

## Le FIA : un nouvel automate permettant l'extraction efficace d'itemsets fréquents dans les *data streams*.

Jean-Émile SYMPHOR\*, Alban MANCHERON\*, Lionel VINCESLAS\* et Pascal PONCELET\*\*

\*GRIMAAG, Université des Antilles et de la Guyane, Martinique, France.  
{je.symphor,alban.mancheron,lionel.vinceslas}@martinique.univ-ag.fr

\*\* EMA-LG2IP/site EERIE, Parc Scientifique Georges Besse, 30035 Nîmes Cedex, France.  
pascal.poncelet@ema.fr

**Résumé.** Nous présentons dans cet article un nouvel automate : le FIA qui permet de traiter de façon efficace la problématique de l'extraction des itemsets fréquents dans les *data streams*. Le FIA est une structure de données très compacte et informative qui, par ailleurs, présente des propriétés incrémentales facilitant grandement la mise à jour avec une granularité très fine. L'algorithme développé pour la mise à jour du FIA effectue un unique passage sur les données qui sont prises en compte par batch, itemset par itemset et pour chaque itemset, item par item. Utilisé dans le cadre d'une approche prédictive, le FIA permet d'indexer les itemsets véritablement fréquents du *stream* en maximisant soit le rappel soit la précision.

### 1 Introduction

L'extraction d'itemsets fréquents constitue une étape importante dans les problématiques de fouille de données pour la recherche de règles d'association, de motifs séquentiels ou encore d'itemsets maximaux. Agrawal et Srikant (1994) furent les premiers à traiter cette question et ont été suivi en ce sens par Han et al. (2000). Toutefois, les différents algorithmes proposés dans la littérature reposent sur des structures de données de type arbre ou encore treillis. C'est le cas notamment des algorithmes bien connus tels que *A-priori* (voir Agrawal et Srikant 1994) et *FP-Growth* (voir Han et al. 2000). On retrouve cette problématique d'extraction de motifs dans d'autres domaines tels que la bioinformatique ou encore l'algorithmique du texte même si les motifs recherchés sont des séquences qui présentent des caractéristiques très différentes de celles des itemsets. En algorithmique du texte par exemple, plusieurs algorithmes reposent sur des structures de données de type automates. Hoshino et al. (2000) ont introduit, un nouvel automate déterministe et acyclique : le SA (Subsequence Automaton) qui permet de reconnaître toutes les sous-séquences d'un ensemble de textes. À partir du SA, Tronicek (2002) a proposé un algorithme permettant la construction d'un automate, également déterministe et acyclique : le CSA (Common Subsequence Automaton) qui reconnaît uniquement l'ensemble des sous-séquences communes à un ensemble de textes. D'une façon générale pour tout les domaines, l'enjeu est de disposer de structures de données suffisamment compactes et informatives afin d'éviter l'explosion combinatoire du nombre de noeuds de l'arbre ou d'états de

l'automate. En effet, c'est l'applicabilité des algorithmes qui peut être remise en question en raison des coûts trop prohibitifs en temps de calcul et en espace mémoire utilisé. Le premier objectif de cet article est d'apporter une réponse à une intéressante et importante question qui reste ouverte : existe-t-il d'autres structures de données qui soient suffisamment informatives et compactes et qui permettent d'extraire de façon plus efficace les *itemsets* fréquents ? Récemment, beaucoup de travaux issus de la communauté des chercheurs en fouille de données, considèrent le cas des flots de données ou *data streams* où l'acquisition des données s'effectue de façon régulière, continue ou incrémentalement et cela sur une durée longue voire éventuellement illimitée. Plusieurs applications, telles que les transactions financières, la navigation sur le Web, la téléphonie mobile relèvent du cas *data streams* et exigent des temps de réponse extrêmement faibles. En conséquence, la problématique d'extraction d'*itemsets* fréquents dans les *data streams* par rapport à la fouille traditionnelle dans les bases de données présente des nouvelles caractéristiques qu'il convient de souligner. En premier lieu, il faut considérer les aspects liés à la mise à jour. En effet, de par la taille très grande voire infinie du *data streams*, la base de données représentant les informations du *stream* est sujette à des mises à jour régulières et fréquentes. Les résultats obtenus pour l'ancienne base ne sont que partiellement valables pour la nouvelle et il n'est pas envisageable de relancer l'algorithme sur toute la base mise à jour. Masségli et al. (2003) développent une approche par mise à jour incrémentale où la stratégie consiste à construire au départ un ensemble de motifs candidats, puis à le maintenir au fur et à mesure des mises à jour successives. Celui-ci est ensuite utilisé pour mettre à jour l'ensemble des motifs fréquents et éviter de relancer l'algorithme depuis le début. En second lieu, l'exigence spatiale et temporelle s'en trouve renforcée et impose de ne s'autoriser qu'un passage sur les données. On parle dans ce cas d'algorithmes *une-passe*. Il faut, par conséquent renoncer à obtenir des résultats exacts, mais autoriser une erreur quant à l'estimation de la fréquence des motifs. Enfin, il faut prendre en compte l'incertitude engendrée par la connaissance toujours incomplète du *data stream*. Les *itemsets* fréquents obtenus ne sont en fait que des *itemsets* fréquents observés. Ainsi, des *itemsets* classés comme non fréquents peuvent le devenir sur une plus longue période d'observation et inversement, des *itemsets* observés fréquents ne sont peut-être plus du tout fréquents sur une plus longue période d'observation du *data stream*. Il est statistiquement impossible de s'affranchir de ces deux sources d'erreurs à partir de la connaissance d'une partie, même très grande du *stream* (voir V. Vapnik 1998). On peut toutefois, chercher à minimiser l'une des sources d'erreurs tout en maintenant l'autre en dessous d'un seuil raisonnable. Notre deuxième objectif, est de proposer une nouvelle structure de données ayant des propriétés incrémentales, permettant de construire et de maintenir de façon efficace l'ensemble des *itemsets* fréquents d'un *data stream* tout en minimisant l'une ou l'autre des sources d'erreur. La suite de l'article est organisée de la manière suivante. La section 2 présente plus formellement la problématique. Dans la section 3, nous présentons un aperçu d'autres travaux abordant cette problématique. La section 4 présente notre approche et nos solutions. Les expérimentations sont décrites dans la section 5 et une conclusion est proposée dans la section 6.

## 2 Problématique

Soit  $\mathbb{I} = \{i_1, i_2, \dots, i_m\}$  un ensemble ordonné d'items utilisé dans une base de données *DB* de transactions, où chaque transaction  $t_r$  identifiée de manière unique est associée à un

ensemble d'items de  $\mathbb{I}$ . Un ensemble  $\mathcal{X} \subseteq \mathbb{I}$  est appelé un  $p$ -itemset et est représenté par  $\{x_1, x_2, \dots, x_p\}$ . L'entier  $p = |\mathcal{X}|$ , est la longueur de  $\mathcal{X}$  et  $Sub(\mathcal{X})$  l'ensemble de tous les sous-itemsets de  $\mathcal{X}$ , c'est à dire les itemsets obtenus en supprimant zéro ou plusieurs items de  $\mathcal{X}$ . Le support d'un itemset  $\mathcal{X}$ , noté  $supp(\mathcal{X})$ , correspond au nombre de transactions dans lesquelles l'itemset apparaît. Un itemset est dit  $\theta$ -fréquent si  $supp(\mathcal{X}) \geq \sigma$ , où  $\sigma = \lceil \theta \times |DB| \rceil$  correspond au support minimal spécifié par l'utilisateur avec  $\theta \in [0; 1]$  et  $|DB|$  la taille de la base de données. Le problème de la recherche des itemsets fréquents consiste à rechercher tous les itemsets dont le support est supérieur ou égal à  $\sigma$  dans  $DB$ . Cette problématique, étendue au cas des *data streams*, peut s'exprimer comme suit. Soit un *Data stream*  $DS = B_{a_i}^{b_i}, B_{a_{i+1}}^{b_{i+1}}, \dots, B_{a_n}^{b_n}$ , un ensemble infini de *batches* où chaque *batch* est associé à une période de temps  $[a_j, b_j]$ , i.e.  $B_{a_j}^{b_j}$  avec  $b_j > a_j$  et  $B_{a_n}^{b_n}$  le plus récent batch obtenu. Chaque batch  $B_{a_j}^{b_j}$  correspond à un ensemble de transactions d'itemsets où  $B_{a_j}^{b_j} = [s_1, s_2, \dots, s_p]$ . Nous supposons également, que les batches n'ont pas nécessairement la même taille. La cardinalité  $k$  du *data stream* ( $|DS|$ ), à un instant donné est définie par  $k = |B_{a_i}^{b_i}| + |B_{a_{i+1}}^{b_{i+1}}| + \dots + |B_{a_n}^{b_n}|$  où  $|B_{a_j}^{b_j}|$  correspond à la cardinalité du batch  $B_{a_j}^{b_j}$ . La fréquence d'un itemset  $\mathcal{X}$  à un instant donné  $t$  est défini comme étant le ratio du nombre de transactions qui contiennent  $\mathcal{X}$  dans les différents batches sur le nombre total de transactions connu à l'instant  $t$ . Ainsi, pour un support minimal fixé par l'utilisateur, le problème de la recherche des itemsets fréquents dans un *data stream* consiste à rechercher tous les itemsets  $\mathcal{X}$  qui vérifient  $\sum_{a_i}^{b_i} supp(\mathcal{X}) = \lceil \theta \times k \rceil$  dans le *data stream*.

$B_0^1$					$B_1^2$		$B_2^3$
$s_1$	$s_2$	$s_3$	$s_4$	$s_5$	$s_6$	$s_7$	$s_8$
abcde	bcef	be	abd	cd	abc	abce	bcef

FIG. 1 – Ensembles de batches  $B_0^1$ ,  $B_1^2$  et  $B_2^3$  construits sur  $\mathbb{I} = \{a, b, c, d, e, f\}$ .

### 3 Travaux liés

Les différents travaux portant sur la problématique d'extraction d'itemsets fréquents dans les *data streams* se déclinent selon trois axes en fonction du modèle de traitement des itemsets du *stream*. Le premier concerne l'extraction à partir de fenêtres à point fixe où on conserve tous les itemsets acquis du *stream* (voir Manku, Motwani 2002) et (voir Li, Lee 2004). Le second axe est différent du précédent simplement par le fait que l'on introduit une distinction entre les itemsets récemment et moins récemment acquis. Chang et Lee (Juin 2004) attribuent un poids décroissant aux transactions en fonction de l'ancienneté de leur acquisition. Autrement dit, les anciennes transactions contribuent moins que les nouvelles quant au calcul de la fréquence des itemsets. Gianella et al (2002) utilisent une structure de type FP-tree pour rechercher des itemsets fréquents à différents niveaux de granularité temporelle. Le dernier axe concerne l'extraction à partir de fenêtres glissantes où on considère plus seulement l'acquisition mais aussi l'enlèvement d'itemsets (voir Chang, Lee juillet 2004). L'approche que nous

développons dans cet article, s'inscrit dans le premier axe. Aussi, nous précisons dans la suite de ce paragraphe, les caractéristiques des algorithmes ainsi que l'erreur et les types d'approximation sur les résultats relatif à cet axe. Manku et Motwani 2002 ont développé un algorithme : *Lossy counting*, basé sur la propriété d'antimonotonie du support. Cet algorithme effectue un seul passage sur les données et utilise une structure à base d'arbres pour représenter les itemsets. Les auteurs introduisent un paramètre d'erreur fixé par l'utilisateur et voulu très inférieur au support afin de minimiser le nombre de résultats faux positifs et afin d'améliorer la valeur de la fréquence obtenue des itemsets. Ils donnent les garanties suivantes sur leurs résultats. Tous les itemsets réellement fréquents sont trouvés. Il n'y a pas de faux négatifs. Tous les itemsets considérés fréquents à tort, i.e. les faux positifs ont une forte valeur de fréquence proche de la fréquence voulue. L'incertitude sur la fréquence des itemsets est fonction du paramètre d'erreur. Li et Lee (2004) proposent d'extraire les itemsets fréquents en partant des plus grands aux plus petits et utilisent une structure très compacte qui résulte d'une extension d'une représentation basée sur des arbres préfixés : le *CFI-tree* (Candidate Frequent Itemset tree). Toutefois, l'algorithme développé : *DSM-FI*, bien qu'effectuant un seul passage sur les données, comprend une phase d'élagage du *CFI-tree* et nécessite plusieurs parcours de la structure pour obtenir l'information sur la fréquence des itemsets. Les garanties apportées, quant aux résultats, indiquent qu'ils n'y a pas de faux négatifs et que l'erreur sur la fréquence des itemsets est bornée.

## 4 Notre Approche

Dans un premier temps, nous nous intéressons à l'extraction des itemsets fréquents dans une base de données. Nous introduisons un nouvel automate : le FIA, l'automate des itemsets fréquents qui constitue une structure de données très compacte et informative permettant d'extraire de façon efficace tous les itemsets fréquents d'une base de données. Dans un second temps, nous montrerons la mise à jour incrémentale du FIA avec l'ajout de nouveaux batches du *data stream*. Pour tenir compte de l'incertitude engendrée par la connaissance toujours incomplète du *stream*, il nous semble pertinent d'un point de vue statistique de développer une approche prédictive (voir Nock 2007). En effet, plutôt que d'extraire des itemsets observés fréquents sur la partie connue du *stream*, il est préférable de prédire les itemsets véritablement fréquents sur tout le *stream* à partir des itemsets connus.

### 4.1 Rappels sur la théorie des automates

Nous rappelons simplement ci-dessous les principaux points de la théorie sur les automates que nous réutiliserons dans la suite. Selon la notation standard des automates finis (6).

**Définition 1.** *Un automate à états finis  $\mathcal{A}$  est un quintuple tel que  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, \mathcal{I}, \mathcal{F})$ , où  $\mathcal{Q}$  est un ensemble fini d'états,  $\Sigma$  un alphabet,  $\delta \subseteq \mathcal{Q} \times \Sigma \times \mathcal{Q}$  est un ensemble de transitions,  $\mathcal{I} \subseteq \mathcal{Q}$  et respectivement  $\mathcal{F} \subseteq \mathcal{Q}$  sont l'ensemble des états initiaux et finaux.*

L'étiquette d'une transition  $d$  passant d'un état  $q$  à un état  $q'$ , noté  $d = (q, \alpha, q')$  est le symbole  $\alpha$ . Un chemin dans  $\mathcal{A}$  est une suite  $c = d_1, \dots, d_n$  de transitions consécutives avec pour étiquette  $|c| = \alpha_1 \dots \alpha_n$ . On écrit également si  $w = |c|$ ,  $c : q_0 \xrightarrow{w} q_n$ . Un chemin  $c : i \rightarrow f$  est dit réussi si  $i \in \mathcal{I}$  et  $f \in \mathcal{F}$ . Un mot est reconnu s'il est l'étiquette d'un

chemin réussi. Le langage reconnu par l'automate  $\mathcal{A}$  est l'ensemble des mots reconnus par  $\mathcal{A}$ , soit  $\mathcal{L}(\mathcal{A}) = \{w \subseteq \Sigma \mid \exists c : i \xrightarrow{w} f, i \in \mathcal{I}, l \in \mathcal{F}\}$ . Un état  $q \in \mathcal{Q}$  d'un automate  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, \mathcal{I}, \mathcal{F})$  est accessible s'il existe un chemin  $c : i \rightarrow q$  avec  $i \in \mathcal{I}$ . De même, l'état  $q$  est coaccessible s'il existe un chemin  $c : q \rightarrow f$  avec  $f \in \mathcal{F}$ . Un automate est émondé si tous ses états sont accessibles et coaccessibles. Soit  $\mathcal{P}$  l'ensemble des états accessibles et coaccessibles et soit  $\mathcal{A}_0 = (\mathcal{P}, \Sigma, \delta \cap (\mathcal{P} \times \Sigma \times \mathcal{P}), \mathcal{I} \cap \mathcal{P}, \mathcal{F} \cap \mathcal{P})$ , l'automate  $\mathcal{A}_0$  est émondé par construction. Comme tout chemin réussi de  $\mathcal{A}$  ne passe que par des états accessibles et coaccessibles, on a  $\mathcal{L}(\mathcal{A}_0) = \mathcal{L}(\mathcal{A})$ . Les automates  $\mathcal{A}_0$  et  $\mathcal{A}$  sont dits équivalents.

**Définition 2.** Un automate à états finis  $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, \mathcal{I}, \mathcal{F})$  est déterministe si et seulement s'il existe un unique état initial ( $|\mathcal{I}| = 1$ ) et si  $\forall (p, \alpha, q), (p, \alpha, q') \in \delta \Rightarrow q = q'$ .

Les définitions 3 et 4 ont été introduites par HOSHINO et al. (2000) pour le SA (Automate des Sous-séquences). Nous les rappelons ci-dessous mais adaptées au cas des itemsets.

**Définition 3.** Soient un ensemble  $\mathcal{S}$  d'itemsets tel que  $\mathcal{S} = \{s_1, \dots, s_k\}$  avec  $s_i \subseteq \mathbb{I}$  pour tout  $i \in [1, k]$ , et une relation d'ordre  $\mathfrak{R}$  sur  $\mathbb{I}$  on définit un point position ou ppos de l'ensemble  $\mathcal{S}$  comme un  $k$ -uplet  $[p_1, \dots, p_k]$ , où  $p_i \in [0, n_i] \cup \{\infty\}$  est un numéro de position dans l'itemset  $s_i$  de longueur  $n_i$  ordonné selon  $\mathfrak{R}$  (noté  $\tilde{s}_i$ ). Si  $p_i = 0$ , cette position correspond à celle d'un itemset vide noté  $\varepsilon$  se trouvant devant le premier item de  $\tilde{s}_i$ . Si  $p_i = \infty$ , cette autre position correspond à celle d'un itemset vide  $\varepsilon$  se trouvant derrière le dernier item de  $\tilde{s}_i$ . Autrement, cette position correspond à la position du  $p_i^{\text{ème}}$  item de  $\tilde{s}_i$ , pour tout  $i \in [1, k]$ .

La position particulière correspondant à  $p_i = 0$  pour tout  $i \in [1, k]$  est appelée point position initial noté  $q_0^k$ . L'autre position particulière correspondant à  $p_i = \infty$  pour tout  $i \in [1, k]$  est appelée point position puit noté  $q_\infty^k$ . On note  $Pos(\mathcal{S})$  l'ensemble des ppos de  $\mathcal{S}$ .

**Définition 4.** Soit un ensemble  $\mathcal{S}$  d'itemsets tel que  $\mathcal{S} = \{s_1, \dots, s_k\}$ , avec  $s_i \subseteq \mathbb{I}$  pour tout  $i \in [1, k]$  soit un ppos  $[p_1, \dots, p_k] \in Pos(\mathcal{S})$  et pour tout item  $i_i \in \mathbb{I}$ , on définit les points position atteints notés  $PPA_{\mathcal{S}}([p_1, \dots, p_k], i_i)$  comme l'ensemble des points position  $[p'_1, p'_2, \dots, p'_k]$  tels que  $\forall i \in [1, k], p'_i = \min(\{j \mid j > p_i \wedge s_i[j] = i_i\} \cup \{\infty\})$ .

## 4.2 L'automate des itemsets fréquents : le FIA

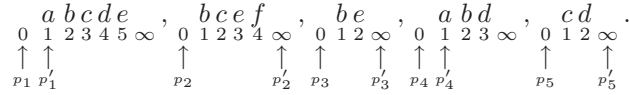
Dans cette section, nous introduisons la définition d'un nouvel automate : le  $FIA_\theta$  qui intègre la notion de fréquence, dont le langage permet de reconnaître l'ensemble des itemsets fréquents d'une base de données.

**Définition 5.** Soit un ensemble  $\mathcal{S}$  d'itemsets  $\mathcal{S} = \{s_1, \dots, s_k\}$  avec  $s_i \subseteq \mathbb{I}$  pour tout  $i \in [1, k]$  et soit un entier  $\sigma$  correspondant à la valeur du support choisi tel que  $1 \leq \sigma \leq k$ , on dit qu'un ppos  $[p_1, \dots, p_k] \in Pos(\mathcal{S})$  satisfait la contrainte de support  $\sigma$  si et seulement si

$$\left| \{p_i \mid p_i < \infty\} \right| \geq \sigma.$$

Ainsi, le ppos  $[p_1, \dots, p_k]$  est dit  $\sigma$ -satisfaisant. L'ensemble de tous les ppos qui sont  $\sigma$ -satisfaisants est noté  $Pos_\sigma(\mathcal{S})$ . C'est un sous-ensemble de  $Pos(\mathcal{S})$ .

**Exemple 1.** Prenons le batch  $B_0^1$  de la figure 1, correspondant à l'ensemble d'*itemsets*  $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5\}$ . À partir de  $q_0^k$ , on obtient (en appliquant la définition 4) que  $PPA_{\mathcal{S}}([0, \dots, 0], a) = [1, \infty, \infty, 1, \infty]$ . Nous représentons schématiquement ci-dessous à titre d'exemple le point position atteint par  $a$  depuis le point position initial.



Cela illustre pour les *itemsets*  $s_1$  et  $s_4$ , que l'item  $a$  est à la première position donc numérotée 1 dans le point position atteint. De même, pour les *itemsets*  $s_2, s_3, s_5$ , cela illustre que l'item  $a$  n'existe pas, ce qui est exprimé par l'utilisation du symbole  $\infty$  dans le point position atteint. Choisissons une valeur support  $\sigma = 2$ , le ppos  $[1, \infty, \infty, 1, \infty]$  est donc  $\sigma$ -satisfaisant. En effet, l'inéquation de la définition 5 est vérifiée car  $p_i$  est inférieur à  $\infty$  2 fois pour ( $s_1$  et  $s_4$ ). Si on choisit une valeur de support  $\sigma = 3$ , ce même ppos n'est pas  $\sigma$ -satisfaisant.

**Définition 6.** Soient un ensemble d'*itemsets*  $\mathcal{S} = \{s_1, \dots, s_k\}$  avec  $s_i \subseteq \mathbb{I}$ , une relation d'ordre  $\Re$  sur  $\mathbb{I}$  pour tout  $i \in [1, k]$  et un entier  $\sigma$  représentant le support, on définit l'automate déterministe des *itemsets* fréquents comme étant le quintuple  $FIA_{\theta}(\mathcal{S}) = (\mathcal{Q}, \mathbb{I}, \delta, \mathcal{I}, \mathcal{F})$  où :

$$\mathcal{Q} = Pos(\mathcal{S}), \quad \mathcal{I} = \{q_0^k\}, \quad \delta = PPA_{\mathcal{S}} \quad \text{et} \quad \mathcal{F} = Pos_{\sigma}(\mathcal{S})$$

L'algorithme <sup>1</sup> (voir Vincelas 2007) que nous avons développé, pour des raisons d'optimisation, ne reprend pas pour les états l'information complète du ppos. On en retient des bits position indiquant simplement la présence ou non d'un item dans l'*itemset* considéré. De plus, la taille des bits position générés est variable selon les états et correspond au support de l'item arrivant à cet état évitant du coup de conserver inutilement les symboles  $\infty$  dans les ppos. L'automate est obtenu en ne construisant que les états  $\sigma$ -satisfaisants. Le  $FIA_{\theta}$ , ainsi construit, est émondé et ne requiert en aucun cas une phase d'élagage. Tous les chemins du  $FIA_{\theta}(\mathcal{S})$  partant de l'état initial vers un état  $q$ , sont étiquetés par des *itemsets* appartenant à  $Sub(s_i)$  pour  $i \in [1, k]$ . Le support de l'*itemset* est obtenu en calculant le nombre de valeurs qui ne sont pas égales à  $\infty$  dans le ppos associé à l'état  $q$ .

**Propriété 1.** Soit un ensemble d'*itemsets*  $\mathcal{S} = \{s_1, \dots, s_k\}$  et soit une valeur support  $\sigma$  fixée où  $\sigma = \lceil \theta \times k \rceil$  avec  $\theta \in [0; 1]$  et  $k = |DB|$ , le  $FIA_{\theta}(\mathcal{S})$  reconnaît tous et seulement tous les *itemsets* appartenant à  $Sub(s_i)$  pour  $i \in [1, k]$  qui respectent la contrainte de support. C'est à dire, tous les *itemsets*  $\theta$ -fréquents de  $\mathcal{S}$ .

**Propriété 2.** Soit un ensemble d'*itemsets*  $\mathcal{S} = \{s_1, \dots, s_k\}$ , soit une valeur support  $\sigma$  fixée où  $\sigma = \lceil \theta \times k \rceil$  avec  $\theta \in [0; 1]$  et  $k = |DB|$ , soit un *itemset*  $\mathcal{X}$  appartenant à  $Sub(s_i)$  pour  $i \in [1, k]$  qui vérifie la contrainte de support, alors tous les *itemsets* appartenant à  $Sub(\mathcal{X})$  ayant la même valeur de support que  $\mathcal{X}$  sont reconnus à un même état du  $FIA_{\theta}(\mathcal{S})$ .

En considérant le batch  $B_0^1$  de la figure 1, nous construisons sur la figure 2 le  $FIA_{\theta}$  émondé pour  $\theta = 0.4$  et donc  $\sigma = 2$ . Les états finaux sont repérés par un double cercle et l'état initial est représenté avec une flèche entrante sans label. Remarquons que l'état initial est également un état final car l'*itemset* vide qui fait partie de  $Sub(s_i)$  pour  $i \in [1, k]$  est reconnu seulement à

<sup>1</sup>La présentation détaillée fait l'objet d'une soumission dans l'atelier démonstration logicielle de EGC'08

l'état initial. Par ailleurs, nous observons que les itemsets  $abd$ ,  $ad$  et  $bd$  avec une même valeur de support à 2, sont reconnus à l'état  $q_7$ . Il en est de même pour les itemsets  $bce$  et  $ce$  de support 2 en  $q_{10}$  et pour  $be$  et  $e$  de support 3 en  $q_5$ . Le  $FIA_\theta$ , du fait de la propriété 2 est une structure très compacte. Si on considère comme pour l'algorithme FP-growth les itemsets classés par ordre de fréquence décroissante des items, le FP-tree résultant compte 11 noeuds mais 10 états pour le FIA dans ce cas.

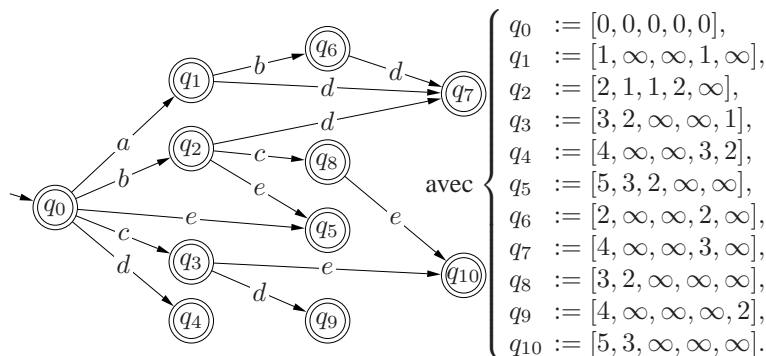


FIG. 2 –  $FIA_{40\%}(\{abcde, bcef, be, abd, cd\})$ .

### 4.3 Le FIA appliqué aux *data streams*

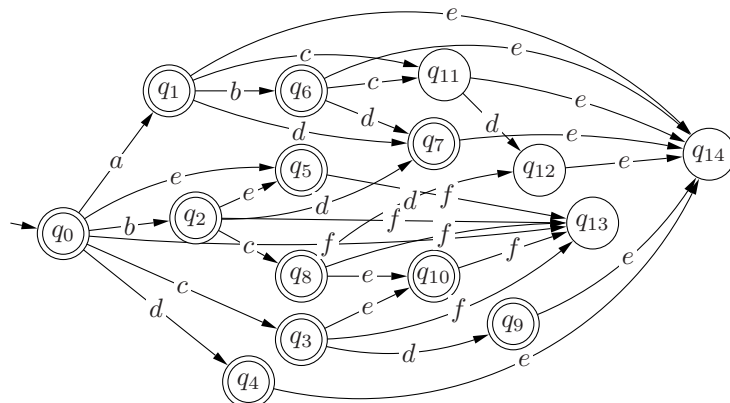
#### 4.3.1 Mise à jour incrémentale du FIA

Hoshino et al. (2000) ont mis en évidence deux propriétés incrémentales du SA concernant l'ajout d'une séquence vide et l'ajout d'un symbole à la dernière position de la dernière séquence ordonnée traitée. Nous reprenons ces deux propriétés appliquées aux itemsets pour effectuer la mise à jour incrémentale du FIA. Notre algorithme<sup>2</sup> de construction et de mise à jour du FIA incrémental (voir Vincelas 2007) est en une-passe sur les données. On prend en compte le nouveau batch, itemset par itemset du premier au dernier et pour chaque itemset, item par item également du premier au dernier. Il convient alors de procéder d'abord à l'extension de dimension des ppos créant ainsi un emplacement dédié à chaque nouvel itemset du batch. Ensuite, pour chaque itemset, il reste à introduire la valeur correspondant à la position de ses sous-itemsets aux emplacements dans les ppos. Afin d'illustrer ces aspects, prenons en compte en plus du batch  $B_0^1$ , le batch  $B_1^2$  de la figure 1 avec  $\theta = 0,4$  et donc  $\sigma = \lceil 0,4 \times 7 \rceil = 3$ . Nous représentons sur la figure 4 le  $FIA_{40\%}$  émondé mis à jour avec la valeur des ppos compte tenu du batch  $B_1^2$ . Les ppos comportent donc deux emplacements supplémentaires, le sixième et le septième réservés respectivement aux itemsets  $s_6 = abc$ ,  $s_7 = abce$  du batch  $B_1^2$ . On a par exemple la valeur 3 dans le ppos  $q_8$  au sixième emplacement pour prendre en compte le sous-itemset  $bc$  faisant partie de  $s_6$ . Il faut souligner que les valeurs aux cinq premiers emplacements demeurent inchangées dans tous les ppos du FIA mis à jour. En considérant le batch  $B_2^3$ , le FIA de la figure 4 demeure inchangé car le sous-itemset  $bce$  de  $s_8$  est déjà reconnue et la prise en compte de l'item  $f$  provoquerait la création d'un état non  $\sigma$ -satisfaisant. Seuls les ppos sont donc mis à jour comme indiqué ci-dessus.

<sup>2</sup>la présentation détaillée fait également l'objet d'une soumission dans l'atelier démonstration logicielle de EGC'08

### 4.3.2 Représentation des bordures statistiques à l'aide du FIA

Appliquée au cas des *data streams* la mise à jour du FIA requiert de connaître l'ensemble des états accessibles, y compris des états non  $\sigma$ -satisfaisants. En effet, mettre à jour le FIA émondé reviendrait à considérer que les itemsets non reconnus par l'automate ont tous un support à 0 ; ce qui engendrerait nécessairement un grand nombre de faux négatifs sur la totalité du *stream*. A l'inverse, en considérant les états non  $\sigma$ -satisfaisants, un grand nombre d'itemsets vrais négatifs serait analysé inutilement. Afin d'illustrer cet aspect, considérons d'une part la représentation du  $FIA_{40\%}$  émondé de la figure 2 et d'autre part la représentation de la figure 3 du  $FIA_{40\%}$  non émondé avec tous ses états accessibles. Les états  $q_{11}, q_{12}, q_{13}, q_{14}$  ne sont pas 2-satisfaisants mais 1-satisfaisants et ne sont donc pas finaux. La question revient à savoir quel est l'automate qu'il convient de considérer pour effectuer la mise à jour du FIA. Il est donc nécessaire de trouver un compromis entre ne conserver aucun état non  $\sigma$ -satisfaisant et tous les états accessibles. Idéalement, seuls les états qui correspondent à des itemsets vrai  $\theta$ -fréquents du *stream* devraient être construits, quand bien même ils ne satisfont pas la contrainte de support. La solution que nous avons adoptée est d'utiliser la bordure statistique supérieure présentée par Symphor et Laur (2006), dans laquelle est maximisé le rappel.



$$\text{avec } \begin{cases} q_0 := [0, 0, 0, 0, 0], & q_5 := [5, 3, 2, \infty, \infty], & q_{10} := [5, 3, \infty, \infty, \infty], \\ q_1 := [1, \infty, \infty, 1, \infty], & q_6 := [2, \infty, \infty, 2, \infty], & q_{11} := [3, \infty, \infty, \infty, \infty], \\ q_2 := [2, 1, 1, 2, \infty], & q_7 := [4, \infty, \infty, 3, \infty], & q_{12} := [4, \infty, \infty, \infty, \infty], \\ q_3 := [3, 2, \infty, \infty, 1], & q_8 := [3, 2, \infty, \infty, \infty], & q_{13} := [\infty, 4, \infty, \infty, \infty], \\ q_4 := [4, \infty, \infty, 3, 2], & q_9 := [4, \infty, \infty, \infty, 2], & q_{14} := [5, \infty, \infty, \infty, \infty]. \end{cases}$$

FIG. 3 –  $FIA_{40\%}(\{abcde, bce, f, be, abd, cd\})$  non émondé.

**Définition 7.**  $\theta'$  est un support statistique pour  $\theta$ ,  $0 < \theta' < 1$ , s'il est utilisé pour approcher les motifs vrais  $\theta$ -fréquents du *stream*.

Le théorème 1 ci-dessous permet d'établir la valeur du support statistique qui permet la construction de la bordure statistique supérieure.

**Théorème 1.**  $\forall k > 0, \forall \theta, 0 \leq \theta \leq 1, \forall \delta, 0 < \delta \leq 1$ , on choisit  $\varepsilon$  tel que :

$$\varepsilon \geq \sqrt{\frac{1}{2k} \ln \frac{|DS|}{\delta}} .$$

Si on fixe  $\theta' = \theta - \varepsilon$  ou  $\theta' = \theta + \varepsilon$  alors on a respectivement le Rappel = 1 ou la Précision = 1 avec une probabilité au moins de  $1 - \delta$ . La valeur  $\delta$  est le risque statistique lié aux data streams fixé par l'utilisateur. Les valeurs  $\theta' = \theta \pm \varepsilon$  sont les supports statistiques au sens de la définition 7.

Les fréquences obtenues ( $\theta' = \theta \pm \varepsilon$ ) sont statistiquement presque optimales. Nous renvoyons aux travaux de Nock et al. (2007) pour la démonstration de ce théorème. Celle-ci repose sur l'utilisation d'inégalités de concentration de variables aléatoires, qui, dans ce cas précis, permettent d'obtenir un résultat statistiquement presque optimal. Par optimalité, nous entendons que toute technique d'estimation obtenant de meilleures bornes est condamnée à se tromper (le critère à maximiser n'est plus égal à un) quel que soit son temps de calcul. Dans le cas de la bordure statistique supérieure correspondant à  $\theta' = \theta - \varepsilon$ , il s'agit de réduire autant que possible le nombre de faux négatifs à savoir les itemsets véritablement fréquents du *stream* et qui ne sont pas retenus comme tels pour la partie observée du *stream*. Lorsque seuls les états de la bordure statistique supérieure ont été conservés, à savoir les états  $\sigma$ -satisfaisants ainsi que les états  $[(\theta - \varepsilon) \times k]$ -satisfaisants, l'automate obtenu après une mise à jour incrémentale est une approximation du  $FIA_\theta$  pour le stream observé (nous le noterons  $\widehat{FIA}_\theta$ ). Toutefois, le théorème 1 permet d'affirmer au risque  $\delta$  que  $\widehat{FIA}_\theta = FIA_\theta$ . De la sorte, on minimise la première source d'erreurs (cf. section 1) avec une forte probabilité  $(1 - \delta)$ . Le langage  $\mathcal{L}(\widehat{FIA}_\theta)$  est le plus petit ensemble possible qui contient tous les itemsets véritablement  $\theta$ -fréquents du *stream* pour sa partie observée. Il n'y a pas de faux négatifs (Rappel = 1) au risque  $\delta$ . Cet ensemble contient également des itemsets faux positifs, c'est à dire  $\theta$ -fréquents pour la partie connue du *stream* mais qui ne le sont peut-être plus sur tout le stream. La figure 4 représente le  $FIA_{40\%}$  émondé mis à jour qui est obtenu à partir du  $FIA_{40\%}$  non émondé de la figure 3 compte tenu du batch  $B_1^2$ . Les ppos des états  $q_7$  et  $q_9$  de la figure 2 après mise à jour, seraient égaux à  $q_7 = [4, \infty, \infty, 3, \infty, \infty, \infty]$ ,  $q_9 = [4, \infty, \infty, \infty, 2, \infty, \infty]$  et ne sont plus  $\sigma$ -satisfaisants. Ces états disparaissent du  $FIA_{40\%}$  émondé mis à jour. Par contre, le point position de l'état  $q_{11}$  vaut après mise à jour  $q_{11} = [3, \infty, \infty, \infty, \infty, 3, 3]$ . Cet état devient  $\sigma$ -satisfaisant et apparaît comme état final du  $FIA_{40\%}$  émondé mis à jour de la figure 4. C'est typiquement un état que nous n'aurions pas obtenu en effectuant la mise à jour à partir du  $FIA_{40\%}$  émondé de la figure 2. En revanche, il a fallu également traiter inutilement les états  $q_{12}$ ,  $q_{13}$ ,  $q_{14}$  alors qu'ils ne sont pas finaux et n'existent pas dans le  $FIA_{40\%}$  émondé mis à jour de la figure 4.

Si on considère, cette fois, seulement les états finaux du  $\widehat{FIA}_\theta$  qui sont  $[(\theta + \varepsilon) \times k]$ -satisfaisants, l'ensemble des itemsets reconnus à ces états représente la bordure statistique inférieure. C'est à dire le plus grand ensemble qui ne contient que des itemsets véritablement  $\theta$ -fréquents du *stream* pour sa partie observée, mais on ne les a pas tous. Il n'y a pas de faux positifs (Précision = 1) au risque  $\delta$ . On minimise ainsi la deuxième source d'erreurs (cf. section 1) avec une forte probabilité  $(1 - \delta)$ .

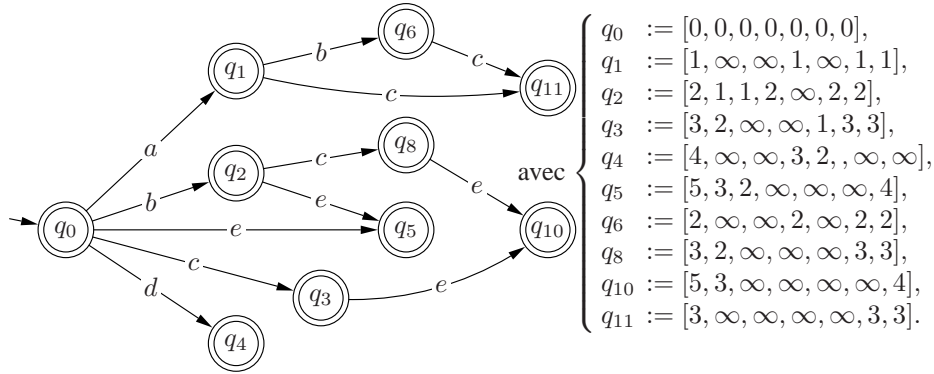


FIG. 4 – FIA<sub>40%</sub>({*abcde, bcef, be, abd, cd, abc, abce*}).

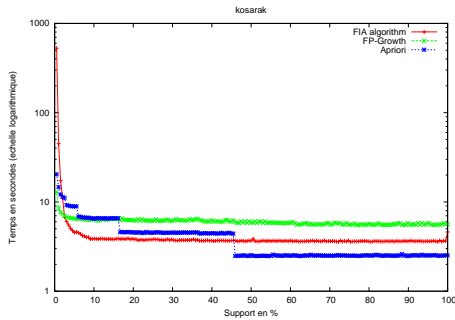


FIG. 5 – Temps requis pour la mise à jour du FIA<sub>θ</sub> avec le jeu d'essai Kosarak.

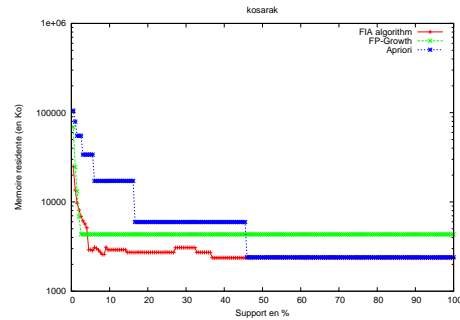
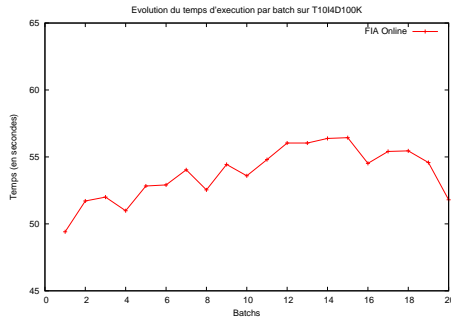


FIG. 6 – Mémoire requise pour la mise à jour du FIA<sub>θ</sub> avec le jeu d'essai Kosarak.

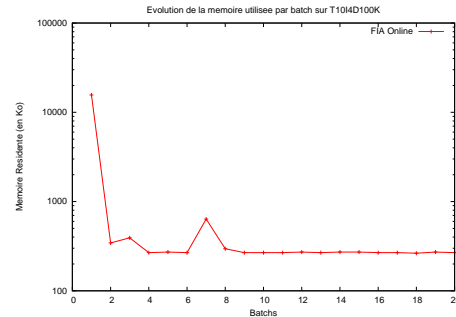
## 5 Expérimentations et comparatifs

Les expérimentations ont été réalisées avec un matériel muni d'un processeur AMD ATHLON 3800<sup>+</sup> (2 × 64) bits, disposant de 1GO de RAM. Les algorithmes ont été écrits à l'aide du langage de programmation C<sup>++</sup>. Les figures 5 et 6 montrent la comparaison entre le temps pris et la mémoire résidente requise en fonction de la fréquence pour les algorithmes FP-Growth, A-priori et du FIA<sup>3</sup> sur le jeu de données Kosarak. Les résultats obtenus avec le FIA sont meilleurs sur une large plage de fréquence tant pour le temps pris que pour la mémoire requise. En outre, lors de la construction des courbes, on note effectivement un écart d'un ordre de grandeur de 10<sup>5</sup> en moins pour le FIA entre le nombre de noeuds du FP-tree par rapport aux nombres d'états du FIA. Sur les figures 7 et 8, nous illustrons les résultats obtenus avec l'algorithme du FIA incrémental en représentant le temps pris et la mémoire requise en fonction de l'insertion de nouveaux batches pour le jeu de données T10I4D100K. Le temps pris et

<sup>3</sup>Comme pour FP-growth, l'algorithme du FIA effectue un premier passage sur les données pour retenir l'ordre des items par ordre de fréquence décroissante



**FIG. 7** – Temps requis pour la mise à jour incrémentale du  $FIA_{\theta}$  avec le jeu d’essai  $T10I4D100K$ .



**FIG. 8** – Mémoire requise pour la mise à jour incrémentale du  $FIA_{\theta}$  avec le jeu d’essai  $T10I4D100K$ .

la mémoire consommée demeurent stables. Cela montre l’applicabilité de l’algorithme du FIA incrémental dans le cas des *data streams*.

## 6 Conclusion

Dans cet article, nous apportons une contribution originale en élaborant un nouvel automate : le FIA qui permet de traiter de façon efficace la problématique de l’extraction des itemsets fréquents dans les *data streams*. A notre connaissance, les automates en tant que structure de données n’ont pas du tout été utilisés pour aborder cette question. Nous montrons que le FIA est une structure très compacte et informative car plusieurs itemsets fréquents ayant la même valeur de support sont reconnus à un même état. Par ailleurs, la structure indexe directement tous les itemsets fréquents sans qu’il soit nécessaire de lui associer un tableau pour finalement obtenir les résultats. Le FIA présente également des propriétés incrémentales qui facilitent grandement la mise à jour dans le cas des *data streams* avec une granularité très fine par batch. Utilisé dans le cadre d’une approche prédictive, le FIA permet d’indexer les itemsets véritablement  $\theta$ -fréquents du *stream* en maximisant soit le rappel soit la précision. L’algorithme développé, pour mettre à jour le FIA, ne requiert qu’un seul passage sur les données qui sont prises en compte par batch, itemset par itemset et pour chaque itemset, item par item. Les expérimentations, avec une analyse en temps de calcul et en mémoire consommée, donnent des résultats satisfaisants qui prouvent l’applicabilité et le passage à l’échelle de l’algorithme. Notre contribution ouvre donc une voie prometteuse avec le FIA, quant à l’utilisation de nouvelles structures de données de type automate, dans les problématiques d’extraction de motifs fréquents dans les *data streams*.

## Références

- [1] AGRAWAL, R., SRIKANT, R. : Fast Algorithms for Mining Association Rules . Proc. 20th Int. Conf. Very Large Data Bases, VLDB (1994) 487–499

- [2] CHANG, J., LEE, W. : Decaying obsolete information in finding recent frequent itemsets over data streams. IEICE Transaction on Information and Systems vol. E87-D N°6 (2004)
- [3] CHANG, J., LEE, W. : A sliding window method for finding recently frequent itemsets over online data streams. Journal of Information Science and Engineering, vol. 20 n°4 (2004)
- [4] GIANNELLA, C., HAN, J. PEI, J., YAN, X., YU, P.-S. : Mining frequent patterns in data streams at multiple time granularities. In Proc. of the NFS workshop on next generation Data mining (2002)
- [5] HAN, J., PEI, J. YIN, Y. : Mining frequent patterns without candidate generation ACM SIGMOD Intl. Conference on Management of Data (2000) 1–12
- [6] HOPCROFT, J.E., ULLMAN, J.D. : Introduction to Automata Theory, Languages, and Computation. ADDISON-WESLEY Longman Publishing Co., Inc., Boston, USA (1990)
- [7] HOSHINO, H., SHINOHARA, A., TAKEDA, M., ARIKAWA, S. : Online Construction of Subsequence Automata for Multiple Texts. In : Proceedings of the 7<sup>th</sup> International Symposium on String Processing and Information Retrieval. (2000) 146–152
- [8] LI, H.-F., LEE, S.-Y., SHAN, M.-K. : An efficient algorithm for mining frequent itemsets over the entire history of data streams. In Proc. of the first Int. Workshop on Knowledge Discovery in Data Streams. (2004)
- [9] MANKU, G., MOTWANI, R. : Approximate frequency counts over data streams In Proc. of the 28<sup>th</sup> International Conference on VLDB . (2002)
- [10] MASSÉGLIA, F., PONCELET, P., TEISSEIRE, M. : Incremental mining of sequential patterns in large databases Data and Knowledge Engineering 46. (2003)
- [11] NOCK, R., LAUR, P.-A., SYMPHOR, J.-E., PONCELET P. : Mining evolving data streams for frequent patterns. Pattern Recognition 40 492–503 (2007)
- [12] SYMPHOR, J.-E., LAUR, P.-A. : Bordures statistiques pour la fouille incrémentale de données dans les data streams. (EGC'2006) volume II 615–626. (2006)
- [13] TRONÍČEK, Z. : Common Subsequence Automaton. In proceedings of the 7<sup>th</sup> International Conference on Implementation and Application of Automata. Number 2608 in (LNCS), Tours, France, SPRINGER-VERLAG (2002) 270–275
- [14] VAPNIK, V. : Statistical learning theory. John Wiley (1998)
- [15] VINCESLAS, L. : Structure d'indexation d'itemsets fréquents. Mémoire de recherche Master 2 ST Université des Antilles et de la Guyane (2007)

## Summary

We present in this paper a new automaton : the FIA which allows to mine efficiently all frequent itemsets in a *data stream*. The FIA is a summary and very informative data structure with incremental properties that make the update processing easier with fine granularity. Our incremental algorithm for updating the FIA only needs one scan over the data which are considered per batch, itemset per itemset and for each itemset, item per item. Used within the framework of a predictive approach, the FIA recognizes the truly frequent itemsets of the *stream*, by maximizing either the recall or the precision.