# VPSP : extraction de motifs séquentiels dans WEKA

D. Kraemer*, L. Di Jorio*, D. Jouve*, A. Serra*,
C. Raïssi**,***, A. Laurent**, M. Teisseire**, P. Poncelet***

* Université Montpellier II, Place Eugène Bataillon, 34000 Montpellier
** LIRMM - CNRS UMR 5506, 161 rue Ada, 34392 Montpellier Cedex 5
*** LGI2P - Ecole des Mines d'Alès, site EERIE - Parc Scientifique Georges Besse, 30035 Nîmes

## Résumé

La problématique de l'extraction de motifs séquentiels dans de grandes bases de données intéresse la communauté Fouille de Données depuis une dizaine d'années. L'environnement WEKA est actuellement très utilisé pour extraire de la connaissance mais il ne propose pas d'algorithmes d'extraction de motifs. L'objectif de cette démonstration est de proposer une nouvelle approche de motifs, appelée VPSP, qui respecte la philosophie de WEKA mais qui en plus possède la particularité d'être très efficace en temps de réponse.

## 1 Introduction

WEKA[1] (*Waikato Environment for Knowledge Analysis*) [WF05] est un projet de recherche mené à l'Université de Waikato (Nouvelle-Zélande) depuis 1994. L'objectif du projet est de proposer un environnement d'extraction de connaissances "open source" qui regroupe un ensemble d'algorithmes de fouille de données. Parmi les algorithmes proposés, nous trouvons des algorithmes de classification, de régression et d'extraction de règles d'association. De manière à faciliter les différentes étapes d'extraction, des outils de pré-traitement des données et de visualisation sont également proposés. Cet environnement (C.f. Figure 1) est à l'heure actuelle de plus en plus utilisé par les chercheurs de la communauté Fouille de Données (il a été classé en 2006 dans les 10 premiers outils d'extraction utilisés[2] et il a reçu en 2005 le prix du "2005 ACM SIGKDD Service Award"[3].

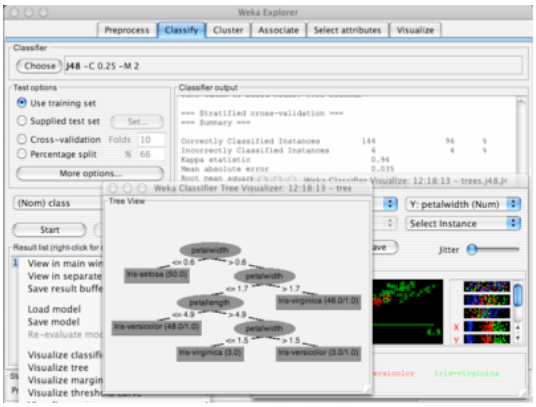


FIG. 1 – L'environnement Weka

L'objectif de cette démonstration est d'enrichir l'environnement WEKA à la prise en compte d'un algorithme de recherche de motifs séquentiels. En effet, introduits dans [AS95] les motifs séquentiels peuvent être vus comme une extension de la notion de règles d'association intégrant diverses contraintes temporelles. La recherche de tels motifs consiste

[1] http ://www.cs.waikato.ac.nz/ ml/weka/index.html

[2] http ://www.kdnuggets.com

[3] http ://www.acm.org/sigs/sigkdd/awards.php

ainsi à extraire des enchaînements d'ensembles d'items, couramment associés sur une période de temps bien spécifiée. En fait, cette recherche met en évidence des associations inter-transactions, contrairement à celle des règles d'association qui extrait des combinaisons intra-transactions. Par exemple, des motifs séquentiels peuvent montrer que *"60% des gens qui achètent une télévision, achètent un magnétoscope dans les deux ans qui suivent"*. Ce problème a donné lieu à de nombreuses contributions au niveau de la communauté Fouille de Données [Zak01, SA96, AFGY02, HPMa$^+$00, MCP98, PHMa$^+$01, PHW02] mais n'est pas encore intégré dans l'environnement Weka.

L'article est organisé de la manière suivante. Dans la section 2 nous précisons la problématique de l'extraction de motifs séquentiels Au cours de la section 3 nous décrivons notre nouvel algorithme et son intégration dans l'environnement Weka. Enfin, nous concluons cet article en présentant des travaux futurs.

## 2 La problématique de l'extraction de motifs séquentiels

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

Un client *supporte* une séquence $s$ si $s$ est incluse dans la séquence de données de ce client. Le *support* d'une séquence $s$ est calculé comme étant le pourcentage des clients qui supportent $s$. Soit $\sigma$ le support minimum fixé par l'utilisateur. Une séquence qui vérifie le support minimum (i.e. dont le support est supérieur à $\sigma$) est une *séquence fréquente*. Soit une base de données $DB$ de séquence de données, le problème de la recherche de motifs séquentiels consiste à extraire toutes les séquences fréquentes qui respectent la contrainte de support minimal.

Depuis la définition de la problématique de nombreuses approches ont été proposées ces dernières années. Elles considèrent généralement : (i) des algorithmes par niveaux (e.g. GSP, PSP, SPAM) en alternant les étapes de génération de candidats, vérification sur la base et suppression des candidats non fréquents et diffèrent principalement sur les structures de données utilisées (arbre de hachage pour GSP [SA96], arbre préfixé pour PSP [MCP98], vecteur de bits pour SPAM [AFGY02]) ; (ii) que la base de données peut tenir en mémoire et utilise des classes d'équivalences (e.g. SPADE) ou des structures préfixées (e.g. PrefixSpan [PHMa$^+$01]).

## 3 L'algorithme Vpsp

Dans cette section nous présentons l'approche Vpsp qui est intégré dans l'environnement Weka et nous montrons quelques expérimentations.

### 3.1 L'approche Vpsp

L'algorithme Vpsp est un algorithme de type "*générer-élaguer*" et combine les avantages principaux des algorithmes PSP et SPADE grâce à une nouvelle structure de données en forme d'arbre prefixée (heritée de PSP) couplée à un chargement de la base de données en représentation verticale. Cette union de structure d'arbre et de transformation de bases de données permet :

1. **d'optimiser l'espace mémoire utilisé**. La différence principale, distinguant notre struc-

ture de celle utilisée dans PSP, provient du fait que la base de données n'est parcourue qu'une seule fois tout au long de l'algorithme permettant l'extraction. Lors de cette unique lecture de la base, le premier étage de l'arbre préfixé est construit. Chaque nœud de profondeur 1, représentant un item de la base de données, garde une trace, pour chaque apparition dans la base, du client et de la date d'apparition correspondante. Ce processus reprend donc la transformation de la base de données opérée dans SPADE, *pour la projeter dans l'arbre préfixé.*

2. **de simplifier l'opération de générations** (inutilité des classes d'équivalences). Cet algorithme reprend les fondements de la génération des candidats effectuée dans PSP. Cependant, dans VPSP lorsqu'un candidat est généré, celui-ci se voit attribué un vecteur d'apparition issu de la base de données qui permet de vérifier plus efficacement le comptage du support.
3. **de simplifier le comptage du support**. Le comptage des séquences candidates tire profit de la structure de données utilisée par VPSP. En effet, grâce à la liste d'apparition dont dispose chaque candidat, nous pouvons déterminer efficacement le nombre de clients qui ont participés à l'incrémentation du support.

### 3.2 Intégration dans WEKA

L'algorithme VPSP est implementé en langage JAVA afin de respecter les spécifications de l'environnement d'extraction WEKA. Via cette implémentation, l'extraction des motifs séquentiels est indépendante du type de support de la base de données (texte brut, format ARFF propre à WEKA ou base de données stockée dans un SGBD). De plus, les résultats peuvent être interpretés et classés à l'aide des outils graphiques de l'environnement WEKA sans transformation préalable. Différentes options sont possibles puisque VPSP permet de choisir, filtrer ou discrétiser les données à fouiller. La figure 2 illustre l'utilisation des motifs séquentiels dans WEKA.

### 3.3 Expérimentations

De manière à analyser l'algorithme VPSP, nous avons effectué différentes expérimentations et nous avons comparé nos résultats avec l'algorithme PSP. En effet, ce dernier fait preuve d'une grande efficacité grâce à la structure d'arbre préfixé qu'il met en œuvre. L'algorithme VPSP en étendant la structure initiale améliore encore l'efficacité de l'extraction de séquences fréquentes. Ainsi, par exemple, le comptage du support qui est effectué, à chaque étage après la génération de candidats, se voit grandement accéléré puisqu'il consiste à effectuer des opérations ensemblistes sur des structures de données choisies à cet effet. Il s'agit d'une méthode inspirée de l'algorithme SPADE qui charge en mémoire une transformation verticale de la base de données. Cependant, la structure d'arbre préfixé dont les nœuds ont une connaissance de la base de données nécessite une occupation mémoire plus elevée que la structure d'arbre préfixé utilisée par l'algorithme PSP. En effet, même si nous appliquons une optimisation qui permet de ne garder que les listes d'apparitions des nœuds qui vont permettre de générer de nouvelles séquences candidates (avec notre approche, seules les séquences fréquentes de longueur 1, les items fréquents, et de longueur $k$ sont nécessaires pour générer des candidats de longueur $k+1$), l'occupation mémoire de notre algorithme est plus importante que celle de PSP. De plus, il est important de noter que si la première étape de notre algorithme permet de déterminer les séquences fréquentes de longueur 1, elle constitue également le seul passage sur la base de données. Au cours de cette lecture, la base de données est entièrement transformée en sa représentation verticale puis projetée dans l'arbre préfixé. Il est donc possible, sur des cas très particuliers de bases de données où il existe un grand nombre d'items mais paradoxalement très peu de séquences fréquentes de longueurs élevées, que notre algorithme ne se montre pas beaucoup

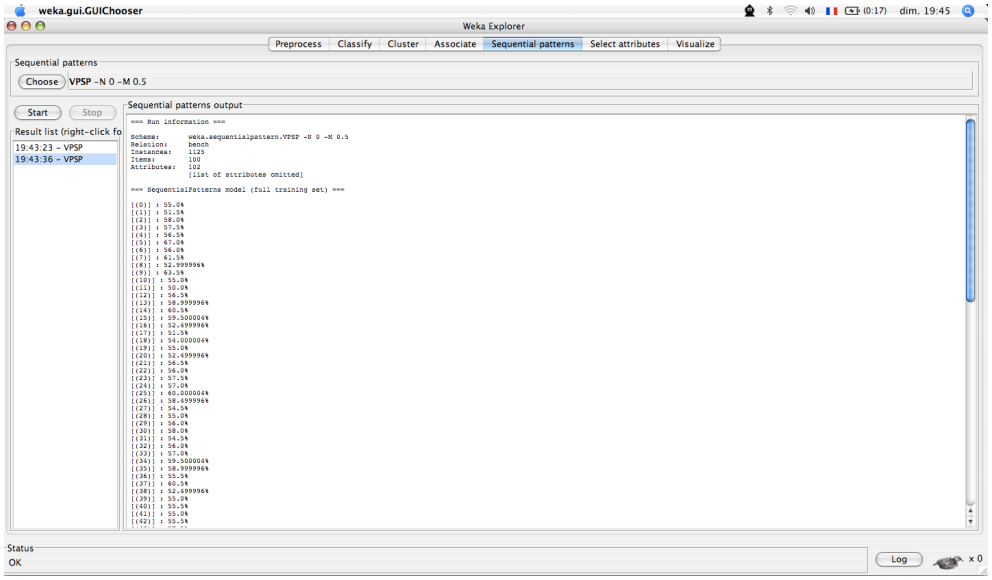


FIG. 2 – Les motifs séquentiels dans WEKA

plus efficace que l'algorithme PSP. Toutefois, les expérimentations menées montrent sur des grandes quantités de données caractéristiques l'excellent comportement de notre algorithme par rapport a celui des algorithmes antérieurs. Ce fait est remarquable étant donné que VPSP a été implémenté en JAVA alors que les algorithmes utilisés pour les expérimentations bénéficiaient de la rapidité d'éxécution apportée par le langage C++.

## 4 Conclusion

Dans cette démonstration, nous avons proposé d'enrichir l'environnement WEKA à la prise en compte d'un nouvel algorithme pour extraire des motifs séquentiels à partir de bases de données. Outre le fait que l'approche poursuit complètement la philosophie de l'environnement, les expérimentations menées ont montré qu'elle était également très efficace malgré le fait qu'elle était développée en Java.
A l'heure actuelle, pour répondre aux besoins de nouvelles applications (Web Usage Mining, données issues de capteurs, télécommunications, ...) dont les données arrivent sous la forme de flots, appelée également Data Streams, il est nécessaire de repenser les algorithmes de motifs séquentiels dans la mesure où les approches traditionnelles ne sont plus adaptées [RPT05]. L'approche VPSP propose des propriétés intéressantes que nous souhaitons considérer dans le cas de flots de données.

## Références

[AFGY02] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using bitmap representation. In *Proc. of KDD'02*, 2002.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of ICDE'95*, 1995.

[HPMa+00] J. Han, J. Pei, B. Mortazavi-asl, Q. Chen, U. Dayal, and M. Hsu. Freespan : Frequent pattern-projected sequential pattern mining. In *Proc. of KDD'00*, 2000.

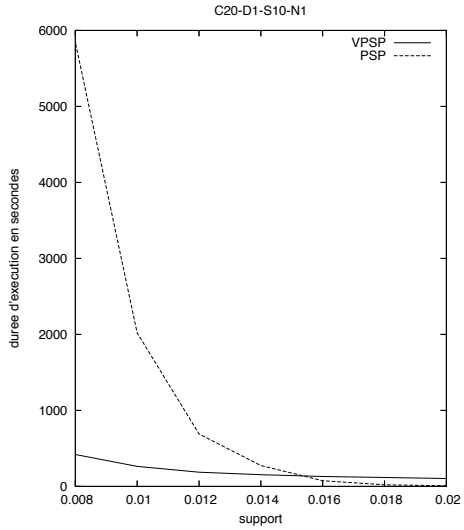


FIG. 3 – Comparaison avec l'algorithme PSP sur la base de données : C20-D1-S10-N1

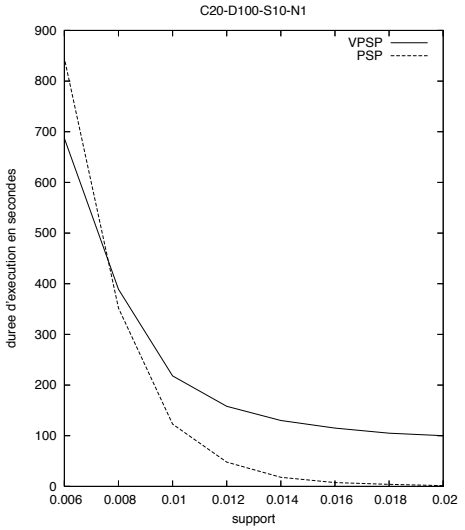


FIG. 4 – Comparaison avec l'algorithme PSP sur la base de données : C20-D100-S10-N1

[MCP98] F. Masseglia, F. Cathala, and P. Poncelet. The PSP approach for mining sequential patterns. In *Proc. of PKDD'98*, 1998.

[PHMa+01] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, and U. Dayal. Prefixspan : Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. of ICDE'01)*, 2001.

[PHW02] J. Pei, J. Han, and W. Wang. Mining sequential patterns with constraints in large databases. In *Proc. of CIKM'02)*, 2002.

[RPT05] C. RaÔssi, P. Poncelet, and M. Teisseire. Need for speed : Mining sequential patterns in data streams. In *Proc. of BDA'05*, 2005.

[SA96] R. Srikant and R. Agrawal. Mining sequential patterns : Generalizations and performance improvements. In *Proc. of EDBT'96*, 1996.

[WF05] I. H. Witten and E. Frank. *Data Mining : Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[Zak01] M.J. Zaki. SPADE : An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42(1) :31–60, 2001.