

Le FIA : un nouvel automate permettant l'extraction efficace d'*itemsets* fréquents dans les *flots de données*

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 (*Frequent Itemset Automaton*) pour traiter de façon efficace la problématique de l'extraction des *itemsets* fréquents dans les *flots de données*. Le FIA est une structure de données très compacte et informative qui présente également des propriétés incrémentales intéressantes pour les mises 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 tout d'abord par batch, i.e. *itemset* par *itemset*, puis pour chaque *itemset*, *item* par *item*. Nous montrons que dans le cadre d'une approche prédictive et par l'intermédiaire de la bordure statistique, le FIA permet d'indexer les *itemsets* véritablement fréquents du *flot* en maximisant le rappel et en fournissant à tout moment une information sur la pertinence statistique des *itemsets* indexés avec la *P*-valeur.

1 Introduction

L'extraction d'*itemsets* fréquents est une problématique de recherche qui intéresse la communauté fouille de données depuis plus d'une dizaine d'années et intervient 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 suivis en ce sens par Han et al. (2000). Traditionnellement, les différents algorithmes proposés dans la littérature reposent sur des structures de données de type arbre ou encore treillis (e.g. A-priori (Agrawal et Srikant 1994), FP-Growth (Han et al. 2000), ...). La problématique de recherche de motifs (i.e. une généralisation des *itemsets*) apparaît dans des domaines aussi variés que la bioinformatique ou la fouille de textes. En ce qui concerne ce dernier, de nouvelles structures de données, basées sur des automates ont fait leur apparition. Par exemple, 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. L'un des problèmes principaux auquel doit faire face une approche d'extraction de motifs est de disposer

de structures qui soient suffisamment compactes et informatives afin de minimiser l'explosion combinatoire liée à d'importants espaces de recherche. En effet, l'applicabilité des algorithmes proposés 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 la question suivante : est-il possible de trouver de nouvelles structures de données, suffisamment informatives et compactes, pour extraire de façon efficace les *itemsets* fréquents ?

Récemment, pour faire face au fait que les données peuvent être disponibles sous la forme de flots qui arrivent de manière continue et sont éventuellement en quantité infinies, les chercheurs de la communauté fouille de données se sont intéressés à l'extraction de connaissance dans de telles conditions. De nombreuses applications (e.g. transactions financières, navigation sur le Web, téléphonie mobile, ...) rentrent dans ce cadre et nécessitent d'obtenir des résultats rapidement. Dans le cas de la problématique d'extraction d'*itemsets* fréquents, la prise en compte de données disponibles sous la forme de flots engendre de nouvelles problématiques. En premier lieu, il est indispensable de considérer les aspects liés à la mise à jour. En effet, étant donné que les données arrivent de manière continue, la base de données est sujette à des mises à jour régulières et fréquentes. Ainsi, la connaissance obtenue pour une base à un moment donné ne sont plus forcément valables lorsque de nouvelles données arrivent et il n'est pas envisageable de relancer l'algorithme sur toute la base mise à jour. La maintenance de connaissance incrémentale a, par exemple, été étudiée dans Massegli et al. (2003) où les auteurs proposent de maintenir au fur et à mesure des mises à jour successives. Malheureusement, tous les travaux de recherche basés sur une approche incrémentale ne sont pas adaptés aux flots dans la mesure où nous ne pouvons pas disposer de la base dans son intégralité. Il est donc nécessaire, en second lieu, de disposer de nouveaux algorithmes qui effectuent une seule passe sur la base, i.e. des algorithmes *une-passe*. Les travaux récents sur les flots ont montré qu'il n'était plus envisageable d'obtenir une réponse exacte (i.e. les *itemsets* réellement fréquents sur le flot) et qu'il fallait accepter une approximation quant à l'estimation de la fréquence des motifs : les *itemsets* obtenus ne sont en fait que des *itemsets* fréquents observés. Il faut donc prendre en compte l'incertitude engendrée par la connaissance toujours incomplète du *flot de données*. Il est important de noter que dans les flots, 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. Dans Vapnik (1998), l'auteur a montré qu'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 flot. On peut toutefois chercher à minimiser l'une des sources d'erreurs tout en maintenant l'autre en dessous d'un seuil raisonnable. La seconde question à laquelle nous nous intéressons dans cet article est la suivante : est-il possible de trouver une structure de données ayant des propriétés incrémentales satisfaisantes et qui permette de construire et de maintenir de façon efficace l'ensemble des *itemsets* fréquents du flot de données tout en minimisant l'une ou l'autre des sources d'erreurs ?

La suite de l'article est organisée de la manière suivante. La section 2 présente plus formellement la problématique. Dans la section 3, nous proposons 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 utilisés 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 θ -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 σ dans DB . Cette problématique, étendue au cas des flots de données, peut s'exprimer comme suit. Soit un flot de données $DS = B_{a_i}^{b_i}, B_{a_{i+1}}^{b_{i+1}}, \dots, B_{a_n}^{b_n}$, 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 flot de données ($|DS|$), 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 flot de données 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 flot.

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 antérieurs

Les différents travaux portant sur la problématique d'extraction d'itemsets fréquents dans les flots de données se déclinent selon trois axes en fonction du modèle de traitement des itemsets du flot. Le premier utilise des fenêtres à point fixe où sont conservés tous les itemsets acquis du flot (e.g. Manku, Motwani 2002, 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 (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 au calcul de la fréquence des itemsets. Par exemple, Gia-

nella 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 l'on ne considère plus seulement l'acquisition mais aussi l'enlèvement d'itemsets comme dans les travaux de Chang et Lee (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 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'il 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 (Frequent Itemset Automaton), 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 étendons cette approche à la prise en compte des flots de données et nous montrons comment mettre à jour incrémentalement le FIA lors de l'ajout de nouveaux batches issus du flot. Cependant, pour tenir compte de l'incertitude engendrée par la connaissance toujours incomplète du flot, nous étudions la représentation de la bordure statistique à l'aide du FIA afin de développer une approche prédictive Nock (2007). En effet, plutôt que d'extraire des itemsets observés fréquents sur la partie connue du flot, nous considérons qu'il est préférable de prédire les itemsets véritablement fréquents sur tout le flot à partir des itemsets connus.

4.1 Rappels sur la théorie des automates

Nous présentons dans cette section, les principes fondamentaux de la théorie sur les automates finis e.g. Hopcroft et Ullman (1990) qui seront utilisés dans la suite.

Définition 1. *Un automate à états finis \mathcal{A} est un quintuple tel que $\mathcal{A} = (Q, \Sigma, \delta, \mathcal{I}, \mathcal{F})$, où Q est un ensemble fini d'états, Σ un alphabet, $\delta \subseteq Q \times \Sigma \times Q$ est un ensemble de transitions, $\mathcal{I} \subseteq Q$ et respectivement $\mathcal{F} \subseteq Q$ 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 α . 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 $\forall (p, \alpha, q), (p, \alpha, q') \in \delta \Rightarrow q = q'$.*

Nous adaptons les définitions 3 et 4 proposées initialement par Hoshino et al. (2000) pour l'automate des sous-séquences (SA), 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 ε se trouvant devant le premier item de \tilde{s}_i . Si $p_i = \infty$, cette autre position correspond à celle d'un itemset vide ε 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_∞^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_θ qui intègre la notion de fréquence et qui 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 σ 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 σ si et seulement si*

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

Automate des Itemsets Fréquents : le FIA

Ainsi, le ppos $[p_1, \dots, p_k]$ est dit σ -satisfaisant. L'ensemble de tous les ppos qui sont σ -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 que $PPA_{\mathcal{S}}([0, \dots, 0], a) = [1, \infty, \infty, 1, \infty]$ (cf. Définition 4). Nous représentons schématiquement ci-dessous à titre d'exemple le point position atteint par a depuis le point position initial.

$$\begin{array}{cccccc} a & b & c & d & e & & b & c & e & f & & b & e & & a & b & d & & c & d & & \\ 0 & 1 & 2 & 3 & 4 & 5 & \infty & 0 & 1 & 2 & 3 & 4 & \infty & 0 & 1 & 2 & \infty & 0 & 1 & 2 & 3 & \infty & 0 & 1 & 2 & \infty \\ \uparrow & \uparrow & & & & & & \uparrow & & & & \uparrow & & \uparrow & \uparrow & & & \\ p_1 & p'_1 & & & & & & p_2 & & & & p'_2 & p_3 & p'_3 & p_4 & p'_4 & & p_5 & p'_5 & & & & & & \end{array}$$

Cela illustre pour les itemsets s_1 et s_4 , que l'item a est à la première position donc numéroté 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 ∞ dans le point position atteint. Choisissons une valeur support $\sigma = 2$, le ppos $[1, \infty, \infty, 1, \infty]$ est donc σ -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 σ -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 \mathfrak{R} sur \mathbb{I} pour tout $i \in [1, k]$ et un entier σ 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'automate est obtenu en ne construisant que les états σ -satisfaisants. Ainsi, Le FIA_θ 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 ∞ 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 σ 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 θ -fréquents de \mathcal{S} .

Propriété 2. Soit un ensemble d'itemsets $\mathcal{S} = \{s_1, \dots, s_k\}$, soit une valeur support σ 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 montrons, sur la figure 2, le FIA_θ 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 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_θ , du fait de la propriété 2 est une structure très compacte. Si l'on choisit la fréquence décroissante comme relation d'ordre \mathfrak{R} sur les items, à l'instar de l'algorithme `FP-growth`, le `FP-tree` résultant compte 11 noeuds mais 10 états pour le FIA dans ce cas.

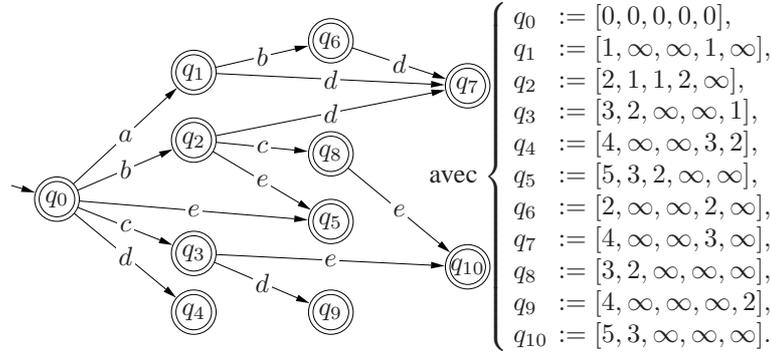


FIG. 2 – $FIA_{40\%}(\{abcde, bcef, be, abd, cd\})$.

4.3 Le FIA appliqué aux flots de données

4.3.1 Mise à jour incrémentale du FIA

Hoshino et al. (2000) ont exploité 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 algorithme¹ de construction et de mise à jour du FIA incrémental n'effectue qu'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. 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 . 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 reconnu et la prise en compte de l'item f provoquerait la création d'un état non σ -satisfaisant.

4.3.2 Représentation des bordures statistiques à l'aide du FIA

Appliquée au cas des flots de données la mise à jour du FIA requiert de connaître l'ensemble des états accessibles, y compris des états non σ -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 flot. À l'inverse, en considérant les états non σ -satisfaisants, un grand nombre d'itemsets vrais négatifs seraient analysés 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 σ -satisfaisant et tous les états accessibles. Idéalement, seuls les états qui correspondent à des itemsets vrai θ -fréquents du flot devraient être construits, quand bien même ils ne satisfont pas la contrainte de support à un instant donné. La solution que nous avons adoptée est d'utiliser la bordure statistique

¹La présentation détaillée fait l'objet d'une soumission dans l'atelier démonstration logicielle de EGC'08.

supérieure présentée par Symphor et Laur (2006), dans laquelle est maximisé le rappel (cf. Définition 7 et Théorème 1 ci-après).

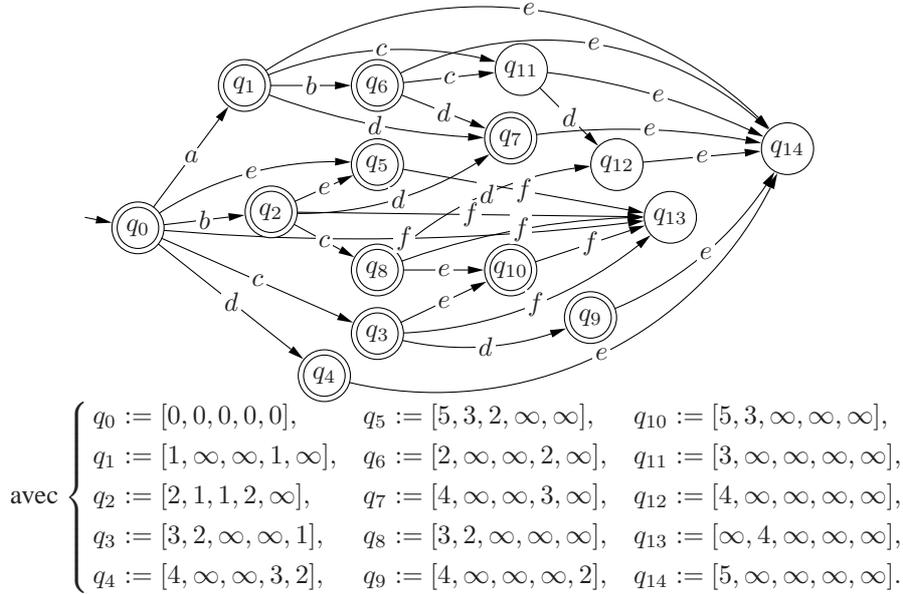


FIG. 3 – FIA_{40%}({*abcde, bcef, be, abd, cd*}) non émondé.

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

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

Théorème 1. $\forall k > 0, \forall \theta, 0 < \theta \leq 1, \forall \delta, 0 < \delta \leq 1$, on choisit ε 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 δ est le risque statistique lié aux flots de données 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) quelque 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 flot et qui ne sont pas retenus comme tels pour la partie observée du flot. Lorsque seuls les états de la bordure statistique supérieure ont été conservés, à savoir les états σ -satisfaisants ainsi que les états $\lfloor (\theta - \varepsilon) \times k \rfloor$ -satisfaisants, l'automate obtenu après une mise à jour incrémentale est une approximation du FIA_θ pour le flot observé (nous le noterons \widehat{FIA}_θ). Toutefois, le Théorème 1 permet d'affirmer au risque δ 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 θ -fréquents du flot pour sa partie observée. Il n'y a pas de faux négatifs ($Rappel = 1$) au risque δ . Cet ensemble contient également des itemsets faux positifs, c'est à dire θ -fréquents pour la partie connue du flot mais qui ne le sont peut-être plus sur tout le flot. 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 σ -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 σ -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.

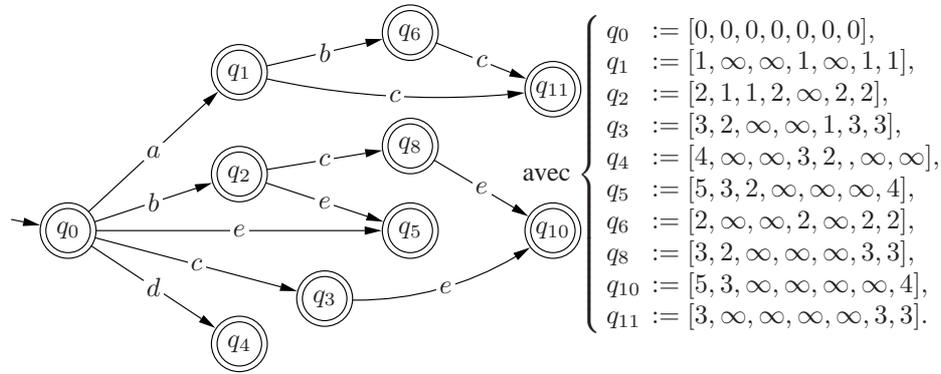


FIG. 4 – $FIA_{40\%}(\{abcde, bcef, be, abd, cd, abc, abce\})$.

Par ailleurs, le Théorème 1 permet d'établir la propriété suivante :

Propriété 3. *Étant donné un itemset i de fréquence observée $\theta' > \theta$, alors la probabilité qu'il existe un itemset de fréquence observée θ' qui ne soit pas θ -fréquent sur la totalité du flot est inférieure à δ avec*

$$\delta = \frac{|DS|}{e^{2k(\theta' - \theta)^2}}$$

Cette probabilité est également appelée la P-valeur de l'itemset i .

La mesure de la P -valeur a été étudiée notamment par Denise et al. (2001). La propriété 3 sur la P -valeur traduit littéralement que la probabilité qu'il existe un faux positif, *i.e.* un itemset de fréquence observée θ' qui ne soit pas θ -fréquent sur tout le flot, est inférieure à δ .

5 Expérimentations

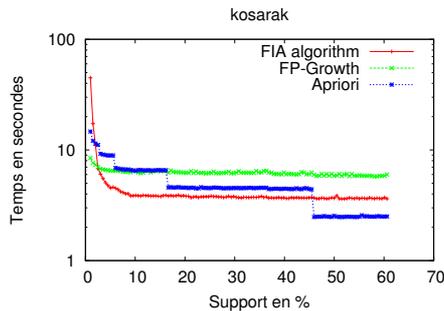


FIG. 5 – Temps requis pour la mise à jour du FIA_{θ} avec le jeu d'essai Kosarak.

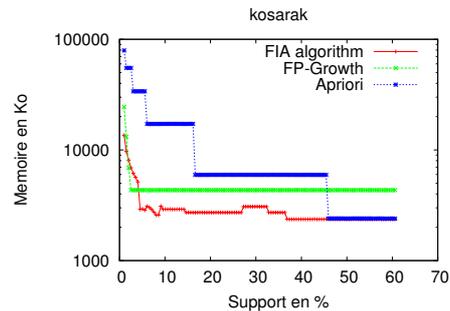


FIG. 6 – Mémoire requise pour la mise à jour du FIA_{θ} avec le jeu d'essai Kosarak.

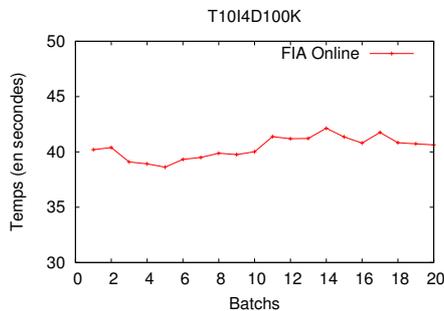


FIG. 7 – Temps requis pour la mise à jour incrémentale du FIA_{θ} avec $\theta = 1\%$ pour le jeu d'essai T10I4D100K.

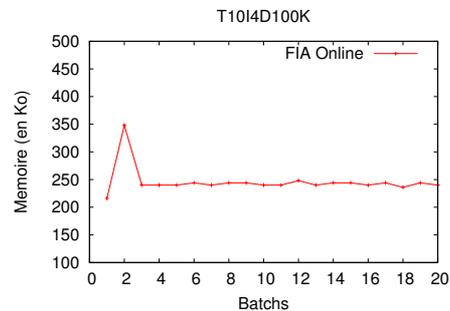


FIG. 8 – Mémoire requise pour la mise à jour incrémentale du FIA_{θ} avec $\theta = 1\%$ pour le jeu d'essai T10I4D100K.

Les expérimentations ont été réalisées sur les jeux de données² Kosarak et T10I4D100K, sur une machine munie d'un processeur AMD ATHLON 3800+(2 × 64) bits disposant de 1Go de RAM. Les algorithmes ont été écrits à l'aide du langage de programmation C++. Les figures 5 et 6 montrent la comparaison entre le temps pris et la mémoire résidente requise en fonction de la fréquence pour les algorithmes³ FP-Growth, A-priori et du FIA sur Kosarak. Notre algorithme de construction du FIA est en une passe, mais pour la comparaison avec

²disponibles à l'URL : <http://fimi.cs.helsinki.fi/data>

³Il s'agit de versions optimisées disponibles à l'URL : <http://www.adrem.ua.ac.be/~goethals/>

notamment FP-Growth, nous avons utilisé une version deux-passes avec un premier passage pour retenir l'ordre des items par ordre de fréquence décroissante. Les résultats obtenus avec le FIA sont meilleurs sur une large plage de fréquence tant pour le temps pris que pour la mémoire requise. On observe effectivement, au-delà des valeurs de fréquence à 5%, un écart en temps de l'ordre de 3sec. et de mémoire de 1 MO en faveur du FIA, qui prend un temps total de 4sec. et consomme 3 MO, par rapport à FP-Growth. Toutefois, les performances du FIA sont dépendantes des données indexées. En effet, cette structure est d'autant plus avantageuse que les itemsets fréquents sont grands. En contrepartie, lorsque les itemsets fréquents sont, majoritairement des singletons, ce qui est le cas pour de très faibles fréquences, le FIA tend à ressembler à un arbre lexicographique et l'algorithme de construction devient inadapté au regard des méthodes classiques de construction de tels arbres. Le FIA est une structure d'autant plus efficace que les données sont denses en informations à extraire. 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 (nombre constant de transactions) pour T10I4D100K. Le temps pris et la mémoire consommée demeurent stables. Cela montre l'applicabilité de l'algorithme du FIA incrémental dans le cas des flots de données.

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 flots de données. À 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 flots de données avec une granularité très fine par batch. Utilisé dans le cadre d'une approche prédictive, le FIA permet d'indexer les itemsets véritablement θ -fréquents du flot en maximisant le rappel et en fournissant à tout moment une information sur leur pertinence statistique avec la P -valeur. Ces deux avantages permettent notamment de construire le FIA par incrément à partir d'une base initiale vide. 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 flots de données.

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] DENISE, A., RÉGNIER, M. VANDENBOGAERT, M. : Assessing the statistical significance of overrepresented oligonucleotides. (LNCS), vol 2149 pages 85-97 (2001)
- [5] 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 DM (2002)
- [6] HAN, J., PEI, J. YIN, Y. : Mining frequent patterns without candidate generation ACM SIGMOD Intl. Conference on Management of Data (2000) 1–12
- [7] HOPCROFT, J.E., ULLMAN, J.D. : Introduction to Automata Theory, Languages, and Computation. ADDISON-WESLEY Longman Publishing Co., Inc., Boston, USA (1990)
- [8] HOSHINO, H., SHINOHARA, A., TAKEDA, M., ARIKAWA, S. : Online Construction of Subsequence Automata for Multiple Texts. In : Proceedings of the 7th International Symposium on String Processing and Information Retrieval. (2000) 146–152
- [9] 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)
- [10] MANKU, G., MOTWANI, R. : Approximate frequency counts over data streams In Proc. of the 28th International Conference on VLDB . (2002)
- [11] MASSÉGLIA, F., PONCELET, P., TEISSEIRE, M. : Incremental mining of sequential patterns in large databases Data and Knowledge Engineering 46. (2003)
- [12] NOCK, R., LAUR, P.-A., SYMPHOR, J.-E., PONCELET P. : Mining evolving data streams for frequent patterns. Pattern Recognition 40 492–503 (2007)
- [13] 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)
- [14] TRONÍČEK, Z. : Common Subsequence Automaton. In proceedings of the 7th International CIAA. Number 2608 in (LNCS), Tours, France, SPRINGER-VERLAG (2002) 270–275
- [15] VAPNIK, V. : Statistical learning theory. John Wiley (1998)

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 and through the statistical border, the FIA recognizes the truly frequent itemsets of the *stream*, by maximizing the recall and by providing, for each information on the statistical relevance of the indexed itemsets, their *P*-value.