, cf*	***ac, af, cf***	*acf*	*ac, af, cf*	***ac, af, cf***	*fac*	*fa, fc, ac*	***fa, fc, ac***
10	*ad*	*ad*	***ad***	*ad*	*ad*	***ad***	*da*	*da*	***da***
11	*abcdef*	*ae, abc, abd, abf, acd, adf, bcf, bdf, cdf, cef, def*	***ae, abc, acd,***	*eacbdf*	*ea, ecf, edf, acb,acd, abd,abf, adf, cbf, cdf, bdf*	***ea, acd***	*bdface*	*ae, bdf, bda, bfa, bfc, bac, dfa, dfc, dfe, dac, fce*	***ae, bdf, bda, bfa, dfa***
12	*bcde*	*bc, bd, ce, de*	***bc, bd, ce, de***	*ecbd*	*ec, ed, cb, bd*	***ec, ed, cb, bd***	*bdce*	*bd, bc, de, ce*	***bd, bc, de, ce***
13	*bf*	*bf*	***bf***	*bf*	*bf*	***bf***	*bf*	*bf*	***bf***
14	*cd*	*cd*	***cd***	*cd*	*cd*	***cd***	*dc*	*dc*	***dc***
15	*df*	*df*	***df***	*df*	*df*	***df***	*df*	*df*	***df***
16	*bef*	*ef*	*ef*	*ebf*	*ef*	*ef*	*bfe*	*fe*	*fe*

- Une classe d'équivalence sans aucun membre représentatif : c'est le cas pour la classe d'équivalence σ suivante constituée de S_1= {*ecf, edf, acb, abd, abf, cbf, bdf*} associée à l'itemset fermé {*eacbdf*} extrait selon l'ordre ascendant de support, du contexte d'extraction *K*, (voir tableau 4.3 ordre ascendant de support).

- En effet, chaque élément de cette classe d'équivalence σ désignée par S_1, contient au moins un générateur minimal non représentatif, par conséquent cette classe d'équivalence est élaguée au départ et n'est pas prise en considération, et ses éléments ne peuvent pas être retrouvés car ils sont impossible à dériver.
- Une classe d'équivalence avec plus d'un candidat comme membre représentatif : c'est le cas pour la classe d'équivalence σ suivante constituée de S_2= {*bdf, bda, bfa, bfc, bac, dfe, fce*} associée à l'itemset fermé {*bdface*} extrait selon l'ordre descendant de support, du contexte d'extraction *K* (voir le tableau 4.3, ordre descendant de support). En effet {*bdf*}, {*bda*} et {*bfa*} sont tous des générateurs minimaux et ont des sous-ensembles générateurs minimaux représentatifs. Même si le choix du générateur minimal représentatif se porte sur le plus petit candidat, la définition donnée par (Dong et al., 2005) a négligé la partie importante qui permet de supprimer le reste des candidats.

Les classes d'équivalences S_1 et S_2 ont exactement la même taille que les itemsets soient extraits selon un ordre ascendant de support ou un ordre descendant de support, seulement la classe S_1 ne possède aucun membre représentatif alors que la classe S_2 en possède trois. Ceci nous prouve que contrairement à ce qui a été affirmé dans (Dong et al., 2005), la taille du système succinct initial des générateurs minimaux dépend étroitement de la relation d'ordre totale appliquée sur les itemsets. Par ailleurs, l'application de la relation $\approx_f$ n'induit pas une relation d'équivalence sur l'ensemble des générateurs minimaux associé à un itemset fermé *f*. En effet si nous considérons l'extraction selon l'ordre alphabétique illustrée par le tableau 4.2, et l'itemset fermé {*abcdef*}, nous avons $abc \approx_{abcdef} bcf \approx_{abcdef} def$ mais *abc* n'est pas $\approx_{abcdef}$ avec *def*. Par conséquent la relation $\approx_f$ n'est pas une relation d'équivalence puisque la condition de transitivité n'est pas vérifiée.

4.2.3 Redéfinition du système succinct des générateurs minimaux initial

Afin de pallier aux lacunes du *SSMG* initial, (Hamrouni et al., 2006) ont proposé une nouvelle relation notée |= permettant de diviser l'ensemble MG_f des générateurs minimaux associé à un itemset fermé f en différentes classes d'équivalences σ via un processus de substitution. Ce processus utilise un opérateur de substitution noté *Subst* permettant de remplacer un sous ensemble Z_1 d'un itemset X par un autre itemset Z_2 appartenant à la même classe d'équivalence de fermeture γ que Z_1 (C'est-à-dire Z_1"= Z_2"). Cet opérateur fonctionne comme suit : $Subst(X, Z_1, Z_2)= (X \setminus Z_1) \cup Z_2$. Il a été démontré dans (Hamrouni et al., 2006) que X et $Subst(X, Z_1, Z_2)$ ont la même fermeture.

Pour chaque classe d'équivalence de fermeture γ, autrement dit pour chaque itemset fermé f, l'opérateur de substitution induit une relation d'équivalence sur l'ensemble MG_f des générateurs minimaux associé à l'itemset fermé f. Le concept de redondance de générateurs minimaux à l'intérieur de la classe d'équivalence σ est par conséquent défini dans (Hamrouni et al., 2006) comme suit :

- **Définition 4.4** (Redondance des générateurs minimaux)

Soient g et g_1 deux générateurs minimaux appartenant à la même classe d'équivalence de fermeture γ. g est dit redondant direct à (ou dérivable de) g_1, noté $g_1 \vdash g$, si $Subst(g_1, g_2, g_3)= g$ avec $g_2 \subset g_1$ et $g_3 \in MG_K$ tel que g_3"= g_2". g est dit redondant transitif à g_1, noté $g_1 \models g$, s'il existe une séquence de n de générateurs minimaux (n>= 2), $gen_1, gen_2, \ldots gen_n$, tel que $gen_i \vdash gen_{i+1}$ (i ∈ [1… (n-1]) Avec $gen_1= g_1$ et $gen_n=g$.

Propriété 4.1

- La relation ⊢ est réflexive, symétrique mais pas nécessairement transitive
- La relation |= est réflexive, symétrique et transitive.

Selon la définition 4.4, si $g \in MG_f$ alors la classe d'équivalence σ de g notée par $[g]$ est un sous-ensemble de MG_F, qui consiste en tous les éléments redondants transitifs à g. Autrement dit $[g] = \{g_1 \in MG_f \mid g \models g_1\}$. Pour définir d'une manière unique le système succinct des

générateurs minimaux, (Hamrouni et al., 2006) ont adopté la même relation d'ordre totale décrite dans définition 4.2. Le plus petit générateur minimal est considéré alors comme générateur minimal représentatif alors que les autres sont étiquetés de générateurs minimaux redondants.

Exemple 4.3

Soit le contexte d'extraction K défini par le tableau 4.1. La relation d'ordre totale $\leq_s$ est l'ordre alphabétique (tableau 4.2). Le générateur minimal *abc* est un générateur minimal succinct associé à l'itemset fermé *abcdef* (puisqu'il est le plus petit de sa classe d'équivalence σ) Nous avons $abc \leq_s abd$ et $abc \leq_s bcf$. Le générateur minimal *abd* est redondant puisque $Subst(abc, bc, bd)=abd \in MG_{abcdef}$ ($abc \vdash abd$ et ainsi $abc \models abd$)

Le système succinct des générateurs minimaux est alors redéfini comme suit :

- **Définition 4.5** (*SSMG* par (Hamrouni et al., 2006))

Un système succinct des générateurs minimaux est l'ensemble de tous les générateurs minimaux succincts de tous les itemsets fermés.

4.2.4 Problèmes du système succinct des générateurs minimaux redéfini

Malheureusement, l'étude mené par les auteurs (Hamrouni et al., 2007) a montré que le *SSMG* redéfini dans (Hamrouni et al., 2006) présente encore un inconvénient majeur qui touche la propriété principale caractérisant l'ensemble des générateurs minimaux d'un contexte K et qui est la propriété d'idéal d'ordre. En effet, si nous considérons l'extraction effectuée sur le contexte K, selon l'ordre ascendant de support telle qu'elle est illustrée dans le tableau 4.2, le générateur minimal *ecf* serait caractérisé comme générateur minimal succinct puisqu'il est le plus petit de sa classe d'équivalence σ.

Tableau 4.4 Ensemble succinct des générateurs minimaux du *SSMG* redéfini

	Ordre **alphabétique**			Ordre **ascendant** de support			Ordre **descendant** de support		
	CI	**MGs**	**MGsuc**	**CI**	**MGs**	**MGsuc**	**CI**	**MGs**	**MGsuc**
1	*Ø*	*Ø*	***Ø***	*Ø*	*Ø*	***Ø***	*Ø*	*Ø*	***Ø***
2	*a*	*a*	***a***	*a*	*a*	***a***	*a*	*a*	***a***
3	*b*	*b*	***b***	*b*	*b*	***b***	*b*	*b*	***b***
4	*c*	*c*	*c*	*c*	*c*	*c*	*c*	*c*	*c*
5	*d*	*d*	***d***	*d*	*d*	***d***	*d*	*d*	***d***
6	*be*	*e*	***e***	*eb*	*e*	*e*	*be*	*e*	***e***
7	*f*	*f*	***f***	*f*	*f*	***f***	*f*	*f*	***f***
8	*ab*	*ab*	***ab***	*ab*	*ab*	***ab***	*ba*	*ba*	***ba***
9	*acf*	*ac, af, cf*	***ac, af, cf***	*acf*	*ac, af, cf*	***ac, af, cf***	*fac*	*fa, fc, ac*	***fa, fc, ac***
10	*ad*	*ad*	***ad***	*ad*	*ad*	***ad***	*da*	*da*	***da***
11	*abcdef*	*ae, abc, abd, abf, acd, adf, bcf, bdf, cdf, cef, def*	***ae, abc, acd,***	*eacbdf*	*ea, ecf, edf, acb,acd, abd,abf, adf, cbf, cdf, bdf*	***ea, ecf, acd***	*bdface*	*ae, bdf, bda, bfa, bfc, bac, dfa, dfc, dfe, dac, fce*	***ae, bdf, dfa***
12	*bcde*	*bc, bd, ce, de*	***bc, bd, ce, de***	*ecbd*	*ec, ed, cb, bd*	***ec, ed, cb, bd***	*bdce*	*bd, bc, de, ce*	***bd, bc, de, ce***
13	*bf*	*bf*	***bf***	*bf*	*bf*	***bf***	*bf*	*bf*	***bf***
14	*cd*	*cd*	***cd***	*cd*	*cd*	***cd***	*dc*	*dc*	***dc***
15	*df*	*df*	***df***	*df*	*df*	***df***	*df*	*df*	***df***
16	*bef*	*ef*	***ef***	*ebf*	*ef*	***ef***	*bfe*	*fe*	***fe***

Toutefois, le sous-ensemble *cf* de *ecf* n'est pas le plus petit de sa classe d'équivalence de fermeture γ, autrement dit *cf* n'est pas générateur minimal représentatif conformément à la définition originale donnée par (Dong et al., 2005). Par conséquent des tests supplémentaires sont nécessaires pour vérifier si un générateur minimal est succinct ou pas. Ainsi la compacité du système succinct des générateurs minimaux est conditionnée par un calcul exhaustif de substitutions entre les générateurs minimaux. Ceci a amené les auteurs à définir une troisième représentation succincte des générateurs minimaux par l'introduction de la famille des *DSFS* (Directed Subtitution free set) notée *FDSFS,* qu'ils affirment être une représentation succincte des générateurs minimaux. Ce système est étudié en détail dans la section suivante puisqu'il constitue une partie de notre travail de recherche

4.3 Représentation succincte des générateurs minimaux sans perte d'information

Afin de pallier aux lacunes du système succinct des générateurs minimaux proposé par (Dong et al., 2005) puis redéfini par (Hamrouni et al., 2006), (Hamrouni et al., 2007) ont introduit un nouveau système succinct de générateurs minimaux qu'ils ont prouvé être une représentation succincte sans perte d'informations de l'ensemble des générateurs minimaux du contexte d'extraction. Lors de la définition du nouveau système, les auteurs ont tenu compte de trois différents facteurs ayant eu un impact majeur dans la définition des deux systèmes succincts précédents, il s'agit de :

- Axiome d'Armstrong de pseudo-transitivité ($X \rightarrow Y$ et $WY \rightarrow Z$ alors $WX \rightarrow Z$)

Dans la substitution utilisée par (Dong et al., 2005) puis formalisée par (Hamrouni et al., 2006), les contraintes de la règle de pseudo-transitivité sont appliquées comme suit :

(i) WY, X et Y représentent les générateurs minimaux

(ii) X et Y appartiennent à la même classe de fermeture γ

(iii) Z représente la fermeture de WY

- Statut du générateur minimal dans sa nouvelle classe d'équivalence induite par la définition d'une fonction qui divisera sa classe d'équivalence de fermeture γ, en plusieurs autres classes d'équivalence.

- Ordre additionnel sur le treillis des itemsets qui induit un idéal d'ordre différent mais relié en quelque sorte à l'idéal d'ordre précédent, et constitué de générateurs minimaux représentatifs. Ce denier est complété par les générateurs minimaux canoniques constituant un sous ensemble de la bordure négative des générateurs minimaux représentatifs et formant ainsi un idéal d'ordre plus grand représentant le système succinct des générateurs minimaux et contenant les deux sortes de générateurs minimaux(les représentatifs et les canoniques).

Selon l'analyse effectuée par (Hamrouni et al., 2007), il est impossible, d'avoir les trois construits alignés. Un système succinct de générateurs minimaux, composés uniquement de générateurs minimaux représentatifs et de générateurs minimaux canoniques, ne permet pas de retrouver tous les générateurs minimaux redondants. Les auteurs proposent donc de compléter la représentation succincte précédemment définie, avec tous les non générateurs minimaux qui forment la bordure négative des générateurs minimaux représentatifs, puisque ces derniers représentent le point unique à partir duquel certains générateurs minimaux redondants peuvent être accessibles par substitution. Une illustration de cette approche est détaillée dans le paragraphe suivant.

4.3.1 Approche *DSFS_MINER*

Soient le contexte d'extraction K et l'ordre ascendant de support sur les items tels que définis dans le tableau 4.4. Considérons l'ensemble de générateurs minimaux associé à l'itemset fermé *eacbdf*. Comme nous l'avons souligné précédemment, il est impossible de dériver le générateur minimal redondant *ecf* à partir du système succinct des générateurs minimaux (tableau 4.4). En effet son sous-ensemble *cf* est un générateur minimal non représentatif de la classe d'équivalence de fermeture γ correspondant à *acf* (ce dernier possède *ac* comme générateur minimal représentatif). Par conséquent, *ecf* reste en dehors du système succinct et aucune chaine de substitution aboutissant à *ecf* ne peut être appliquée à partir de *ea* ni de

acd. La seule substitution qui parait raisonnablement possible de *cf* est *cf/ac*. Ceci produit l'itemset *eac*, non générateur minimal mais dont tous les sous-ensembles sont générateurs minimaux représentatifs (par conséquent *eac* appartient à la bordure négative de l'idéal correspondant). Ajouter *eac* au système succinct des générateurs minimaux, rétablit donc sa complétude et son intégralité. Ceci nécessite une définition plus large de la canonicité qui est définie par (Hamrouni et al., 2007) comme suit :

- **Définition 4.6** (Bordure négative des générateurs minimaux représentatifs)

Soit $MGrep_K$ l'ensemble générateurs minimaux représentatifs pouvant être extraits du contexte $K(O, I, R)$. La bordure négative des générateurs minimaux représentatifs est définie comme suit :

$$Bordure(MGrep_K) = \{X \subseteq I \mid \forall\ Y \subset X, Y \in MGrep_K \wedge X \notin MGrep_K\}$$

Comme les itemsets canoniques nouvellement redéfinis, forment la bordure négative des générateurs minimaux représentatifs, les générateurs minimaux précédemment définis comme canoniques dans (Dong et al., 2005), sont par conséquent contenus dans cette bordure ($MGcan_K \subseteq Bordure(MGrep_K)$) tout comme les itemsets canoniques non générateurs minimaux. Afin de formaliser le statut d'irréductibilité des systèmes succincts de générateurs minimaux déjà proposés dans la littérature, (Hamrouni et al., 2007) se sont appuyés sur la contrainte d'opérateur de substitution. Deux types de substitutions ont été distingués par les auteurs. Ces types sont complémentaires et dépendent du statut des itemsets concernés par la substitution. Ainsi, une substitution positive (respectivement négative) sur un itemset X consiste à remplacer un sous ensemble Z_1 de X par un sous ensemble Z_2 ayant la même fermeture que Z_1 mais étant supérieur (respectivement inférieur) selon la relation d'ordre totale $\preceq_s$ appliquée sur le treillis des itemsets.

- **Définition 4.7** (Substitution positive versus Substitution négative)

Soient $X, Y \subseteq I$, $Z_1 \subset X$ et $Z_2 \subseteq I$ tel que $Z_1 \neq Z_2$, $Z_1'' = Z_2''$, et $Subst(X, Z_1, Z_2) = Y$. Les relations de substitution positive/négative sont définies comme suit :

1) $X|\!-^{+} Y$ si et seulement si $Z_1 \leq_s Z_2$.

2) $X|\!-^{-} Y$ si et seulement si $Z_2 \leq_s Z_1$

Selon cette définition, il est clair que toute substitution est soit positive, soit négative, c'est-à-dire qu'il n'existe pas de substitution neutre. Par ailleurs, les substitutions positives produisent des résultats plus grands selon la relation d'ordre $\leq_s$ appliquée sur le treillis des itemsets, que les itemsets initiaux, alors que les substitutions négatives ont un effet opposé. En particulier, si l'itemset remplacé est un représentatif, alors la substitution est nécessairement positive alors que si c'est le représentatif qui remplace un autre itemset, il s'agit donc d'une substitution négative. Les itemsets irréductibles pour la substitution négative ou itemsets pour lesquels la substitution négative ne peut pas s'appliquer, sont appelés par (Hamrouni et al., 2007) « Directed substitution-free sets » (noté : DSFSs).

- **Définition 4.8** ($DSFS_K$)

soit $DSFS_K$ une collection des *DSFS*s pouvant être extraits d'un contexte K.

$DSFS_K = \{X \subseteq I \mid \forall X_1 \subset X, \forall X_2 \subseteq I, (X_1' = X_2' \rightarrow X_1 \leq_s X_2)\}$

Exemple 4.4

Considérons le contexte *K* avec l'ordre ascendant de support tel que spécifié par le tableau 4.4. L'itemset *eac* est un *DSFS*, comme mentionné ci-dessus bien que la famille comprend *ea* et *acd* et non *ecf*. Il est clair que l'ensemble *DSFS*s est égal à l'union des générateurs minimaux représentatifs et de leur bordure négative ($DSFS_K = MGrep_K$ U Border ($MGrep_K$)). Suite à cette définition, La proposition suivante fut immédiate par (Hamrouni et al., 2007) :l'ensemble $DSFS_K$ est un idéal d'ordre du treillis booléen $2^I = (\mathcal{P}(I), \subseteq)$.

Étant donnée cette structure, la famille de *DSFS* peut facilement être construite par un algorithme à niveau qui, énumère séquentiellement les itemsets dans l'ordre $\leq_s$ appliquée sur le treillis des itemsets. Ainsi tous les *DSFS*s à un niveau donné sont facilement reconnaissables puisque tous leurs sous-ensembles (en particulier les maximaux) appartiennent à la partie de la famille déjà découverte. Un effort additionnel est nécessaire

pour identifier les générateurs minimaux représentatifs parmi les membres de la famille. À cette fin, les propriétés d'ordre sont exploitées : En effet le générateur minimal représentatif est le premier de sa classe d'équivalence de fermeture γ à être examiné, donc pour vérifier si le générateur minimal candidat est un représentatif, il suffit de vérifier si sa fermeture a déjà été produite par un *DSFS* précédemment extrait. Selon (Hamrouni et al., 2007) n'importe quel ordre total sur les itemsets, génère un idéal dans le treillis booléen qui fonctionne comme un noyau irréductible pour échanger des sous-ensembles avec d'autres sous-ensembles équivalents. Inversement il y'a le processus d'expansion qui commence avec un *DSFS* et retrouve la famille entière de générateurs minimaux .En effet, selon (Hamrouni et al., 2007), avec une suite de substitutions positives sur les *DSFS*s constituant l'ensemble $DSFS_K$, la famille entière des générateurs minimaux du contexte K est reconstituée, autrement dit, tous les générateurs minimaux redondants peuvent être dérivés à partir des *DSFSs* via une chaine de substitutions positives sur les *DSFS*s composant $DSFS_K$. D'où le théorème suivant énoncé par (Hamrouni et al., 2007)

- **Théorème 4.1**

L'ensemble $DSFS_K$ des *DSFS*s est une représentation succincte de l'ensemble des générateurs minimaux du contexte K.

4.3.2 Algorithme *DSFS_MINER*

Dans cette section, nous esquissons les idées clés lié à un algorithme permettant l'extraction des *DSFS*s appelé *DSFS_Miner*, proposé par (Hamrouni et al., 2007). Cet algorithme parcourt le treillis des itemsets par niveaux et traite par conséquent les générateurs minimaux candidats par ordre ascendant de taille. Pour une taille donnée, les générateurs minimaux candidats associés, sont triés selon la relation d'ordre total $\preceq_s$ appliquée sur les itemsets. Ceci est obtenu naturellement du moment que la relation d'ordre total $\preceq_s$ est appliquée d'abord sur les items puis étendue sur les itemsets.

Algorithme 4.1 *DSFS_Miner*

Entrées K : contexte d'extraction où les items sont triés selon un ordre total $\leq_s$, et *minSupport* : Seuil minimal de support défini par l'utilisateur

Sortie $FDSFS_K$: ensemble des DSFSs fréquents

1) $FDSFS_K = \{\emptyset\}$;
2) $FCI_K \leftarrow \{\emptyset"\}$;
3) $FMGrep_1 \leftarrow I \setminus \{\emptyset"\}$;
4) **pour** ($i \leftarrow 2$; $FMGrep_i \neq \emptyset$; i++) **faire**
5) // Déterminer la fermeture et le support des MGrep candidats
6) $FMGrep_i \leftarrow$ GEN-Fermeture ($FMGrep_{i-1}$)
7) // Déterminer les DSFS fréquents et les M$Grep_i$ fréquents à utiliser pour former
8) // les M $Grep_{i+1}$ candidats
9) **pour chaque** (c $\in$ $FMGrep_i$) **faire**
10) // Vérifier si le candidat est fréquent sinon l'élaguer dans MGrep et dans FDSFS
11) **si** (c.Supp < *minSupport*) **alors**
12) $FMGrep_i = FMGrep_i \setminus \{c\}$;
13) **sinon**
14) $FDSFS_K = FDSFS_K \cup \{c\}$;
15) **si** (c.fermé $\notin$ FCI_K) **alors**
16) $FCI_K = FCI_K \cup \{c.fermé\}$
17) **sinon**
18) $FMGrep_i = FMGrep_i \setminus \{c\}$;
19) **fin si**
20) **fin si**
21) $FMGrep_{i+1}$ = GEN-Représentative($FMGrep_i$);
22) **fin pour**
23) **Retourner** $FDSFS_K$
24) **Fin**

Algorithme 4.2 *GEN-Représentative*

Entrées $FMGrep_i$: L'ensemble des générateurs minimaux représentatifs fréquents de taille i
Sortie $FMGrep_{i+1}$: ensemble des générateurs minimaux représentatifs candidats de taille i+1
1) // Jointure des candidats possédant un préfixe commun de taille *i*-2
2) $FMGrep_{i+1} \leftarrow \emptyset$
3) **pour** chaque paire d'*itemsets* $I_1, I_2 \in FMGrep_i$ ne différant que par leur dernier *item* **faire**
4) $ci \leftarrow I_1 \cup I_2$;
5) $FMGrep_{i+1} \leftarrow FMGrep_{i+1} \cup \{ci\}$
6) **fin pour**
7) **pour** chaque *itemset* candidat $c \in FMGrep_{i+1}$ **faire**
8) Is-deleted =0; /*Cette variable vérifie si un des sous ensemble de c n'est pas fréquent
9) Is-covered =0; /* Cette variable vérifie si c est inclus dans la fermeture d'un de
10) ses sous-ensembles immédiat et si oui il n'est donc pas MG mais peut être un DSFS*/
11) **pour** chaque sous-ensemble $s \in$ c.subsets **faire**
12) **si** ($s \notin FMGrep_i$) **alors**
13) supprimer *c* de $FMGrep_{i+1}$
14) Is-deleted = 1; // ce n'est pas un DSFS
15) **sinon**
16) **si** (c $\subseteq$ s.fermé) **alors**
17) is-covered =1; // ce n'est pas un MGrep mais c'est un DSFS
18) **fin si**
19) **fin si**
20) **fin pour**
21) **si** (Is-deleted =0 et Is-covered =1) **alors**
22) $FDSFS_K = FDSFS_K \cup \{c\}$;
23) supprimer *c* de $FMGrep_{i+1}$
24) **fin si**
25) **fin pour**
26) Retourner $FMGrep_{i+1}$:

Tableau 4.5 Ensemble des *DSFS*s fréquents extraits du contexte *K* du tableau 4.1 (les MGs représentatifs sont en gras et les *DSFS* non MGs ∈ Bordure négative sont en rouge)

	Ordre **alphabétique**			Ordre **ascendant** de support			Ordre **descendant** de support		
	CI	**MGs**	**DSFS**	**CI**	**MGs**	**DSFS**	**CI**	**MGs**	**DSFS**
1	*Ø*	*Ø*	***Ø***	*Ø*	*Ø*	***Ø***	*Ø*	*Ø*	***Ø***
2	*a*	*a*	***a***	*a*	*a*	***a***	*a*	*a*	***a***
3	*b*	*b*	***b***	*b*	*b*	***b***	*b*	*b*	***b***
4	*c*	*c*	***c***	*c*	*c*	***c***	*c*	*c*	***c***
5	*d*	*d*	***d***	*d*	*d*	***d***	*d*	*d*	***d***
6	*be*	*e*	***e***, *be*	*eb*	*e*	***e***, *eb*	*be*	*e*	***e***, *be*
7	*f*	*f*	***f***	*f*	*f*	***f***	*f*	*f*	***f***
8	*ab*	*ab*	***ab***	*ab*	*ab*	***ab***	*ba*	*ba*	***ba***
9	*acf*	*ac, af, cf*	***ac***, *af, cf*	*acf*	*ac, af, cf*	***ac***, *af, cf*	*fac*	*fa, fc, ac*	***fa***, *fc, ac*
10	*ad*	*ad*	***ad***	*ad*	*ad*	***ad***	*da*	*da*	***da***
11	*abcdef*	*ae, abc, abd, abf, acd, adf, bcf, bdf, cdf, cef, def*	***ae***, *abc, acd,*	*eacbdf*	*ea, ecf, edf, acb,acd, abd,abf, adf, cbf, cdf, bdf*	***ea***, *acd, eac*	*bdface*	*ae, bdf, bda, bfa, bfc, bac, dfa, dfc, dfe, dac, fce*	***ae***, *bdf, bda, bfa, dfa, aef*
12	*bcde*	*bc, bd, ce, de*	***bc***, *bd, ce, de*	*ecbd*	*ec, ed, cb, bd*	***ec***, *ed, cb, bd*	*bdce*	*bd, bc, de, ce*	***bd***, *bc, de, ce*
13	*bf*	*bf*	***bf***	*bf*	*bf*	***bf***	*bf*	*bf*	***bf***
14	*cd*	*cd*	***cd***	*cd*	*cd*	***cd***	*dc*	*dc*	***dc***
15	*df*	*df*	***df***	*df*	*df*	***df***	*df*	*df*	***df***
16	*bef*	*ef*	***ef***	*ebf*	*ef*	***ef***	*bfe*	*fe*	***fe***

Ainsi, la procédure utilisée pour générer les générateurs minimaux candidats de taille i+1, à partir de ceux de taille i, respecte la relation d'ordre total puisque à chaque fois elle combine deux itemsets X et Y tels que $X \prec_s Y$ et X et Y partageant les (i-1) premiers items en commun. Ces derniers items sont augmentés par l'item restant dans X puis par l'item restant dans Y formant ainsi un générateur minimal candidat de taille i+1. Par conséquent la relation d'ordre total est toujours respectée. Le pseudo-code de *DSFS_Miner* proposé par (Hamrouni et al., 2007) est donné par l'algorithme 4.1

4.3.2.1 Extraction des *DSFS*s avec *DSFS_Miner*

L'algorithme *DSFS_Miner*, proposé par (Hamrouni et al., 2007) est un algorithme itératif d'extraction des *DSFS*s fréquents qui parcourt le treillis des itemsets fréquents par niveaux. Durant chaque itération m, un ensemble $FMGrep_m$ de m-générateur représentatif candidats est considéré, chaque élément de cet ensemble est composé de trois éléments : sa liste des (m-1) sous-ensembles du générateur candidat, sa fermeture et son support. À chaque itération m, l'algorithme met à jour l'ensemble des $FDSFS_K$ des *DSFS*s du contexte K, avec les *DSFS*s de taille m fréquents. Un exemple d'extraction des *DSFS*s à partir du contexte K du tableau 4.1 est donné par le tableau 4.5.

L'algorithme commence par initialiser l'ensemble $FMGrep_1$ des 1-générateurs avec les 1-itemsets (ligne 3) du contexte, c'est-à-dire avec les items de l'ensemble I du contexte d'extraction. Chacune des itérations m suivantes (lignes 4 - 22) consiste en trois phases.

La première phase consiste à déterminer les fermetures des m-générateurs dans $FMGrep_m$ et calculer leur support (ligne 6).Cette phase est réalisée par l'algorithme *Gen-Fermeture* (décrit par l'algorithme 3.2). Dans la phase deux, uniquement les éléments de $FMGrep_m$ qui ont un support au moins égal à minSupport (les fréquents) sont insérés dans l'ensemble $FDSFS_K$ des *DSFS*s fréquents du contexte K (lignes 11-20). La dernière phase consiste à générer les (m+1)-générateurs candidats à partir des m-générateurs fréquents de $FMGrep_m$ (ligne 21). Cette phase est réalisée par l'algorithme *Gen-Représentative* (décrit par l'algorithme 4.2). Ces itérations sont répétées jusqu'à ce qu'aucun nouveau générateur minimal représentatif candidat ne peut être généré, c'est-à-dire jusqu'à ce que $FMGrep_m$ soit vide.

Les générateurs candidats sont construits par l'algorithme *GEN-Représentative*. Ce dernier reçoit en entrée l'ensemble $FMGrep_m$ des *m*-itemsets fréquents, il retourne l'ensemble $FMGrep_{m+1}$ des (*m*+1)-candidats contenant les *(m+1)-générateurs* qui sont utilisés durant l'itération *m*+1. Cette procédure est constituée de quatre phases. Durant la première phase toutes les jointures des m-générateurs de $FMGrep_m$ possédant un préfixe commun de taille *m*-1 sont effectués. Les résultats de ces jointures sont des (*m*+1)- candidats générateurs qui sont insérés dans $FMGrep_{m+1}$(lignes 3-6).

La seconde phase consiste à vérifier pour chaque générateur potentiel créé, la présence de tous ses sous-ensembles de taille m dans $FMGrep_m$ (lignes 12 -15). Si ce test est vérifié alors le (*m*+1)-générateur candidat est élagué de $FMGrep_{m+1}$ parce qu'il est sur-ensemble d'un itemset non fréquent. La troisième phase consiste à vérifier pour chaque générateur candidat potentiel créé, s'il est inclus dans la fermeture de l'un de ses sous-ensembles et le supprime s'il vérifie cette condition car selon la définition de générateur minimal il ne serait pas générateur minimal (lignes 17-20). La dernière phase consiste à ajouter à l'ensemble des *DSFS* fréquents ($FDSFS_K$), les non-générateurs minimaux appartenant à la bordure négative des générateurs minimaux représentatifs qui sont facilement identifiables puisqu'ils représentent les candidats de l'ensemble $FMGrep_{m+1}$ des générateurs minimaux représentatifs, élagués dans la phase trois de l'algorithme (lignes 22-25).

4.4 Extraction des règls d'assoiation succinctes et informatives

(Bastide et al., 2000) ont montré que le couple (*BI*, *BG*) constituait une représentation conservatrice, valide et informative de l'ensemble des règles d'association valides du contexte. Cependant (Hamrouni et al., 2007) ont prouvé que les générateurs minimaux composant les prémisses et conclusions de ces bases contiennent encore de la redondance. Afin d'extraire des ensembles plus compactes de règles d'association, (Hamrouni et al., 2007) ont proposé d'intégrer le concept de la représentation succincte des générateurs minimaux dans le cadre des bases génériques. (Hamrouni et al., 2007) ont donc utilisé la définition de la base (*BI*, *BG*) proposée par (Bastide et al., 2000) pour définir une nouvelle base (*SBI*, *SBG*) succincte et informative des règles d'association.

- **Définition 4.9** (Base générique succincte *SGB* des règles d'association exactes)

Soient FCI_K l'ensemble des itemsets fermés extraits du contexte *K*, et $FMG_{Suc\,f}$ l'ensemble des généraeurs minimaux succincts fréquents associé à chaque itemset fermé *f* de FCI_K *(les DSFS générateurs minimaux associés à f)*.La base générique succincte pour les règles d'association exactes est définie comme suit : $SGB = \{R : g \rightarrow (f \backslash g) \mid f \in FCI_K \wedge g \in FMG_{Suc\,f} \wedge g \neq f\}$.

- **Définition 4.10** (Réduction transitive succincte *SBI* des règles d'association approximatives)

Soit FMG_{SucK} l'ensemble des générateurs minimaux succincts fréquents extrait du contexte *K* (*sous ensemble de l'ensemble* $FDSFS_K$ *des DSFSs extraits du contexte K*).La réduction transitive succincte des règles d'association approximatives est définie comme suit : $SBI = \{R : g \rightarrow (f \backslash g) \mid f \in FCI_K \wedge g \in FMG_{SucK} \wedge f_1 \prec^6 f$ où $f_1 = g''\}$.

Tout comme les règles génériques exactes définies dans (Bastide et al., 2000), une règle générique exacte succincte est une association intra-classe avec une valeur de confiance égale à 1 à l'intérieur d'une classe d'équivalence de fermeture γ du treillis (Hamrouni et al., 2007).

Et tout comme les règles génériques approximatives définies dans (Bastide et al., 2000), une règle générique approximative succincte est une association interclasses avec une mesure de confiance inférieure à 1, entre une classe de fermeture γ et une autre appartenant à sa couverture supérieure (Hamrouni et al., 2007).

4.5 Conclusion

Nous avons présenté dans ce chapitre un état de l'art des systèmes succincts de générateurs minimaux proposés dans la littérature afin d'éliminer la redondance localisée au niveau des générateurs minimaux composant les prémisses des bases génériques et informatives des

---

[6] $\prec$ représente l'opérateur de couverture ($f \in$ à la couverture de f_1)

règles d'association. Un premier système succinct des générateurs minimaux (*SSMG*) a été introduit par (Dong et al., 2005) et constituait une première étape vers une réduction sans perte d'informations de cette redondance. Cependant, il a été montré dans (Hamrouni et al., 2006), que les affirmations faites dans (Dong et al., 2005) sur la nature de représentation sans perte d'informations du système succinct défini, n'étaient pas correctes. Pour remédier aux problèmes du système succinct des générateurs minimaux initial, (Hamrouni et al., 2006) ont donc proposé un deuxième système succinct de générateurs minimaux qui vient améliorer l'ancien et restaurer sa perte d'information, par l'ajout d'éléments supplémentaires à la base du système SSMG initial. Mais encore la, même avec cette redéfinition effectuée dans (Hamrouni et al., 2006), le nouveau *SSMG* présenterait selon (Hamrouni et al., 2007) un inconvénient majeur puisque le nouvel ensemble des générateurs minimaux le constituant ne préserve plus la propriété principale de l'ensemble des générateurs minimaux et qui est sa structure d'idéal d'ordre. Un troisième système succinct a été alors proposé par (Hamrouni et al., 2007), il s'agit de la famille des *DSFS*s. Cette famille est considérée jusqu'à aujourd'hui comme étant le seul système succinct à constituer une représentation succincte sans perte d'informations de l'ensemble des générateurs minimaux du contexte. L'intégration de ce concept de représentation succincte des générateurs minimaux dans le cadre des bases génériques a permis à (Hamrouni et al., 2007) de définir un sous-ensemble du couple (*BI*, *BG*) proposé par (Bastide et al., 2000), pour représenter la base générique informative succincte (*SBI*, *SBG*) des règles d'association valides.

Dans le cadre de ce projet de mémoire , un processus d'expansion est proposé et sera appliqué à la nouvelle représentation succincte des générateurs minimaux pour dériver l'ensemble de tous les générateurs minimaux redondants afin de constituer avec la représentation succincte des générateurs minimaux (Famille des *DSFS*s), la famille entière des générateurs minimaux pouvant être extraits du contexte. L'expansion est décrite d'une façon détaillée dans le chapitre suivant. Selon (Hamrouni et al., 2007), avec l'utilisation de ces générateurs minimaux redondants, toutes les règles d'association redondantes pourront être dérivées de la base générique informative succincte (*SBI*, *SBG*).

CHAPITRE V

DÉRIVATION DES GÉNÉRATEURS MINIMAUX REDONDANTS DE LA REPRÉSENTATION SUCCINCTE DES GÉNÉRATEURS MINIMAUX

5.1 Introduction

La collection $FDSFS_K$ proposé par (Hamrouni et al., 2007) forme, une représentation succincte sans perte d'informations de l'ensemble des générateurs minimaux fréquents d'un contexte *K*. Ce chapitre est consacré à l'expansion de cette représentation succincte des générateurs minimaux qui constitue notre contribution. Elle consiste à définir un processus produisant à partir de la représentation succincte des générateurs minimaux fréquents, l'ensemble de tous les générateurs minimaux redondants fréquents.

Une description détaillée du processus d'expansion est présentée dans la suite. Elle est suivie de la description de l'algorithme *DSFS_Expander* que nous proposons, lequel réalise notre processus d'expansion. Nous le validons avec les résultats de nos expérimentations présentés à la dernière section de ce présent chapitre.

5.2 Principe de l'expansion

Hamrouni et al., 2007), ont défini l'ensemble $DSFS_K$ (respectivement $FDSFS_K$)[7] des *DSFS*s (respectivement *DSFS*s fréquents). Ils ont prouvé que cet ensemble constitue un idéal d'ordre dans $(2^I,\subseteq)$ et ils ont démontré qu'il est une représentation succincte sans perte d'informations des générateurs minimaux (respectivement générateurs minimaux fréquents)

[7] DSFS $\in$ $FDSFS_K$ est un *DSFS* fréquent du contexte K

pouvant être extraits du contexte *K*. Ils ont défini $DSFS_K$ comme étant l'union de l'ensemble des générateurs minimaux représentatifs du contexte *K* et de la bordure négative de l'ensemble des générateurs minimaux représentatifs.
La collection $FDSFS_K$ est générée par l'algorithme *DSFS_Miner*, à laquelle va être appliqué l'algorithme que nous proposons dans le cadre de l'expansion, afin de retrouver la famille entière des générateurs minimaux fréquents MG_K.

La famille MG_K est constituée par l'ensemble de tous les générateurs minimaux fréquents associés à chacun des itemsets fermés fréquents. Elle peut être construite progressivement par notre algorithme. Ce dernier va parcourir la collection $FDSFS_K$, et traiter chaque *DSFS* fréquent contenu dans la collection, dans l'ordre dans lequel il est généré par *DSFS_Miner*, et va générer au fur et à mesure à partir du *DSFS* fréquent traité, les éléments composants la collection MG_K.

Le *DSFS* fréquent appartenant à la collection $FDSFS_K$ est stocké dans un enregistrement ayant les trois champs suivants (Support, fermeture, liste de sous ensembles immédiats de l'itemset générateur), par conséquent l'itemset générateur n'est pas spécifié explicitement dans le *DSFS*, il est plutôt défini par la liste de ses sous ensembles immédiats composant le troisième champ du *DSFS*.
Tout *DSFS* fréquent appartenant à la collection $FDSFS_K$[8], est soit un générateur minimal représentatif fréquent , soit un générateur minimal fréquent ou un non-générateur minimal appartenant à la bordure négative des générateurs minimaux représentatifs fréquents associés aux itemsets fermés fréquents du contexte *K*.

Pour chaque itemset fermé fréquent donné, il existe unique *DSFS* dans $FDSFS_K$ contenant son générateur minimal représentatif. Il correspond au premier *DSFS* rencontré dont la fermeture coïncide avec le fermé en question, lors du parcours de la collection $FDSFS_K$ par notre algorithme d'expansion. En effet, ce dernier a été le premier à produire sa fermeture et par conséquent le premier à constituer le *DSFS* pour l'itemset fermé. Il est donc le premier

[8] Dans la suite, nous reférons aux éléments DSFS fréquents de l'ensemble $FDSFS_K$ par DSFS uniquement.

DSFS fréquent avec pour l'itemset fermé à être inséré dans la collection $FDSFS_K$ par l'algorithme d'extraction des *DSFS*s fréquents.

Chacun des éléments composant la liste des sous-ensembles immédiats, d'un *DSFS* non générateur minimal représentatif (donc appartenant à la bordure négative des générateurs minimaux représentatifs), est un générateur minimal représentatif de sa propre fermeture.

Le *DSFS* non générateur minimal, appartenant à la bordure négative, est utilisé uniquement pour dériver les générateurs minimaux redondants fréquents et il devrait donc être élagué par notre algorithme d'expansion, une fois que tous les générateurs minimaux redondants fréquents pouvant en être dérivés, sont générés.

Le processus d'expansion doit démarrer du *DSFS* fréquent produit par l'algorithme *DSFS-Miner*, pour retrouver la famille entière de générateurs minimaux fréquents. L'opérateur de substitution positive $\vdash^+$ tel que définit par (Hamrouni et al., 2007) et rappelé ci-dessous, serait utilisé à cette fin.

- **Définition 5.1** (Relation de substitution positive)

(Cette définition est une partie de celle définie dans définition 4.7)
Soient $X, Y \subseteq I, Z_1 \subset X$ et $Z_2 \subseteq I$ tel que $Z_1 \neq Z_2$, $Z_1'' = Z_2''$, et $Subst\ (X, Z_1, Z_2)=Y$. La relation de substitution positive est définie comme suit :
$X \vdash^+ Y$ si et seulement si $Z_1 \lesssim Z_2$.

Nous affirmons que chaque itemset dans le treillis booléen est nécessairement accessible par au moins une chaîne de substitutions positives en partant d'un *DSFS*, particulièrement les générateurs minimaux redondants. En effet, chaque générateur minimal redondant peut être dérivé à partir d'un *DSFS* de la même fermeture en utilisant une suite successive de substitutions positives. Plus spécifiquement, en partant du *DSFS* X, et opérant des substitutions successives d'un sous ensemble Z_1 représentatif par un ensemble Z_2 appartenant à la même classe d'équivalence γ que Z_1, nous arriverons nécessairement à la génération de la famille entière des générateurs minimaux. Cependant, il peut arriver que de

différentes suites de substitutions positives produisent le même générateur minimal redondant. C'est pourquoi, nous proposons d'effectuer dans notre algorithme d'expansion, une validation de chaque résultat de substitution positive et élaguer le générateur minimal redondant s'il s'avère qu'il a déjà été produit.

Le processus de récupération doit aussi assurer l'exactitude du mécanisme, c'est-à-dire, la garanti que seulement les générateurs minimaux redondants fréquents sont dérivés. À cet effet un test sur le support, doit être effectué : Un itemset est un générateur minimal si et seulement si son support est strictement inférieur aux supports de tous ses sous-ensembles. Pour des raisons d'efficacité, le test sera limité aux sous-ensembles maximaux seulement. Ce test doit être effectué aussi bien sur le *DSFS* (avant de procéder à l'opération de substitution pour vérifier s'il est un non générateur minimal de la bordure négative, que sur l'itemset obtenu après chaque opération de substitution. Normalement, un test sur le support exige un balayage par niveau du treillis Booléen. Ceci est une approche classique pour la recherche d'itemsets fréquents, mais dans notre approche, afin d'éviter ce balayage, nous proposons d'utiliser la propriété des générateurs minimaux (Pasquier., 2010) ainsi que les propriétés de l'ensemble $FDSFS_K$, telles que prouvées dans (Hamrouni et al., 2007) :

- **Propriété 5.1** (Pasquier., 2000): Les sous ensembles d'un générateur minimal fréquent g sont aussi des générateurs minimaux fréquents.

- **Propriété 5.2** (Hamrouni et al., 2007) : X et $Subst(X, Z_1, Z_2)$ ont la même fermeture Donc : Si $X \vdash Y$ alors $X'' = Y''$ et support (X) = Support $(Y$

- **Propriété 5.3** (Hamrouni et al., 2006): Les Propriétés d'ordre sont respectées dans l'ensemble $FDSFS_K$ des *DSFS*s fréquents extraits du contexte K.

Le générateur minimal le plus petit d'une classe de fermeture donnée (et donc son représentatif) est le premier *DSFS* produit avec la fermeture en question, il est donc le premier de sa classe d'équivalence à être énuméré dans la collection $FDSFS_K$.Notre algorithme d'expansion peut parcourir la collection $FDSFS_K$ produite par l'algorithme

d'extraction des *DSFS*s, contenant les *DSFS*s fréquents, énumérés suivant la relation d'ordre totale sur les itemsets $\preceq_s$, puis peut appliquer, tout en respectant le même ordre, une chaine de substitution positive sur chacun des *DSFS*s. L'algorithme va ainsi générer une collection MG_K dont la cardinalité est égale à celle des itemsets fermés fréquents du contexte K, et composée d'éléments renfermant chacun un itemset fermé fréquent, son support, son générateur minimal représentatif ainsi que la liste globale de tous les générateurs minimaux fréquents qui lui sont associés. Par conséquent, il est clair que lors du parcourt de la collection $FDSFS_K$ et de la construction de la collection MG_K, les générateurs minimaux les plus petits associés à un itemset fermé donné, sont atteints en premiers et sont ainsi ajoutés en premier, à la liste des générateurs minimaux fréquents associés à un itemset fermé fréquent donné. Donc pour savoir si un itemset Y obtenu via substitution positive de X, est un générateur minimal, il suffit de vérifier si les conditions suivantes sont satisfaisantes en utilisant les propriétés décrites ci-dessus :

- Déterminer la fermeture ainsi que le support de Y : selon la propriété 2, la fermeture et le support de Y sont exactement les mêmes que ceux de X qui sont déjà connus puisqu'ils composent le *DSFS* (le premier élément de la chaine de substitution).

- Y est générateur minimal fréquent que si ses sous-ensembles sont générateurs minimaux fréquents n'ayant pas sa fermeture. Comme les sous ensembles immédiats de Y sont extraits du contexte avant Y et leurs fermetures sont produites avant la fermeture de Y, ils sont donc plus petits que Y par rapport à la relation d'ordre $\preceq_s$. S'ils sont des *DSFS*s, ils sont les premiers énumérés dans $FDSFS_K$ .S'ils ne sont pas des *DSFS*s, ils sont les premiers produits à partir d'opérations de substitutions. Par conséquent les sous-ensembles de Y sont les premiers traités et ajoutés par *DSFS_Expander*, dans la liste des générateurs minimaux fréquents associée à un itemset fermé fréquent. À partir de ces sous-ensembles, et de la propriété 1, le statut de Y (générateur minimal ou non générateur minimal) est déterminé.

L'algorithme *DSFS_Expander* est un algorithme itératif qui parcourt la collection $FDSFS_K$ séquentiellement et traite chacun des *DSFS*s de la collection dans l'ordre dans lequel il est

énuméré dans la collection. L'algorithme effectue une chaine de substitutions positives sur chaque *DSFS* traité afin de générer tous les générateurs minimaux redondants pouvant être dérivés du *DSFS* en question. Cette chaine de substitutions positives est effectuée sur le *DSFS* en appliquant sur ce dernier les substitutions positives d'une façon récursive. Autrement dit, une première substitution positive est effectuée sur le *DSFS* puis une nouvelle substitution positive est appliquée à chaque résultat produit de la substitution positive jusqu'à ce que plus aucune substitution ne soit possible.

Une validation est effectuée par notre algorithme d'expansion pour vérifier le résultat produit par chaque substitution positive. Parmi ces résultats, uniquement ceux satisfaisant la condition d'être des générateurs minimaux sont considérés par notre algorithme comme générateurs minimaux redondants et sont par conséquents utilisés pour y appliquer de nouvelles substitutions positives afin de générer d'autres générateurs minimaux redondants. Autrement dit une chaine de substitution effectuée récursivement sur un *DSFS* s'arrête dès que le résultat de la substitution positive produise un non générateur minimal. Ainsi le principe général de notre algorithme peut être résumé en l'application d'une substitution positive sur le *DSFS* et sur tout résultat de substitution positive non déjà produit, et satisfaisant la condition d'être générateur minimal. La description de notre algorithme d'expansion ainsi que son pseudo-code sont présentés dans la sous-section suivante.

5.2.1 Algorithme d'expansion

La génération de toute la famille des générateurs minimaux fréquents à partir de la collection $FDSFS_K$ est effectuée par *DSFS_Expander*. Ce dernier est constitué de deux autres procédures auxquelles il fait appel respectivement pour la validation des générateurs minimaux et pour la génération des générateurs minimaux redondants et que nous avons décrits comme deux algorithmes indépendants nommés respectivement par *EstUnMg* et *RedundantMgGen*.

Dans le reste du document tous nos exemples sont basés sur les données du tableau 4.5 des *DSFS*s extraits à partir du contexte d'extraction *K*.

5.2.1.1 Algorithme *DSFS_Expander*

L'algorithme *DSFS_Expander* est un algorithme séquentiel procédant d'une manière itérative en parcourant l'ensemble $FDSFS_K$ des *DSFS*s fréquents du contexte *K*, dans l'ordre dans lequel ils sont énumérés et en construisant au fur et à mesure la collection MG_K. Cette dernière contient pour chaque itemset fermé fréquent, son support, son générateur minimal représentatif fréquent et sa liste globale de tous les générateurs minimaux fréquents qui lui sont associés, produisant ainsi une collection MG_K ayant une cardinalité identique à celle des itemsets fermés fréquents du contexte *K*.

Un *DSFS* fréquent appartenant à la collection $FDSFS_K$ est caractérisé par trois champs (*Sup* : support, *CI*: fermeture, *MGSubsets* :liste des sous ensembles immédiats), et Un élément *C* de MG_K produit par *DSFS_Expander* est caractérisé par quatre champs (*Sup* : le Support, *CI*: l'itemset fermé fréquent, *MGRep* : le générateur minimal fréquent représentatif de sa classe d'équivalence de fermeture, *MGsSet* : l'ensemble de tous les générateurs minimaux fréquents associés à l'itemset fermé *CI*), par exemple, l'élément avec l'itemset fermé *abcdef* utilisant l'ordre alphabétique du tableau 4.5, serait inséré dans MG_K par *DSFS_Expander* sous la forme suivante : (1, *abcdef*, *ae*, {*ae, abc, abd, abf, acd, adf, bcf, bdf, cdf, cef, def*}).

L'ensemble FCI_K est utilisé par l'algorithme pour contenir tous les itemsets fermés fréquents du contexte *K*, accessibles via les *DSFS*s de $FDSFS_K$. Cet ensemble est mis à jour avec un nouvel itemset fermé fréquent, à chaque fois qu'un *DSFS* avec nouvel itemset fermé fréquent est atteint lors du parcours de $FDSFS_K$. Autrement dit, la mise à jour de FCI_K est effectuée à chaque fois que le *DSFS* atteint, renferme un générateur minimal fréquent représentatif, puisque le premier *DSFS* inséré dans la collection $FDSFS_K$ pour un itemset fermé fréquent donné, est celui renfermant le générateur minimal représentatif de sa classe d'équivalence de fermeture. En effet nous utilisons FCI_K à cette fin, il nous permet de savoir si le *DSFS* courant, renferme un générateur minimal représentatif puisque ce dernier est atteint juste une seule fois pour chaque itemset fermé fréquent. Il nous informe par conséquent l'action requise sur MG_K (un ajout ou juste une mise à jour du champ *MGsSet*).

Un ensemble $FMGrep_K$ est aussi utilisé pour contenir l'ensemble de tous les générateurs minimaux représentatifs fréquents du contexte *K*.

Pour des raisons d'efficacité, l'algorithme utilise deux structures de tables de hachages pour stocker l'ensemble des itemsets fermés fréquents (FCI_K) et l'ensemble des générateurs minimaux fréquents représentatifs($FMGrep_K$) au fur et à mesure qu'il construise MG_K. En effet l'itemset fermé et le générateur minimal représentatif sont uniques à chaque élément de MG_K. La table de hachage des générateurs minimaux représentatifs permet l'accès direct à l'itemset fermé à partir du générateur minimal représentatif alors que la table de hachage stockant les itemsets fermés permet l'accès direct à l'élément entier de MG_K à partir d'une fermeture.

L'algorithme fonctionne donc comme suit : Pour chaque *DSFS* atteint, un générateur minimal candidat *Mgc*, est constitué à partir des sous ensembles immédiats du *DSFS(Dsfs.MgsSubset)* et un contrôle est effectué pour vérifier si le *DSFS* renferme un générateur minimal représentatif ou bien un itemset appartenant à la bordure négative.

Cette vérification se fait avec l'itemset fermé contenu dans le *DSFS* (*Dsfs.CI*) qui est comparé avec le contenu de FCI_K et si la fermeture du *DSFS* n'existe pas dans FCI_K, nous déduisons qu'il s'agit d'un premier *DSFS* avec l'itemset fermé en question et que par conséquent l'itemset générateur du *DSFS* est bien le générateur minimal représentatif de sa classe de fermeture. L'itemset fermé contenu dans le *DSFS* est alors ajouté à la collection MG_K ($MG_K.CI$=*DSFS.CI*) avec son support ($MG_K.SUP$=*DSFS*.SUP), le générateur minimal candidat comme générateur minimal représentatif *MGRep* ($MG_K.MGRep$=*Mgc*), et aussi un premier élément dans son ensemble de générateurs minimaux *MGsSet* ($MG_K.MGsSet_1$ = {*Mgc*}) associé.

Si par contre l'itemset fermé du *DSFS* atteint, existe déjà dans FCI_K nous déduisons qu'il s'agit d'un itemset (générateur ou pas) appartenant à la bordure négative des représentatifs du

Algorithme 5.1 *DSFS_Expander*

Entrées $FDSFS_K$: Ensemble des DSFSs fréquents triés selon un ordre total $\preceq_s$

Sortie MG_K : ensemble des générateurs minimaux associés aux itemsets fermés fréquents

1) $FCI_K \leftarrow \{\varnothing"\}$;
2) $MG_K \leftarrow \{\varnothing\}$;
3) $FMGrep_K \leftarrow \{\varnothing\}$;
4) **Pour chaque** ($Dsfs \in FDSFS_K$) **faire**
5) $Mgc \leftarrow$ U (*Dsfs. MGSubsets*); // Constituer l'itemset générateur du DSFS
6) // Vérifier si le DSFS renferme un MG représentatif ou MG∈ bordure(MGRep)
7) **si** ($Dsfs.CI \notin FCI_K$ *ou* $MGC = \varnothing$) **alors** // Le DSFS renferme un MG représentatif
8) MG_K-*i*.CI← *Dsfs.CI;*
9) MG_K-*i*.Sup← *Dsfs.Sup;*
10) MG_K-*i*.MGSet← {*Mgc*};
11) MG_K-*i*.MGRep ← {*Mgc*};
12) $MG_K \leftarrow MG_K U \{MG_K\text{-}i\}$;
13) $FCI_K \leftarrow FCI_K U \{(Dsfs.CI\}$;// table de hachage qui donne accès à MG_{Ki}
14) $FMGrep_K = FMGrep_K U \{Mgc\}$;// table de hachage donnant accès à fermeture
15) **sinon** // Le DSFS renferme un MG ∈ bordure(MGRep)
16) MG_K-*i*← *obtenir de MGK(Dsfs.CI)*; /* accéder directement à l'élément de MG_K
17) contenant la fermeture du DSFS via la table de hachage FCI*/
18) **si** *EstUnMG* (*Dsfs*, MG_K) **alors** // Vérifier si l'itemset générateur du DSFS est MG
19) MG_K-*i. MGsSet*← MG_K-*i. MGsSet* U {*Mgc*};
20) **fin si**
21) **fin si**
22) // ajouter à l'ensemble des générateurs minimaux associé à la fermeture du *dsfs*
23) // tous les générateurs minimaux redondants pouvant être dérivés du *dsfs*
24) MG_K-i.*MGsSet* ← MG_K-i.*MGsSet* U *RedundantMGGen* (Dsfs, MG_K);
25) Retourner MG_K

contexte K et que par conséquent l'itemset fermé existe déjà dans MG_K. Il suffit dans ce cas, de le retrouver et mettre à jour l'ensemble des générateurs minimaux qui lui est associé. Avec la structure de table de hachage, l'élément de MG_k contenant la fermeture du *DSFS*, est accédé directement et l'ensemble MG_{Ki}.*MGsSet* associé à l'itemset fermé en question est mis à jour avec l'ajout de l'itemset générateur si ce dernier est validé par la fonction *EstUnMG*. Sinon ce dernier est élagué.

Que l'itemset générateur renfermé par le *DSFS* atteint, soit un générateur minimal représentatif ou appartenant à la bordure négative, une fois ajouté à (ou élagué de), MG_{Ki}.*MGsSet* associé sa fermeture, il est utilisé dans l'appel de la fonction *RedundantMGGen* pour compléter l'ensemble des générateurs minimaux lui étant associé, avec l'ensemble des générateurs minimaux redondants dérivés du *DSFS*. L'ensemble des générateurs minimaux redondants produit par la fonction *RedundantMGGen* est alors ajouté à l'ensemble des générateurs minimaux associés à la fermeture du *DSFS* atteint.

Prenons l'exemple du *DSFS eac* obtenu par rapport à l'ordre ascendant de support du tableau 4.5, qui est sous la forme ~~(1, *eacbdf*, {*ea, ec, ac*}), dans $FDSFS_K$.~~ Lorsque *DSFS_Expander* atteint ce *DSFS*, il n'est pas ajouté à la collection MG_K puisque sa fermeture existe déjà avec le générateur minimal représentatif *ea*.

Pour le *DSFS* courant *eac* l'algorithme met juste à jour l'ensemble MG_K.*MGsSet* des générateurs minimaux correspondant à sa fermeture MG_K.*CI=eacbdf* qui est défini à ce stade, comme suit : (MG_K.*Sup*=1, MG_K.*CI=eacbdf*, MG_K.*MGset*= {ea}). *eac* n'est pas un générateur minimal puisque son sous ensemble *ea* a la même fermeture et la fonction *EstUnMG* retourne 1. Par conséquent *eac* n'est pas ajouté à l'ensemble MG_K.*MGset* correspondant à sa fermeture *eacbdf*. Cependant il est utilisé pour générer les générateurs minimaux redondants via l'appel à *RedundantMGGen*. *RedundantMGGen* qui retourne en effet l'ensemble : *RedundantMgSet = {ecf, edf, cbf, bdf, acb, abd,abf}*. Cet ensemble retourné est ajouté à l'ensemble MG_K.*MGset* de l'itemset fermé MG_K.*CI=eacbdf*, et ce dernier est donc mis à jour comme suit (MG_K.*Sup*=1, MG_K.*CI=eacbdf*, MG_K.*MGSet*= {*ea ecf, edf, cbf, bdf, acb*

abd, abf}). Cette liste est complétée avec le reste des générateurs minimaux lors du traitement du DSFS *acd*.

5.2.1.2 Algorithme *EstUnMg*

Cette fonction reçoit en entrée un *DSFS*, pouvant contenir un itemset appartenant à la bordure négative des représentatifs ou alors un itemset obtenu via l'opération de substitution appliquée sur un générateur minimal. Elle reçoit aussi l'ensemble MG_K en cours de construction. Cette fonction consiste à valider si l'itemset contenu dans le *DSFS* reçu en entrée, est un générateur minimal.

Pour vérifier si le *DSFS* est un générateur minimal, *EstUnMG* considère chacun des sous ensembles immédiats composant le *DSFS* courant, qui sont déjà traités par *DSFS_Expander* puisqu'ils sont plus petits que le *DSFS* en question, et vérifie s'ils sont générateurs minimaux. Comme le sous-ensemble immédiat du *DSFS* est traité par *DSFS_Expander* avant le *DSFS* le contenant, alors s'il est générateur minimal, *DSFS_Expander* l'a définitivement déjà inséré dans la collection MG_K et par conséquent, il doit déjà appartenir à l'ensemble des générateurs minimaux *MGsSet* associé à un des itemsets fermés de la collection MG_K.

Cette fonction procède donc comme suit: elle utilise une variable booléenne *EstMG* initialisée à vrai ou à zéro, parcourt pour chacun des sous ensembles immédiats du *DSFS* (appartenant à *DSFS.MGsSet*) la collection MG_K, vérifie si le sous ensemble en question appartient à l'ensemble des générateurs minimaux *MGsSet* associé à un des itemsets fermés de MG_K, et retourne faux (ou 1) via la variable *EstMG* si le sous ensemble n'existe dans aucun ensemble des générateurs minimaux de MG_K (autre que celui associé à la fermeture du DSFS). Si au moins un des sous-ensembles n'est pas générateur minimal d'une fermeture autre que celle du *DSFS*, alors le *DSFS* est validé comme étant un non générateur minimal.
Par exemple lorsque *DSFS_Expander* atteint le *DSFS eac* (généré selon l'ordre ascendant de support tel que défini dans le tableau 4.5), il fait appel à la fonction *EstUnMG*, en lui passant les paramètres suivants : *DSFS* (1, *eacbdf*, {*ea, ec,ac*}), MG_K). La fonction initialise la variable booléenne *EstMg* à vrai et parcourt MG_K avec le premier sous-ensemble *ea* qu'elle

compare avec le contenu de l'ensemble *MGsSet* associé aux différents itemsets fermés de MG_K atteints. *ea* appartient à *MGsSet* associé à l'itemset *eacbdf* qui est l'itemset fermé du *DSFS* courant, ce qui amène la mise à jour à faux de *EstMG* et par conséquent le retour de *EstUnMG* avec l'affirmation que le *DSFS eac* n'est pas générateur minimal.

Algorithme 5.2 *EstUnMg*

Entrées *Dsfs :* élément de $FDSFS_K$
MG_K : ensemble des générateurs minimaux associés aux itemsets fermés
Sortie *EstMG* : 0 ou 1
1) *EstMG* ← 0;
2) **pour Chaque** (*MGC* ∈ *Dsfs.MGSubsets* et *EstMG)* **faire**
3) i ← 1;
4) // recherche dans la liste déjà découverte de MGs le sous-ensemble dsfs
5) **tant que** (i ≤ |MG_K| et MGC ∉ MG_{Ki}.*MGsSet*) **faire**
6) i ← i++
7) **fin tant que**
8) /* Si sous ensemble n'existe pas comme MG ou alors existe avec la même
9) fermeture que le dsfs */
10) **si** (i > |MG_K| ou MG_{Ki}.*CI* =*Dsfs.CI*) **alors**
11) *EstMG*←1;
12) **fin si**
13) **fin pour**
14) **retourner** *EstMG*

5.2.1.3 Algorithme *RedundantMGGen*

Cette fonction reçoit en entrée la collection des générateurs minimaux associés à chacun des itemsets fermés contenus dans les *DSFS*s déjà traités (MG_K en cours de conception par *DSFS_Expander*), ainsi qu'un *DSFS* à utiliser pour dériver tous les générateurs minimaux

redondants. La fonction retourne un ensemble *RedundantMGSet*. *RedundantMGSet* correspond en effet à l'ensemble des générateurs minimaux redondants dérivés du *DSFS* reçu en argument.

La fonction utilise l'opérateur de substitution positive pour générer les générateurs minimaux redondants. La substitution positive sur un *DSFS* consiste à remplacer un sous ensemble représentatif Z_1 du *DSFS* par un ensemble Z_2 de la même classe de fermeture que Z_1. Comme le *DSFS* est défini implicitement via ses sous ensembles immédiats spécifiés dans le champ *MGSubsets*, nous appliquons, l'opérateur de substitution, sur l'ensemble des sous-ensembles immédiats composant le *DSFS* en question. Par conséquent, pour dériver les générateurs minimaux à partir d'un *DSFS*, il suffit de recenser parmi ses sous-ensembles immédiats, ceux qui sont générateurs minimaux représentatifs. Par la suite, nous procédons à la substitution du *DSFS* en effectuant un remplacement de chacun de ces générateurs minimaux représentatifs, par des générateurs minimaux appartenant à la même classe d'équivalence. Le résultat de chacune de ces substitutions donne naissance à un générateur minimal redondant candidat qu'il faut valider avec la fonction *EstUnMG*, afin de pouvoir l'ajouter (ou l'élaguer) à l'ensemble *RedundantMGSet*.

Si le générateur minimal candidat est validé par *EstUnMG* comme non générateur minimal, alors il n'est plus utilisé pour générer d'autres candidats. En plus, plusieurs substitutions positives différentes peuvent donner un même candidat et c'est pourquoi dans notre algorithme nous vérifions l'existence du candidat obtenu après chaque opération de substitution positive.

Les sous-ensembles *MGSubsets* d'un *DSFS*, comme nous l'avons déjà mentionné ci-dessus, sont plus petits par rapport à la relation d'ordre que l'itemset renfermé par le *DSFS*. Par conséquent, les sous-ensembles du *DSFS* courant sont déjà traités par *DSFS_Expander* et existent donc dans MG_K. Donc pour effectuer la la substitution positive sur un *DSFS*, il suffit d'effectuer pour chacun de ses sous ensembles immédiats $Dsfs.MgSubsets_j$, les étapes successives suivantes :

Algorithme 5.3 *RedundantMGGen*

Entrées MG_K : ensemble des générateurs minimaux associés aux itemsets fermés fréquents
Dsfs : sous forme (*Sup*, *Ci*, *MGSubsets*)
Sortie *RedundantMgSet* : ensemble des générateurs minimaux redondants dérivé du Dsfs

1) *RedundantMgSet* ← {∅"};
2) MG_K ← {∅};
3) **Pour chaque** (*subDsfs* ∈ *Dsfs.MgSubsets)* **faire**
4) /*Utiliser la table de hachage de $FMGRep_K$ pour l'accès direct a un élément de MG_K
5) ayant comme un générateur minimal représentatif = sous-ensemble immédiat du Dsfs */
6) MG_K-*i*← *obtenir de MGK (subDsfs)*
7) **si** (MG_K-*i* ≠ *null* et | MG_K-*i.MGsSet* | > 1) **alors** // vérifier si la substitution est faisable
9) /* Opérer la substitution en remplaçant le sous ensemble immédiat du DSFS
10) (***subDsfs***) par chacun des générateurs minimaux de *MGK-i.MGsSet*/
12) **Pour chaque** (*Mg* ∈ MG_K-*i.MGsSet* ∧ *Mg* ≠ *subDsfs*) **faire**
13) *RedundantMg* ← (Dsfs \ *subDsfs*) ∪ *Mg* // opération de substitution
15) /* *RedundMGC a* la même structure que le DSFS et contient donc
16) le résultat de la substitution *RedundantMg* (pour l'appel récursif) */
17) *RedundMGC*.CI← *Dsfs.CI;*
18) *RedundMGC.Sup*← *Dsfs.Sup*
19) *RedundMGC.MgSubsets*←*subsets*(*RedundantMg*)
20) * Vérifier si le résultat de la substitution n'a pas été déjà produit
21) et s'il est générateur minimal */
22) **si** (*RedundantMg* ∉ *RedundantMgSet* ∧ *EstUnMg(RedundMGC)*)
23) *RedundantMgSet* = *RedundantMgSet* ∪ *RedundantMg* ∪
24) *RedundantMGGen(RedundMgC, MGK)*
25) **fin si**
26) **fin pour**
27) **fin si**
28) **fin pour**
29) **retourner** *RedundantMgSet*

1. Recherche dans MG_K s'il existe un élément MG_{Ki} possédant les caractéristiques suivantes : Un itemset fermé différent de l'itemset fermé du *DSFS* courant ($MG_{Ki}.CI <> DSFS.CI$), un générateur minimal représentatif égal au sous ensemble immédiat du *DSFS* courant ($MG_{Ki}.MGRep = Dsfs.MgSubsets_j$) et enfin la cardinalité de ensemble de générateurs minimaux associé à son itemset fermé $MG_{KI}.MGsSet$ est supérieure à 1. En effet, la substitution est opérée sur les générateurs minimaux représentatifs et donc pour pouvoir remplacer un sous ensemble immédiat d'un *DSFS* par un autre générateur minimal, il faut que le sous ensemble immédiat soit représentatif. Il est donc le plus petit d'une autre classe d'équivalence. Pour que son remplacement par un autre générateur minimal de la même fermeture soit possible, il faut que ce générateur minimal représentatif ne soit pas unique de sa classe d'équivalence.

   Par exemple, lors du traitement par *DSFS_Expander* du *DSFS* ***abc*** du tableau 4.5, produit selon la relation d'ordre totale alphabétique et définit Dans $FDSFS_K$ comme (**1, *abcdef***, {***ab, ac, bc***}), un appel vers la fonction *RedundantMGGen* est effectué par *DSFS_expander* avec le *DSFS* **(1, *abcdef*, {*ab, ac, bc*}).** Comme expliquée ci-dessus, *RedundantMGGen* procède à la substitution en considérant chacun des sous-ensembles immédiats du *DSFS*. Pour ce cas, il va donc traiter ***ab***, ensuite ***ac*** et enfin ***bc***. Lors du traitement de ***ab***, il va chercher dans MG_K, un MG_{Ki} qui a un itemset fermé $MG_{Ki}.CI$ différent du fermé de ***abc***, *Dsfs.CI =abcdef*, et un générateur minimal représentatif $MG_{Ki}.MGRep$ égal à ***ab*** et dont la cardinalité de l'ensemble de générateurs minimaux est supérieure à1. Ici aucune substitution ne peut être faite sur *ab*. Pourtant il existe bien un MG_{Ki} avec l'itemset fermé ***ab*** possédant un générateur minimal représentatif égal à ***ab,*** mais il est le seul générateur minimal de sa classe, donc il n'existe pas d'autres générateurs qui peuvent se substituer à ***ab.*** Par contre lors du traitement de ***ac***, la substitution est possible, puisqu'il existe un MG_{Ki} avec l'itemset fermé MG_{Ki}.CI=***acf***, possédant un générateur minimal représentatif $MG_{Ki}.MGRep$=***ac*** et dont $MG_{Ki}.MGsSet$ a une cardinalité de trois. Donc le sous ensemble ***ac*** du *DSFS* courant peut être remplacé par les deux générateurs minimaux

partageant la même fermeture, soit ***af*** et ***cf***. Après substitution, ils donnent naissance à des candidats pour *abc*,

2. Une fois l'étape1 complétée, la substitution devient possible sur le *DSFS* courant, avec le sous-ensemble immédiat ayant satisfait les conditions. La substitution est alors effectuée, en remplaçant le sous-ensemble immédiat représentatif par chacun des générateurs minimaux appartenant à la même classe d'équivalence. Le résultat de chacune de ces substitutions donne naissance à un générateur candidat *RedundMgC* qu'il faut valider avec l'appel à la fonction ***EstUnMG***. Si la fonction ***EstUnMG*** valide le résultat de la substitution comme étant un générateur minimal alors ce dernier est ajouté (s'il n'existe pas déjà), à l'ensemble des générateurs minimaux redondants du *DSFS*. Par la suite, il est utilisé pour dériver d'autres générateurs minimaux redondants du *DSFS*. En effet, il peut avoir des sous-ensembles représentatifs à remplacer par d'autres générateurs minimaux.

   Si nous prenons l'exemple précédent du *DSFS* ***abc*** du tableau 4.5 (selon la relation d'ordre totale alphabétique), avec le sous-ensemble immédiat de ***ac***, comme mentionné ci-dessus, la recherche du MG_{Ki} est positive et nous fournit donc le MG_{Ki} ayant l'itemset fermé $MG_{Ki}.CI=acf$, le générateur minimal représentatif $MG_{Ki}.MGRep=ac$ et l'ensemble des générateurs minimaux $MG_{Ki}.MGsSet=\{ac,af,cf\}$. Nous avons donc lors de la première substitution qui remplace ***ac*** par ***af***, le générateur minimal candidat suivant : *RedundMgC .MgSubsets*={*ab*, *af*, *bf*} dont l'union donne un itemset générateur minimal candidat *abf* qui est validé par EstUnMg. ***abf*** est donc ajouté à l'ensemble des générateurs minimaux redondants dérivé de *abc*. Ensuite un appel récursif est fait à la fonction ***MGRedundantGen*** pour dériver des générateurs minimaux redondants à partir de ***abf***. Cet appel ne donne aucun générateur dérivé de *abf*. Nous passons donc à la deuxième substitution à effectuer, toujours avec *ac,* mais cette fois-ci avec le deuxième générateur minimal, ***cf***. Le résultat de cette deuxième substitution donne le générateur minimal redondant candidat *bcf* qui est validé par ***EstUnMg*** comme générateur minimal. *bcf* est donc ajouté à l'ensemble des générateurs minimaux redondants *RedundantMgSet={abf,*

bcf} Ensuite un appel récursif est fait à la fonction ***MGRedundantGen*** pour dériver des générateurs à partir de *bcf*. Contrairement au précédent, il produit, suite à une chaine successive de substitutions, un *RedundantMgSet= {bdf, cef,def}* qui est ajouté à l'ensemble précédent pour produire *{ abf, bcf, bdf, cef,def}*. Cette substitution n'est pas complète pour *abc*, puisqu'elle a été effectuée que sur le sous-ensemble représentatif ***ac***. Avec d'autres substitutions sur le sous-ensemble immédiat représentatif ***bc***, nous obtenons le générateur minimal redondant ***abd*** qui est ajouté à l'ensemble ***RedundantMgSet*** pour représenter l'ensemble des générateurs minimaux redondants ***RedundantMgSet***={ *abf, bcf, bdf, cef,def, abd*} du *DSFS* (*1, abcded, abc*).

5.3 Résultats expérimentaux

La validation du processus complet d'extraction d'une représentation succincte des générateurs minimaux et d'une dérivation à partir de celle-ci, de la famille entière des générateurs minimaux, a été effectuée via des expérimentations et des implémentations de deux algorithmes *DSFS_Miner* et *DSFS_Expander*. Nous avons utilisé comme base de comparaison l'algorithme *Close*. Ces algorithmes ont été implémentés en Java, et ont été appliquées sur des jeux de données fréquemment utilisés afin de comparer les performances des algorithmes. Il s'agit des données secondaires provenant des fichiers de données du domaine public. Des tests de performance et de fiabilité ont été effectués et portaient sur les temps d'exécution effectués par chacun des trois algorithmes. Les résultats d'expérimentation ont été répertoriés et sont présentées dans cette section sous forme de courbes de variations. Une comparaison des résultats d'expérimentation nous a permis de vérifier les résultats théoriques et hypothèses annoncées.

Toutefois, les courbes représentant la performance en temps d'exécution de *DSFS_Miner* dans les figures 5.2, 5.3, 5.5 et 5.6 résultent de notre propre implémentation de l'algorithme car la version optimale telle qu'elle a été définie par (Hamrouni et al., 2007) n'était pas disponible.

5.3.1 Jeux de donnés

Les tests ont été effectués à partir de deux fichiers de données du domaine public correspondant respectivement à des données denses (*mushrooms*) et à des données éparses (gaz10k).
Nous rappelons que les données éparses sont caractérisées par des items de faible support (les ensembles d'objets associés à ces items sont de taille réduite). Les données denses quand à elles, se distinguent par leur support élevé (les ensembles d'objets associés à ces items sont de grande taille).

Les caractéristiques de ces jeux de données sont énoncées dans le tableau ci-dessous.

Tableau 5.1 Caractéristiques des jeux de données

Jeux de données	Nombre d'items	Nombre d'objets	Taille moyenne des objets
mushrooms	128	8416	23
Gaz10k	46 300	10 000	40

5.3.2 Résultats des expérimentations

5.3.2.1 Série de tests sur les données denses (Mushrooms)

- Un premier test de cette série, a été effectué pour déterminer le nombre de *DSFS*s versus le nombre de représentatifs extraits par *DSFS_Miner*, sur des données denses. La figure 5.1 décrit en effet le nombre de *DSFS*s et le nombre de Générateurs représentatifs extraits par *DSFS_Miner* en fonction du pourcentage du nombre total d'objets du contexte Mushrooms avec un support fixé à 10.

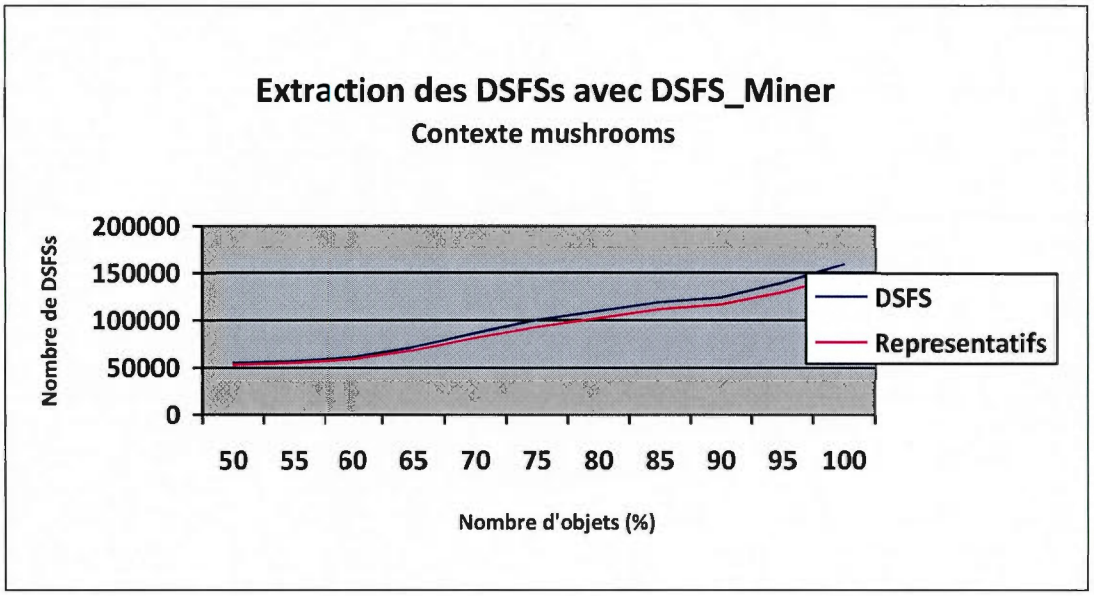

Figure 5.1 Courbes de DSFSs extraits de "mushrooms" avec *DSFS_Miner* (% Obj)

- Le deuxième test concerne la comparaison des algorithmes d'extraction des générateurs minimaux, l'algorithme *Close* avec l'union de *DSFS_Miner* et *DSFS_Expander*, sur des données denses telles que le contexte d'extraction "mushrooms". La figure 5.2 décrit le temps d'exécution effectué par chacun des quatre algorithmes (*Close*, *DSFS_Miner*, *DSFS_Expander*, et *DSFS_Miner+DSFS_Expander*) pour extraire les générateurs minimaux et leurs fermetures en fonction du pourcentage du nombre total d'objets du contexte Mushrooms et avec un support de fréquence fixé à 10. La figure 5.3 décrit par contre le temps d'exécution effectué par chacun des quatre algorithmes sur la totalité des données du contexte "Mushrooms" en fonction du support fixé par l'utilisateur

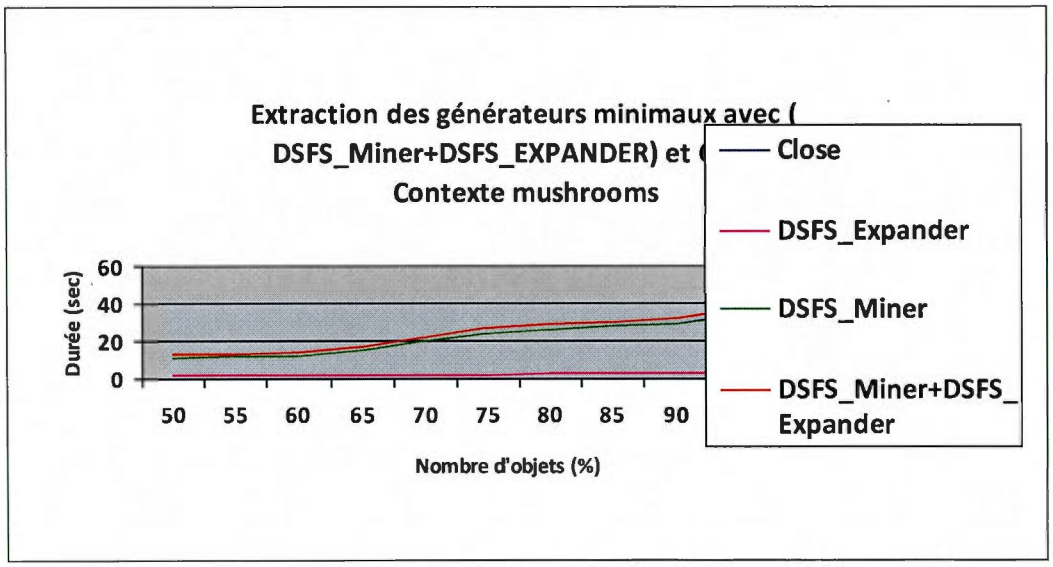

Figure 5.2 Courbes de MGs et IFFs extraits avec *Close*, *DSFS_Miner*, *DSFS_Expander*(1)

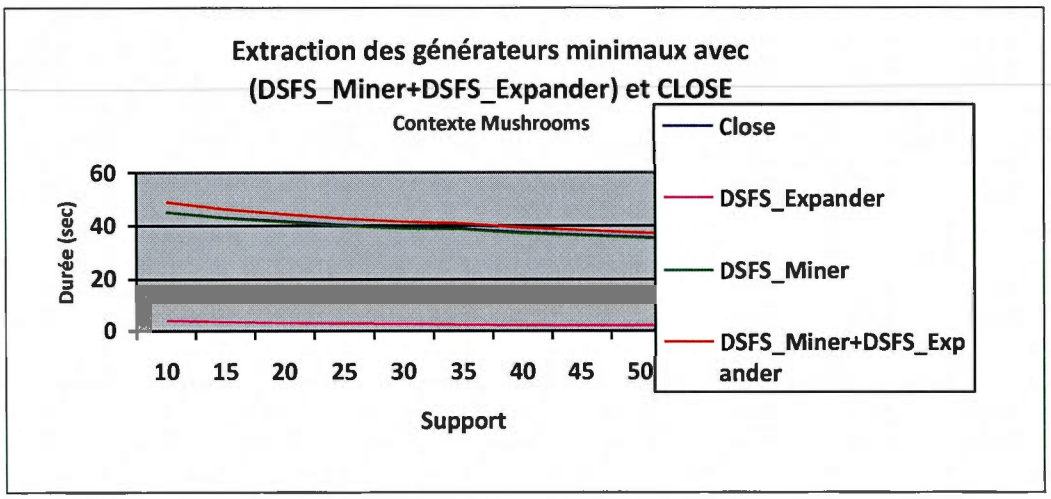

Figure 5.3 Courbes de MGs et des IFF extraits avec *Close*, *DSFS_Miner*, *DSFS_Expander*(2)

L'algorithme *DSFS_Expander* s'avère très performant dans l'extraction des générateurs minimaux sur des contextes denses, puisqu'il surpasse l'algorithme *Close* dans l'extraction des générateurs minimaux et les fermetures de ces derniers. Les expériences sur les données denses montrent qu'il est quarante fois plus rapide que *Close* pour un support de dix.

La performance de *DSFS_Expander* est justifiée d'une part par le nombre de balayages effectués par ce dernier qui se résume à un seul et unique balayage de l'ensemble des *DSFS*s et d'autre part par le calcul de support qui nécessite aucun traitement puisque ce dernier est dérivé directement du *DSFS* utilisé.

La performance en termes de temps d'exécution, de l'algorithme d'extraction des *DSFS*s (*DSFS_Miner*), dépasse légèrement celle de l'algorithme d'extraction des générateurs minimaux (*Close*) parce que le nombre de générateurs miminaux extraits par *Close* est généralement supérieur au nombre de *DSFS*s

La courbe de performance de l'algorithme d'extraction des générateurs minimaux (*DSFS_Miner* + *DSFS_Expander*), est comparable à celle de *Close* en terme de temps d'exécution sur Mushrooms. Performance de *Close* dépasse légèrement celle de (*DSFS_Miner* + *DSFS_Expander*).

5.3.2.2 Série de tests sur les données éparses (Gaz10k)

- Le premier test effectué dans cette deuxième série de tests concerne le nombre de *DSFS*s versus le nombre de représentatifs extraits par *DSFS_Miner* dans un contexte de données éparses. La figure 5.4 décrit en effet le nombre de *DSFS*s et le nombre de Générateurs représentatifs extraits par *DSFS_Miner* en fonction du pourcentage du nombre total d'objets du contexte gaz10k avec un support fixé à 10.

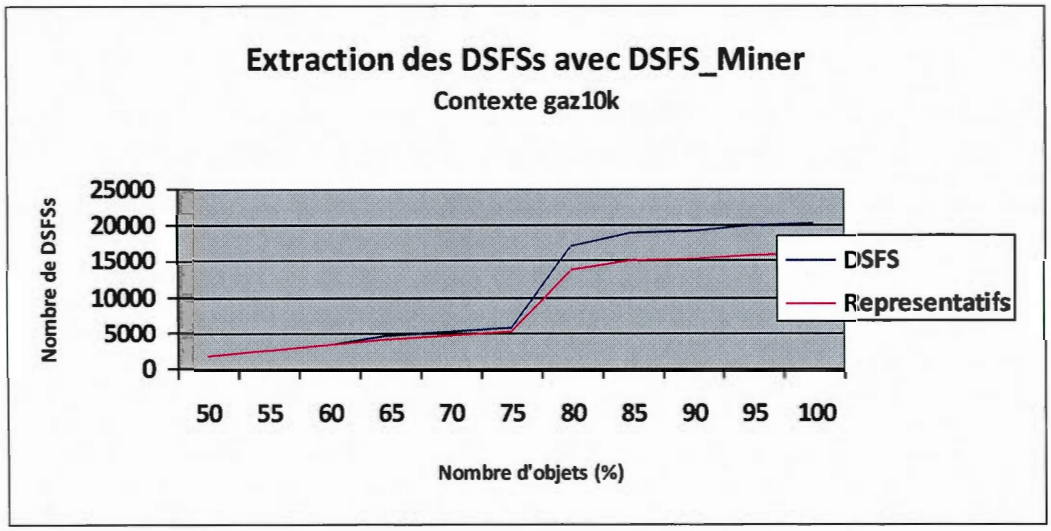

Figure 5.4 Courbes de DSFSs extraits de "gaz10k" avec *DSFS_Miner* (%Obj)

- Le deuxième test de cette série concerne la comparaison de l'algorithme *Close* avec les deux algorithmes réunis *DSFS_Miner* et *DSFS_Expander*, sur des données éparses telles que le contexte d'extraction "gaz10k" dans l'extraction des générateurs minimaux et des itemsets fermés. La figure 5.5 décrit le temps effectué par chacun des quatre algorithmes (*Close*, *DSFS_Miner*, *DSFS_Expander*, et *DSFS_Miner+DSFS_Expander*) pour extraire les générateurs minimaux et leurs fermetures respectives en fonction du pourcentage du nombre total de données de "gaz10k" et avec un support fixé à 10 pour les quatre extractions, alors que la figure 5.6 décrit le temps d'exécution effectué par chacun des quatre algorithmes sur la totalité des données du contexte "gaz10k" en fonction du support fixé par l'utilisateur.

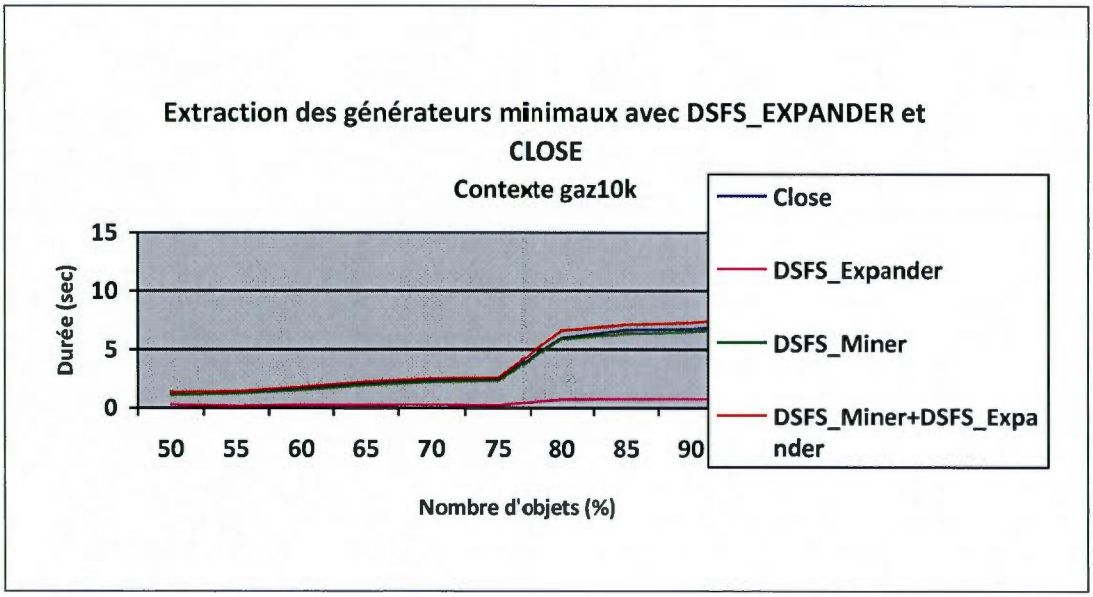

Figure 5.5 Courbes de MGs et IFFs extraits avec *Close*, *DSFS_Miner*, *DSFS_Expander*(3)

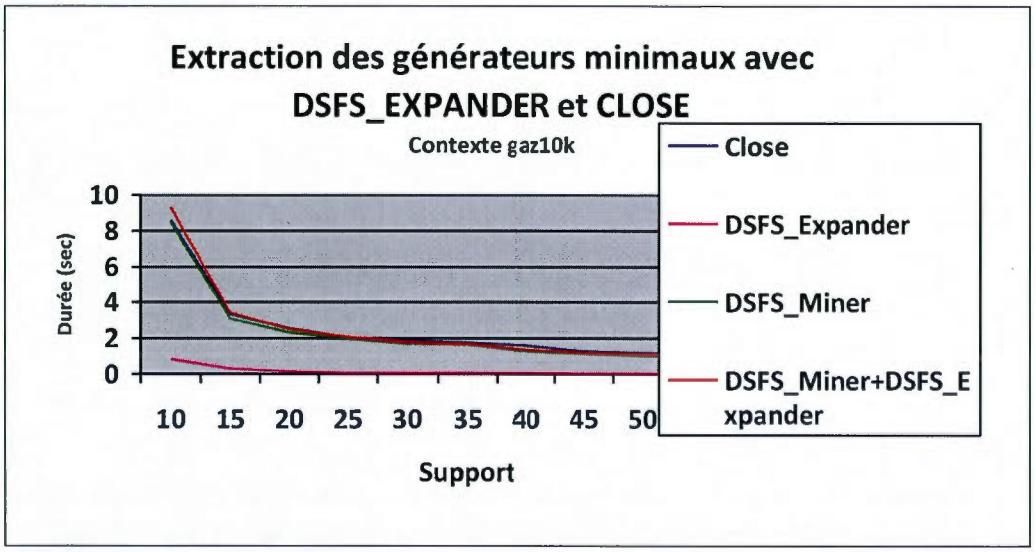

Figure 5.6 Courbes de MGs et IFFs extraits avec *Close*, *DSFS_Miner*, *DSFS_Expander*(4)

Encore la, l'algorithme *DSFS_Expander* dépasse l'algorithme *Close* en temps d'extraction, nos résultats d'expérimentation montrent en effet que pour un support faible, le temps d'exécution de *DSFS_Expander* est soixante fois plus rapide que *Close*, ceci se justifie également par le seul balayage de la collection des *DSFS*s effectué par ce dernier comparativement à celui effectué par *Close* qui est égal à la taille du plus long itemset fermé fréquent extrait du contexte.

La courbe de performance de *DSFS_Miner* dépasse légèrement celle de *Close* sur un contexte de données éparses parce que le nombre de *DSFS*s extraits est généralement inférieur à celui des générateurs minimaux.

La courbe de performance en terme de temps d'exécution de l'algorithme d'extraction de générateurs minimaux (*DSFS_Miner*+ *DSFS_Expander*) dépasse légèrement celle de *Close* sur un contexte de données éparses.

5.4 Dérivation des règles redondantes de (*BI, BG*) à partir de (*SBI, SBG*)

(Hamrouni et al., 2007) ont appliqué les *DSFS*s générateurs minimaux générés par *DSFS_Miner*, à la base de règles d'association (*BI, BG*) définie par (Bastide et al., 2000) et ont obtenu une nouvelle base (*SBI, SBG*) succincte et informative des règles d'association, et sous-ensemble de la base (*BI, BG*). Les règles d'association appartenant à (*SBI, SBG*) sont des implications entre les générateurs minimaux succincts fréquents (*DSFS*s générateurs minimaux) et les itemsets fermés fréquents. Par conséquent, pour dériver le couple (*BI, BG*) à partir de (*SBI, SBG*), il est nécessaire de déduire auparavant les générateurs minimaux redondants puisque ces derniers composent les prémisses des règles d'association génériques redondantes (règles d'association appartenant à (*BI, BG*) et élagués de (*SBI, SBG*)). Ces générateurs minimaux redondants sont ceux qui sont générés par notre algorithme *DSFS_Expander*. Ainsi, partant d'une règle d'association R: $g \rightarrow f \backslash g$ appartenant à (*SBI, SBG*), où $g \in FMGsuc_K$ et $f \in FCI_K$, l'ensemble des règles d'association génériques redondantes est donné par : $Regles\_Assoc\_Gen\_Red_R = \{R1: g_1 \rightarrow f \backslash g_1 \mid g_1 \in FMGred_K$ tel que $g \vdash + g_1\}$.

5.5 Conclusion

Nous avons présenté dans ce chapitre le processus d'expansion de la représentation succincte des générateurs minimaux proposé par (Hamrouni et al., 2007) et nous avons proposé notre algorithme d'expansion *DSFS_Expander* qui permet de dériver à partir de cette représentation, l'ensemble de tous les générateurs minimaux redondants pour constituer la famille entière des générateurs minimaux fréquents d'un contexte d'extraction donné.

L'algorithme *DSFS_Expander* est un algorithme d'extraction des générateurs minimaux avec leurs fermetures respectives à partir de la représentation succincte des générateurs minimaux. Une analyse empirique a montré qu'il est très performant aussi bien sur les données éparses que sur les données denses, comparé aux algorithmes d'extraction des générateurs minimaux à niveau existants.

Les résultats obtenus de nos expérimentations nous ont montré que le temps d'extraction des générateurs minimaux par *DSFS_Expander* est plus court que le temps d'extraction des générateurs minimaux par Close.

En effet, contrairement aux autres algorithmes d'extraction des générateurs minimaux, *DSFS_Expander* fait un seul balayage de la base des DSFSs et ne nécessite aucun traitement pour le calcul de support, ce dernier est dérivé directement du DSFS utilisé pour la dérivation de tous les générateurs minimaux redondants.

Les générateurs minimaux redondants extraits par *DSFS_Expander* vont composer les prémisses des règles d'association redondantes de la base (*BI, BG*) dérivées à partir de la base succincte et informative (*SBI, SBG)* proposée par (Hamrouni et al., 2007).

CONCLUSION

La première partie de ce mémoire a été consacrée à l'approche classique de la découverte des règles d'association et à ses limites. Le problème majeur de cette approche (consistant en la génération de règles d'association à partir de l'extraction des itemsets fréquents), réside principalement dans la phase de découverte des itemsets fréquents. En effet, le temps d'extraction dans cette étape est très coûteux de par le nombre de balayages du contexte réalisé et le nombre d'itemsets candidats considéré. Ce dernier augmente exponentiellement avec la taille de l'ensemble d'items du contexte d'extraction. D'autre part, comme la génération des règles d'association est exponentielle au nombre d'itemsets fréquents extraits, l'ensemble des règles d'association fortes produites est très volumineux et sa taille peut compliquer l'interprétation par l'utilisateur.

À partir des résultats de l'analyse formelle des concepts, nous avons présenté dans le troisième chapitre, l'approche de la découverte de règles d'association par l'extraction des itemsets fermés. L'utilisation de ce nouvel ensemble a permis non seulement de diminuer le temps d'extraction des itemsets fréquents en réduisant l'espace de recherche des itemsets fréquents à celui des itemsets fermés mais aussi de préserver l'information véhiculée par les itemsets fréquents et par les règles d'association valides du contexte. D'autre part, l'utilisation de l'ensemble des itemsets fermés a permis de définir un sous ensemble de l'ensemble des règles d'associations valides appelés bases informatives. Ces bases sont définies à partir des générateurs minimaux composant leurs prémisses minimales, et d'itemsets fermés composant leurs conclusions maximales. Ces bases constituent un noyau irréductible des règles d'association permettant une meilleure exploitation des résultats par l'utilisateur.

Ces bases informatives ont constitué le noyau compact et irréductible des règles d'associations jusqu'a ce qu'une étude menée par (Dong et al., 2005) affirme en effet que les bases en question contiennent encore de la redondance au niveau des générateurs minimaux composant leurs prémisses. Dans ce contexte de réduction, nous avons présenté dans le quatrième chapitre l'approche de la représentation succincte des générateurs minimaux

définie dans la littérature pour éliminer cette redondance qui persiste dans les bases informatives des règles d'association. Parmi les systèmes succincts des générateurs minimaux proposés dans la littérature, l'ensemble des *DSFS*s défini par (Hamrouni et al., 2007) représente jusqu'à aujourd'hui l'unique système sans perte d'informations.

Dans le chapitre cinq, nous avons défini le processus d'expansion qui permet de dériver à partir de la représentation succincte des générateurs minimaux définie par (Hamrouni et al., 2007), l'ensemble des générateurs minimaux redondants du contexte, constituant ainsi la famille entière des générateurs minimaux du contexte. Nous avons proposé l'algorithme *DSFS_Expander* qui réalise l'expansion, qui s'est avérée selon les résultats de nos expérimentations être très performant en termes de temps d'exécution comparé à un algorithme d'extraction des générateurs minimaux à niveau (*Close*).

L'utilisation de la représentation succincte des générateurs minimaux dans la génération de bases informatives permet de produire un ensemble de bases de règles d'association de taille plus réduite que l'ensemble de bases informatives générées avec l'ensemble des générateurs minimaux extraits. Cette réduction de l'ensemble de règles permet de faciliter leur interprétation par l'utilisateur, et lui fournit toute l'information véhiculée par l'ensemble de toutes les règles d'association. Toutefois, les résultats obtenus de nos expérimentations nous montrent que l'extraction des *DSFS*s versus l'extraction de la famille entière des générateurs minimaux est intéressante lorsque la taille de l'ensemble des générateurs minimaux associé à aux itemsets fermés du contexte est importante. En effet, les *DSFS*s ne sont pas tous des générateurs minimaux. Par conséquent lorsque les *DSFS*s ne peuvent être utilisés pour la dérivation des générateurs minimaux redondants parce que la redondance est faible, nous pensons que l'extraction des *DSFS*s est moins avantageuse comparativement à l'extraction de l'ensemble des générateurs minimaux puisque le nombre de *DSFS*s dans ce cas peut dépasser le nombre de générateurs minimaux à cause des *DSFS*s non générateurs minimaux qui nécessitent une opération supplémentaire pour les élaguer.

Notre étude s'est concentrée sur la génération efficace de la famille des *DSFS*s et sur l'expansion efficiente de cette dernière pour retrouver la famille entière des générateurs minimaux. Les algorithmes proposés dans le cadre de la génération et l'expansion des *DSFS*s

appartiennent à la famille des algorithmes à niveaux. Dans les prochaines étapes, d'autres algorithmes d'extraction de générateurs minimaux existants dans la littérature, et appartenant à la famille d'algorithmes verticaux ou à la famille d'algorithmes hybrides, pourraient être adaptés pour l'extraction des *DSFSs*. Des algorithmes d'expansion verticaux ou hybrides pourraient aussi être conçus à cet effet pour dériver de ces DSFSs, les générateurs minimaux redondants. Il serait par la suite interessant, de comparer les performances de ces algorithmes sur différents contextes de données, denses et éparses.

BIBLIOGRAPHIE

Agrawal, R., et Srikant R., « Fast Algorithms forMining 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.

Agrawal R., Mannila H., Srikant R., Toivonen H., and Verkamo A. I., « Fast discovery of association rules », In *Advances in knowledge discovery and data mining*, pages 307_328., American Association for Artificial Intelligence, 1996.

Aho A.V., Hopcroft J. E., and Ullman J. D., « Data structures and algorithms. » Addison Wesley, 1985.

Armstrong W.W, « Dependency structures of database relationships. » In *IFIP Congress*, pages 580–583, September 1974.

Assaghir Zainab, «Analyse formelle de concepts et fusion d'informations : application à l'estimation et au contrôle d'incertitude des indicateurs agro-environnementaux», Thèse de d*octorat d'Université, Université INPL-Nancy (*spécialité Science agronomique*)*, France, Avril 2011

Barbut, M., Monjardet, B., « Ordre et Classification : Algèbre et combinatoire », Hachette, 1970.

Bastide Y., Paquier N., Taouil R., Lakhal L., Stumme G., « Mining minimal non-redundant association rules using frequent closed itemsets », *Proceedings of the Intl. Conference DOOD'2000, LNCS, Springer-verlag*, July 2000, p. 972-986.

Bastide Y., Paquier N., Taouil R., Lakhal L., Stumme G., « Mining frequent patterns with counting inference », *SIGKDD Explorations*, vol. 2, no 2, 2000, p. 66-75.

Bayardo R.J., Agrawal R., Gunopulos D., «Constraint-based rule mining in large, dense databases. », *Proc. ICDE conf.*, pp 188–197, March 1999.

Benyahia S., Cherif C. L., Mineau G., Jaoua A., « Découverte des règles associatives non redondantes : application aux corpus textuels », *Revue d'Intelligence Artificielle (special issue of Intl. Conference of Journées francophones d'Extraction et Gestion des Connaissances(EGC'2003)*, Lyon, France, vol. 17, no 1-2-3, 2003, p. 131-143.

Birkhoff, G., «Lattice theory» In *Colloquium Publications*, volume 25, American Mathematical Society, 1967, Third edition.

Brin Sergey, Motwani Rajeev, Ullman Jeffrey D., and Shalom Tsur, « Dynamic itemset counting and implication rules for market basket data », *Proceedings of the 1997 ACM SIGMOD international conference on Management of data - SIGMOD '97*, pages 255–264, 1997.

Brin Sergey, Motwani Rajeev, Silverstein C., « Beyond market baskets: Generalizing association rules to correlation. », *Proc. SIGMOD conf.*, pp 265–276, May 1997.

Davey, B. A., Priestley, H. A., « Introduction to lattices and order », *Cambridge University Press*, 1992.

Dong, G., Jiang, C., Pei, J., Li,J., Wong, L.: « Mining succinct systems of minimal generators of formel concepts. », *In proceedings of the 10th International Conference on Dtabase Systems for advanced Applications*(DASFAA 2005), Springer-Verlag, LNCS, Volume 3453, Beijing, China. (2005) 175-187

Duquenne V., Guigues J.-L, « Famille minimale d'implications informatives résultant d'un tableau de données binaires. », *Mathématiques et SciencesHumaines,* 24(95):5–18, 1986.

Fayyad, U.M., Piatetsky-Shapiro,G., and Smyth, P. « From Data mining knowledge discovery: An overview. » In U.M. Fayyad, G. Piatetsky-Shapiro, P.Smyth, and R.Uthurusamy, editors, *Advances in Knowledge Discovery and data mining*, pages 1-30 AAAI Press, 1996

Fayyad, U.M., Piatetsky-Shapiro,G., Smyth, P., Uthurrusamy, R.,editors, « Advances in Knowledge Discovery and data mining », pages 1-30 *AAAI Press*, 1996.

Ganter B., Wille R., « Formal Concept Analysis: Mathematical foundations. », *Springer*, 1999.

Gasmi G., «Proposition d'une nouvelle base générique de règles d'association», Rapport de mastère, Faculté des sciences de Tunis, Faculté des sciences de Tunis, Département Informatique, July 2005.

Gasmi G., BenYahia S., Nguifo E. M. et Slimani Y., « Discovering "factual" and "implicative" generic association rules. », In *Proceedings of the Intl. Conference CAP'05, Nice, France. Presses Universitaires de Grenoble*, pages 329–344, Juin 2005.

Gasmi G., BenYahia S., Nguifo E. M. et Slimani Y., « A new informative generic base of association rules. », In *Proceedings of the Ninth PacificAsia Knowledge Discovery and Data Mining Conference (PAKDD'05),Hanoi, Vietnam*, pages 81–90, May 2005.

Gasmi G. , Ben Yahia S., Mephu Nguifoy E. and Slimani Y., *IGB*: « une nouvelle base générique informative des règles d'association », *Tunisien CMCU 05G1412,* 2007

Godin, R., Saunders, E., Gecsel, J., « Lattice Model of Browsable Data Spaces. », *Information Sciences*, 40, 1986, p. 89-116.

Godin, R., Missaoui R., « An incremental Concept Formation Approach for learning from Databases. », In *Theoretical Computer Science*, 133:378-419, 1994

Han J., Fu Y., « Discovery of multiple-level association rules from large databases. », *Proc. VLDB conf.*, pp 420–431, September 1995.

Han J., Pei J., and Yin Y., « Mining frequent patterns without candidate generation.», *In SIGMOD '00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 1_12. ACM Press, 2000.

Heckerman D., « *Bayesian networks for knowledge discovery.* », *Advances in Knowledge Discovery and Data Mining*, pp 273–305. AAAI Press, 1996

Hamrouni, T., Ben Yahia, S., Mephu Nguifo, «E.: Redundancy-free generic bases of association rules. », *In: Proceedings of the 8th French Conference on Machine Learning (CAP 2006), Presses Universitaires de Grenoble, Trégastel*, France. (2006) 363–378

Hamrouni, T., Ben Yahia, S., Mephu Nguifo, E.: Succinct system of minimal generators: A thorough study, limitations and new definitions. In: Proceedings of the 4th International Conference on Concept Lattices and their Applications (CLA 2006), Hammamet, Tunisia. (2006) 139–153

Hamrouni T., «Itemsets fréquents et règles d'associations : extraction et réduction », *Presse université d'Artois de Lenz, LGI2A* Novembre 2007.

Hamrouni T., Valtchev P., Ben yahia S., Mephu Nguifo E., «About the Lossless Reduction of the Minimal Generator Family of a Context», *ICFCA 2007, LNAI 4390*, pp. 130–150.

Houtsma M., and Swami A., «Set-oriented data mining in relational databases», *Data & Knowledge Engineering*, 17 :245–262, 1995.

Klemettinen M., Mannila H., Ronkainen P., Toivonen H., Verkamo A. I., « Finding interesting rules from large sets of discovered association rules. », *Proc. CIKM conf.*, pp 401–407, November 1994.

Kryszkiewicz M., « Representative association rules. », In *Research andDevelopment in Knowledge Discovery and Data Mining. Proc. of SecondPacific-Asia Conference (PAKDD).* Melbourne, Australia, 1998.

Kryszkiewicz M., «Concise representations of association rules. », In D. J.Hand, N.M. Adams et R.J. Bolton, éditeurs, *Proceedings of Pattern Detection and Discovery, ESF Exploratory Workshop, London, UK*, volume 2447 of *Lecture Notes in Computer Science*, pages 92–109.Springer-Verlag, September 2002.

Lakshmanan V. S., Ng R. T., Han J., Pang A., « Exploratory mining and pruning optimizations of constrained association rules. » *Proc. SIGMOD conf.*, pp 13–24, June 1998.

Lefloch Amélie, « extraction des règles d'association dans le contexte de l'analyse formelle des concepts », *Mémoire Université du Québec à Montréal, Décembre 2003.*

Lin D-I., and Kedem Z. M, « Pincer search : A new algorithm for discovering the maximum frequent set », In *International Conference on Extending Database Technology*, pages 385–392, Valencia, Spain, 1998.

Lin JL. « Mining association rules: Anti-skew algorithms. », *Data Engineering, 1998. Proceedings.* Pages 486–493, 1998.

Luong V. Phan, « Raisonnement sur les règles d'association», In *Proceedings 17ème Journées Bases de Données Avancées BDA'2001, Agadir(Maroc), Cépaduès Edition*, pages 299–310, November 2001.

Luxenburger M., « Implications partielles dans un contexte », *Mathématiques,Informatique et Sciences Humaines*, 29(113):35–55, 1991.

Mannila H., Toivonen H., « Levelwise Search and Borders of Theories in Knowledge Discovery », *Data Mining and Knowledge Discovery*, vol. 1, no 3, p. 241-258, pages 181-192, September 1997.

Meo R., Psaila G., Ceri S., « A new SQL-Like operator for mining association rules », In *Proceedings of 22nd international conference on Very Large Data Bases* In *Data Mining and Knowledge Discovery (VLDB96)*, pages. 122-133, Morgan Kaufmann, September 1996.

Mueller Andreas, « Fast Sequential and Parallel Algorithms for Association RuleMining : Table of Contents », *Knowledge Creation Diffusion Utilization*, 0057(August) :1–76, 1995.

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.

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.

Pasquier N., Bastide Y., Taouil R., and Lakhal L., « Closed set based discovery of small covers for association rules. », *In Proc. 15emes Journees Bases de Donnees Avancees*, BDA, pages 361_381, 1999.

Pasquier N., Bastide Y., Taouil R., and Lakhal L., « Discovering frequent closed itemsets for association rules. », *Lecture Notes in Computer Science*, 1540:398_416, 1999.

Pasquier N., Bastide Y., Taouil R., Lakhal L., « Efficient Mining of Association Rules using Closed Itemset Lattices », *Journal of Information Systems*, vol. 24, no 1, 1999,p. 25-46, Elsevier Science.

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

Pei J., Han J., and Mao R., « CLOSET: An efficient algorithm for mining frequent closed itemsets. », *In ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21_30, 2000.

Piatetsky-Shapiro G., « Discovery, analysis and presentation of strong rules. », *Knowledge Discovery in Databases*, pp 229–248. AAAI Press, 1991.

Piatetsky-Shapiro G., Matheus C.J., « The interestingness of deviations. », *AAAI KDD workshop*, pp 25–36, July 1994.

Silberschatz A., Tuzhilin A., « What makes patterns interesting in knowledge discovery systems. », *IEEE Transactions on Knowledge and Data Engineering*, 8(6):970–974, December 1996.

Silverstein C., Brin S., Motwani R., « Beyond market baskets : Generalizing association rules to dependence rules. », *Data Mining and Knowledge Discovery*, 2(1):39–68, January 1998.

Silverstein C., Brin S., Motwani R., Ullman J.D., « Scalable techniques for mining causal structures. », In *Proceedings of 24th international conference on Very Large Data Bases (VLDB98),* pages 594-605. Morgan Kaufmann, August 1998.

Srikant R., Agrawal R., « Mining generalized association rules.», *Proc. VLDB conf.*, pp 407–419, September 1995.

Srikant R., Vu Q., Agrawal R., « Mining association rules with item constraints. », *Proc. KDD conf.*, pp 67–73, August 1997.

Stumme G., Taouil R., Bastide Y., Pasquier N., Lakhal L., « Fast Computation of Concept Lattices Using Data Mining Techniques », Bouzeghoub M., Klusch M., Nutt W., Sattler U., Eds., *Proceedings of 7th Intl. Workshop on Knowledge Representation Meets Databases (KRDB'00)*, Berlin, Germany, 2000, p. 129-139.

Szathmary Laszlo, « Méthodes symboliques de fouille de données avec la plate-forme Coron », Thèse de d*octorat d'Université, Université Henri-Poincaré-Nancy 1(spécialité informatique)*, France, Novembre 2008

Toivonen H., Klemettinen M., Ronkainen P., Hatonen K., Mannila H., « Pruning and grouping discovered association rules», *ECML MLnet workshop*, pp 47–52, April 1995.

Toivonen Hannu, « Sampling large databases for association rules », In *Proceedings of the twenty-second international Conference on Very Large Data Bases, September 3–6, 1996, Mumbai (Bombay),India*, pages 134–145, pub-MORGAN-KAUFMANN: adr, 1996.Morgan Kaufmann Publishers.

Valtchev P., Missaoui R., Godin R., Meridji M., « Generating frequent itemsets incrementally : two novel approaches based on Galois lattice theory », *J. Expt. Theoretical Artificial Intelligence*, vol. 14, no 1, 2002, p. 115-142.

Wang J., Han J., and Pei J., « CLOSET+: searching for the best strategies for mining frequent closed itemsets. », In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 236_245. ACM Press, 2003.

Wille R., « Restructuring lattice theory : an approach based on hierarchies of concepts. », *Ordered sets*, pages 445_470, 1982.

Zaki M. J., Parthasarathy S., Ogihara M., and Li W., « New Algorithms for Fast Discovery of Association Rules. », *In Proceedings of the 3rd Int'l Conference on Knowledge Discovery in Databases*, pages 283_286, August 1997.

Zaki, M. J., Hsiao, C.-J., « CHARM: An efficient algorithm for closed association rule mining. », *Technical Report 99-10, Renssaeler Polytechnic Institute*, Troy, New York, 1999.

Zaki M. J., «Scalable Algorithms for Association Mining. », *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372_390, 2000.

Zaki M. J., and Hsiao C.-J., « CHARM: An Efficient Algorithm for Closed Itemset Mining», *In SIAM International Conference on Data Mining SDM'02*, pages 33_43, Apr 2002.

Zaki M. J., and Gouda K., «Fast vertical mining using diffsets. », *In KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 326_335, New York, NY, USA, 2003. ACM Press.

Zaki M. J., «Mining non-redundant association rules. », *Data Mining andKnowledge Discovery : An International Journal(DMKDJ'04)*. Pages 223–248, November 2004.