Extraction de motifs séquentiels

Problèmes et méthodes

F. Masseglia [*] — **M. Teisseire** [**] — **P. Poncelet** [***]

[*] *INRIA Sophia Antipolis*
[**] *LIRMM - Université de Montpellier II - ISIM*
[***] *EMA/LGI2P - Nîmes*

SYNOPSIS. *Dans un premier temps, le problème de l'extraction de motifs séquentiels peut sembler proche de celui de l'extraction de règles d'association. Ce rapprochement s'avère cependant très fragile en raison d'un élément clé qui est propre à l'extraction de motifs séquentiels : la temporalité. Cette notion permet à la fois de distinguer à l'intérieur des enregistrements un ordre d'apparition mais aussi de regrouper certains éléments. En effet si les règles d'association s'appliquent à des données de type itemsets (et permettent l'extraction de règles intra-transaction), la recherche de motifs séquentiels s'applique à des données de type séquences d'itemsets (et permet donc l'extraction de règles inter-transactions). Nous proposons dans cet article de faire un pas en avant dans la compréhension du comportement des principaux algorithmes d'extraction de motifs séquentiels en expliquant et illustrant leur fonctionnement. De plus nous proposons de considérer les problématiques qui ont émergé depuis les motifs séquentiels comme l'extraction incrémentale et interactive, l'extraction sous contraintes, les motifs séquentiels fermés, les motifs séquentiels approximatifs et d'autres problématiques très proches.*

MOTS-CLÉS : *motifs séquentiels, extraction, approches incrémentales, breadth-first, depth-first, motifs séquentiels fermés*

1. Introduction

Introduits dans [AGR 95b] et largement étudiés dans [MAS 02], les motifs séquentiels peuvent être vus comme une extension de la notion de règles d'association intégrant diverses contraintes temporelles. La recherche de tels motifs consiste ainsi à extraire des enchaînements d'ensembles d'items, couramment associés sur une période de temps bien spécifiée. En fait, cette recherche met en évidence des associations inter-transactions, contrairement à celle des règles d'association qui extrait des combinaisons intra-transactions. Par exemple, des motifs séquentiels peuvent montrer que "60% des gens qui achètent une télévision, achètent un magnétoscope dans les deux ans qui suivent". Ce problème, posé à l'origine dans un contexte de marketing, intéresse à présent des domaines aussi variés que les télécommunications (détection de fraudes), la finance, ou encore la médecine (identification des symptômes précédant les maladies).

Règles d'association vs Motifs séquentiels : Elaborés en premiers lieux, les algorithmes de recherche de règles d'association connaissent de grandes difficultés d'adaptation aux problèmes d'extraction de motifs séquentiels. En effet, si le problème de la recherche de règles d'association est proche de celui des motifs séquentiels, les études dans ce sens [AGR 95b] montrent que, lorsque l'adaptation est possible, c'est au prix de temps de réponse inacceptables. Depuis la définition du problème dans [AGR 95b], de nombreuses approches destinées à résoudre la problèmatique de l'extraction de motifs séquentiels ont été proposées.

Dans cet article nous présentons les problèmes et méthodes liés à l'extraction de motifs séquentiels. De manière chronologique les premières méthodes proposées pour résoudre ce problème d'extraction étaient basées sur la principe d'une recherche "en largeur d'abord" (*breadth-first*) qui correspond au modèle d'Apriori (C.f. Section 2). Nous présenterons GSP, qui est l'algorithme pionnier de la recherche de motifs séquentiels généralisés ainsi que SPADE. La communauté s'est ensuite penchée sur le développement de techniques dites "en profondeur d'abord" (*depth-first*) très bien adaptées à cette problématique (C.f. Section 4). Nous ferons le point sur les méthodes implémentant ce principe comme PSP, PREFIXSPAN, ou encore SPAM qui est le premier à extraire des motifs séquentiels à l'aide de bitmaps. Nous présenterons également dans la section 4.4 les méthodes CLOSPAN et BIDE qui ont pour objectif l'extraction de motifs séquentiels fermés (*closed sequential patterns*). Nous donnerons une définition de ces motifs et détaillerons le fonctionnement de ces méthodes qui permettent d'élaguer l'espace de recherche. La recherche incrémentale de motifs séquentiels sera ensuite abordée dans la section 5 avec les méthodes ISE, FASTUP et KISP, ou encore ISM et IUS qui étudient les bénéfices de la bordure négative. Enfin de nombreuses contributions relatives aux motifs séquentiels, présentées section 6, concernent des extensions de cette problématique, avec par exemple l'extraction de motifs sous contraintes (de temps, d'expressions régulières) ou encore la recherche de motifs séquentiels approximatifs ou sans support minimum..

2. Recherche de motifs séquentiels : problématique et définitions

La problématique de l'extraction de motifs séquentiels peut être perçue comme une extension de celle de l'extraction de règles d'association. En effet la prise en compte de la temporalité dans les enregistrements à étudier permet une plus grande précision dans les résultats, mais implique aussi un plus grand nombre de calculs et de contraintes.

Le problème de la recherche de séquences dans une base de données de transactions est présenté dans [AGR 95b] de la façon suivante (nous gardons, au niveau des définitions, les concepts de clients et d'achats) :

Définition 1 Une *transaction*[1] constitue, pour un client C, l'ensemble des items achetés par C à une même date. Dans une base de données client, une transaction s'écrit sous la forme d'un ensemble : id-client, id-date, itemset. Un *itemset* est un ensemble d'items non vide noté $(i_1 i_2 \dots i_k)$ où i_j, avec j de 1 à k, est un *item* (il s'agit de la représentation d'une transaction non datée). Une *séquence* est une liste ordonnée, non vide, d'itemsets notée $< s_1 s_2 \dots s_n >$ où s_j est un itemset (une séquence est donc une suite de transactions qui apporte une relation d'ordre entre les transactions). Une *séquence de données* est une séquence représentant les achats d'un client. Soit $T_1, T_2, \dots, T_n$ les transactions d'un client, ordonnées par dates d'achat croissantes et soit itemset(T_i) l'ensemble des items correspondants à T_i, alors la séquence de données de ce client est $< itemset(T_1)\ itemset(T_2) \dots\ itemset(T_n) >$.

Exemple 1 *Soit C un client et S=$< (C)\ (D\ E)\ (H) >$, la séquence de données représentant les achats de ce client. S peut être interprétée par "C a acheté l'item C, puis en même temps les items D et E et enfi n l'item H".*

Définition 2 Soit $s_1 =< a_1 a_2 \dots a_n >$ et $s_2 =< b_1 b_2 \dots b_m >$ deux séquences de données. s_1 est *incluse* dans s_2 ($s_1 \prec s_2$) si et seulement si il existe $i_1 < i_2 < \dots < i_n$ des entiers tels que $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots a_n \subseteq b_{i_n}$.

Exemple 2 *La séquence $s1 =< (C)\ (D\ E)\ (H) >$ est incluse dans la séquence $s2 =< (G)\ (C\ H)\ (I)\ (D\ E\ F)(H) >$ (i.e. $s1 \prec s2$) car $(C) \subseteq (C\ H)$, $(D\ E) \subseteq (D\ E\ F)$ et $(H) \subseteq (H)$. En revanche $< (C)\ (E) > \not< (C\ E) >$ (et vice versa).*

Définition 3 Un client *supporte* une séquence s (fait partie du support pour s) si s est incluse dans la séquence de données de ce client. Le *support* d'une séquence s est calculé comme étant le pourcentage des clients qui supportent s. Soit $minSupp$ le support minimum fixé par l'utilisateur. Une séquence qui vérifie le support minimum (i.e. dont le support est supérieur à $minSupp$) est une *séquence fréquente*.

1. Nous considérons ici le terme de transaction dans le sens d'une transaction fi nancière et non pas dans celui d'une transaction dans une base de données.

Remarque 1 *Une séquence de données n'est prise en compte qu'une seule fois pour calculer le support d'une séquence fréquente, i.e. elle peut présenter plusieurs fois le même comportement, le processus de recherche de séquences considère qu'elle produit ce comportement sans tenir compte du nombre de ses apparitions dans la séquence de données.*

Définition 4 Soit une base de données DB, l'ensemble L^{DB} des *séquences fréquentes maximales* (également notées *motifs séquentiels*) est constitué de toutes les séquences fréquentes telles que pour chaque s dans L^{DB}, s n'est incluse dans aucune autre séquence de L^{DB}. Le problème de la recherche de séquences maximales ($sequential\ patterns$ dans [AGR 95b]) consiste à trouver l'ensemble L^{DB}.

Les deux propriétés suivantes considèrent le cas des sous-ensembles par rapport aux calculs du support et de l'inclusion.

propriété 1 *Soit s_1 et s_2 deux séquences. Si $s_1 \prec s_2$ alors $support(s_1) \geq support(s_2)$.*

propriété 2 *Soit s_1 une séquence non fréquente. Quelle que soit s_2 telle que $s1 \prec s_2$, s_2 est une séquence non fréquente.*

La propriété 1 se justifie par le fait que toute séquence de données d dans DB supportant s_2 supporte obligatoirement s_1 (l'inverse ne se vérifie pas). La propriété 2 est une conséquence de la propriété 1. En effet, d'après cette propriété, $support(B) \leq support(A) < minSupp$, donc B n'est pas fréquent.

Exemple 3 *Considérons la base de données DB illustrée par la figure 1. Avec un support minimum de 50% (i.e. pour qu'une séquence soit retenue, il faut que deux clients dans la base de données supportent cette séquence), les séquences fréquentes maximales sont alors les suivantes : $< (A) >$, $< (F) >$, $< (B)\ (I) >$, $< (C)\ (I) >$ et $< (C)\ (D, G) >$. La première fait partie des achats de C_2 et C_3, alors que la dernière apparaît dans les séquences de données des clients C_2 et C_4.*

3. Méthodes basées sur Apriori (*breadth-first*)

La méthode GSP (*Generalized Sequential Patterns*) [SRI 96] a été l'une des premières propositions pour résoudre la problématique des motifs séquentiels (ce travail fait suite à [AGR 95b]). Les auteurs, en définissant la problématique de l'extraction de motifs séquentiels, ont également proposé un algorithme reprenant les principes d'Apriori, conçu pour l'extraction de règles d'association. Cependant, les difficultés relatives à la prise en compte de la temporalité ont rapidement conduit à la mise en

Client	Date	Items
C_1	01/01/2004	B,F
C_1	02/02/2004	B
C_1	04/02/2004	C
C_1	18/02/2004	H,I
C_2	11/01/2004	A
C_2	12/01/2004	C
C_2	29/01/2004	D,F,G
C_3	05/01/2004	C,E,G
C_3	12/02/2004	A,B
C_4	06/02/2004	B,C
C_4	07/02/2004	D,G
C_4	08/02/2004	I

Figure 1. *Base de données exemple*

place d'une méthode de génération de candidats adaptée à ce contexte. Celle-ci maintient cependant les principes d'une recherche "en largeur d'abord" puisque les candidats sont générés en fonction de leur longueur et non de leur préfixe. Avec le même principe d'exploration de l'espace de recherche, l'auteur de SPADE [ZAK 01] a proposé une méthode qui élague l'espace de recherche en regroupant les candidats par catégorie. Nous présentons dans cette section les algorithmes GSP et SPADE, ainsi que les structures de données mises en jeu.

3.1. *Algorithme pionnier :* GSP *et sa structure*

Dans [AGR 95a] nous trouvons un résumé des techniques mises en œuvre depuis le début du projet Quest d'IBM. Ce projet est à l'origine de l'algorithme GSP [SRI 96], extension de Apriori, lui-même destiné à reprendre l'algorithme AIS présenté dans [AGR 93].

GSP est un algorithme basé sur la méthode générer-élaguer mise en place depuis Apriori et destinée à effectuer un nombre de passes raisonnable sur la base de données. La technique généralement utilisée par les algorithmes de recherche de séquences est basée sur une création de candidats, suivie du test de ces candidats pour confirmer leur fréquence dans la base. Bénéficiant de propriétés relatives aux séquences et à leur fréquence d'apparition, ces techniques sont tout de même contraintes "d'essayer" des séquences avant de les déterminer fréquentes (ou non).

GSP utilise deux fonctions :

– COMPTERSUPPORT. Cette fonction est destinée à incrémenter le support des candidats contenus dans C_k à partir de la séquence de données d.

– GENERERCANDIDATS. Après avoir éliminé les candidats non fréquents issus de COMPTERSUPPORT (en fonction de $minSup$, cette fonction a pour but de créer tous

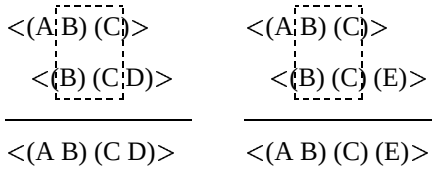

Figure 2. *2 exemples de jointure entre candidats dans* GSP

les k-candidats (candidats de longueur k) susceptibles d'être fréquents (donc tous les k-candidats susceptibles de devenir des k-fréquents, ou fréquents de longueur k) à partir d'un ensemble de (k-1)-fréquents.

Le bénéfice obtenu est le nombre de passes effectuées sur la base de données à la fin du processus qui est exactement égal à la taille du plus grand fréquent. Pour générer les candidats, l'algorithme GSP procède de la façon suivante : soit k la longueur des candidats à générer, plusieurs cas peuvent alors se présenter.

– $k = 1$. Pour générer les fréquents de taille 1, GSP énumère tous les items de la base et détermine en une passe lesquels ont une fréquence supérieure au support minimum.

– $k = 2$. A partir des items fréquents, GSP génère les candidats de taille 2 de la façon suivante : pour tout couple $x,\ y$ dans l'ensemble des 1-séquences fréquentes (les items fréquents), alors si $x = y$ le 2-candidat $< (x)\ (y) >$ est généré et si $x \neq y$, alors les 2-candidats $< (x)\ (y) >$ et $< (x\ y) >$ sont générés.

– $k > 2$. C_k est obtenu par auto-jointure sur L_{k-1}. La relation *jointure*($s_1,\ s_2$) s'établit si la sous-séquence obtenue en supprimant le premier élément (plus petit item du premier itemset) de s_1 est la même que la sous-séquence obtenue en supprimant le dernier élément (plus grand item du dernier itemset) de s_2. La séquence candidate obtenue par *jointure*($s_1,\ s_2$) est la séquence s_1 étendue avec le dernier item de s_2. L'item ajouté fait partie du dernier itemset s'il était dans un itemset de taille supérieure à 1 dans s_2 et devient un nouvel itemset s'il était dans un itemset de taille 1 dans s_2 (C.f. figure 2).

Le théorème suivant (dont la preuve est donnée dans [SRI 96]) garantit que, pour toutes les longueurs de candidats générés, si un candidat est susceptible d'être fréquent alors il sera généré.

Théorème 1 *Soit L_{k-1}, l'ensemble de (k-1)-fréquents. La génération des candidats construit un sur-ensemble de L_k, l'ensemble des k-fréquents.*

Pour évaluer le support de chaque candidat en fonction d'une séquence de données, GSP utilise une structure d'arbre de hachage destinée à organiser les candidats. Les candidats sont stockés en fonction de leur préfixe. Pour ajouter un candidat dans l'arbre des séquences candidates, GSP parcourt ce candidat et effectue la descente

Client	Itemset	Items
1	10	A B
	20	B
	30	A B
2	20	A C
	30	A B C
	50	B

Client	Itemset	Items
3	10	A
	30	B
	40	A
4	30	A B
	40	A
	50	B

Figure 3. *$DBspade$, base de données exemple pour* SPADE

correspondante dans l'arbre. Pour trouver quelles séquences candidates sont incluses dans une séquence de données, GSP parcourt l'arbre en appliquant une fonction de hachage sur chaque item de la séquence de données. Quand une feuille est atteinte, elle contient des candidats potentiels pour la séquence de données.

Exemple 4 *On peut trouver un exemple de structure de hachage dans la figure 5 (arbre de gauche). Nous remarquons que la feuille qui porte l'étiquette C est utilisée pour stocker les séquences candidates* $< (A)\ (C)\ (B\ D) >$ *et* $< (A\ C)\ (D)\ (B) >$. *Quand* GSP *atteint cette feuille, il n'a aucun moyen de savoir si c'est la sous-séquence* $< (A)\ (C) >$ *ou bien* $< (A\ C) >$ *qui l'a conduit jusqu'à cette feuille. C'est pour cette raison que* GSP *doit tester les séquences candidates contenues dans les feuilles atteintes afin de savoir quels supports incrémenter.*

3.2. *Gestion des données en mémoire :* SPADE

SPADE, présenté dans [ZAK 01], se classe dans la catégorie des algorithmes qui cherchent à réduire l'espace de recherche en regroupant les motifs séquentiels par catégorie. Pour SPADE, les motifs fréquents présentent des préfixes communs, qui permettent de décomposer le problème en sous-problèmes qui seront traités en mémoire.

Le calcul de F_2 (les fréquents de taille 2) par SPADE, passe par une inversion de la base, qui la transforme d'un format vertical vers un format horizontal. Les auteurs considèrent que cette opération peut être simplifiée si la base peut-être chargée en mémoire vive. SPADE gère les candidats et les séquences fréquentes à l'aide de classes d'équivalence comme suit :
Deux k-séquences appartiennent à la même classe si elles présentent un préfixe commun de taille $(k-1)$. Plus formellement, soit $\mathcal{P}_{k-1}(\alpha)$ la séquence de taille k-1 qui préfixe la séquence α. Comme α est fréquente, $\mathcal{P}_{k-1}(\alpha) \in F_{k-1}$, avec F_{k-1} les fréquents de taille $k-1$. Une classe d'équivalence est définie de la manière suivante :

$$[\rho \in F_{k-1}] = \{\alpha \in F_k | \mathcal{P}_{k-1}(\alpha) = \rho\}$$

Chacune de ces classes d'équivalence contient alors deux types d'éléments : $[\rho.l_1] =< \rho\,(x) >$ ou bien $[\rho.l_2] =< \rho x >$, selon que l'item x appartient ou pas à la même transaction que le dernier item de ρ.

Les candidats seront ensuite générés selon trois méthodes :

– Auto-jointure ($[\rho.l_1] \times [\rho.l_1]$).
– Auto-jointure ($[\rho.l_2] \times [\rho.l_2]$).
– Jointure ($[\rho.l_1] \times [\rho.l_2]$).

Le reste de l'algorithme, à savoir le comptage du support pour les candidats générés, repose sur la re-écriture préalable de la base de données. En effet la transformation consiste à associer à chaque k-séquence l'ensemble des couples (client, itemset) qui lui correspondent dans la base. L'exemple 5 illustre le résultat de cette transformation, et la façon dont la table obtenue est utilisée lors du calcul du support.

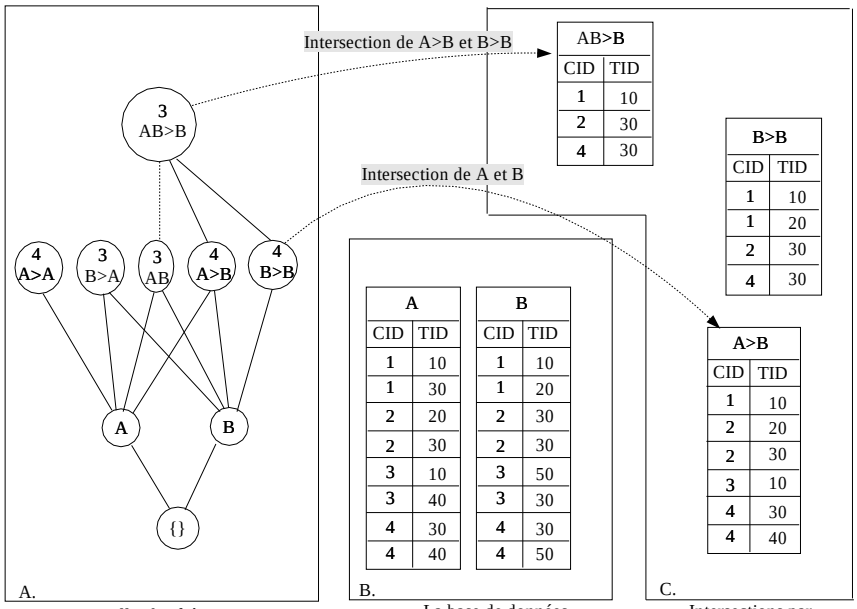

Figure 4. *Intersections de listes d'itemsets dans* SPADE*, avec la base de données de la figure 3*

Exemple 5 *La figure 3 représente une base de données, représentée selon le format vertical classique. Après transformation selon les besoins de l'algorithme* SPADE*, la*

base de données DB_{spade} est alors décrite dans le cadre "B." de la fi gure 4. Une fois la base de données ainsi transformée, l'algorithme peut alors accéder aux supports des candidats de taille 2, grâce aux listes d'itemsets et clients supportant les items fréquents, en procédant à une intersection de ces listes. Considérons l'enchaînement "A>B" qui signifi e "A est suivi par B" ou encore < (A) (B) >. En utilisant un algorithme d'intersection des listes de A et de B, SPADE *peut déduire la liste d'itemsets de < (A) (B) >.*

La façon dont SPADE gère les intersections est détaillée dans [ZAK 01]. Dans [LEL 04], l'auteur propose une extension de SPADE pour des données contenant des répétitions successives.

4. Méthodes basées sur le principe *depth-fi rst*

La prise en compte de la temporalité dans les transactions a conduit de nombreux auteurs à privilégier des méthodes de recherche dites "en profondeur d'abord" pour extraire les motifs séquentiels. C'est le cas de PSP [MAS 98], qui met en place et exploite un arbre de préfixes pour gérer les candidats. C'est aussi le principe adopté par [HAN 00] avec FREESPAN et amélioré par [PEI 01] avec l'algorithme PREFIXSPAN. PREFIXSPAN implémente de plus un principe de re-écriture de la base de données en fonction des préfixes des motifs séquentiels fréquents découverts (ou d'une indexation en fonction de la mémoire disponible). Ce principe a été repris dans de nombreuses contributions (comme [YAN 03] par exemple). Nous présentons dans cette section une sélection de méthodes basées sur ce principe de la recherche "en profondeur d'abord".

4.1. PSP

Les auteurs de [MAS 98] estiment que l'arbre de hachage utilisé dans [AGR 95b, SRI 96] présente un défaut qu'il est facile de constater. En effet lors de la recherche des feuilles susceptibles de contenir des candidats inclus dans la séquence analysée, la structure utilisée ne tient pas compte des changements de date entre les items de la séquence qui servent à la navigation. Par exemple, avec la séquence $<$ (A C) (B D) $>$, l'algorithme va atteindre la feuille du sommet C (fils de A), alors que cette feuille peut contenir deux types de candidats :

– ceux qui commencent par $<$ (A) (C)... d'un côté

– et ceux qui commencent par $<$ (A C) ... de l'autre.

Le but est alors de mettre en place une structure d'arbre de préfixes, pour gérer les candidats. L'algorithme PSP (*Prefi x Tree for Sequential Pattern*) [MAS 98], destiné à exploiter cette structure, est basé sur la méthode générer-élaguer. Le principe de base de cette structure consiste à factoriser les séquences candidates en fonction de leur préfixe. Cette factorisation, inspirée de celle mise en place dans [AGR 95b],

pousse un peu plus loin l'exploitation des préfixes communs que présentent les candidats. En effet les auteurs proposent de prendre en compte les changements d'itemsets dans cette factorisation. L'arbre de préfixes ainsi proposé ne stocke plus les candidats dans les feuilles, mais permet de retrouver les candidats de la façon suivante : tout chemin de la racine à une feuille représente un candidat et tout candidat est représenté par un chemin de la racine à une feuille. De plus, pour prendre en compte le changement d'itemset, l'arbre est doté de deux types de branches. Le premier type, entre deux items, signifie que les items sont dans le même itemset alors que le second signifie qu'il y a un changement d'itemset entre ces deux items. L'exemple 6 donne une illustration de la structure d'arbre de préfixes.

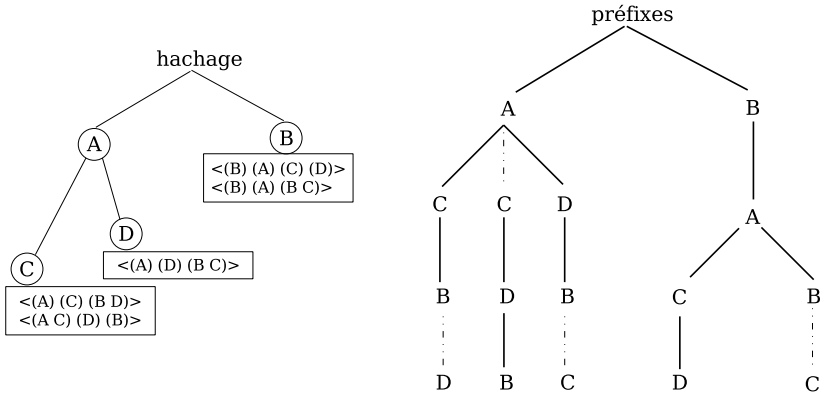

Figure 5. *Comparaison entre la structure de hachage et l'arbre de préfixes*

Exemple 6 *La figure 5 illustre la façon dont les candidats sont stockés dans l'arbre de préfixes. La figure propose une comparaison du stockage des candidats selon que l'on utilise la structure de hachage ou l'arbre de préfixes. Dans ce dernier, nous pouvons distinguer les items appartenant à un même itemset (branche en pointillés) ou bien les changements d'itemsets (branche en trait plein).*

La structure préfixée présente alors deux avantages :

– Une complexité en mémoire plus faible. En effet, que le nombre d'éléments à utiliser pour représenter l'arbre est inférieur au nombre d'éléments que présentent les candidats au total. Le nombre d'éléments nécessaires pour représenter l'arbre est de l'ordre du nombre d'éléments que présentent les candidats moins les éléments appartenant à leur préfixes communs.

– Une plus grande rapidité de détection des inclusions. Intuitivement, il suffit d'observer la figure 5 et de simuler la détection des candidats inclus dans la séquence de données $<$ (B) (D) (B C) $>$ pour s'en rendre compte. Une structure de hachage aurait stocké les candidats qui commencent par B dans une même feuille. Cette feuille

atteinte il aurait fallu, à nouveau, lancer une comparaison entre les séquences de cette feuille et la séquence de données qui y a conduit. Dans l'arbre de préfixes, l'algorithme est guidé par la séquence jusqu'à ce que le parcours échoue, signe de non inclusion des séquences représentées par les feuilles qui sont descendantes du sommet de cet échec. Si le parcours réussit alors les séquences, représentées par les chemins de la racine aux feuilles sur lesquelles l'algorithme aboutit, sont des candidats inclus dans la séquence de données testée.

4.2. PREFIXSPAN

Dans [HAN 00], les auteurs proposent l'algorithme FREESPAN (*Frequent pattern-projected Sequential pattern mining*). L'idée générale est de proposer des projections récursives de la base de données en fonction des items fréquents. La base est alors projetée en plusieurs bases plus petites et les séquences fréquentes grandissent avec le nombre de projections. Les temps de réponses sont alors améliorés car chaque base projetée est plus petite et facile à traiter. Ce travail est le point de départ d'autres études sur la projection de bases de données en recherche de motifs séquentiels. FREESPAN présente tout de même un défaut selon ses auteurs : une sous-séquence peut être générée par n'importe quelle combinaison dans une séquence, donc FREESPAN doit conserver la totalité de la séquence dans la base d'origine sans réduire sa taille.

La méthode PREFIXSPAN, présentée dans [PEI 01], se base sur une étude du nombre de candidats qu'un algorithme de recherche de motifs séquentiels peut avoir à produire afin de déterminer les séquences fréquentes. En effet, selon les auteurs, pour envisager d'utiliser un algorithme comme GSP il faut s'attendre à devoir générer, uniquement pour la seconde passe, pas moins de $n^2 + \frac{n \times (n-1)}{2}$ candidats de taille 2 à partir des n items trouvés fréquents lors de la première passe. L'objectif des auteurs est alors de réduire le nombre de candidats générés. Pour parvenir à cet objectif, PREFIXSPAN propose (à l'instar de PSP avec les candidats) d'analyser les préfixes communs que présentent les séquences de données de la base à traiter. À partir de cette analyse, l'algorithme construit des bases de données intermédiaires qui sont des projections de la base d'origine déduites à partir des préfixes identifiés. Ensuite, dans chaque base obtenue, PREFIXSPAN cherche à faire croître la taille des motifs séquentiels découverts en appliquant la même méthode de manière récursive.
Deux sortes de projections sont alors mises en place pour réaliser cette méthode : la projection dite "niveau par niveau" et la "bi-projection". Au final, les auteurs proposent une méthode d'indexation permettant de considérer plusieurs bases virtuelles à partir d'une seule, dans le cas où les bases générées ne pourraient être maintenues en mémoire en raison de leurs tailles.

PREFIXSPAN fonctionne également avec une écriture de la base différente de celle utilisée par GSP. En effet cet algorithme requiert un format qui présente sur une ligne le numéro de client suivi de toutes ses transactions sous forme de séquence de données. Ce format nécessite une re-écriture de la base de données avant de procéder à

Client	Séquence
10	< (a) (a b c) (a c) (d) (c f) >
20	< (a d) (c) (b c) (a e) >
30	< (e f) (a b) (d f) (c) (b) >
40	< (e) (g) (a f) (c) (b) (c) >

Figure 6. *$DBspan$, base de données exemple pour* PREFIXSPAN

Préfixe	base projetée (suffixes)	motifs séquentiels
<a>	<(abc)(ac)(d)(cf)>, (_d)(c)(bc)(ae)>, <(_b)(df)(c)(b)>, <(_f)(c)(b)(c)>	<a>, <(a)(a)>, <(a)(b)>, <(a)(bc)>, <(a)(bc)(a)>, <(a)(b)(a)>, <(a)(b)(c)>, <(ab)>, <(ab)(c)>, <(ab)(d)>, <(ab)(f)>, <(ab)(d)(c)>, <(a)(c)(a)>, <(a)(c)(b)>, <(a)(c)(c)>, <(a)(d)>, <(a)(d)(c)>, <(a)(f)>
<b>	<(_c)(ac)(d)(cf)>, <(_c)(ae)>, <(df)(c)(b)>, <c>	<b>, <(b)(a)>, <(b)(c)>, <(bc)>, <(bc)(a)>, <(b)(d)>, <(b)(d)(c)>, <(b)(f)>
⋮	⋮	⋮

Figure 7. *Résultat de* PREFIXSPAN *sur la base de données de la fi gure 6*

l'étape de fouille de données. Nous illustrons, à présent, un exemple de fouille avec PREFIXSPAN sur la base de données $DBspan$ de la figure 6.

La méthode de projection préfixée va permettre de procéder à une extraction des motifs séquentiels avec un support minimum de deux clients, en appliquant les étapes suivantes :
Étape 1 : Trouver les items fréquents. Pour cela, une passe sur la base de données va permettre de collecter le nombre de séquences supportant chaque item rencontré et donc d'évaluer le support des items de la base. Les items trouvés sont (sous la forme <item> :support) : <a> :4 ,<b> :4 ,<c> :4 ,<d> :3 ,<e> :3 ,<f> :3 .
Étape 2 : Diviser l'espace de recherche. L'espace de recherche complet peut être divisé en six sous-ensembles, puisqu'il y a six préfixes de taille 1 dans la base (i.e. les six items fréquents). Ces sous-ensembles seront : (1) les motifs séquentiels ayant pour préfixe <a>, (2) ceux ayant pour préfixe <b>, ... et (6) ceux ayant pour préfixe <f>.
Étape 3 : Trouver les sous-ensembles de motifs séquentiels. Les sous-ensembles de motifs séquentiels peuvent être trouvés en construisant les projections préfixées des bases obtenues et en réappliquant l'algorithme de fouille de manière récursive. Les bases ainsi projetées et les motifs obtenus sont alors en partie donnés à la figure 7.

Le processus de découverte des motifs séquentiels fréquents, sur les bases projetées, se déroule alors de la manière suivante :
Tout d'abord, prefixSpan cherche les sous-séquences des séquences de données ayant

pour préfixe <a>. Seules les séquences contenant <a> sont à prendre en compte. De plus dans chacune de ces séquences, seul le suffixe doit être considéré. Par exemple avec la séquence < (e f) (a b) (d f) (c) (b) >, seule la sous-séquence (le suffixe) < (_b) (d f) (c) (b) > sera pris en compte (dans cette séquence le caractère "_" signifie que le préfixe était contenu dans le même itemset que "b").

Les séquences de $DBspan$ qui contiennent <a> sont alors projetées pour former $DBspan_{<a>}$, qui contient quatre suffixes : <(abc)(ac)(d)(cf)> <(_d)(c)(bc)(ae)> <(_b)(df)(c)(b)> et <(_f)(c)(b)(c)>. Une passe sur $DBspan_{<a>}$ permet alors d'obtenir les motifs séquentiels de longueur 2 ayant <a> pour préfixe commun : <(a)(a)> :2, <(a)(b)> :4, <(ab)> :2, <(a)(c)> :4, <(a)(d)> :2 et <(a)(f)> :2.
De façon récursive, toutes les séquences ayant pour préfixe <a> peuvent être partitionnées en 6 sous-ensembles : (1) celles qui ont pour préfixe <(a)(a)>, (2) celles qui ont pour préfixe <(a)(b)>... et (6) celles qui ont pour préfixe <(a)(f)>. Ces motifs peuvent alors former de nouvelles bases projetées, et chacune de ces bases peut être utilisée en entrée de l'algorithme, toujours de manière récursive.

4.3. *Exploiter les bitmaps :* SPAM

Assez proche de SPADE, mais avec une méthode de représentation différente, l'algorithme SPAM [AYR 02] gère la présence ou l'absence d'un item dans une séquence par l'intermédiaire de vecteurs de bits. Ainsi, un arbre de représentation est créé respectivement par *S-extension* (ajout d'un item dans une autre transaction) ou par *I-extension* (ajout d'un item dans la même transaction) et les résultats sont stockés sous la forme de vecteurs de bits. La vérification des candidats est immédiate car il suffit de compter les bits positionnés à 1 dans la structure et de les comparer avec le support minimum.

4.4. *Recherche des motifs séquentiels fermés*

L'extraction de motifs séquentiels devient problématique selon la longueur des motifs séquentiels extraits. Les auteurs de [YAN 03] illustre ce problème avec l'exemple d'une base de données ne contenant qu'un seul motif : $< (a_1)(a_2)...(a_{100}) >$. Dans ce cas, il faudra générer $2^{100} - 1$ sous-séquences fréquentes avec un support minimum de 1. Ces sous-séquences seront redondantes car elles auront toutes le même support que $< (a_1)(a_2)...(a_{100}) >$. Dans [YAN 03], les auteurs définissent donc la problématique de la recherche des motifs séquentiels fermés (*closed sequential patterns*), inspirée de la recherche d'itemsets fermés. Ils proposent CLOSPAN, le premier algorithme capable de résoudre ce problème et optimisé pour cela. Dans [WAN 04], l'approche BIDE utilise une nouvelle manière d'étendre les séquences et optimise l'espace de recherche en analysant à l'avance les motifs à étendre.

Client	Séquence
10	< (a f) (d) (e) (a) >
20	< (e) (a) (b) >
30	< (e) (a b f) (b d e) >

Figure 8. *$DBClose$, base de données exemple pour* CLOSPAN

Les motifs séquentiels fermés peuvent être définis de la façon suivante :

Définition 5 *Soit $minSup$, le support minimum et FS l'ensemble des motifs séquentiels fréquents correspondants. L'ensemble des motifs séquentiels fermés CS est défi ni comme : $CS = \{s/s \in FS \ et \ \nexists s' \ telle \ que \ s' \subseteq s \ et \ support(s') = support(s)\}$.*

4.4.1. CLOSPAN

CLOSPAN [YAN 03] est une méthode basée sur le principe depth-first et implémente l'algorithme PREFIXSPAN. En fait, il s'agit d'une optimisation de ce dernier, destinée à élaguer l'espace de recherche en évitant de parcourir certaines branches dans le processus de divisions récursives (en détectant par avance les motifs séquentiels non fermés). Le principe de CLOSPAN repose sur deux éléments essentiels : l'ordre lexicographique des séquences et la détection de liens systématiques entre deux items (i.e. "β apparaît toujours avant γ dans la base de données"). Considérons la base de données $DBClose$ de la figure 8. A l'aide de cette base, nous illustrons les principales optimisations mises en place par CLOSPAN :

Exemple 7 *Puisque $a < f$ on scanne la base projetée de a avant celle de f. De plus on peut détecter que a apparaît toujours avant f dans la base de données $DBClose$. L'idée de* CLOSPAN *consiste à dire que toute séquence de la forme $< (f) + \alpha >$, avec α une séquence ajoutée en tant que I-extension ou S-extension (C.f.* SPAM *Section 4.3), aura le même support que $< (af) + \alpha >$. Il n'est donc pas nécessaire de scanner la base projetée de f et l'on ne garde que les résultats issus de la base projetée de a.*

Exemple 8 *Voici le deuxième cas possible : puisque $b < e$ on scanne la base projetée de b avant celle de e. De plus on peut détecter que e apparaît toujours avant b dans la base de données $DBClose$. Dans ce cas, toute séquence de la forme $< (b) + \alpha >$ aura le même support que $< (e)(b) + \alpha >$. Il n'est donc pas nécessaire de scanner la base projetée de e, mais dans ce cas, on peut ajouter (e) devant tous les résultats issus du scan sur la base projetée de b et garder les séquences de la forme $< (e)(b) + \alpha >$ comme étant le résultat fermé.*

La principale difficulté de cette méthode réside dans la détection efficace des relations du type "β apparaît toujours avant γ dans D" avec D une base de données projetée. Soit $\mathcal{I}(D)$ le nombre total d'items dans une base de données D (i.e.

$\mathcal{I}(D)$=$\sum_{i=1}^{n} l(s_i)$, par exemple $\mathcal{I}(DBClose)$=15). Pour implémenter leur principe de manière efficace les auteurs de CLOSPAN s'appuient sur le théorème suivant :

Théorème 2 *soient s et s' deux séquences telles que $s \subseteq s'$. Soient D_s et D'_s les bases de données projetées selon les préfixes s et s'. Alors on a :*

$$D_s = D'_s \Leftrightarrow \mathcal{I}(D_s) = \mathcal{I}(D'_s).$$

A partir de ce théorème, les auteurs proposent ensuite deux conditions d'arrêt précoce pour l'algorithme d'extraction, selon les relations entre s et s'.

Condition 1 (*backward sub-pattern*) : $s < s'$, $s \supseteq s'$ et $\mathcal{I}(D_s) = \mathcal{I}(D'_s)$ alors on peut éviter d'explorer les descendants de s'. Reprenons l'exemple 7. Nous avons $(f) < (a)(f)$ et $(a)(f) \supseteq (f)$. De plus $\mathcal{I}(D_{(a)(f)}) = \mathcal{I}(D_{(f)}) = 6$ et $D_{(a)(f)} = D_{(f)} = \{< (d)(e)(a) >, < (b)(d)(c) >\}$. Nous pouvons en déduire que l'ensemble des descendants de (f) est équivalent à l'ensemble des descendants de $(a)(f)$ et donc que tout motif séquentiel qui commence par (f) sera absorbé par un motif séquentiel qui commence par $(a)(f)$.

Condition 2 (*backward super-pattern*) : $s < s'$, $s \subseteq s'$ et $\mathcal{I}(D_s) = \mathcal{I}(D'_s)$ alors on peut éviter d'explorer les descendants de s'. Reprenons l'exemple 8. Nous avons $D_{(b)} = D_{(e)(b)}$. De plus on a déjà exploré $D_{(b)}$ et on sait que l'ensemble des descendants de $(e)(b)$ est équivalent à l'ensemble des descendants de (b). Il est donc inutile d'explorer $D_{(e)(b)}$ et $(e)(b)$ peut absorber (b).

Des optimisations sur les structures de données, en particulier une modification de l'arbre de préfixes contenant les séquences afin de le représenter sous forme de treillis sont également proposées dans [YAN 03]. Ce treillis contient les raccourcis découverts par CLOSPAN en fonction des conditions d'arrêt précoce.

4.4.2. BIDE

Etant donné que CLOSPAN conserve l'historique des séquences candidates, il ne s'avère pas efficace dans le cas de bases contenant de trop nombreuses séquences fermées. Pour pallier ce problème, une nouvelle approche, BIDE (*BI-Directional Extension*) est proposée dans [WAN 04]. L'idée générale est d'étendre les séquences dans les deux directions, i.e. en avant (*forward extension*) et en arrière (*backward extension*). En effet, considérons une séquence $S = i_1 i_2 ... i_n$, celle-ci peut être étendue de trois manières possibles : ajout d'un item après i_n, ajout d'un item entre $i_1 i_2 ... i_n$, ajout d'un item avant i_1. La première correspond à une extension en avant et les deux dernières à une extension en arrière. Ainsi, les auteurs montrent que pour une séquence S, s'il n'y a pas d'extension avant ni d'extension arrière alors S est une séquence fermée. Comme dans CLOSPAN, une base projetée est constituée. Pour une séquence S, son ensemble d'items extensibles en avant, i.e. les items qui peuvent être ajoutés à la fin de S, est constitué par les items locaux dont le support est égal à celui de la séquence. Ces items locaux sont simplement trouvés en parcourant la base projetée pour ce préfixe et en comptant le nombre d'items. Pour effectuer rapidement cette opération, la projection utilisée est une pseudo projection comme dans [PEI 02]. De manière à définir

les extensions possibles en arrière, il faut dans un premier temps rechercher, pour les items d'une séquence, quelles sont les extensions en arrière possibles. Pour cela, il est nécessaire de remonter dans la séquence pour examiner avec quel item il est possible de l'étendre. Nous ne détaillons pas ici comment récupérer ces extensions (le lecteur intéressé peut se reporter à [WAN 04]), toutefois nous en illustrons le principe avec l'exemple 9

Exemple 9 *Considérons la séquence $S = CA_1A_2BC_2DA_3C_3E$ et la séquence $S_p = ABC$. Nous pouvons obtenir les séquences résultantes de la première apparition de S_p dans S et de sa dernière apparition. Elles correspondent respectivement à $FI = CA_1A_2BC_2$ et $LI = CA_1A_2BC_2DA_3C_3$. Nous pouvons également obtenir pour chaque item de S_p sa dernière apparition dans LI qui respecte l'ordre de la séquence S_p. Nous avons alors : $LL_1 = A_2$, $LL_2 = B$ et $LL_3 = C_3$. À partir du moment où nous avons obtenu ces ensembles, l'idée consiste à rechercher quelles sont les périodes maximales (une période correspond à une sous-séquence de S) dans la séquence S dans lesquelles les extensions possibles peuvent se trouver. Ainsi, si nous considérons $LL_1 = A_2$, la période maximale dans laquelle on peut rechercher des extensions arrière se situe dans la partie de S avant LL_1, i.e. la sous séquence CA_1. Pour $LL_2 = B$, les extensions possibles se situent dans la période de la séquence A_2. Enfi n, pour LL_3, la 3ième période maximale est C_2DA_3. En d'autres termes, la période correspond à la fenêtre maximale pour laquelle on a besoin de remonter en arrière pour un item. Par exemple, nous voyons bien que pour LL_3, une extension ne peut avoir lieu qu'entre sa position et l'item qui le précède, i.e. LL_2, mais dans la séquence S.*

À partir du moment où il est possible de trouver les périodes maximales de recherche pour un item, les auteurs ont montré que si un item i' apparaissait dans chacune des ièmes périodes maximales d'une séquence S, il était possible d'obtenir une nouvelle séquence plus englobante, i.e. qui absorbe S, et que cet item i' est donc une extension arrière. De manière à améliorer les performances et à diminuer l'espace de recherche, les auteurs proposent également de repérer à l'avance quelles sont les séquences qu'il convient de ne pas étendre. L'idée consiste à affiner les périodes maximales pour un item en semi périodes maximales en tenant compte de la notion d'ordre dans la séquence et du fait qu'un item peut être répété plusieurs fois dans une séquence. Ainsi, si un item apparaît dans toutes les semi périodes les auteurs montrent qu'il n'est plus nécessaire de poursuivre l'extension en arrière.

Exemple 10 *A partir de l'exemple précédent, il est possible de rechercher pour chacun des items de S_p sa dernière apparition dans la première apparition de S_p dans S. Nous avons alors : $LF_1 = A_2$, $LF_2 = B$ et $LF_3 = C_2$. Nous pouvons ainsi obtenir pour chacun des items des semi périodes maximales qui sont donc déterminés par rapport à la première apparition de S_p. Dans notre cadre, nous avons : $SMP_1 = C_1A_1$, $SMP_2 = A_2$ et $SMP_3 = \emptyset$.*

5. Recherche incrémentale de motifs séquentiels

Nous proposons dans cette section d'étudier comment tenir compte des connaissances extraites lors d'une phase précédente de manière à optimiser la fouille lorsque des modifications (ajout de clients, ajout d'items, ...) interviennent sur les données sources. Deux grandes tendances existent à l'heure actuelle pour répondre à cette démarche incrémentale. La première approche consiste à conserver des connaissances supplémentaires lors de la phase d'extraction. Nous retrouvons ainsi des algorithmes comme ISM [PAR 99] ou IUS [ZHE 02] qui sont basés sur la notion de bordure négative introduite par [MAN 94]. D'autres approches comme FASTUP [LIN 98] ou son extension plus récente KISP [LIN 03] gèrent quand à elles une vraie base de connaissances. La seconde tendance cherche quand à elle à minimiser l'espace de stockage des résultats. Elle ne considère ainsi que les séquences fréquentes et examine à partir de celles-ci les nouvelles séquences qui peuvent intervenir lors de l'ajout d'items mais également celles qui ne restent pas fréquentes dans le cas de l'ajout de transactions ou bien de changement de valeur de support. Les algorithmes comme SUFFIXTREE [WAN 96] et ISE [MAS 03] rentrent également dans cette catégorie.

5.1. *Description du problème*

Soient DB une base de données, $minSupp$ le support minimum spécifié par l'utilisateur lors d'une phase de fouille de données et L^{DB} l'ensemble des séquences fréquentes découvertes sur DB vérifiant $minSupp$. Soit db la base de données incrément contenant de nouvelles transactions et/ou de nouveaux clients. Nous considérons que chaque transaction de db est triée par client et par date. Soit $U = DB \cup db$ la base de données résultant de la mise à jour (i.e. intégrant les clients et transactions de db ajoutés à ceux de DB).
Le problème de l'*extraction incrémentale de motifs séquentiels* consiste à découvrir l'ensemble des séquences fréquentes (au sens de la définition 4) sur U, noté L^U, qui respectent le support $minSupp$. De plus l'approche incrémentale doit utiliser les résultats de la phase de fouille de données précédente (i.e. L^{DB}) pour éviter de recalculer une (ou certaines) partie(s) du résultat final.

Considérons la base de données DB de la figure 9. Pour une valeur de support de 50%, nous avons $L^{DB} = \{< (A\ B)\ (C) >, < (A\ B)\ (D) >\}$. Examinons, à présent, l'ajout dans db de nouvelles transactions mais aussi de nouveaux clients. Avec la même valeur de support, comme un nouveau client ($C5$) a été ajouté, pour être considérée comme fréquente, une séquence doit maintenant être supportée par au moins trois clients. Dans cette nouvelle configuration l'ensemble des séquences fréquentes sur DB (L^{DB}) devient : $\{< (A\ B) >\}$ car les séquences $< (A\ B)\ (C) >$ et $< (A\ B)\ (D) >$ ne sont supportées que par les clients $C2$ et $C3$. Cependant la séquence $< (A\ B)\ (E) >$ devient fréquente car elle apparaît dans les séquences de données des clients $C1$, $C2$ et $C3$. En prenant maintenant en considération l'ensemble des mises à jour, l'ensemble des séquences fréquentes dans $U = DB \cup db$ devient

Client	Itemsets		
$C1$	A B	B	E G
$C2$	A B	C	D
$C3$	A B	D	C
$C4$	F	I	
$C5$			

(DB)

Itemsets	
$E\ F\ G$	$H\ J$
$E\ F$	$H\ I$
$A\ D$	$G\ H$

(db)

Figure 9. *Une base de données (DB) et son incrément constitué de nouvelles transactions et de nouveaux clients (db).*

$L^U = \{< (A\ B)\ (E) >, < (A)\ (G) >, < (A)\ (H) >, < (D)\ (H) >, < (F) >\}$. Observons plus attentivement la séquence $< (A\ B)\ (E) >$. Cette séquence pouvait être detectée pour le client $C1$ de la base d'origine DB mais son support n'était pas suffisant pour la considérer comme fréquente. Néanmoins, comme l'item E devient fréquent grâce aux mises à jour, cette séquence est maintenant supportée par les clients $C1$, $C2$ et $C3$. De la même façon, la séquence $< (A)\ (G) >$ devient fréquente dans la mesure où, avec l'incrément, elle est supportée par les clients $C1$, $C2$ et le nouveau client $C5$.

5.2. *A partir des séquences fréquentes*

Dans [WAN 96], les auteurs proposent une solution au problème de l'extraction incrémentale de motifs séquentiels via l'utilisation d'arbres suffixés. L'idée générale est d'acquérir les données au fur et à mesure de la lecture et de les stocker directement dans un arbre suffixé. La structure utilisée permet ainsi de construire tous les fréquents en une seule passe sur la base. Cette méthode s'applique très bien à la problématique de recherche incrémentale en permettant de continuer la construction de l'arbre en "poursuivant" la lecture des nouvelles données. Une fois cette lecture achevée et l'arbre construit, les auteurs obtiennent les nouveaux fréquents. Même si l'efficacité en temps d'exécution d'une telle méthode ne peut qu'être reconnue, il faut toutefois être conscient de la complexité en espace d'une telle méthode. En effet, l'algorithme présenté par [WAN 96] (tout comme ISM présenté plus loin) affiche une complexité en espace dépendante de la taille de la base de données à traiter.

Dans [MAS 03], un nouvel algorithme appelé ISE a été proposé pour minimiser l'espace de stockage des solutions, i.e. ne conserver que les connaissances minimales lors de la phase d'extraction précédente. Le principe général de l'approche est donc de ne conserver que les fréquents obtenus précédemment et de reconsidérer la méthode générer-élaguer dans ce contexte.
La première étape de cet algorithme consiste à rechercher les items fréquents qui appartiennent à db pour construire un ensemble de base, appelé $source$. Ce dernier est

en fait constitué des fréquents de DB auxquels un item fréquent sur db a été ajouté. Si la séquence qui résulte de cette opération est fréquente sur U alors elle est ajoutée à $source$. Une passe sur U permet de déterminer les extensions fréquentes de taille 2 et les séquences appartenant à $source$. À partir des extensions fréquentes de taille j, les extensions candidates de taille $j+1$ sont construites. De plus, à partir des extensions fréquentes de taille j et de l'ensemble source, les candidats à l'incrémental sont construits. Une passe sur U détermine les nouvelles extensions fréquentes ainsi que les nouveaux fréquents découverts de manière incrémentale. Le principe général d'ISE est illustré figure 10.

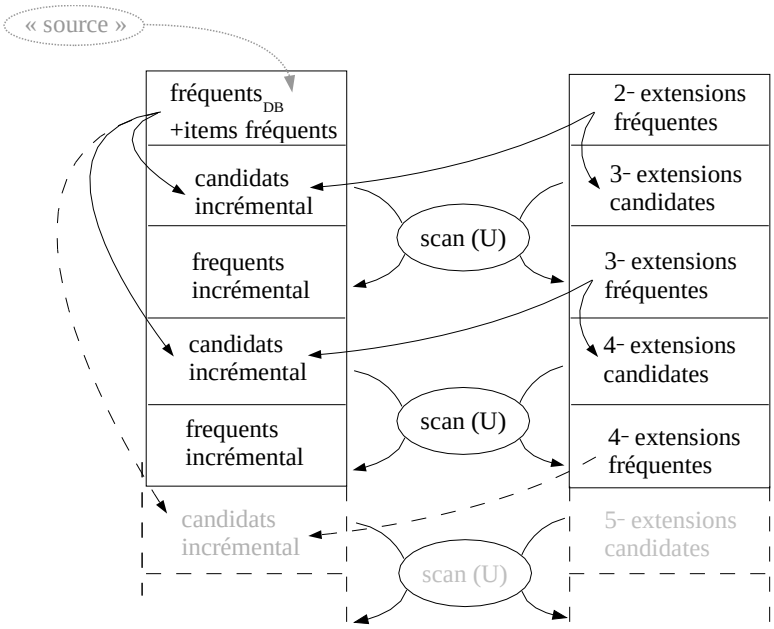

Figure 10. *Principe général d'*ISE

Dans [MAS 03] différentes optimisations sont également proposées pour réduire le nombre de candidats à tester. En outre, l'utilisation d'une approche incrémentale dans le cadre initial de l'extraction est étudiée. La problématique consiste alors à examiner s'il est plus efficace de découper la base initiale en DB+db, d'appliquer GSP sur DB et ISE sur U plutôt que d'utiliser directement GSP sur U. Des expériences menées sur des bases de données non fortement corrélées et pour des séquences de longueurs restreintes, ont montré que les temps de réponse étaient considérablement améliorés avec un tel principe.

Client	Itemset	Items
1	10	A B
	20	B
	30	A B
	100	**A C**
2	20	A C
	30	A B C
	50	B

Client	Itemset	Items
3	10	A
	30	B
	40	A
	110	**C**
	120	**B**
4	30	A B
	40	A
	50	B
	140	**C**

Figure 11. *$DBspade$, après la mise à jour*

5.3. *Utiliser la bordure négative*

Certaines approches considérent qu'il est possible d'obtenir, lors d'une extraction initiale, d'autres connaissances que la liste des séquences fréquentes. C'est dans ce cadre que se situent les travaux basés sur la bordure négative. Dans cette section, nous présentons principalement l'algorithme ISM [PAR 99] qui est une extension de SPADE (C.f. Section 3.2) et qui prend en compte la mise à jour à l'aide d'une bordure négative et d'une re-écriture de la base.

La figure 11 présente la base de données $DBspade$ de la figure 3 après une mise à jour. La première exécution de SPADE sur $DBspade$ est décrite dans le rectangle "A" du treillis de la figure 4. Lors de la construction de ce treillis, SPADE a proposé des candidats et les a inséré dans le treillis, à chaque étape de l'algorithme, soit pour chaque taille j (avec $1 \leq j \leq k$ et j la taille maximale définie par l'utilisateur des fréquents obtenus). L'idée d'ISM consiste à conserver dans ce treillis la bordure négative (NB) qui est composée des j-candidats les plus bas de la hiérarchie d'inclusion qui n'ont pas été retenus. En d'autres termes, soit s une séquence appartenant à NB, alors $\nexists s'/s'$ est fils de s et $s' \in NB$, et plus précisemment NB est l'ensemble des séquences qui ne sont pas fréquentes mais dont les sous-séquences qui l'ont générée sont fréquentes. La figure 12 illustre un exemple de bordure négative (zone grisée) pour la base de données $DBSpade$ de la figure 3. Les liens en pointillés représentent une génération de candidats qui n'a pas donnée lieu à de nouveaux fréquents. Par exemple pour le fréquent $<$ (A B) (B) $>$, seules les séquences fréquentes $<$ (A) (B) $>$ et $<$ (B) (B) $>$ lui permettent de devenir fréquent et servent à calculer son support.

La première étape d'ISM, consiste à supprimer de l'ensemble des séquences fréquentes, celles qui ne le sont plus après la mise à jour. Via une passe sur la base, le treillis ainsi que la bordure négative sont mis à jour pour prendre en compte le nouveau support. Au cours de cette étape ISM détermine les séquences de la bordure négative

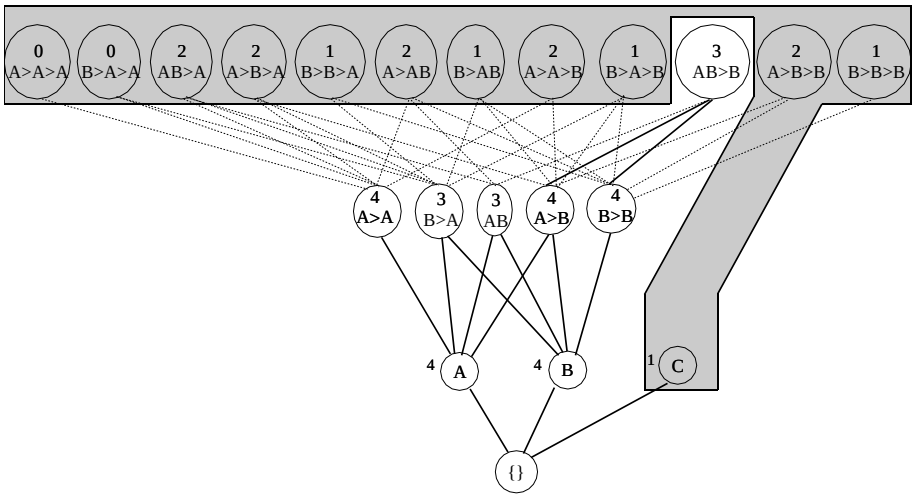

Figure 12. *La bordure négative, considérée par* ISM *après exécution de* SPADE *sur la base de données de la fi gure 3*

qui vont migrer de NB vers FS (où FS désigne ici l'ensemble des séquences fréquentes). ISM obtient donc les séquences qui, en quelque sorte, vont être "dégrisées" dans le treillis. De plus, ISM met à jour l'ensemble des items fréquents à la fin de cette passe, afin d'insérer d'éventuels nouveaux items dans le treillis.
La seconde étape d'ISM, consiste à reprendre les nouveaux fréquents (les items ajoutés au treillis, ou les séquences qui ont migré de NB vers FS) un par un, afin de faire progresser l'information dans le treillis en reprenant le principe de génération de SPADE. Le champ d'exploration que doit considérer ISM est alors limité aux éléments nouveaux, et des optimisations sont mises en place compte tenu des caractéristiques particulières que présentent les nouveaux fréquents (pour plus de détails sur les caractéristiques que les nouveaux fréquents peuvent présenter, le lecteur peut se reporter à [PAR 99]).
À l'issue de cette dernière étape, le treillis contenant le nouvel ensemble de séquences fréquentes ainsi que la nouvelle bordure négative sont mis à jour.

Dans [ZHE 02], l'algorithme IUS améliore l'utilisation de la bordure négative en spécifiant un nouveau support minimum supplémentaire qui permet de ne plus conser-

ver la totalité de la bordure mais seulement une partie correspondant à cette nouvelle valeur. L'avantage de cette approche est alors de considérablement réduire l'espace de stockage. Cependant cette valeur de support n'est en fait définissable que de manière expérimentale et est donc tout à fait dépendante des types de sources utilisées.

5.4. *De l'incrémental à l'intéractif*

Certaines approches étendent encore plus la connaissance extraite. Dans le cadre de FASTUP [LIN 98] et de son extension KISP [LIN 03], l'idée générale est de proposer une sorte de GSP amélioré qui tienne compte des résultats obtenus lors d'une précédente fouille de données sur la base avant de construire, proposer et valider des candidats, selon le schéma générer-élaguer. En fait, ces approches se situent plus dans une démarche *intéractive* qu'incrémentale dans la mesure où elles s'intéressent plus particulièrement aux variations de support. La notion d'interactivité avait initialement été proposée dans [PAR 99].

Le principe général est le suivant. Une première recherche des motifs séquentiels est effectuée sur DB via GSP. La base fondamentale des connaissances KB est construite, en même temps, en comptant le support des l-séquences. KB est ensuite mise à jour pour chaque passe effectuée par GSP. A la fin de l'exécution de GSP, KB contient toutes les séquences (fréquentes ou candidates) dont le support à été calculé. KB conserve cette information pour deux raisons. La première réside dans le fait que lors d'une diminution de support, certains candidats non fréquents initialement peuvent devenir fréquents et l'obtention de ces motifs ne nécessite plus d'accéder à la base DB. Deuxièmement pour retrouver les motifs fréquents, GSP compte le support de nombreux candidats même si ceux-ci se révèlent non fréquents par la suite. Via KB il est alors possible d'éviter de générer de tels candidats et d'avoir alors dans GSP une phase de comptage de support plus efficace et un arbre de hachage plus restreint. En d'autres termes, soit sup_{KB} le support pour lequel KB a été calculée. Dans le cas d'une nouvelle extraction avec $minSupp > sup_{KB}$, il suffit d'examiner dans KB quels sont les motifs qui satisfont le nouveau support. KB n'est pas modifié et aucun accès à DB n'est effectué. Dans le cas où $minSupp < sup_{KB}$, il est nécessaire d'appliquer une extraction sur DB pour retrouver les nouveaux motifs qui ne sont pas contenus dans KB. L'amélioration apportée à GSP est que l'on connaît à l'avance les candidats à tester, i.e. ceux qui ne sont pas contenus dans KB. A l'issue de l'extraction, les nouveaux candidats sont intégrés dans KB pour une extraction future et sup_{KB} prend pour valeur $minSupp$. Au fur et à mesure des calculs, le contenu de KB est étendu de manière incrémentale. L'efficacité en terme de réduction de comptage de support de KB va donc en s'accroissant. Les principes utilisés par KISP étendent les travaux menés autour de la bordure négative. La question de la taille de la base de connaissance et de son évolution reste cependant entière. Par exemple, dans le cadre de motifs séquentiels long et avec un support élevé cette dernière aura rapidement une taille supérieure à celle de la bordure négative.

6. Problématiques étendues

Comme nous l'avons vu dans les sections précédentes, la problématique de la recherche des motifs séquentiels a donné lieu à un grand nombre de travaux de recherche. Ces dernières années, motivés par l'utilisation que l'on peut faire de ces motifs, de nouveaux travaux étendent la problématique initiale, notamment à la prise en compte de différentes contraintes ou à d'autres types de motifs.

6.1. *L'extraction de motifs séquentiels sous contraintes*

Dans [PEI 02], les auteurs proposent une contribution ayant deux aspects. Le premier concerne un travail de synthèse sur l'exploitation des contraintes dans l'extraction de motifs séquentiels. Le second se place au niveau de l'extraction de motifs à l'aide des techniques de bases projetées (C.f. Section 4.2) qui intègrent ces contraintes. Nous invitons le lecteur intéressé par la gestion des contraintes en extraction de motifs séquentiels à trouver plus de détails dans [PEI 02]. Cependant nous proposons ici un bref aperçu des avantages qu'offre ce domaine et des contraintes que l'on peut définir.

Pour comprendre l'intérêt de l'extraction de motifs séquentiels sous contraintes, considérons l'exemple suivant (issu de [PEI 02]). Pour identifier une nouvelle maladie, des chercheurs peuvent avoir besoin de trouver des motifs séquentiels de cette forme : " 2 à 7 jours de toux, suivis de fièvre entre 37,5 et 39 degrès pendant 2 à 5 jours avec une température moyenne de 38 degrès ($\pm$0,2), tous ces symptomes étant contenus sur 2 semaines". On peut alors énumérer plusieurs types de contraintes (en partie illustrées par l'exemple précédent) :

– *contrainte d'item* : permet un restriction sur l'ensemble des items contenus dans les résultats (par exemple dans le cadre de l'analyse du comportement des utilisateurs d'un site Web (i.e. Web Usage Mining), se limiter aux pages d'enseignement d'une personne précise).

– *contrainte de longueur* : permet d'obtenir des motifs séquentiels d'une longueur particulière (par exemple une longueur minimum, en termes d'items ou d'itemsets).

– *contrainte de sur-séquence* : permet d'obtenir des résultats qui contiennent un motif séquentiel précis (par exemple tous les motifs qui contiennent l'achat d'un PC suivi par l'achat d'un caméscope).

– *contrainte d'agrégation* : permet d'appliquer des fonctions de regroupement sur les motifs extraits (par exemple les motifs tels que le prix moyen des articles vendus dépasse 100 Euros).

– *contrainte d'intervalle de temps* : permet d'obtenir des motifs séquentiels vérifiés par des séquences de données (donc participant au support de ce motif) qui respectent certains intervalles. Ce type de contrainte est utile pour trouver des comportements à long terme, à court terme ou bien pour extrapoler les données temporelles contenues dans la base.

– *contrainte d'expressions régulières* : prend la forme d'une expression régulière

sur l'ensemble des items. Un motif séquentiel n'est accepté que s'il respecte l'automate représentant l'expression considérée. Par exemple, il est possible d'extraire uniquement les motifs qui respectent la contrainte (*permanent* | *thesard*) (*publications*) (*revues* | *conf_internationale* | *chap_livre*) qui s'interprèterait comme la recherche de comportement d'utilisateurs ayant navigué, dans l'ordre, sur la page d'accueil d'un thésard ou d'un permanent, suivi d'une rubrique publications, puis d'une page concernant ses publications en revue ou en conférence internationale ou comme chapitre de livre.

Nous ne détaillerons pas dans cet article les différentes contributions associées. Cependant, nous invitons le lecteur intéressé par la gestion des contraintes de temps à consulter les propositions de [MAS 04, LIN 02]. Concernant l'extraction de motifs séquentiels sous contraintes d'expressions régulières, le lecteur est invité à consulter les propositions de [GAR 02, ALB 03].

6.2. *Problématiques proches*

6.2.1. APPROXMAP *: motifs approximatifs*

Dans [KUM 03], les auteurs proposent une méthode d'extraction de motifs séquentiels approximatifs. L'originalité de cette proposition se situe sur deux plans :

– Une approche de décomposition du problème à l'aide de classification sur les séquences de données à traiter.

– Une approche d'extraction de motifs séquentiels approximatifs sur les clusters obtenus.

La première idée des auteurs consiste à proposer des clusters pour les séquences de données contenues dans la base. Dans cette optique, un algorithme des plus proches voisins (k-NN) est utilisé avec pour mesure de distance le coût des opérations (insertion, suppression, remplacement) nécessaires pour transformer une séquence $s1$ en $s2$. Les détails des opérations envisagées, des coût mis en jeu et de l'algorithme de classification développé sont donnés dans [KUM 03]. L'originalité de cette approche se situe dans la définition des motifs séquentiels approximatifs. Il s'agit en effet d'extraire des motifs séquentiels qui, s'ils ne sont pas exacts ou complets, s'approchent de la réalité et permettent de comprendre les données d'un point de vue différent. Considérons le cluster donné à la figure 13 et issu de [KUM 03]. Les auteurs proposent l'alignement exprimé dans la colonne "Alignement" (figure 13) pour les séquences de la colonne "Sequence". Cet alignement permet de calculer une séquence pondérée (colonne "Weighted sequence") qui permet alors de calculer les motifs approximatifs. Avec un support minimum de 80% , le motif obtenu est : < (a) (b) (d) >. En effet la séquence pondérée indique que chacun de ces items a un support supérieur ou égal à 4/5. De la même façon, un support de 60% permet d'obtenir le motif : < (a) (b c) (d e) > (qui correspond à la séquence pondérée filtrée sur les items ayant un support supérieur ou égal à 3/5). Il est intéressant de remarquer que ce motif ne correspond à aucune des séquences enregistrées dans la base. Cependant il reflète une tendance

Seq-id	Sequence	Alignement				
S1	<(ag)(f)(bc)(ae)(h)>	<(ag)	(f)	(bc)	(ae)	(h)>
S2	<(ae)(h)(b)(d)>	<(ae)	(h)	(b)	(d)	>
S3	<(a)(b)(de)>	<(a)		(b)	(de)	>
S4	<(a)(bcg)(d)>	<(a)		(bcg)	(d)	>
S5	<(bci)(de)>	<		(bci)	(de)	>
Weighted sequence		< (a:4, e:1, g:1):4	(f:1, h:1):2	(b:5, c:3, g:1, i:1):5	(a:1, d:4, e:3):5	(h:1):1> :5

Figure 13. *Cluster obtenu par la première phase d'ApproxMAP et motifs extraits*

dans cette base et peut donc s'avérer utile. Le principal intérêt de cette approche se situant dans l'utilisation de la séquence pondérée qui permet d'éviter tout calcul en cas de changement de support (une application du filtre permet de trouver le nouveau motif).

6.2.2. TSP *: Top-k sequential patterns*

Dans [TZV 03], les auteurs proposent le problème de la recherche des *Top-k Sequential Patterns*. Il s'agit d'une variante du problème de la recherche de motifs séquentiels dans la mesure où la méthode accepte les paramètres suivants : une base de données D, une longueur de motif minimum min_l et le nombre de motifs k. A partir de ces informations l'algorithme doit fournir le support minimum correspondant et les motifs trouvés pour ces critères (les k motifs les plus fréquents de longueur au moins min_l). De plus, dans ce contexte, l'algorithme TSP a pour objectif de fournir les motifs séquentiels fermés afin d'optimiser les temps de réponse. Cette problématique présente un intérêt majeur pour certains types d'applications. En effet, pour certaines applications, il est fréquent de procéder à une phase d'extraction de motifs peu satisfaisante (motifs courts ou évidents par exemple) et de recommencer en baissant le support minimum jusqu'à obtenir des motifs séquentiels intéressants (mais ils restent encore trop nombreux). L'idée consiste donc à dire que l'algorithme doit être capable de fournir des motifs séquentiels en ne connaissant que leur longueur minimum et le nombre désiré. Dans les grandes lignes, voici le principe de TSP : 1) pour chaque item, appliquer PREFIXSPAN avec min_sup=1 2) continuer PREFIXSPAN de manière récursive jusqu'à l'obtention d'une séquence de longueur $> min\_l$ 3) élever min_sup en prenant le motif le moins fréquent obtenu en 2) et 4) recommencer de manière récursive sur les branches non explorées avec le nouveau min_sup.

7. Conclusion

Après l'engouement des travaux de recherche pour les règles d'association, les motifs séquentiels ont été très étudiés ces dernières années. Nous avons pu constater au cours de cet article que depuis la définition de la problématique en 1995, de nombreux travaux de recherche ont été menés. Initialement, les premiers travaux ont consisté à améliorer les performances des propositions. Dans ce cadre, de nouvelles structures de données ou de nouvelles représentations des données sources ont été mises au point.

D'autres extensions ont considéré qu'avec les nouvelles techniques de stockage, il devenait possible de gérer la base en mémoire vive et proposent ainsi de nouveaux algorithmes. Même si les propositions ont permis de considérablement améliorer les temps de réponse, un premier constat a emergé. De la même manière que pour les règles d'association, il n'est pas possible de dire quel algorithme est meilleur que les autres, dans la mesure où leurs performances sont étroitement liées aux types de données manipulées. Ainsi, les algorithmes du type GSP ou PSP seront très efficaces dans le cas de grandes bases de données avec des séquences moyennement longues. Par contre, les algorithmes comme SPADE, SPAM ou PREFIXSPAN seront eux très efficaces dans le cas où un très grand nombre de candidats de même taille sont générés. L'utilisation des algorithmes sur différents domaines d'application et le fait que les données sources peuvent varier rapidement ont donné lieu à de nombreux travaux de recherche autour de la fouille de données incrémentale.

L'une des raisons essentielles du développement des approches autour des motifs séquentiels est leur capacité d'adaptation à de très nombreux problèmes. Pour faciliter cette adaptation, les derniers travaux intègrent d'ailleurs de plus en plus de contraintes et offrent plus de souplesse sur la définition des motifs séquentiels. Parmi les nouveaux problèmes utilisant des techniques directes ou issues des motifs séquentiels nous pouvons citer :

– le *Web Mining* [KOS 00] qui consiste à analyser les sites Web en fonction de leurs structures et de leurs contenus (Web Content Mining) mais également en fonction de leurs usages (Web Usage Mining).

– le *Schema mining* [CON 02] qui consiste à rechercher dans de grandes bases de données d'objets complexes des structures typiques pour par exemple créer de manière automatique des index ou des vues,

– le *Text Mining* dans lequel les motifs séquentiels peuvent servir à analyser des tendances au cours du temps dans des textes [LEN 97],

– et bien d'autres problèmes relatifs à la bio-informatiques (séquences de gènes, relevés de capteurs bio-médicaux, ...) [ZAK 03], à l'accidentologie, la sécurité (détection d'intrusions, détection de fraudes), etc.

8. Bibliographie

[AGR 93] AGRAWAL R., IMIELINSKI T., SWAMI A., « Mining Association Rules between Sets of Items in Large Databases », *Proceedings of the 1993 ACM SIGMOD Conference*, Washington DC, USA, May 1993, p. 207-216.

[AGR 95a] AGRAWAL R., MEHTA M., SHAFER J., SRIKANT R., ARNING A., BOLLINGER T., « The Quest Data Mining System », *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining (KDD'95)*, Montreal, Canada, August 1995.

[AGR 95b] AGRAWAL R., SRIKANT R., « Mining Sequential Patterns », *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, Tapei, Taiwan, March 1995.

[ALB 03] ALBERT-LORINCZ H., BOULICAUT J.-F., « Mining Frequent Sequential Patterns under Regular Expressions : a Highly Adaptative Strategy for Pushing Constraints », *Proceedings of SIAM DM'03*, San Francisco, May 2003, p. 316-320.

[AYR 02] AYRES J., GEHRKE J., YIU T., FLANNICK J., « Sequential Pattern Mining Using Bitmap Representation », *Proocedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 2002.

[CON 02] CON G., YI L., LIU B., WANG K., « Discovering Frequent Substructures from Hierarchical Semi-structured Data », *Proceedings of the Second SIAM International Conference on Data Mining (SDM-2002)*, Arlington, VA, USA, April 2002.

[GAR 02] GAROFALAKIS M., RASTOGI R., , SHIM K., « Mining Sequential Patterns with Regular Expression Constraints », *IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 3*, May/June 2002, p. 530-552.

[HAN 00] HAN J., PEI J., MORTAZAVI-ASL B., CHEN Q., DAYAL U., HSU M., « FreeSpan : Frequent Pattern-Projected Sequential Pattern Mining », *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD'00)*, Boston, MA, August 2000, p. 355-359.

[KOS 00] KOSALA R., BLOCKEEL H., « Web mining research : A survey », *ACM SIGKDD Explorations*, vol. 2, 2000, p. 1-15.

[KUM 03] KUM H.-C., PEI J., WANG W., DUNCAN D., « ApproxMAP : Approximate Mining of Consensus Sequential Patterns », *Proceedings of SIAM DM'03*, San Francisco, May 2003.

[LEL 04] LELEU M., « Extraction de Motifs Séquentiels Sous Contraintes dans des Données Contenant des Répétitions Consécutives », *PHD Dissertation, INSA Lyon*, 2004.

[LEN 97] LENT B., AGRAWAL R., SRIKANT R., « Discovering Trends in Text Databases », *Proceedings of the 3rd International Conference on Knowledge Discovery in Databases and Data Mining (KDD'97)*, Newport Beach, California, August 1997.

[LIN 98] LIN M., LEE S., « Incremental Update on Sequential Patterns in Large Databases », *Proceedings of the Tools for Artificial Intelligence Conference (TAI'98)*, May 1998, p. 24-31.

[LIN 02] LIN M.-Y., LEE S.-Y., WANG S.-S., « DELISP : Effi cient Discovery of Generalized Sequential Patterns by Delimited Pattern-Growth Technology », *Proceedings of the 6th Pacific-Asia Conference (PAKDD 2002)*, Taipei Taiwan, May 2002.

[LIN 03] LIN M.-Y., LEE S.-Y., « Improving the Effi ciency of Interactive Sequential Pattern Mining by Incremental Pattern Discovery », *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, Big Island, Hawaii, January 2003.

[MAN 94] MANNILA H., TOIVONEN H., VERKANO A. I., « Improved Methods for Finding Association Rules », *Proceedings of the AAAI Worshop on Knowledge Discovery*, Washington, July 1994, p. 181-192.

[MAS 98] MASSEGLIA F., CATHALA F., PONCELET P., « The PSP Approach for Mining Sequential Patterns », *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98), LNAI, Vol. 1510*, Nantes, France, September 1998, p. 176-184.

[MAS 02] MASSEGLIA F., « Algorithmes et applications pour l'extraction de motifs séquentiels dans le domaine de la fouille de données : de l'incrémental au temps réel », *PHD Dissertation*, Université de versailles St Quentin - France, 2002.

[MAS 03] MASSEGLIA F., PONCELET P., TEISSEIRE M., « Incremental Mining of Sequential Patterns in Large Databases », *Data and Knowledge Engineering*, vol. 46, n° 1, 2003, p. 97-121.

[MAS 04] MASSEGLIA F., TEISSEIRE M., PONCELET P., « Pre-Processing Time Constraints for Effi ciently Mining Generalized Sequential Patterns », *Proceedings of the 11th International Symposium on Temporal Representation and Reasoning (TIME'04)*, Tatihou France, July 2004.

[PAR 99] PARTHASARATHY S., ZAKI M., OGIHARA M., DWARKADAS S., « Incremental and Interactive Sequence Mining », *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM'99)*, Kansas City, MO, USA, November 1999, p. 251-258.

[PEI 01] PEI J., HAN J., MORTAZAVI-ASL B., PINTO H., CHEN Q., DAYAL U., HSU M., « Prefi xSpan : Mining Sequential Patterns Effi ciently by Prefi x-Projected Pattern Growth », *Proceedings of 17th International Conference on Data Engineering (ICDE'01)*, 2001.

[PEI 02] PEI J., HAN J., WANG W., « Mining Sequential Patterns with Constraints in Large Databases », *Proceedings 2002 Int. Conf. on Information and Knowledge Management (CIKM'02)*, Washington, November 2002.

[SRI 96] SRIKANT R., AGRAWAL R., « Mining Sequential Patterns : Generalizations and Performance Improvements », *Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96)*, Avignon, France, September 1996, p. 3-17.

[TZV 03] TZVETKOV P., YAN X., HAN J., « TSP : Mining Top-K Closed Sequential Patterns », *Proceedings of the Third International Conference on Data Mining (ICDM'03)*, Melbourne, November 2003, p. 347-355.

[WAN 96] WANG K., TAN J., « Incremental Discovery of Sequential Patterns », *Proceedings of ACM SIGMOD'96 Data Mining Workshop*, Montréal, Canada, June 1996, p. 95-102.

[WAN 04] WANG J., HAN J., « BIDE : Effi cient Mining of Frequent Closed Sequences », *Proceedings of the International Conference on Data Engineering (ICDE'04)*, Boston, M.A., March 2004.

[YAN 03] YAN X., HAN J., AFSHAR R., « CloSpan : Mining Closed Sequential Patterns in Large Databases », *SDM'03*, San Fransisco, CA, May 2003.

[ZAK 01] ZAKI D., « SPADE :An Effi cient Algorithm for Mining Frequent Sequences », *Machine Learning*, vol. 42, 2001, p. 31-60, Kluwer Academic Publishers.

[ZAK 03] ZAKI M., « Mining Data in Bioinformatics », *in Nong Ye (ed.) Handbook of Data Mining*, Lawrence Earlbaum Associates, 2003, p. 573-596.

[ZHE 02] ZHENG Q., KSU K., MA S., LV W., « The Algorithms of Updating Sequential Patterns », *Proceedings of the International Conference on Data Mining (ICDM'02)*, USA, 2002.