Une approche pour l'extraction des itemsets (fermés) fréquents

A. Bouchahda — S. Ben Yahia — Y. Slimani

Université de Tunis El Manar, Faculté des sciences de Tunis
Département des Sciences de l'Informatique
Tunis,
Tunisia
ahlem_bouchahda @yahoo.fr, {sadok.benyahia,yahya.slimani }@fst.rnu.tn

RÉSUMÉ. Il est connu que les performances des algorithmes d'extraction des itemsets fermés laissent à désirer lorsqu'ils sont appliqués sur des contextes éparses. Aussi, dans ce papier, nous proposons une approche qui s'inscrit dans l'optique d'appliquer au type du contexte, selon sa densité (dense ou éparse), le type d'algorithme le plus approprié du point de vue performances. L'approche proposée utilise la technique "diviser-pour-générer" pour diviser récursivement le contexte d'extraction en sous-contextes, sur lesquels l'algorithme approprié sera appliqué. Un certain nombre de résultats d'expérimentations, menés sur des benchmarks typiques, montrent que les performances sont améliorées.

ABSTRACT. Extracting "condensed" patterns is grasping the interest of the Data Mining community. In fact, avoiding the extraction of an overwhelming knowledge is of primary importance as it guaranties extra value knowledge, usefulness and reliability. Extracting frequent closed itemsets seems to be a promising approach to provide a faithful nuclei of condensed knowledge. As a drawback, frequent closed itemset extraction algorithms behave badly on sparse datasets, on the contrary of frequent itemset extraction algorithms. In this paper, we introduce a novel approach aiming to apply the appropriate extraction approach depending of extraction context density. The hybrid approach, based on the "Divide-and-conquer" technique, recursively divides the extraction context into sub-contexts, on which the appropriate frequent (closed) itemset extraction process is applied. Results of the experiments carried out on real-life datasets have shown that this approach performances outperform those of the traditional approaches.

MOTS-CLÉS : Fouille de données, Règles associatives, Itemsets fermés fréquents, Trie, Bases denses, Bases éparses, Performance

KEYWORDS : Data Mining, Association rules, Closed frequent itemsets, Trie, Dense datasets, Sparse datasets, Performance

1. Introduction

Les deux dernières décennies ont renforcé le fait que notre capacité à collecter des données est inversement proportionnelle à celle d'en extraire des connaissances à valeur ajoutée intéressante. Pour faire face à ce défi, une panoplie d'algorithmes d'extraction de motifs à partir de ces gisements de données ont été développés. L'idée est de permettre un passage à l'échelle en effectuant le minimum de calcul pour extraire **tous** les motifs intéressants. Cette économie dans l'effort s'est matérialisée par deux aspects complémentaires : (*i*) De plus en plus d'algorithmes se proposent d'adopter une structure de données avancée où le contexte d'extraction peut être compressé (en fusionnant les objets se partageant une description (*e.g.*, des items)) pour arriver à achever le processus d'extraction de motifs (*e.g.* structures FP-tree [1], CATS [2]) ; (*ii*) La proposition d'algorithmes pour l'extraction de motifs fermés [3]. L'extraction de ces motifs, générateurs du reste des autres motifs, a permis d'outrepasser les algorithmes basés sur APRIORI, qui traitent essentiellement des contextes où les éléments sont fortement corrélés. Cependant, leurs performances laissent à désirer quand ils sont appliqués sur des contextes épars.
Dans ce papier, nous proposons une nouvelle approche qui consiste à appliquer la bonne approche d'extraction en fonction du type de contexte de données (dense ou épar). Ainsi, en utilisant la technique d'extraction "Diviser-pour-générer", le contexte d'extraction est divisé récursivement en sous-contextes. À chaque sous-contexte l'approche d'extraction appropriée est appliquée selon l'estimation de la densité. Ne disposant pas d'un moyen non coûteux pour l'estimation de la densité, nous avons procédé par des approximations empiriques. Aussi, nous proposons deux algorithmes appelés, respectivement, FELINE-F et FELINE-H. Le premier est une extension de l'algorithme FELINE [2], qui exploite la structure de données CATS, pour l'extraction des motifs fermés. Le deuxième, à savoir l'algorithme FELINE-H, implémente une approche pour l'extraction de motifs fermés fréquents et/ou motifs fréquents.

L'article est organisé comme suit. La deuxième section présente la structure de données CATS ainsi que l'algorithme FELINE. Dans la troisième section, nous introduisons l'algorithme FELINE-F pour l'extraction des motifs ou itemsets fermés fréquents. La quatrième section est dédiée à la présentation de l'approche proposée via l'algorithme FELINE-H. Les résultats des expérimentations montrant l'utilité de l'approche proposée sont présentés dans la cinquième section. La conclusion et les travaux futurs font l'objet de la sixième section.

2. Structure de données CATS et l'algorithme FELINE

2.1. L'arbre CATS

L'arbre CATS est un arbre préfixé contenant tous les composants d'un arbre FP-TREE [4]. À chaque item correspond un nœud dans la table entête, auquel on associe la fréquence totale de cet item dans la base de transactions. De plus, à chaque nœud de l'entête est associé un pointeur vers le premier nœud dans l'arbre CATS portant le même label. Chaque nœud de l'arbre CATS contient le label de l'item, sa fréquence, un pointeur vers son père, des pointeurs vers ses fils, un pointeur vers le nœud qui le précède et qui porte le même label que lui, ainsi qu'un pointeur vers le nœud successeur portant le même label que lui. Les fils d'un nœud dans un arbre CATS sont rangés par ordre décroissant de leurs fréquences. Les chemins partant de la racine vers les feuilles représentent l'ensemble des transactions de la base de transactions [2]. La figure 1(Droite) représente un arbre CATS relatif à la base de transactions donnée par la figure 1(Gauche).

TID	ITEMS
1	F,A,C,D,G,I,M,P
2	A,B,C,F,L,M,O
3	B,F,H,J,O
4	B,C,K,S,P
5	A,F,C,E,L,P,M,N

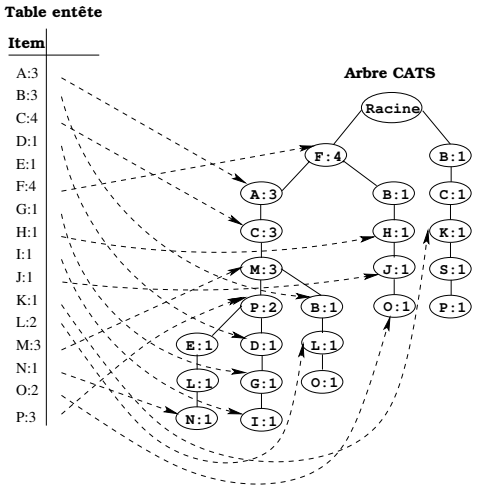

Figure 1. *(**Gauche**) Base de transactions. (**Droite**) Arbre* CATS *associé.*

2.2. L'algorithme FELINE

Contrairement à l'arbre FP-TREE qui ne représente que les items fréquents d'une base de transactions, l'arbre CATS représente tous les items indépendamment de leurs fréquences. Par conséquent, une fois l'arbre CATS construit, nous pouvons conduire plusieurs processus d'extraction d'itemsets fréquents avec des valeurs différentes de support sans avoir à reconstruire pour chaque valeur la structure d'arbre correspondante (cas de l'arbre FP-TREE). D'une façon similaire à l'algorithme FP-GROWTH, l'algorithme FELINE utilise une méthode basée sur l'approche *"Diviser-pour-générer"* afin de générer les itemsets fréquents en minimisant la génération d'itemsets candidats.

Pour un itemset p, l'arbre CATS conditionnel associé est un arbre CATS construit à partir de toutes les transactions contenant l'itemset p. Un arbre CATS *condensé* est un arbre CATS où tous les items non fréquents sont supprimés [2]. Comme pour le cas de FP-TREE, seuls les items fréquents possèdent des nœuds dans l'arbre CATS condensé. En revanche, les items au sein d'une branche d'un arbre CATS condensé sont rangés en se basant sur les fréquences locales des items au sein de la branche, au lieu des fréquences globales utilisées dans FP-TREE. Cependant, bien qu'un arbre CATS conditionnel condensé soit très similaire à un FP-TREE conditionnel, l'algorithme FP-GROWTH ne peut pas y être appliqué directement [2].

3. L'algorithme FELINE-F

3.1. Motivations

Notre objectif consiste à extraire les itemsets fermés fréquents à partir d'un arbre CATS. Pour cela, nous allons utiliser l'approche *"Diviser-pour-générer"* et la stratégie de recherche *en profondeur d'abord*. À la différence de l'extraction des itemsets fréquents, durant le processus d'extraction des itemsets fermés, il peut y avoir des itemsets préfixes qui ne sont pas susceptibles de produire des itemsets fermés fréquents. Ainsi, ils ne peuvent pas être utilisés pour étendre des itemsets fermés et nous devons ainsi les détecter et les éliminer. Pour élaguer l'espace de recherche et accélérer da-

vantage le processus d'extraction des itemsets fermés fréquents, nous avons adopté les heuristiques suivantes :

– **Élagage des arbres conditionnels condensés CATS :** Avant de construire l'arbre conditionnel condensé CATS relatif à un item fréquent, nous allons vérifier en premier lieu si l'item correspondant existe dans l'arbre des itemsets fermés fréquents. Si tel est le cas, nous ne construisons pas cet arbre car tous les itemsets fermés fréquents que nous allons extraire ont été déjà extraits et nous passons à l'item fréquent suivant dans la liste des items fréquents. Ainsi, nous économisons le temps de construction de l'arbre conditionnel condensé CATS et le temps de son exploration.

– **Élagage des sous-itemsets :** Soit *X* l'itemset fréquent courant à considérer. Si *X* est un sous-ensemble d'un itemset fermé fréquent déjà découvert *Y* et $sup(\mathrm{X}) = sup(\mathrm{Y})$, alors *X* et tous ses descendants ne peuvent pas être des itemsets fermés fréquents et par conséquent ils peuvent être élagués.

Puisque FELINE-F est un algorithme qui suit l'approche d'exploration de l'espace de recherche *en profondeur d'abord*, alors un itemset fréquent ne peut être un sous-ensemble que d'un itemset fermé fréquent déjà découvert. Dans cet algorithme, nous allons utiliser une structure de données globale à savoir l'arbre des itemsets fermés fréquents, appelé CFI-TREE (*Closed Frequent Itemset Tree*) (ou arbre CFI), pour stocker les itemsets fermés fréquents ayant été découverts avec leurs supports respectifs et tester la fermeture des itemsets trouvés. Cette structure est une autre variante de l'arbre FP-TREE introduite dans [5]. Dans un arbre CFI, tous les nœuds portant le même label sont liés ensemble de façon similaire à l'arbre FP-TREE. L'insertion d'un itemset fermé fréquent dans un arbre CFI est similaire à l'insertion d'une transaction dans un arbre FP-TREE, sauf que le support d'un nœud n'est pas incrémenté mais remplacé par un support supérieur, *i.e.,* celui du nouvel itemset fermé fréquent découvert [5].

Un itemset fréquent nouvellement découvert est inséré dans l'arbre des itemsets fermés fréquents seulement si il n'est pas un sous-ensemble d'un itemset figurant dans l'arbre. Un itemset fréquent *Y*, nouvellement trouvé contenant *X*, est comparé avec toutes les branches de l'arbre CFI incluant l'item *X*. Si, dans l'arbre CFI, il n'y a pas un sur-ensemble de *Y* ayant le même support que *Y* alors *Y* est un itemset fermé.

3.2. Principe de l'algorithme FELINE-F

Le pseudo-code de l'algorithme FELINE-F est donné par l'Algorithme 1. Il prend en entrée un arbre CATS qui constitue l'espace de recherche de l'algorithme. Ensuite, l'espace de recherche est divisé en sous-espaces de recherche conformément à la liste des items fréquents. L'exploration de chaque sous-espace de recherche donnera lieu à un ensemble d'itemsets fermés candidats. Par la suite, nous devons vérifier l'existence de ces candidats dans l'arbre CFI afin de le mettre à jour.

4. L'algorithme FELINE-H

Les expérimentations, que nous avons menées, ont montré que les arbres CATS, relatifs à des contextes denses, sont des arbres compacts, du fait que les transactions présentent plusieurs préfixes communs. Les arbres CATS conditionnels condensés qui leur sont associés sont beaucoup plus compacts et de taille réduite. Par conséquent, le parcours de ces arbres est très rapide. Dans ce cas, l'extraction des itemsets fermés fréquents est plus intéressante puisque nous pouvons avoir un nombre assez important d'itemsets fréquents. Quant aux bases éparses, les arbres CATS sont le plus souvent très grands du fait qu'ils n'ont pas plusieurs préfixes communs à partager et ceci ralentit leur parcours. Par conséquent, l'espace de recherche des itemsets fermés fréquents tend à se confondre avec celui des itemsets fréquents [6]. Ainsi, nous trouvons qu'il est plus judicieux, quand nous traitons une

Algorithme 1 FELINE-F

Entrée: Un arbre CATS, minSup
Sortie: Un arbre CFI contenant l'ensemble complet des itemsets fermés fréquents

```
1: Trier les items de l'entête par ordre décroissant
2: Pour tout item fréquent I faire
3:    Si (il n'existe pas un nœud dans CFI portant le label I avec un support supérieur ou égal à
      celui de l'item I) alors
4:       Construire I-Tree
         {/* l'arbre CATS conditionnel condensé de l'item I */}
5:       Si (I-Tree est un chemin unique) alors
6:          Générer tous les itemsets fermés fréquents à partir de I-Tree ;
7:          Pour tout itemset fermé fréquent X généré faire
8:             Si (X n'existe pas dans CFI) alors
9:                Mettre à jour l'arbre CFI
10:            sinon
11:               MINE-FERME(I-Tree, minSup)
12:            Fin Si
13:         Fin Pour
14:      Fin Si
15:   Fin Si
16: Fin Pour
17: Retourner L'arbre CFI
```

base éparse, d'extraire les itemsets fréquents. Ceci dit, il s'avère intéressant d'avoir un algorithme "*hybride*" qui s'adapte à la nature de l'arbre conditionnel condensé CATS. Cet algorithme réalise l'extraction des itemsets fermés fréquents lorsqu'il s'agit d'une base dense et seulement l'extraction des itemsets fréquents lorsque la base est éparse. Seulement, nous sommes en face du problème suivant : *"Comment décider de la nature d'une base"*. Un élément de réponse [7] consisterait à calculer la moyenne des supports des items fréquents de la base relative à un arbre conditionnel condensé CATS. Si cette moyenne est *largement* supérieure à *minSup*, alors nous considérons que la base est dense. Le problème est comment quantifier le terme *"largement"* (coefficient m). Comme nous ne pouvons déterminer ce coefficient de manière absolue, nous allons tenter de le déterminer expérimentalement. Si la moyenne des supports des items fréquents relatifs à un arbre conditionnel condensé CATS est supérieure à m fois le *minSup*, nous concluons que la base conditionnelle correspondante est dense.

Le pseudo-code de l'algorithme hybride, appelé FELINE-H, est donné par l'Algorithme 3. D'une manière similaire à l'algorithme FELINE-F, l'algorithme FELINE-H prend en entrée un arbre CATS qui constitue l'espace de recherche de l'algorithme. Ensuite, un arbre conditionnel condensé CATS est construit pour chaque item fréquent. Pour chacun de ces arbres, nous calculons la moyenne des supports des items fréquents relatifs à cet arbre. Ensuite, nous opérons de la manière suivante :

– Si cette moyenne est supérieure à $(m \times minSup)$, alors l'algorithme FELINE-F est appliqué.

– Sinon, l'algorithme FELINE est appliqué.

5. Evaluation expérimentale

Toutes nos expérimentations ont été réalisées sur un PC Pentium IV avec un processeur de 2 GHz et 512 Mega octets de mémoire centrale fonctionnant sous le système d'exploitation Mandrake

Algorithme 2 Procédure MINE-FERME

Entrée: *I-Tree* : l'arbre conditionnel condensé de l'item *I*, *minSup*
Sortie: L'arbre CFI mis à jour

```
1: Si (support(I)> minSup) alors
2:    s ← ∅ {/ * Ensemble des items traités*/}
3:    Pour tout item I_i ∈ I-Tree et I_i ∉ s faire
4:       s← I_i ;
5:       Construire I_i Tree {/ * l'arbre CATS conditionnel condensé de l'item I_i */}
6:       MINE-FERME(I_i-Tree,minSup){/*Générer tous les itemsets fermés fréquents*/}
7:       Pour tout itemset fermé fréquent X généré faire
8:          Si (X n'est pas inclus dans CFI) alors
9:             Mettre à jour l'arbre CFI ;
10:         Fin Si
11:      Fin Pour
12:   Fin Pour
13: Fin Si
14: Retourner L'arbre CFI mis à jour
```

Linux 9.1. Les programmes ont été écrits en C++. Dans le cadre de nos expérimentations, nous avons utilisé deux types de bases : des bases éparses et des bases denses[1].

5.1. Résultats expérimentaux sur FELINE-F

Les temps d'exécution des algorithmes FELINE-F et CLOSET sur les bases éparses sont présentés par la figure 2. Notons que l'exécution de l'algorithme CLOSET pour la base T40I10D100k pour

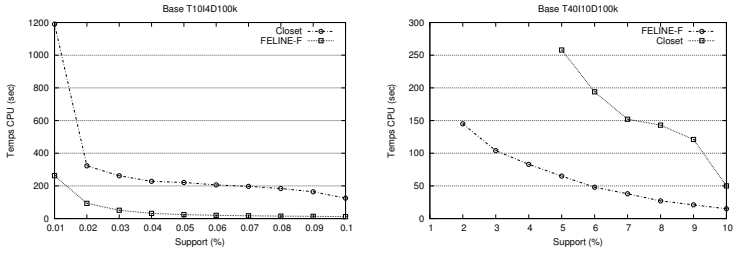

Figure 2. FELINE-F *vs* CLOSET *pour les bases de test éparses*

des valeurs de support comprises entre 1% et 4% a donné le message "aborted". Comme le montre la figure 2, FELINE-F est largement plus performant que CLOSET. Les temps d'exécution des algorithmes FELINE-F et CLOSET sur les bases denses sont présentés par la figure 3. Comme le montre cette figure, FELINE-F est plus performant que CLOSET. Il est à noter que FELINE-F est beaucoup moins sensible à la variation de *minSup* que CLOSET surtout pour les bases denses. Afin de tester la mise à l'échelle (scalability) de Feline-H par rapport à la taille de la base et la comparer avec CLOSET, nous avons généré des bases T10I4DxP1K en faisant varier le nombre de transactions de $200K$ transactions à $1400K$ transactions, tout en fixant le nombre d'items à $1K$. Nous avons généré ces bases en utilisant le générateur de bases synthétiques d'IBM[2]. Le *minSup* a été fixé à 0.01%.

1. L'ensemble de ces bases est disponible aux adresses suivantes : http ://fimi.cs.helsinki.fi/data et http ://dep-tinfo.unice.fr/˜ pasquier
2. http ://www.almaden.ibm.com/cs/quest/syndata.html

Algorithme 3 FELINE-H

Entrée: Un arbre CATS, minSup
Sortie: L'arbre CFI contenant l'ensemble complet des itemsets fréquents

```
1: Trier les items de l'entête par ordre décroissant ;
2: Pour tout item fréquent I faire
3:    Construire I-Tree ;{/*l'arbre CATS conditionnel condensé de l'item I*/}
4:    Si I-Tree est un chemin unique alors
5:       Générer tous les itemsets fermés fréquents à partir de I-Tree
6:       Mettre à jour l'arbre CFI
7:    sinon
8:       Calculer la moyenne moy des supports des items de I-Tree
9:       Si moy > m× minSup alors
10:         MINE-FERME(I-Tree, minSup)
11:         Pour tout itemset fermé fréquent généré faire
12:            Générer l'ensemble de tous les itemsets fréquents qui lui sont associés ;
13:         Fin Pour
14:      sinon
15:         MINE-FREQUENT(I-Tree, minSup)
16:         Mettre à jour l'arbre CFI
17:      Fin Si
18:   Fin Si
19: Fin Pour
20: Retourner L'arbre CFI
```

L'exécution de l'algorithme CLOSET pour toutes les bases utilisées donne le message "aborted". Les résultats ainsi obtenus confirment que FELINE-F est beaucoup moins sensible à la variation des items et/ou de *minSup* et qu'il est largement plus performant que CLOSET.

5.2. Résultats expérimentaux sur FELINE-H

5.2.1. Résultats expérimentaux sur les bases denses

Comme le montre la figure 4, l'algorithme FELINE-H exécuté sur des bases denses permet d'améliorer les temps de réponse comparativement à FELINE-F. Il est à remarquer que la différence au niveau temps d'exécution entre l'algorithme FELINE-F et l'algorithme FELINE-H décroît au fur et à mesure que la valeur du *minSup* croît. Ceci peut s'expliquer par le fait que, lorsque la valeur du *minSup* devient de plus en plus grande, le nombre d'items vérifiant le *minSup* devient de plus en plus petit. Par conséquent, l'arbre CATS associé va contenir de moins en moins de nœuds et sa taille devient de plus en plus réduite. Ces expérimentations nous ont permis aussi de déterminer approximativement le coefficient *m* qui est compris, dans la plupart des cas, entre 3 et 5.

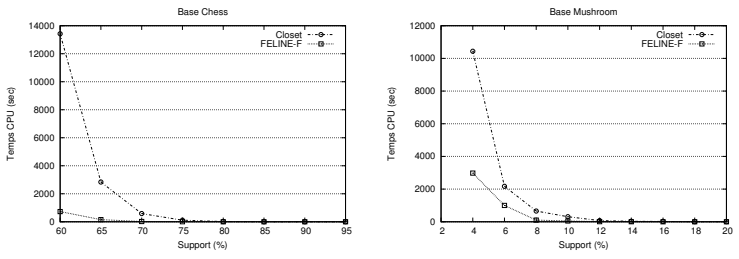

Figure 3. FELINE-F *vs* CLOSET *pour les bases de test denses*

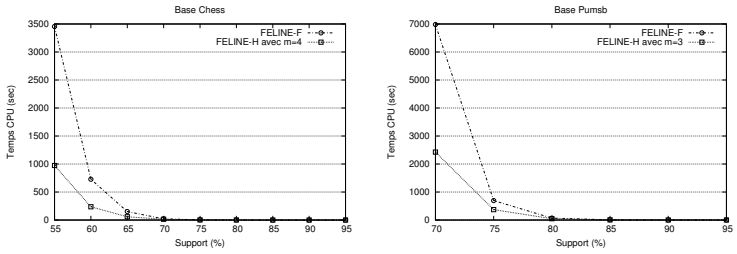

Figure 4. FELINE-F *vs* FELINE-H *pour les bases de test denses*

5.2.2. Résultats expérimentaux sur les bases éparses

Les gains en terme de temps de réponse, suite à l'exécution de l'algorithme FELINE-H sur des bases éparses, sont insignifiants. En effet, les temps d'exécution des algorithmes FELINE-F et FELINE-H sont équivalents. Ceci se justifie par le fait que la plupart des arbres conditionnels condensés CATS sont constitués d'un chemin unique ce qui implique que, pour les bases éparses, appliquer l'algorithme FELINE-H revient à appliquer l'algorithme FELINE-F.

6. Conclusion

Dans ce papier, nous avons proposé deux nouveaux algorithmes : (i) FELINE-F qui permet d'extraire les itemsets fermés fréquents ; (ii) FELINE-H permettant de s'adapter à la densité d'une base. Les expérimentations que nous avons réalisées ont montré que l'algorithme FELINE-F est plus performant que CLOSET et qu'il est scalable pour des bases de grandes tailles. L'avantage majeur de l'algorithme FELINE-H est qu'il permet d'améliorer les temps de réponse et ce en prenant en compte la nature de la base en cours d'exploration (dense ou éparse). Les perspectives de travaux futurs concernent l'étude d'un modèle théorique pour l'estimation de la densité du contexte d'extraction.

7. Bibliographie

[1] J. HAN, J. PEI, Y. YIN « Mining frequent patterns without candidate generation », *Proceedings of the ACM-SIGMOD Intl. Conference on Management of Data (SIGMOD'00),Dallas, Texas*, 2000.

[2] W. CHEUNG, O.R. ZAIANE « Incremental Mining of Frequent Patterns Without Candidate Generation or Support Constraint », *Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS 2003), Hong Kong, China*, 2003.

[3] J. PEI, J. HAN,, R. MAO « CLOSET : An efficient algorithm for mining frequent closed itemsets », *SIGMOD, Dallas, Tx, USA*, 2000.

[4] J. PEI, J. HAN, Y. YIWEN « Mining Frequent Patterns without Candidate Generation », *SIGMOD, Washington D.C., USA*, 2000.

[5] G. GRAHNE, J. ZHU « Efficiently Using Prefix-trees in Mining Frequent Itemsets », *FIMI'03 Workshop on Frequent Itemset Mining Implementations, Melbourne, Florida, USA*, 2003.

[6] S. BENYAHIA, E. MEPHU NGUIFO « Approches d'extraction de règles d'association basées sur la correspondance de GALOIS », *Ingénierie des Systèmes d'Information (ISI)*, vol. 3–4, n° 9, 2004.

[7] S. ORLANDO, P. PALMERINI, R. PEREGO « Statistical Properties of Transactional Databases », *Proceedings of the 2004 ACM symposium on Applied computing, Nicosia, Cyprus*, 2004.