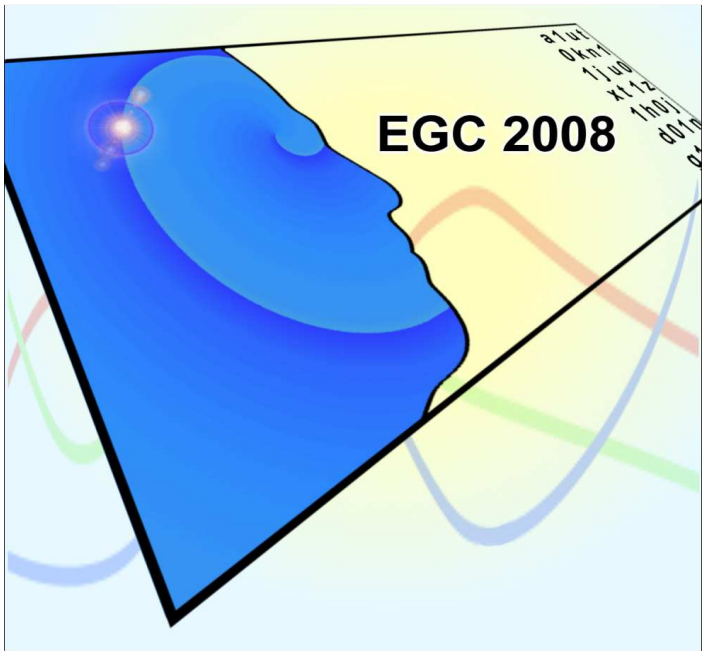
8èmes Journées Francophones
Extraction et Gestion des Connaissances
Sophia Antipolis 29 janvier 2008

Atelier



Visualisation et Extraction de Connaissances

Organisateurs :

- François Poulet (IRISA, Univ. Rennes I)
- Bénédicte Le Grand (LIP6/CNRS)
- Thanh-Nghi Do (INRIA Futurs)

Responsables des Ateliers EGC :

Alzennyr Da Silva (INRIA, Rocquencourt)
Alice Marascu (INRIA, Sophia Antipolis)
Florent Masseglia (INRIA, Sophia Antipolis)

http://www-sop.inria.fr/axis/egc08

EGC

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche SOPHIA ANTIPOLIS - MÉDITERRANÉE

Avant-propos

Les outils de visualisation contribuent à l'efficacité des processus mis en œuvre en extraction de connaissances en offrant aux utilisateurs des représentations intelligibles et facilitant l'interaction.

La visualisation intervient à différentes étapes de la chaîne de traitement : dans les phases amonts pour appréhender les données et effectuer les premières sélections, lors du processus de fouille, et dans la phase aval pour évaluer les résultats obtenus et les communiquer. Du fait de l'importance croissante accordée au rôle de l'utilisateur en fouille de données, les outils de visualisation sont devenus des composantes majeures des logiciels qui s'utilisent de plus en plus en coopération étroite avec des méthodes automatiques, à la fois en pré et post - traitement.

La fouille visuelle de données ("Visual Data Mining") qui vise à développer des outils interactifs adaptés au traitement des données et des connaissances associées intègre par essence des concepts issus de disciplines diverses : perception visuelle, psychologie cognitive, métaphores de visualisation, visualisation scientifique ou d'information, etc.

Le but de l'atelier Visualisation et Extraction des Connaissances est de fournir un lieu d'échange et de présentation de méthodes nouvelles, d'axes de recherche, de développements dans le domaine de la visualisation en extraction de connaissances et de la fouille visuelle de données.

Les éditions précédentes de l'atelier ont illustré le besoin et l'intérêt grandissant pour la visualisation en extraction de connaissances. Elles ont également permis d'aborder le délicat problème de l'évaluation des méthodes visuelles proposées.

Cette sixième édition s'inscrit dans cette continuité, avec une attention grandissante sur les problèmes posés par les données volumineuses et par le rôle essentiel de l'interactivité.

L'atelier est organisé en 3 sessions:
- la première session traite de la visualisation de graphes et de règles d'association et aborde des aspects théoriques,
- la deuxième session s'intéresse à des applications originales de la visualisation au Web Usage Mining et à l'analyse d'images,
- la dernière session est dédiée à la présentation d'un environnement de programmation visuelle et à une discussion.

Table des matières

Recherche de motifs quasi-similaires dans des graphes

Fanny Chevalier*, Maylis Delest*
Jean-Philippe Domenger*

*Université de Bordeaux, 351 Cours de la Libération, 33400 TALENCE
nom.prenom@labri.fr,
http://www.labri.fr/

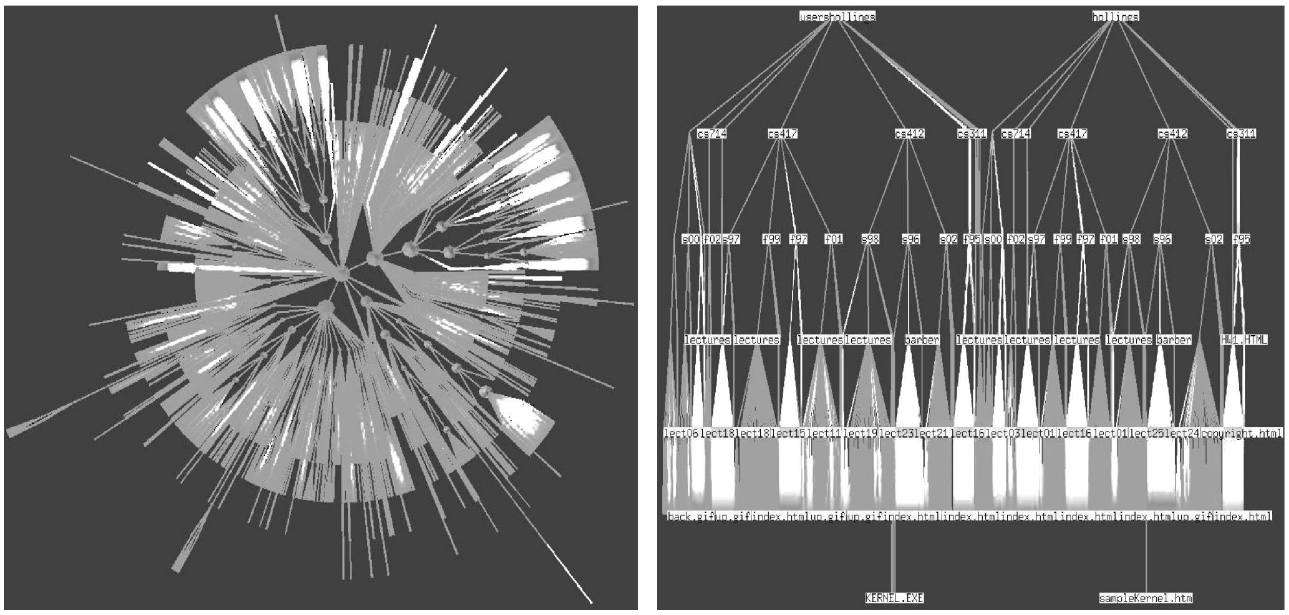
Résumé. Nous décrivons un algorithme basé sur des métriques intrinsèques de graphes permettant de découvrir des motifs communs et similaires entre plusieurs graphes. Nous montrons des applications à la recherche d'image dans une collection et à l'interprétation de données géographiques.

1 Introduction

Il existe un très large panel de méthodes et algorithmes d'analyse, manipulation, navigation, dessin, visualisation pour l'étude et la représentation des graphes. L'une des tâches les plus importantes liées à l'utilisation des graphes est leur comparaison. Ainsi, le problème de mise en correspondance de graphes et de sous-graphes fait l'objet d'une recherche intensive. Le problème d'isomorphisme de graphes et de sous-graphes étant coûteux, un grand nombre de techniques proposées exploitent les spécificités des données du domaine d'application afin d'optimiser les traitements en limitant les comparaisons. D'autre part, selon l'application visée, il est souvent pertinent de détecter dans des structures de graphes des éléments qui se ressemblent fortement, sans pour autant être identiques. En effet, la détection de motifs ressemblants que l'on appellera *motifs quasi-similaires* peut se révéler une source d'information importante. En vision par ordinateur, Lladós et al. (2001) ont proposé une méthode de reconnaissance de symboles graphiques ou de diagrammes manuscrits. Dans leur approche, un graphe d'adjacence des régions est associé à chaque symbole. La reconnaissance s'effectue par comparaison de graphes en utilisant une méthode de coût d'édition. En chimie, on peut citer les travaux de Yan et al. (2006) pour la recherche de sous-structures dans des molécules chimiques. Chaque molécule est dans un premier temps associée à un graphe. Ce graphe est ensuite décomposé en un ensemble de motifs élémentaires prédéfinis. Ces motifs servent d'index pour retrouver une molécule dans une base de données. Dans le domaine du multimédia, de nombreux travaux sont basés sur les techniques de mise en correspondance de graphes. On peut citer Gomila et Meyer (2003) qui utilisent des méthodes de relaxation probabiliste pour le suivi d'objets segmentés dans la vidéo. Demirci et al. (2006) proposent une mise en correspondance des sommets telle que plusieurs régions d'une image requête (associées à plusieurs sommets dans le graphe d'adjacence des régions), peuvent être mises en correspondance avec une seule et même région (donc un unique sommet dans le graphe d'adjacence) de l'image cible. En maintenance de code informatique, les arbres syntaxiques correspondant au code source sont utilisés pour la détection de duplication de code. Baxter et al. (1998) ont proposé

une méthode basée sur la classification des sommets par similarité (au vu de la composition du sous-arbre). Cette classification implique que deux sommets classés dans la même catégorie correspondent à deux sommets racines de sous-arbres similaires.

La méthode, que nous détaillons ici, s'appuie sur une heuristique utilisant les caractéristiques structurelles des sommets pour la mise en correspondance. Le résultat de la mise en correspondance de motifs quasi-similaires est présenté comme une coloration des sommets telle que deux motifs de la même couleur sont des motifs quasi-similaires. A l'origine, cette méthode a été développée pour répondre aux tâches concernant la comparaison d'arborescences de fichiers. L'un des objectifs fixés par le concours Infovis (Fekete et Plaisant (2003)) était de proposer une méthode de détection visuelle des similarités et/ou changements entre deux versions différentes d'une même arborescence. La solution logicielle EVAT de Auber et al. (2003) a été proposée dans ce cadre. L'interface de visualisation permet à l'utilisateur d'identifier visuellement, par la coloration des sommets et arêtes des arborescences (ici en blanc), si des changements ont eu lieu entre deux versions d'un système arborescent de fichiers. Une même couleur suggère des motifs quasi-similaires. Sur la figure 1.a, on remarque la présence de deux sous-arbres proches [1]. En montrant le détail de ces deux sous-arbres (Fig. 1.b) on constate en effet que les répertoires nommés `hollings` et `usershollings` sont très proches en terme de structure.



(a) (b)

FIG. 1 – *EVAT : Visualisation de motifs communs dans un système de fichiers.*

[1] Pour plus de lisibilité, on recommande vivement au lecteur de consulter la version couleur de l'article en ligne à l'adresse http ://www.labri.fr/publications/mabiovis/. Dans cette version, des motifs rouges, roses et bleus sont visibles dans les deux sous-arbres

Par la suite, Auber et al. (2006) ont repris la méthode dans un algorithme pour le dessin de structures secondaires d'ARN. Le principe est d'identifier le plus grand motif similaire entre deux structures. Ce motif est alors dessiné à une même place et avec la même orientation dans chacune des fenêtres dédiées à la représentation des ARNs. Ainsi, le motif similaire permet à l'utilisateur d'avoir un point de référence pour la comparaison. Enfin, dans le domaine de la visualisation de l'information, Chevalier et al. (2007a) ont proposé une technique de visualisation permettant d'identifier visuellement les changements au cours du temps des blocs de codes en génie logiciel.

Nous proposons dans cet article une formalisation de cette heuristique généralisée aux graphes. L'heuristique procède en deux phases. Dans un premier temps, les sommets des graphes sont classifiés en différentes familles de sommets partageant les mêmes caractéristiques. La deuxième phase concerne la construction des motifs quasi-similaires : l'algorithme se base sur la comparaison des étiquetages des voisinages des sommets à comparer.

Nous avons introduit l'arbre de couverture de Beygelzimer et al. (2006) pour optimiser la phase de construction des familles de sommets aux mêmes propriétés. D'autre part, nous définissons ici deux variantes pour la reconnaissance de motifs quasi-similaires. Ces variantes se distinguent par la stratégie de comparaison du voisinage. L'ensemble des sommets considérés pour la comparaison de motifs, selon la variante, est constitué de :

– une partie du voisinage direct des sommets
– l'intégralité du voisinage direct des sommets

Dans ce qui suit nous présentons tout d'abord la classification des sommets puis dans la section suivante la contruction des motifs. La section 4 est consacré aux applications. Enfin, nous proposons une conclusion en section 5.

2 Classification des sommets

La première étape de la méthode est consacrée à la classification des sommets des graphes selon une mesure de similarité basée sur leurs propriétés structurelles. Les caractéristiques structurelles considérées sont liées au type d'application. Cette classification est une première étape à la comparaison des sous-structures des graphes : deux sous-graphes dont les sommets racines ont des caractéristiques structurelles (locales et globales) proches sont intéressants à comparer.

La classification des sommets en familles distinctes s'appuie sur un ensemble de caractéristiques associées à chaque sommet.

Les métriques utilisées ici pour évaluer la similarité entre deux sommets correspondent à des métriques intrinsèques, calculées sur les sommet d'un graphe, indépendamment des informations propres aux données étudiées. En effet nous pensons que quelle que soit l'application visée, la structure représente une caractéristique pertinente qu'il est important de considérer pour la comparaison. Parmi les caractéristiques structurelles, on peut citer, pour un sommet u, le degré, le nombre de sommets à distance n de u, le nombre de cliques de taille k qui contiennent u ; si l'on considère des arbres, la distance à la racine, la taille du sous-arbre enraciné en u, le nombre de strahler (Auber et al. (2004)), *etc*.

Soit G (resp. G') un graphe d'ensemble de sommets V (resp. V') et d'arêtes E (resp. E'). Pour une métrique μ, la valeur associée à un sommet $u \in V \cup V'$, correspond à la valeur

normalisée $\tilde{\mu}(u)$ de la métrique μ selon la formule :

$$\tilde{\mu}(u) = \frac{\mu(u) - \mu_{min}}{\mu_{max} - \mu_{min}} \quad (1)$$

avec $\mu_{min} = \min_{v \in V \cup V'} \mu(v)$ et $\mu_{max} = \max_{v \in V \cup V'} \mu(v)$.

Ainsi, pour un ensemble de n métriques $\{\mu_i\}_{i \in \{1,\cdots,n\}}$, nous calculons pour chaque sommet $u \in V \cup V'$, le vecteur caractéristique associé $(\tilde{\mu}_1(u), \tilde{\mu}_2(u), \cdots, \tilde{\mu}_n(u))$.

A cette étape, nous construisons les différentes familles $\mathcal{F}$ contenant des sommets aux propriétés similaires. Nous définissons un étiquetage λ des sommets tel que si deux sommets u et v appartiennent à une même famille $\mathcal{F}$, alors $\lambda(u) = \lambda(v)$.

Définition 1 (Sommets ϵ-similaires) *Soient μ_i, $i \in \{1, \cdots, n\}$ l'ensemble des métriques considérées et soit le seuil de similarité $\epsilon \in [0, 1]$. On dit que deux sommets u et v sont ϵ-similaires s'ils vérifient :*

$$dist(u, v) < \epsilon \quad (2)$$

Dans cet article, $dist(u, v)$ correspond à la distance euclidienne entre les vecteurs. La construction des familles de sommets ϵ-similaires se base donc sur une comparaison deux à deux des sommets des graphes à laquelle nous intégrons une stratégie de partitionnement plus efficace pour la construction des familles. L'algorithme est basé sur une variante de la méthode d'*arbre de couverture* décrite dans Beygelzimer et al. (2006). Pour définir l'arbre de couverture, il est nécessaire d'introduire dans un premier temps les deux définitions de *séparabilité* et de *couverture* suivantes :

Définition 2 (Séparabilité) *Soit un ensemble S d'éléments définis dans un espace métrique. Soit* dist *une distance sur cet espace et soit θ un réel.*

On dit que l'ensemble S vérifie la condition de séparabilité de seuil θ si, pour tout $u, v \in S$, on a $dist(u, v) > \theta$.

Définition 3 (Couverture) *Soient un ensemble S d'éléments définis dans un espace métrique et S' un sous-ensemble de S. Soit* dist *une distance sur cet espace et soit θ un réel.*

On dit que l'ensemble S' vérifie la condition de couverture (de seuil θ) de l'ensemble S si, pour tout $u \in S$, il existe $v \in S'$ tel que $dist(u, v) \leq \theta$.

Un arbre de couverture T défini pour un ensemble d'éléments S est un arbre multi-niveaux. Le niveau 0 correspond aux feuilles de l'arbre, et le niveau le plus élevé (dénoté i_{max}) correspond à la racine. A chaque niveau i, est associé un réel θ_i tel que $\theta_i < \theta_{i+1}$. Soit S_i l'ensemble des éléments au niveau i.

Définition 4 (Arbre de couverture) *L'arbre de couverture T défini pour l'ensemble d'éléments S vérifie les propriétés suivantes pour tout i :*

- *$S_i \subset S_{i-1}$*
- *S_i est une couverture de l'ensemble S_{i-1}*
- *S_i vérifie la propriété de séparabilté*

L'arbre de couverture n'est pas nécessairement unique : le contenu des familles peut varier en fonction de l'ordre dans lequel les sommets sont traités car la construction de l'arbre de couverture dépend de l'ordre d'insertion. Auber (2003) propose un algorithme de dessin de graphes incrémental pour la visualisation progressive de la structure à afficher. Le principe est de calculer un ordre sur les sommets afin d'afficher progressivement (en proposant à l'utilisateur le rendu à différentes étapes de calcul du dessin) la structure du graphe. Dans ce contexte, il est important de faire apparaître dès les premiers calculs la forme générale du graphe afin d'offrir à l'utilisateur une vision globale de la structure qu'il souhaite visualiser. Pour ce faire, il utilise un ordre sur les sommets : en considérant les sommets par ordre inverse de leur nombre de Strahler, il semble que la vue schématique de l'image finale (moins de 10% des éléments affichés) donne une bonne représentation de l'allure générale du graphe. Nous avons donc choisi, pour construire l'arbre de couverture d'un graphe, de traiter les sommets dans l'ordre inverse de leur nombre de Strahler, positionnant ainsi les sommets les plus importants comme représentants des familles de sommets.

La procédure d'étiquetage des sommets est décrite par l'algorithme 1,

- T_u désigne le sous-arbre enraciné en u,
- $prof_T(u)$ désigne la distance du sommet u à la racine de l'arbre T
- $\sigma(u)$ correspond au nombre de Strahler du sommet u

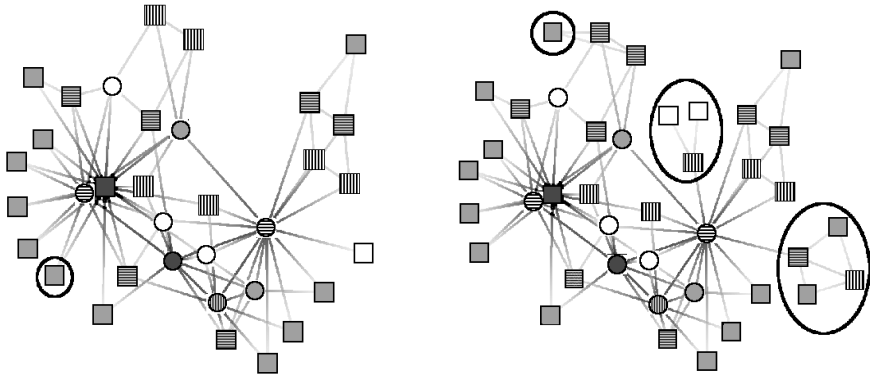
```
1  S := V ∪ V'
2  trier S par σ décroissant;
3  T := arbre de couverture de l'ensemble trié S
4  R := {u ∈ T | prof_T(u) = i}
5  Entier L := 1;
6  Pour u ∈ R Faire
7      U := v ∈ T_u
8      Pour v ∈ U Faire
9          λ(v) := L
10     FinPour
11     L := L + 1;
12 FinPour
```

Algorithme 1 – *Procédure d'étiquetage des familles*

Nous n'avons pas abordé dans ces travaux le problème de l'évolution des graphes. La modification des graphes nécessite de recalculer la classification des sommets. Cependant, seuls les sommets dont le vecteur de caractéristiques a été altéré (et leurs successeurs dans l'arbre de couverture) doivent être reconsidérés pour la classification. Ainsi, il est possible d'adapter la méthode à des graphes dynamiques en reconstruisant partiellement l'arbre de couverture pour la classification des sommets.

Par la suite, on note $\mathcal{F}_l$ la famille de sommets u d'étiquette $\lambda(u) = l$. La figure 2 montre un exemple de calcul de familles de sommets ϵ-équivalents sur l'ensemble des sommets de deux versions du classique graphe "club de karaté" de Zachary (1977). Les formes texturées représentent les différentes valeurs d'étiquettes, une même forme texturée correspondant à une même étiquette. Le premier (Figure 2.a) correspond aux données originales, que nous avons légèrement modifié manuellement en ajoutant des sommets et des arêtes (les éléments

qui diffèrent sont détourés dans les figures) pour obtenir un second graphe (Fig. 2.b). Nous pouvons d'ores et déjà remarquer des ressemblances au niveau du contenu [2].



(a) (b)

FIG. 2 – *Exemple de classification en familles de sommets ϵ-équivalent. (a) le graphe original du club de karaté et (b) le même graphe après modifications.*

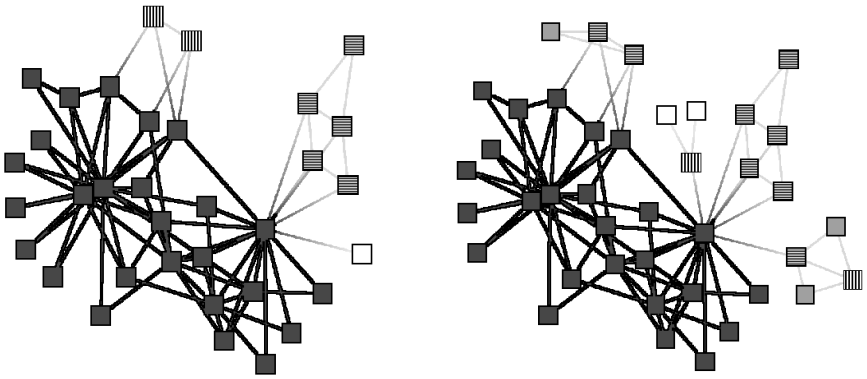
Complexité Il a été prouvé par Don (2006) que l'algorithme de construction de l'arbre de couverture d'un ensemble de points S défini sur un espace métrique E a pour complexité $O(2^{2dd} n \log A)$, avec dd la dimension doublante de E associée à l'arbre, et $A = \frac{d_{max}}{d_{min}}$ l'*aspect ratio* de l'ensemble S (d_{min} et d_{max} étant les distances minimale et maximale respectivement entre les paires d'éléments distincts de S).

La phase de classification des sommets des graphes de l'exemple de la figure 6 (135 et 170 sommets) des réseaux de migrations alternantes 40 ms sur un processeur de 1,8 GHz. Pour deux graphes représentant le réseau aérien mondial de 1542 et 1781 sommets respectivement, le temps de calcul est de 820 ms.

3 Construction des motifs similaires

Si l'on considère $u \in G$ et $u' \in G'$ tels que $\lambda(u) = \lambda(u')$, alors les caractéristiques structurelles de u sont proches de celles de u' car les sommets sont dits ϵ-similaires. Il est probable que les sous-graphes G_u issu de u et $G_{u'}$ issu de u' soient similaires. La procédure de reconnaissance de motifs considère donc de telles paires de sommets $\{u, u'\}$. La construction du motif similaire se fait par propagation : l'algorithme traite les sommets depuis les sommets considérés u et u' vers les voisins identifiés comme appartenant au motif, et récursivement sur les voisins des voisins. Les sommets considérés au départ sont les sommets appartenant à la famille dont le représentant a la plus forte valeur de Strahler. On note que toute décision est définitive : si un ensemble de sommets est considéré comme appartenant à un motif, alors on ne reviendra pas sur cette décision. La figure 3 montre un exemple de propagation. Le principe général de l'heuristique consiste, pour chaque paire de sommets ϵ-similaires, à évaluer

[2] Le lecteur est invité à consulter la version couleur de l'article pour plus de lisibilité.



(a) (b)

FIG. 3 – *Exemple de reconnaissance des motifs correspondant à la classification de la figure 2*

la *similarité* de leur voisinage. On utilise pour cela une mesure de dissimilarité $D(C, C')$ entre deux ensembles de sommets C et C' définie, pour chacune des deux variantes. Cette mesure de dissimilarité se base sur la distribution des étiquettes des sommets de C et C'. Intuitivement, elle peut être assimilée à une distance d'édition entre les étiquetages des sommets de C et C' en considérant les opérations d'ajout et de suppression uniquement.

Si $D(C, C')$ est inférieure à un seuil de similarité τ fixé, alors on considère que les deux étiquetages sont cohérents et les ensembles de sommets sont dits τ-similaires : C et C' correspondent à un même motif. On énonce la définition de motifs τ-similaires comme suit :

Définition 5 (Motifs τ-similaires) *On dit que deux ensembles de sommets étiquetés C et C' sont τ-similaires si la mesure de dissimilarité $D(C, C')$ vérifie :*

$$D(C, C') \leq \tau$$

La procédure de reconnaissance est appliquée récursivement sur les sommets nouvellement inclus dans le motif.

Nous proposons ici deux versions de l'heuristique pour l'identification de motifs par propagation, correspondant à deux différentes stratégies de comparaison sur la similarité des sous-structures. Une troisième a été proposée par Chevalier et al. (2007a) dans le cadre des applications au géniel logiciel. Cette troisième version correspond à la figure 4.d.

Dans la première version, nous cherchons à mettre en correspondance les portions des voisinages directs par famille. Dans l'exemple de la figure 4.a, on considère qu'il existe un trop grand écart de cardinalité entre les ensembles de voisins de u et de v étiquetés par un rond. Ainsi, ces sommets voisins ne sont pas inclus dans le motif représenté par la zone de la figure 4.b. Par contre, les voisinages étiquetés d'un triangle sont assez proches en terme de cardinalité pour être inclus dans le motif. Ainsi nous avons :

Définition 6 *Soient $C_l(u)$ et $C_l(u')$ les ensembles des voisins de u et u' respectivement d'étiquette l. La distance d'édition d'étiquetage D_1 se calcule comme suit :*

$$D_1(C_l(u), C_l(u')) = abs(|C_l(u)| - |C_l(u')|)$$

Dans la deuxième version, on examine les ensembles des sommets voisins dans leur intégralité : il est nécessaire que les voisinages complets soient proches. Dans cette version, la construction des motifs similaires se fait “par niveaux” : on propage toujours sur l’ensemble des voisinages si ces derniers se correspondent. Ainsi, pour une paire de sommets, soit tous les voisins sont inclus dans le motif, soit aucun. Dans l’exemple de la figure 4.a, les ensembles des voisins de u et de v sont trop différents pour être considérés comme appartenant à des motifs similaires. Par contre, les compositions des voisinages de u_3 et u'_5 sont suffisamment proches pour constituer des motifs similaires (voir Fig. 4.c).

On utilise alors la mesure de dissimilarité suivante

Définition 7 *Soient $C(u)$ et $C(u')$ les ensembles des voisins de u et u' respectivement, quelle que soit leur étiquette. La mesure de dissimilarité D_2 est définie par*

$$D_2(C(u), C(u')) = \sum_{l \in [1,\cdots,L]} D_1(C_l(u), C_l(u'))$$

Complexité Nous n’avons pas prouvé la complexité globale de l’étape de propagation des sommets. Nous donnons cependant quelques indications.

Soient $G = (V, E)$ et $G' = (V', E')$ deux graphes et soit L le nombre de familles de sommets ϵ-similaires définies sur G et G'. Le nombre de comparaisons de sommets pour la phase de propagation est au pire $\sum_{i \in [1,L]} |\mathcal{F}_i \cap V||\mathcal{F}_i \cap V'|$. Dans ce cas, toutes les paires de sommets sont testées, ce qui implique que la propagation échoue systématiquement. Dans la pratique, ce cas extrême est très peu probable, ainsi, dès qu’une étape de propagation est effectuée, l’ensemble des sommets appartenant au motif nouvellement identifié sont exclus du processus de propagation par la suite. Ce qui a pour effet de réduire le nombre de comparaisons.

Pour une étape de propagation de motifs issus des sommets u et u', on effectue au pire L comparaisons (on compare le nombre de successeurs de u et de v appartenant à chacune des classes). Si la propagation est validée, alors le processus est récursif sur les sommets nouvellement aggrégés au motif, mais ces sommets ne seront plus considérés pour la propagation par la suite.

La phase de reconnaissance de motifs par propagation dans les graphes de l’exemple de la figure 6 (135 et 170 sommets) a été effectuée en 1s sur un processeur de 1,8 GHz. Pour deux graphes correspondant au réseau aérien mondial (1542 et 1781 sommets), le temps de calcul est de 8s.

Dans la section qui suit, nous présentons quelques applications de la méthode que nous venons de présenter.

4 Exemples d’application

L’heuristique de reconnaissance de motifs similaires présentée dans ce papier est générique et peut être adaptée à de nombreux domaines d’applications. Cette méthode a été utilisée pour la reconnaissance d’objets extraits de flux vidéo à très basse résolution (Chevalier et al. (2007b)). Le principe de l’application est de retourner à l’utilisateur les objets de la base de données qui correspondent le mieux à un objet requête. Un exemple de l’application sur deux requêtes (vignettes détourées) est montré sur la figure 5. Dans cet exemple, les 5 objets les plus

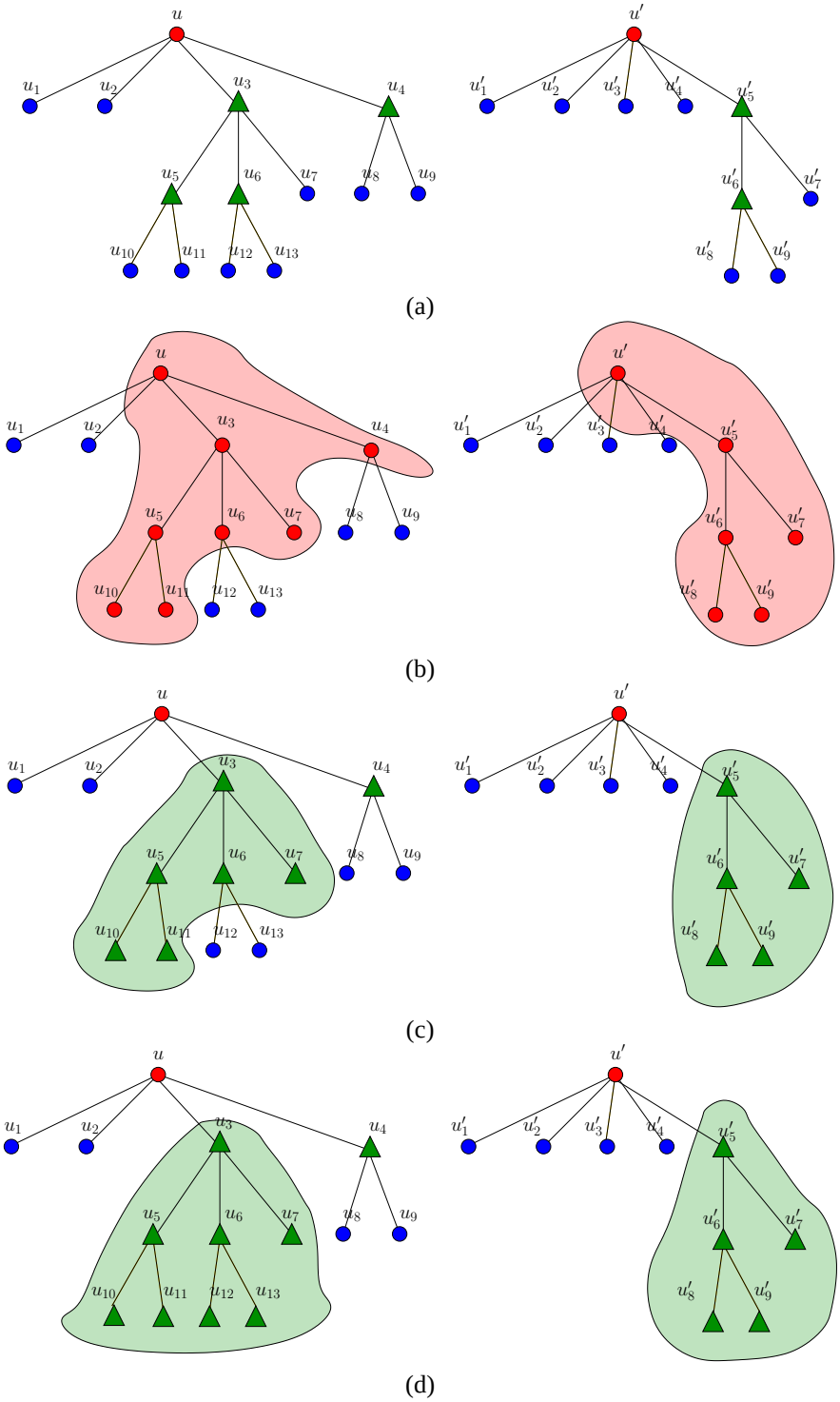


FIG. 4 – *Les différentes versions de propagation d'étiquette. (a) Deux graphes étiquetés en familles de sommets ϵ-similaires et (b) Version 1 : propagation sur les voisins par classe, (c) Version 2 : propagation sur l'ensemble des voisins et (d) Version 3 : propagation sur le sous-DAG (ou sous-arbre).*

similaires sont proposés à l'utilisateur. Dans cette application, on construit les graphes par segmentation à partir des objets de la vidéo et on applique la deuxième variante de l'heuristique pour déterminer les portions communes entre deux objets.

FIG. 5 – *Exemple d'une requête d'objet dans une base de données vidéo.*

Dans le cadre du projet ANR SPANGEO, nous avons appliqué cette méthode à l'analyse des migrations professionnelles en 1975 et en 1982 dans la région Provence Alpes Côte d'Azur. Les graphes sont orientés et les arêtes sont dirigées de la ville de résidence vers la ville de travail. Nous n'avons conservé que les arêtes du graphe telles que l'effectif dépasse 8 personnes. La première variante de la méthode est utilisée. La figure 6 montre le résultat de la reconnaissance de motifs quasi-similaires entre les années 1975 (Fig. 6.a) et 1982 (Fig. 6.b). On peut par exemple remarquer la présence d'un large motif commun entre les deux réseaux. Sur la figure 6, les sommets appartenant à un motif quasi-similaire sont représentés en foncé. Sur la figure 7, nous avons isolé le plus large motif quasi-similaire (en violet dans la version couleur). Dans cet exemple, 50 % des sommets sont communs aux deux motifs, avec notamment des villes pivots comme Le Cannet, Saint-Laurent-du-Var, Carros et Villeneuve-Loubet. Les géographes du projet SPANGEO ont considéré les résultats comme étant pertinents et utiles à leur analyse.

5 Conclusion et perspectives

Nous avons présenté dans cet article une méthode pour la reconnaissance de motifs similaires dans des graphes. En proposant deux variantes correspondant à deux différentes stratégies de tolérance sur la notion de similarité de structure, nous avons défini une méthode générique qui peut être adaptée à de nombreuses applications dans divers domaines tels que la biologie (Auber et al. (2006)) ou le multimédia (Chevalier et al. (2007b)).

Les applications basées sur la méthode ayant donné lieu à des publications concernent dans tous les cas des structures orientées sans cycles (DAGs ou arbres) dans lesquelles un parcours hiérarchique peut être effectué. Nous envisageons, en perspective de ces travaux, d'approfondir la méthode pour des applications nécessitant la comparaison de graphes non orientés tels que les réseaux géographiques, ou les réseaux sociaux, et l'optimisation des algorithmes pour permettre de traiter efficacement les graphes dynamiques en limitant les calculs nécessaires à l'introduction de nouvelles données.

Un autre point important à soulever dans le cadre de cette étude concerne l'évaluation et la validation de la méthode. En effet, si sur des applications particulières les résultats des comparaisons permettent de déterminer les performances de la méthode dans chaque cas précis,

il reste difficile de valider l'efficacité de l'algorithme dans le cas général de la comparaison de graphes. En effet, il n'existe pas à notre connaissance de jeu de données de référence pour lequel les motifs à retrouver sont exhaustivement définis, si ce n'est dans le cadre d'une application précise dans un domaine d'application particulier. Nous envisageons de constituer un tel jeu de données de référence pour permettre une évaluation précise des performances et limitations de la méthode proposée, ainsi que des ses avantages et inconvénients par rapport aux méthodes existantes.

Références

Auber, D. (2003). *Graph drawing software*, Chapter Tulip - A Huge Graph Visualization framework, pp. 105–126. Verlag.

Auber, D., M. Delest, J. Domenger, et S. Dulucq (2006). Efficient drawing and comparison of rna secondary structure. *Journal of Graph Algorithms and Applications 10*(2), 329–351.

Auber, D., M. Delest, J. Domenger, P. Ferraro, et R. Strandh (2003). EVAT : Environment for visualization and analysis of trees. In *IEEE Symposition on Information Visualisation Contest*, Volume www.cs.umd.edu/hcil/iv03contest/, pp. 124–126.

Auber, D., M. Delest, J. Fédou, J. Domenger, et P. Duchon (2004). New strahler numbers for rooted plane trees. In M. Drmota, P. Flajolet, D. Gardy, et B. Gittenberger (Eds.), *Third Colloquium on Mathematics and Computer Science, Algorithms, Trees, Combinatorics and Probabilities*, Trends in Mathematics, pp. 203–215. Vienna University of Technology : Birkhauser.

Baxter, I., A. Yahin, L. M. ans M. Sant'Anna, et L. Bier (1998). Clone detection using abstract syntax trees. In *IEEE International Conference on Software Maintenance*, Bethesda, MD, USA, pp. 368–377.

Beygelzimer, A., S. Kakade, et J. Langford (2006). Cover trees for nearest neighbor. In *ACM International Conference Proceeding Series, Proceedings of the 23rd international conference on Machine learning*, Volume 148, Pittsburgh, Pennsylvania, pp. 97–104. ACM Press.

Chevalier, F., D. Auber, et A. Telea (2007a). Structural analysis and visualization of c++ code evolution using syntax trees. In *9th International Workshop on Principles of Software Evolution (IWPSE'07)*.

Chevalier, F., M. Delest, et J. Domenger (2007b). A heuristic for the retrieval of objects in video in the framework of the rough indexing paradigm. *Signal Processing : Image Communication on Content-Based Multimedia Indexing (SPIC) 22*, 622–634.

Demirci, M. F., A. Shokoufandeh, Y. Keselman, L. Bretzner, et S. Dickinson (2006). Object recognition as many-to-many feature matching. *International Journal of Computer Vision 69*(2), 203–222.

Don, A. (2006). *Indexation et navigation dans les contenus visuels : approches basÃl'es sur les graphes*. Ph. D. thesis, University of Bordeaux 1.

Fekete, J. et C. Plaisant (2003). Infovis contest 2003 - visualization and pair wise comparison of trees. In IEEE (Ed.), *IEEE Symposium on Information Visualization*, Volume www.cs.umd.edu/hcil/iv03contest/.

Gomila, C. et F. Meyer (2003). Graph-based object tracking. In *IEEE International Conference on Image Processing (ICIP)*, Volume 2, pp. 41–44.

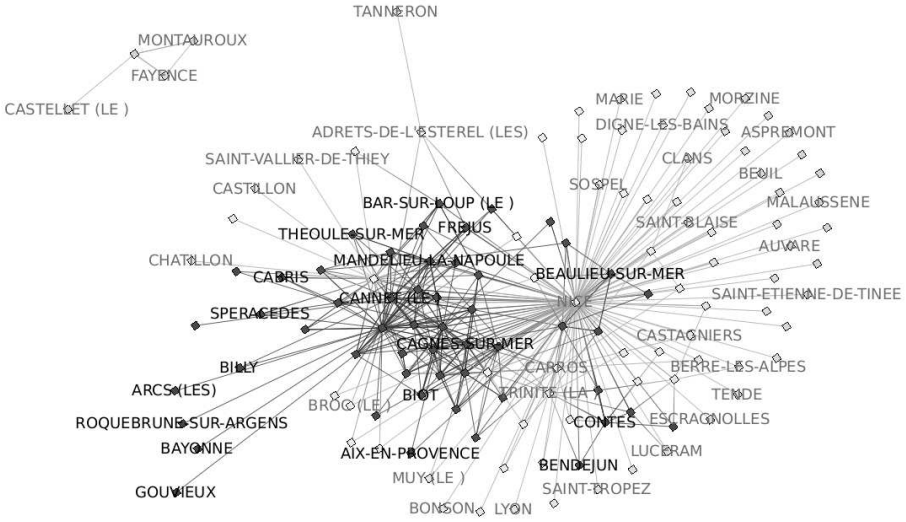
Lladós, J., E. Martí, et J. Villanueva (2001). Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*(10), 1137–1143.

Yan, X., F. Zhu, J. Han, et P. Yu (2006). Searching substructures with superimposed distance. In *ICDE '06. Proceedings of the 22nd International Conference on Data Engineering*, pp. 88–97.

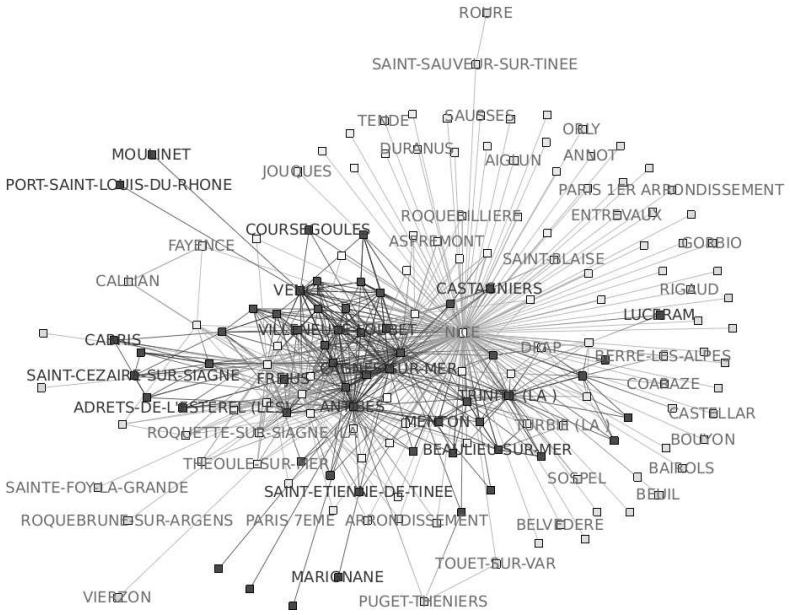
Zachary, W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research 33*, 452–473.

Summary

Detecting similar patterns in graphs is known to be a difficult task. We tackle this problem by designing a heuristic based on intrinsic metrics for graphs, looking for patterns where metric values are coupled with topology. Our algorithm allows us to detect similar patterns common to a series of graphs. We give applications to geographical data and to feature detection in video data.

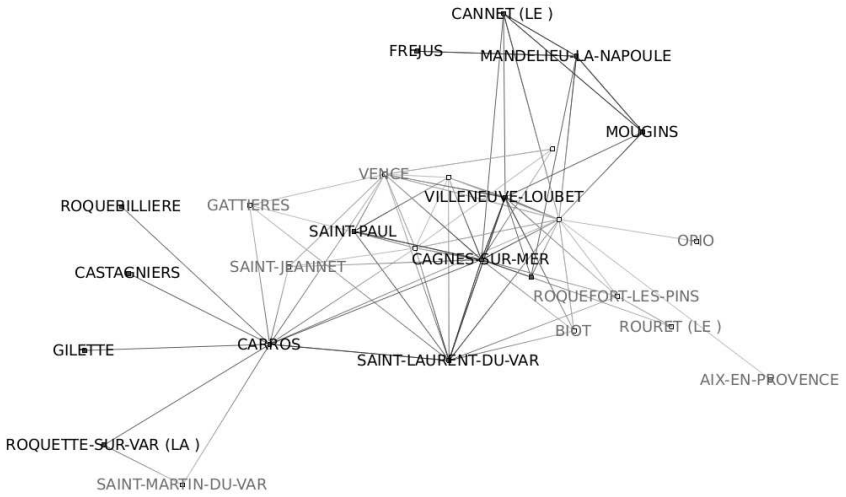


(a) 1975

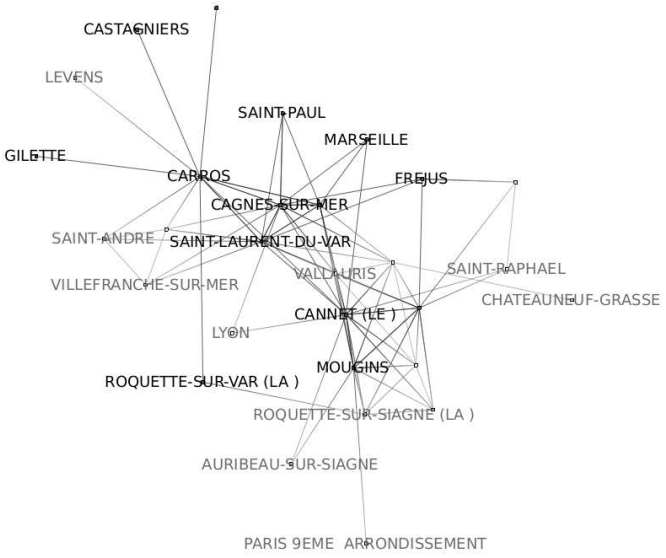


(a) 1982

FIG. 6 – *Réseaux de migrations professionnelles : reconnaissance de motifs par propagation d'étiquette. Les motifs similaires sont représentés en foncé (dans la version couleur, une même couleur représente un motif quasi-similaire).*



(a) 1975



(a) 1982

FIG. 7 – *Réseaux de migrations professionnelles : un motif quasi-similaire. Les sommets communs aux deux graphes sont représentés en foncé (en vert dans la version couleur).*

DagMap : exploration interactive de relations d'héritage

P-Y. Koenig* *** et G. Melançon** ***

*CNRS UMR 5506 LIRMM, Montpellier, France
Pierre-Yves.Koenig@lirmm.fr,
http ://www.lirmm.fr/∼koenig/
**CNRS UMR 5800 LaBRI, Talence (Bordeaux), France
Guy.Melancon@labri.fr
http ://www.labri.fr/perso/melancon
***INRIA Bordeaux – Sud-Ouest, France

Résumé. Cet article décrit le DagMap, une variante du TreeMap qui permet d'explorer des hiérarchies décrivant des relations d'héritage multiple. Ces relations apparaissent naturellement lorsque l'on décrit l'architecture de systèmes orientés objets. Les relations entre les sociétés mères et leurs filiales (à tous niveaux) fournit un autre exemple. Dans ces deux cas, le graphe sous-jacent forme un graphe orienté acyclique (DAG). L'exploration de ces hiérarchies se fait à l'aide d'un TreeMap construit à partir d'un DAG. La gestion du niveau de détails de la visualisation est obtenue en reprenant les idées originales de Furnas et van Ham & van WIjk, étendues aux DagMap. Des interactions adaptées permettent ainsi de faire varier le niveau de détail, d'identifier l'héritage multiple et les généalogies communes.

1 Introduction

Les relations d'héritage apparaissent souvent quand on décrit la conception d'applications orientées objets. Les objets de bas niveau (les classes) spécialisent des classes plus générales (plus abstraites). Ces classes abstraites donnent lieu à plusieurs spécialisations possibles ou classes filles. Typiquement, une classe peut hériter des propriétés de plusieurs classes plus abstraites. Les relations d'héritage apparaissent aussi dans la description de relations entre entités dans d'autres domaines d'application. Par exemple, les relations entre des sociétés et leurs filiales (à tous niveaux). Une société peut avoir différentes filiales et une filiale peut être contrôlée par plusieurs sociétés "mère".

Les relations d'héritage entre entités (objet, sociétés, etc) peuvent être décrites formellement par un graphe $G = (V, E)$. Par définition, les relations d'héritage sont orientées vers les entités de bas niveau ; par conséquence, le graphe résultant est un graphe orienté acyclique (DAG). Le dessin et la visualisation interactive de DAGs est un défits en soit. Les algorithmes de dessin de DAG forment une branche importante de la littérature de dessin de graphe (Graph Drawing Battista et al. (1998); Kaufmann et Wagner (2001)). Ces algorithmes dessinent le plus souvent les sommets sur différents niveaux (cf Fig. 1). Les entités les plus générales – sommets

source – correspondant aux sommets sans ancêtre sont dessinées en haut de la hiérarchie, tandis que les autres sommets sont dessinés suivant leur distance aux sommets sources. La qualité et la lisibilité de ces représentations noeud-lien sont le plus souvent mesuré en fonction de leur capacité à éviter le croisement d'arête (cf Gutwenger et Mutzel. (2004)).

Bien que les diagrammes noeuds-liens soient utiles pour représenter les DAGs, ils sont peu adaptés lorsque l'on veut représenter des données sémantiques à l'aide de la taille et de la couleur des sommets, par exemple. Même si on dessine le DAG avec des sommets de même taille, les différents niveaux doivent être suffisamment éloignés pour assurer une lisibilité du diagramme.

Un espace supplémentaire est nécessaire quand on traite des sommets de taille différente. La lisibilité des arêtes impose aussi de garder un éloignement suffisant entre les niveaux afin d'éviter des arêtes trop à l'horizontal. Même avec des DAGs de taille moyenne, éviter le chevauchement de sommets voisins se fait au détriment de la comparaison de leur taille.

Il est à noter que cela est déjà le cas pour les représentations classiques des arbres. Dans le cas des arbres comme dans celui des DAGs, le dessin est partiellement consacré à la description de la structure du graphe (relation dominante) laissant un espace précieux vide entre les niveaux. Les approches de pavage ("space-filling" Shneiderman (1992)) utilisant tout l'espace apportent une solution pour visualiser les attributs des sommets d'un arbre sacrifiant la représentation de la structure au profit des attributs des feuilles. Le DagMap que nous décrivons ici vise à adapter les approches par pavage aux DAGs, à la fois à l'aide du dessin mais aussi d'une interaction adaptée.

La représentation du DAG par le DagMap requière dans un premier temps de le déployer en un arbre. En revanche, cela nécessite de concevoir des interactions spécifiques afin de recouvrer la structure du DAG et de permettre sa navigation à tous les niveaux. Les cellules du DagMap sont alors liées à travers sa représentation graphique. L'accès aux attributs d'un ancêtre commun, tout en préservant ceux des sommets fils est rendu possible. Cela permet de mettre en évidence le fait qu'un élément de la hiérarchie joue des rôles différents selon sa relation avec chacun de ses ancêtres.

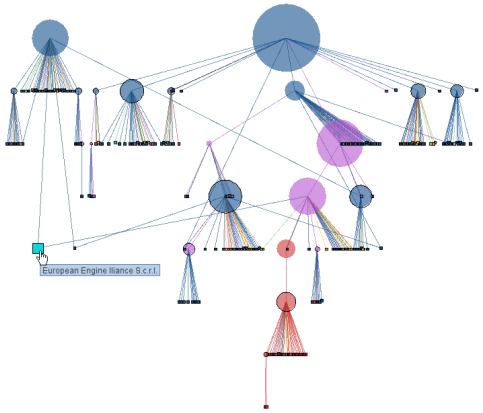


FIG. 1 – *Dessin noeud-lien classique de DAG, avec un attribut mis en évidence par la taille et la couleur des sommets.*

1.1 Extension des TreeMaps aux graphes orientés acycliques

Les TreeMaps sont des techniques de visualisation pour représenter des hiérarchies d'information sur un espace 2D Shneiderman (1992). Les TreeMaps suivent une approche de pavage de l'espace ("space-filling") représentant les sommets feuilles d'un arbre sur des aires contiguës du plan, avec différents artifices visuels rendant compte d'attributs des données. [L'aire elle même peut être calculée selon un des attributs des feuilles.] Cette approche diffère radicalement des représentations classiques noeuds-liens des arbres (voir Battista et al. (1998); Kaufmann et Wagner (2001)) où l'accent est mis sur la position relative des sommets reflétant la *structure* de la hiérarchie, par opposition à la sémantique des données (des sommets feuilles).

Les nombreuses applications exploitant la TreeMap, notamment les applications commerciales, apporte une preuve de son utilisté et de son utilisabilité. Une meilleure interactivité Chintalapani et al. (2004) et une plus grande versatilité Vliegen et al. (2006) en font un bon choix pour la mise en place de système de visualisation de données hiérarchiques (arborescentes dans le cas de Vliegen et al. (2006)).

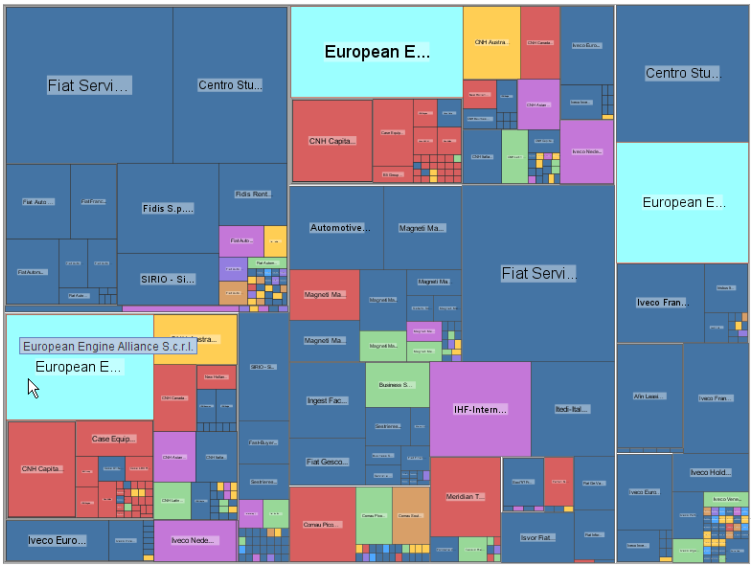


FIG. 2 – *Le DagMap permet de visualiser un DAG au travers d'une TreeMap (ici "squarified" Bruls et al. (2000)). En cliquant sur une cellule, les cellules correspondant à la même entité sont sélectionnées – voir les cellules turquoises avec une aide contextuelle. La filiale sélectionnée apparaît trois fois, montrant qu'elle est contrôlée par trois sociétés de plus haut niveau. Le contrôle de la filiale par ses sociétés mères est relativement équilibré comme le montre la taille comparable des différentes aires.*

Notre travail a été initié par la nécessité d'avoir une visualisation centrée sur les attributs comme les TreeMaps pour explorer des données économiques. En collaboration avec des géographes, nous avons collecté des données décrivant les liens entre différentes sociétés majeures

et leurs filiales (et sous-filiales, etc.). L'exploration de ces données permet aux géographes de comprendre les différents types de relations existant entre ces sociétés et leurs filiales. En visualisant simultanément la localisation géographique ainsi que de la relative importance des sociétés (nombre de filiales, capital relatif, ...) ils ont pu examiner les politiques territoriales contribuant ou s'opposant aux stratégies économiques des sociétés les unes par rapport aux autres. Les relations d'héritage apportant une vue claire sur les attributs géographiques et économiques sont à la base de l'exploration de ces données.

Les questions résolues par la conception du DagMap sont avant tout posées par les données elles-mêmes. Les sociétés et leurs filiales ne sont pas organisées en arbre, mais plutôt en graphe orienté acyclique (DAG), du fait qu'une sociétés peut être contrôlée par différentes sociétés mères. Si la TreeMap apparaît comme le bon choix pour notre visualisation (montrant les attributs que nous devons afficher), on ne peut pas oublier la structure du DAG et simplement extraire un arbre de celui-ci de façon naïve. Nous avons élaboré le DagMap afin de faciliter l'extraction de connaissances et aider les géographes à explorer comment les sociétés organisent et contrôlent leurs activités ou développent leurs stratégies économiques et territoriales à travers leurs filiales. Le DagMap muni d'interactions simples se révèle utile quand on se pose des questions telles que :

- Les sociétés contrôlant une filiale (ou un ensemble de filiales) sont-elles réparties sur plusieurs régions du monde ou concentrées dans une région spécifique ?
- Le contrôle est-il partagé entre sociétés mères et filiales de moyenne importance, ou est-il concentré a la tête de la hiérarchie ? Y a t-il un niveau de la hiérarchie auquel se concentre le contrôle ?
- Les sociétés mère d'une filiale donnée ont-elles une stratégie de développement similaire (sont-elles présentent dans la même zone géographique, couvrent-elles le même secteur industriel, etc.) ?
- La distribution des filiales dans une région du monde obéit-elle à certaines régularités ?

Le jeu de données utilisé ici a été étudié en collaboration avec des géographes [1], nous donnant l'opportunité de travailler en étroite collaboration avec nos utilisateurs-finaux (experts). Le DagMap s'est révélé utile pour l'analyse de ce jeu de données spécifique comme atteste les travaux de nos collègues montrant l'exploitation des représentations et des applications fournies Gautier (2007); Bohan et al. (2007).

Comme le montrent les questions précédentes, le visualisation doit représenter simultanément la structure hiérarchique et rendre compte des différents attributs (localisation géographique, atouts des sociétés, etc.). Les attributs sont représentés de façon efficace sur le DagMap, tandis que la structure peut être explorée au moyen d'interacteurs spécifiques. La possibilité de faire varier le niveau de détail souhaité dans le DagMap permet une exploration complète de la hiérarchie (voir sections 3, 4).

La section suivante introduit la notion du DagMap extension de TreeMaps au graphe orienté acyclique(DAGs). Nous allons par la suite revenir sur les travaux de Furnas (1986) et van Wijk et van Ham (2004) et décrire comment le niveau de détail ou le degré d'abstraction (DOA), d'un DAG peut être déterminé par la structure même du DAG (section 3.1). Différentes interactions sur le DagMap vont être décrites, basée sur le mécanisme du DOA (section 3.1) sur la structure du DAG ...(section 4). Le DagMap et les interactions ont été implémentés en utilisant la boîte à outil *prefuse* Heer et al. (2005). Les interactions ont été

[1] Dans le cadre du projet ANR MDCO SPANGEO. Voir l'URL www.s4.org

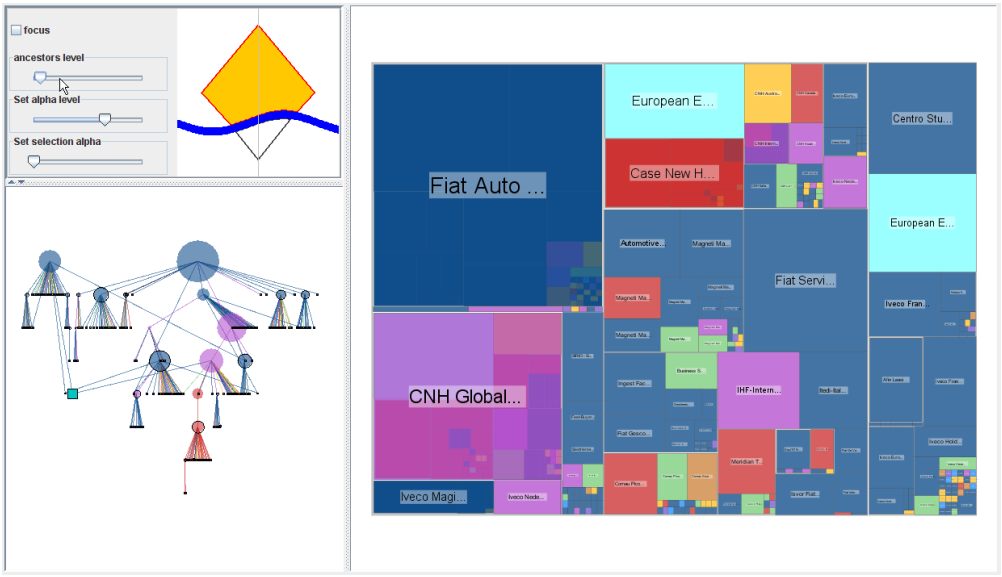


FIG. 3 – *L'application DagMap. La taille relative des fenêtres de droite et de gauche peut être ajustée. Les vues des différentes fenêtres sont synchronisées. Lorsqu'on sélectionne un sommet, celui-ci est animé et voit sa saturation varier pour faciliter sa localisation. Les attributs des sommets peuvent être facilement distingués dans la fenêtre de droite. La partie en haut à gauche est dédiée au contrôle et à l'interaction des différentes vues (voir section 3).*

réalisés et testés pour notre jeu de données en collaboration avec nos experts. Les interactions sont décrites ici d'un point de vue algorithmique et analytique, mais répondent à des besoins de nos utilisateurs experts.

2 Calcul du DagMap

L'algorithme de dessin des TreeMaps peut être adapté afin de pouvoir gérer des DAGs. Le DAG est déployé en un arbre en dupliquant les sommets ayant plusieurs pères afin que chaque fils aient un seul père (Fig. 4). Il est à noter qu'on obtient pas un arbre à proprement parler mais plutôt un ensemble d'arbres (une forêt : un arbre pour chaque sommet source). La description plus formelle de l'algorithme de déploiement du DAG ne sera pas présenter ici (voir Koenig et al. (2007)).

Du fait de la duplication de sommets, le DagMap doit être équipé d'interactions de base afin de permettre à l'utilisateur de visualiser combien un élément est reparti dans la hiérarchie. Quand on sélectionne un élément – une cellule – du DagMap, il est mis en évidence ainsi que toute les cellules du DagMap correspondantes.

Dans notre exemple (voir Fig. 2), une filiale révèle sa présence dans différentes régions du DagMap quand le contrôle est distribué par différentes sociétés mères distinctes – ceci est par-

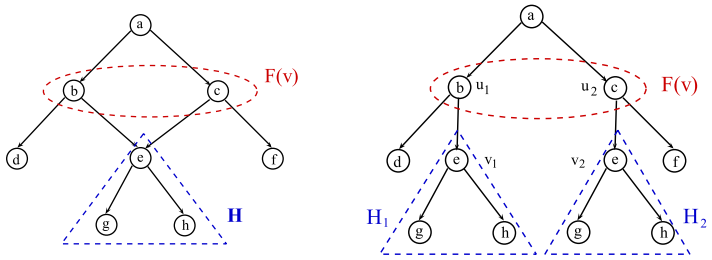


FIG. 4 – *Un arbre étiqueté est obtenu du DAG par duplication des sommets (étiquetés) avec des ancêtres multiples.*

ticulièrement utile quand les filiales voisines appartiennent à différentes régions géographique. Le code couleur utilisé dans la Fig. 2 a été établi par les géographes (bleu pour L'Europe, rouge pour L'Amérique du nord, jaune pour l'Asie, vert pour L'Amérique du sud et marron pour L'Afrique). Certaines cellules sont en fuschia pour indiquer l'appartenance à un paradis fiscal).

La taille de l'arbre obtenue du DAG peut être potentiellement très large, dépendant du nombre d'arêtes transversales. Ce problème n'est pas gênant quand on traite des DAGs peu denses (ce qui est notre cas). Il est à noter, comme c'était le cas dans DynaDags North et Woodhull (2002), que la plus part des techniques traitant des DAGs souffrent de cette limitation (voir aussi Melançon et Herman (2000), Gutwenger et Mutzel. (2004)). Cette limitation est contournée par le fait que l'arbre n'est calculé qu'à partir d'une partie du DAG (voir section 3.1), permettant de garder la taille de l'arbre sous contrôle.

3 Visualisation par niveau de détail

Le premier ingrédient que nous utilisons pour notre visualisation repose sur la notion de "coupe" dans le DAG. Le DagMap est construit à partir d'une sélection de certains éléments du DAG se trouvant à des niveaux contigus (mais pas forcément situés sur un *même* niveau) et recouvrant toute sa largeur. Un deuxième ingrédient permet à l'utilisateur d'obtenir plus de détail autour d'un élément du DAG.

Le calcul de cette "coupe" repose sur différents paramètres qui peuvent être modifiés par le biais d'un interacteur en forme de losange. (Cet interacteur apparaît en haut à gauche de l'application, Fig. 3). La forme de cette interacteur reflète la forme général d'un DAG. La ligne incurvée couvrant l'interacteur peut être déplacée de haut en bas et est utilisée pour faire varier la coupe et la sélection du niveau de détail auquel le DagMap sera construit (voir Fig 5 gauche).

L'interacteur permet d'explorer les données en suivant des scénarios naturels. Sur notre exemple de sociétés et filiales, supposons que l'utilisateur s'intéresse aux filiales, mais seulement à celles se situant à une certaine distance des sociétés de plus haut niveau. Ceci peut être obtenu facilement en déplaçant la ligne incurvée vers le haut ou vers le bas. En cliquant sur la partie au dessus de la ligne incurvée, le DagMap est construit à partir des sommets de la coupe

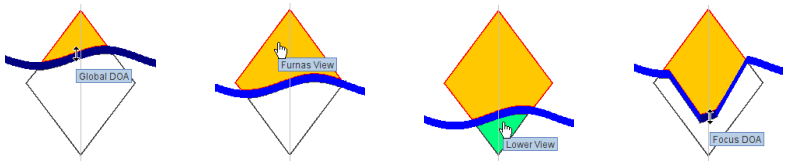


FIG. 5 – *Un interacteur en forme de losange est utilisé pour induire une coupe du DAG, en le déplaçant de haut en bas – image de gauche. (La forme de l'interacteur reflète la forme général d'un DAG.) Le DagMap est alors calculé en partant des éléments de la coupe jusqu'aux éléments au niveau le plus haut (sommet racine) – image de droite.*

et l'ensemble des sommets au dessus de celle-ci, jusqu'aux sociétés de de plus haut niveau (Fig. 5 seconde image). La distance ici n'a pas été mesurée seulement seulement en terme de sociétés intermédiaires rencontrées jusqu'aux sociétés de plus haut niveau, mais dépend du % de contrôle exercé à chaque niveau. Ceci explique pourquoi les sommets ne sont pas choisis par niveau dans la hiérarchie mais aussi selon leurs attributs (groupe de sociétés, % de contrôle, etc.). Le % de contrôle exercée par une société sur une filiale est une information contenue dans le jeu de données et peut donc être composée sur plusieurs niveaux. Nous reviendrons sur cet aspect plus loin.

(L'interacteur permet notamment de cliquer sur la partie en dessous de la ligne incurvée, afin de construire le DagMap à partir des sommets de la coupe et l'ensemble des sommet en dessous de celle-ci (Fig. 5 troisième image)).

Maintenant, supposons que l'utilisateur porte une attention particulière sur un élément et en fait son centre d'intérêt, afin d'obtenir plus de détail le concernant. Cela se traduit, dans le calcul de la coupe, par le besoin de donner priorité aux éléments proches de cet élément sélectionné, tout en donnant moins d'importance aux éléments éloignés.

Cette situation est modélisée dans l'interacteur en permettant à l'utilisateur de jouer avec la forme de la ligne incurvée, permettant à la coupe de plonger autour du sommet sélectionné tout en gardant un niveau de détail élevé pour les éléments éloignés de celui-ci (Fig. 5 droite). L'application se met en mode "focus" en cochant un bouton de contrôle dans l'interface (voir Fig. 3), l'utilisateur peut jouer avec la ligne incurvée et définir combien de détail est requis autour de l'élément focus.

3.1 Niveaux, statistiques sur les sommets et "coupe"

Nous allons maintenant détailler comment les paramètres sont contrôlés par l'interacteur en forme de losange, expliquant comment on étend l'idée de Furnas (1986) et de van Wijk et van Ham (2004) aux graphes orientés acycliques. Observons que la coupe et la sélection de la partie au dessus de l'interacteur est utilisée pour filtrer nos données. Cette technique de filtrage que nous avons implémeté requière une hiérarchisation des données. Dans la plus pare des cas, cette hiérarchisation est réalisée sous la forme d'un arbre (binaire). Un exemple utilisé par Furnas est celui d'un code source C, contenant implicitement une hiérarchisation en bloque d'instructions.

La situation étudiée par van Wijk et van Ham (2004), de même que par Quigley et Eades (2000) ou encore Schaffer et al. (1996), est un graphe muni d'un clustering hiérarchique. Le graphe est équipé d'une structure d'arbre où les éléments du graphe d'origine correspondent aux feuilles de l'arbre. Les sommets internes correspondent à des clusters regroupant les feuilles du sous arbres dont il est la racine.

Nous allons montrer comment ces mécanismes développés pour les arbres fonctionnent aussi lorsque la hiérarchie est un DAG. Un exemple de coupe est illustrée dans la figure Fig. 6. Revenons sur le cas des arbres.

La structure d'arbre joue un rôle fondamentale dans le processus de suppression dans le but d'obtenir une vue abstraite des données, et repose sur le calcule de statistique (indice) pour les sommets de l'arbre. Dans le cas de Furnas, la statistique associée à un sommet v, est appelé le niveau *à priori* d'intérêt (*API*) et correspond à son niveau $\ell(v)$ dans l'arbre (exprimé par des valeurs négatives 0, -1, -2, ...), reflétant combien une instruction est profonde dans le code. Une vue abstraite du code source peut être obtenue en sélectionnant des instructions avec un niveau supérieur à un certain seuil. Suivant une stratégie similaire, Herman et al. (1998) ont utilisé différentes statistiques sur les sommets [2] pour déclencher dynamiquement le déploiement du sous-arbre induit par un sommet donné, produisant ainsi une vue abstraite de la hiérarchie. Ces deux cas sont toutefois particuliers puisque la structure d'arbre utilisé pour la visualisation *coïncide* avec structure de données à visualiser.

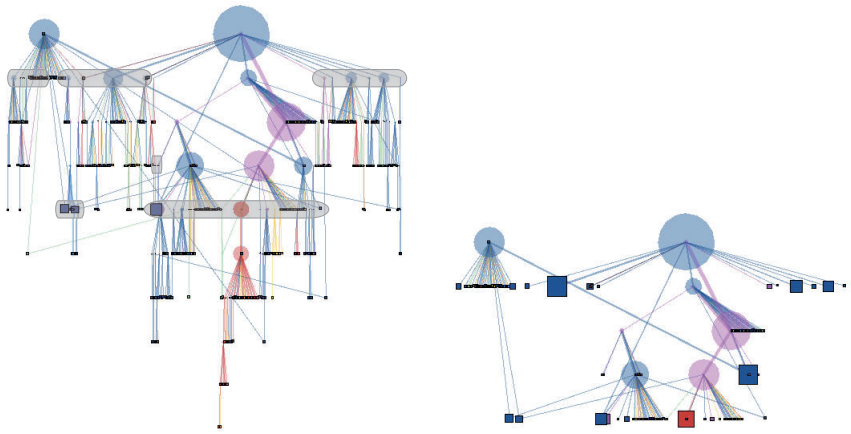


FIG. 6 – *La région grise (image de gauche) entour les sommets identifiés comme membre de la coupe (ici avec un $DOA = 0.3$). L'image de droite montre le DAG après avoir pris la partie* supérieur *du DAG "à la Furnas" pour construire le DagMap*

Cette situation est différente dans le cas de van Wijk et van Ham (2004) qui produisent une vue abstraite pour un graphe basée sur une structure d'arbre distinct du graphe visualisé (à l'exception des sommets feuilles qui correspondent aux sommets du graphe d'origine) (van Wijk et van Ham, 2004, Section 4.1) utilisent une valeur DOA, avec $DOA \in [0, 1]$, afin de

[2] Une statistique possible est le nombre de feuilles d'un sous-arbre, ou encore le nombre de Strahler d'un sommet d'un arbre.

selectionner les clusters formant ainsi une abstraction du graph. Un DOA de 0 correspond à la vue la plus détaillée du graphe et sélectionne l'ensemble des sommets du graphe d'origine (le niveau le plus bas correspond aux feuilles dans l'arbre). Un DOA de 1 retourne la vue la plus abstraite du graph constituée des clusters les plus haut (racine de l'arbre de clustering). Pour un $DOA \in [0, 1]$, les clusters C' (dont le père est noté C) sont ceux qui satisfont l'inégalité suivante :

$$d_{C'} \leq d_{root} \cdot DOA < d_C. \quad (1)$$

La racine étant l'élément le plus abstrait de l'arbre ($DOA = 1$), le DOA choisi est appliqué à la valeur de la racine ($d_{root} \cdot DOA$). Les clusters C' sélectionnées sont ceux se trouvant immédiatement au-dessous de cette valeur ($d_{C'} \leq d_{root} \cdot DOA$), tandis que leur père C sont au-dessus ($d_{root} \cdot DOA < d_C$).

[Observons que l'approche de van Ham & van Wijk ne diffère pas fondamentalement de celle de Furnas. En effet, nous pouvons transformer l'exemple de Furnas en attribuant à chaque sommet v dans l'arbre (une instruction) une valeur égale à $d_v = 1/(1 - \ell(v))$, où $\ell(v)$ est le niveau du sommet v dans l'arbre. La racine r a donc la valeur $d_r = 1$, les sommets au niveau $\ell(v) = -1$ ont pour valeur 1/2, ceux au niveau $\ell(v) = -2$ ont pour valeur $1/3$ ... Pour chaque valeur du DOA tel que $1/a < DOA \leq 1/(a + 1)$ sélectionne les sommets au niveau $\ell(v) = -a$.]

L'équation Eq. (1) peut être directement appliquée aux DAGs pour déterminer une coupe à un certain degré d'abstraction (DOA). Étant donné une statistique (un indice) sur les sommets C dans le DAG (croissante des feuilles aux racines), et une valeur de DOA implicitement choisie en utilisant l'interacteur (Fig. 5 gauche), les éléments satisfaisant Eq. (1) forment une coupe. Il y a une grande différence entre les deux configurations : la coupe calculée à partir d'un arbre forme une *antichaine* – si l'arbre est vu comme un diagramme de Hasse d'un ensemble ordonné, ce qui n'est pas le cas dans un DAG. La coupe dans un DAG couvre simplement l'ensemble des sommets puits (sans successeur).

van Ham & van Wijk montre comment les inéquations (1) et (2) peuvent être adaptées pour tenir compte d'un sommet *focus* dans l'arbre (van Wijk et van Ham, 2004, Section 4.2). Supposons que l'utilisateur a sélectionné un élément f, qui peut être un élément visible sélectionné directement à l'écran ou à travers une requête sur les données. L'inégalité peut être adaptée comme suit :

$$d_{C'} \leq d_{root} \cdot DOA(|C' - f|) \; ; \; d_{root} \cdot DOA(|C - f|) < d_C \quad (2)$$

afin de sélectionner des éléments de bas niveau qui sont proches du focus tout en sélectionnant des éléments de plus haut niveau plus éloignés du focus. Afin de décider si un sommet C' doit être sélectionné, on prend en compte sa distance $|C' - f|$ au focus. En d'autres mots, on utilise ici une fonction *croissante* $DOA : [0, \infty[to[0, 1]$ qui varie selon la distance au focus f (voir van Wijk et van Ham (2004) pour une discutions plus détaillée sur les options possibles.)

Après avoir sélectionné un élément, l'utilisateur peut jouer avec l'interacteur, variant sa forme, afin d'influer le DOA utilisé pour la sélection de la coupe.

3.2 Variation sur les statistiques des sommets

Différents statistiques (indices) sur les sommets du DAG (appelé d_C dans les équations Eqs. (1) et (2)) peuvent être calculées. Le niveau d'un sommet v dans le DAG en est un exemple

(égale à la longueur du chemin le plus long entre un sommet source et le sommet v). Le nombre de Strahler étendu aux DAGs Herman et al. (1999) est un autre bon candidat.

4 Exploration de la structure hiérarchique à travers le DagMap

Notre expérience avec des utilisateurs finaux a clairement montré la nécessité de visualiser la structure de la hiérarchie directement sur le DagMap. Nous permettons donc à l'utilisateur de visualiser comment les sommets ancêtres couvrent les cellules dans le DagMap. Après avoir sélectionné une coupe, l'utilisateur peut définir une "bande" rassemblant tout les éléments à une certain distance au dessus de la coupe. Ce faisant, l'utilisateur peut visualiser comment les éléments dépendent de leurs ancêtres.

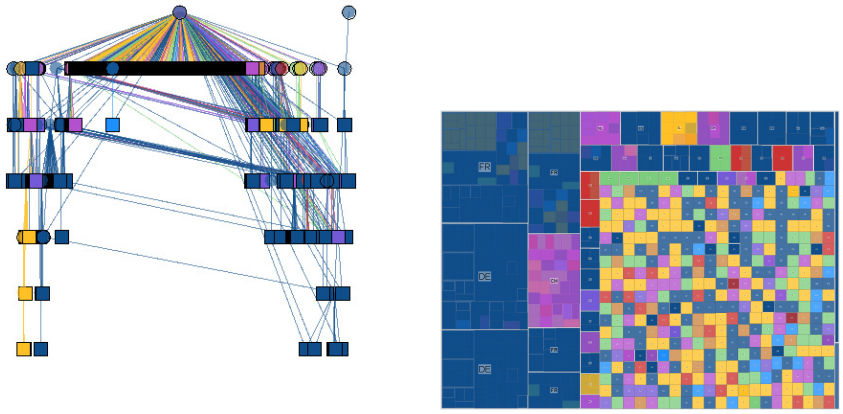


FIG. 7 – *Le DAG décrit les liens entre Neslé et ses filiales. Même si le DAG n'est pas dense, le diagramme noeud-lien ne permet pas de savoir comment les filiales sont distribuées dans les différentes régions du monde, ou comment les sociétés de haut niveau exercent leur contrôle sur les filiales de bas niveau. Le DagMap facilite la détection de motifs (formes particulières) formé par les attributs des sommets (couleur / aire). Le voile transparent mis sur des cellules de haut niveau sur le DagMap permet de faire des hypothèses sur les stratégies de gestion conduites par des sociétés de niveau plus haut.*

En faisant varier la hauteur de la bande, l'utilisateur obtient des informations sur les attributs des cellules d'ancêtre et la taille de son voisinage. Cette fonctionalité s'est révélée très utile sur notre jeu de données, rendant évident le contrôle de filiales par des sociétés jouant le rôle de paradis fiscaux (cellules de couleur fuchsia), comme le montre l'exemple de la figure Fig 10. Dans d'autres situations, les géographes ont pu constaté que des filiales sud-américaines et/ou asiatiques se trouvant au niveau le plus bas de la hiérarchie étaient gardées sous le contrôle des sièges sociaux européens. Comparez par exemple la région gauche du DagMap dans les figures Fig. 2, Fig. 10 et Fig. 11.

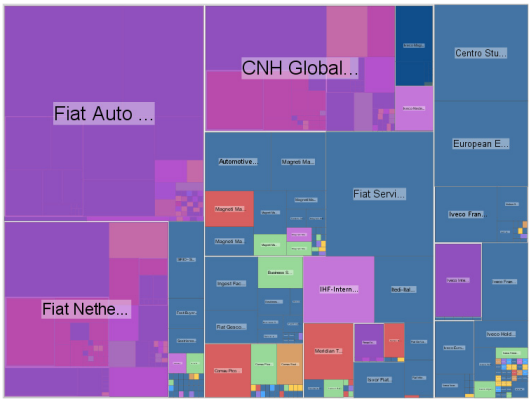


FIG. 8 – *Le DagMap montré ici est obtenue à partir de celui de la figure Fig 2 auquel on a rajouté une bande (Fig 11). L'emboîtement des filiales devient clair quand on peut visualiser les sociétés mère se trouvant au dessus d'elles. Le code couleur indique que les sociétés sont contrôlées par des sociétés mère suspectées de jouer le rôle de paradis fiscaux (Fiat Netherland, en bas à gauche ; CNH Global, en haut au milieu).*

5 Conclusion et travaux futurs

Nous avons présenté le DagMap généralisant les TreeMaps aux graphes orientés acycliques, vus en tant que structures d'héritage générales. Le DagMap ainsi que les interactions que nous avons décrites ont été conçus avec l'aide d'utilisateurs experts. Notre cas d'étude est particulièrement adapté aux techniques présentées ici puisque l'ensemble de données est intrinsèquement codé en DAG, mais également parce que les besoins des utilisateurs requis le développement d'une technique combinant astucieusement la visualisation et la comparaison des attributs des données et de la structure hiérarchisée dans une même visualisation.

Nous avons également appliqué le DagMap pour visualiser les dépendances entre les modules dans la distribution Ubuntu (Linux). Les modules sont structurés de manière hiérarchiques, avec les modules de base et essentiels au bas de la hiérarchie (comme les bibliothèques C) – voir le Fig. 13. Cet exemple diffère de notre cas d'étude principal parce que le DAG d'Ubuntu est plus dense, rendant une visualisation noeud-lien complètement inutilisable. L'aire des cellules dans le DagMap indique combien d'espace disque est exigé lors de l'installation d'un package (et les sous-packages requis). Quand l'espace disque est un critère central, la visualisation aide à faire la différence entre les package incontournables et ceux qui sont facultatifs, aidant potentiellement à faire un choix entre les environnements de bureau possibles.

Notre travail est dans une certaine mesure proche des Flexible TreeMaps proposés par Chintalapani et al. (2004), du fait que les deux techniques permettent à l'utilisateur de changer dynamiquement la visualisation lorsque l'exploration évolue. Nous voyons également le DagMap, concernant les graphes orientés acycliques comme un compétiteur d'autres techniques associant les TreeMaps (codant un arbre couvrant) avec des représentations noeuds-liens pour des graphes (voir Fekete et al. (2003)).

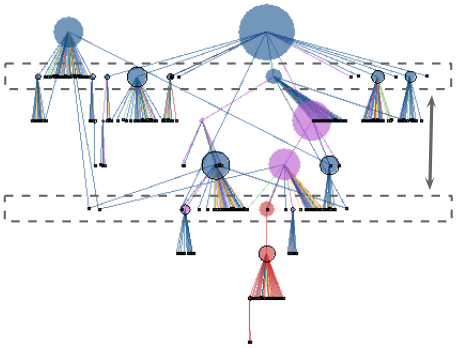


FIG. 9 – *Après sélection d'un coupe, l'utilisateur peut définir une bande situé au dessus de la coupe et visualisée sur le DagMap en tant que cellules colorées superposées. La variation de la hauteur de cette bande fournit un retour visuel direct sur le DagMap.*

Les interactions que nous avons développées peuvent être employées comme des outils du niveau de détail sur le DagMap. Un interacteur en forme de losange intuitif, implémentant un mécanisme de filtrage de DOA adapté de van Wijk et van Ham (2004), ont été conçus et employés avec succès. Bien que le DagMap soit fondamentalement une approche de remplissage d'espace se concentrant sur des attributs de sommet, la structure du DAG peut être récupérée en recouvrant d'un voile les cellules au-dessus d'autres.

Le fait que nos collègues géographes ont employé le DagMap de façon quasi-quotidienne pour explorer leurs données et diffuser leurs résultats démontre en quelque sorte l'utilité de notre technique. Nous prévoyons néanmoins effectuer une expérimentation contrôlée pour établir objectivement les avantages et inconvénients de notre méthode.

Dans un futur proche, nous prévoyons d'étudier les situations où des données sont groupées avec des recouvrement qui s'imbriquent et se chevauchent. En effet, une hiérarchie de tels groupe peut être décrite à l'aide d'un DAG. Un exemple classique est celui où des concepts de base (ou des mots-clés) apparaissent comme des cas particuliers de concepts de plus haut niveau. L'ambiguïté intrinsèque de la langue implique qu'un concept ou un mots-clé appartient à plusieurs groupes distincts. Pouvoir observer directement cette ambiguïté ou la dispersion d'un concept dans une collection de documents pourrait être utile pour percevoir la structure logique des documents et comprendre comment les concepts sont organisés.

Remerciements. Nous souhaitons exprimer ici nos remerciements à notre collègue Laurent Perrier qui a effectué la collecte du jeu de données, et à nos collègues Bérengère Gautier, Charles Bohan et Céline Rozenblat qui ont utilisé notre application avec grand enthousiasme. Nous tenons aussi à remercier la société BVD qui a mis à notre disposition les données sous forme brute.

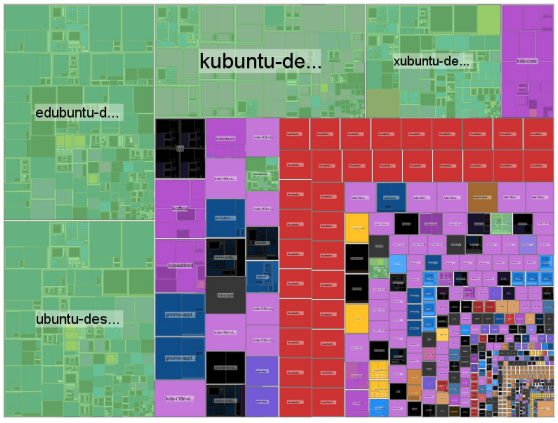


FIG. 10 – *Après sélection d'une coupe, l'utilisateur peut définir une bande se trouvant au-dessus de la coupe et visualisée sur le DagMap en tant que cellules colorées superposées.*

Références

Battista, G. d., P. Eades, R. Tamassia, et I. G. Tollis (1998). *Graph Drawing : Algorithms for the Visualisation of Graphs*. Prentice Hall.

Bohan, C., B. Gautier, C. Rozenblat, D. Auber, et P.-Y. Koenig (2007). Cities networks through multinational firms networks : A multi-level graph approach. In *15th European Colloquium on Theoretical and Quantitative Geography*, Zurich, Switzerland.

Bruls, M., K. Huizing, et J. J. van Wijk (2000). Squarified treemaps. In W. d. Leeuw et R. v. Liere (Eds.), *Joint Eurographics and IEEE TCVG Symposium on Visualization (Data Visualization '00)*, Amsterdam, pp. 33–43. Springer-Verlag.

Chintalapani, G., C. Plaisant, et B. Shneiderman (2004). Extending the utility of treemaps with flexible hierarchy. In *Eighth International Conference on Information Visualisation (IV'04)*, pp. 335–344. IEEE Computer Society.

Fekete, J., D. Wang, N. Dang, A. Aris, et C. Plaisant (2003). Overlaying graph links on treemaps. In *IEEE Information Visualization 2003 Symposium Poster Compendium*, pp. 82–83. IEEE Press.

Furnas, G. W. (1986). Generalized fisheye views. In *Human Factors in Computing Systems CHI '86*, pp. 16–23. ACM Press.

Gautier, B. (2007). Integration of the moroccan cities by the multinational firms of agro-alimentary sector. In *IGU Commission of "Monitoring cities of tomorrow"*, Guangzhou, China, pp. 193–209. Sun Yat Sen University Press.

Gutwenger, C. et P. Mutzel. (2004). An experimental study of crossing minimization heuristics. In B. Liotta (Ed.), *Graph Drawing 2003*, Volume 2912 of *Lecture Notes in Computer Science*, pp. 13–24. Springer-Verlag.

Heer, J., S. K. Card, et J. A. Landay (2005). prefuse : a toolkit for interactive information

visualization. In *SIGCHI conference on Human factors in computing systems*, Portland, Oregon, USA, pp. 421–430. ACM Press.

Herman, I., M. S. Marshall, G. Melançon, D. J. Duke, M. Delest, et J.-P. Domenger (1999). Skeletal images as visual cues in graph visualization. In E. Gröller, H. Löffelmann, et W. Ribarsky (Eds.), *Data Visualization '99, Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pp. 13–22. Springer ŰVerlag.

Herman, I., G. Melançon, et M. Delest (1998). Tree visualisation and navigation clues for information visualisation. *Computer Graphics Forum 17*(2), 153–165.

Kaufmann, M. et D. Wagner (Eds.) (2001). *Drawing Graphs, Methods and Models*, Volume 2025 of *Lecture Notes in Computer Science*. Springer.

Koenig, P.-Y., G. Melançon, C. Bohan, et B. Gautier (2007). Combining dagmaps and sugiyama layout for the navigation of hierarchical data. In *IV '07 : Proceedings of the 11th International Conference Information Visualization*, pp. 447–452. IEEE Computer Society.

Melançon, G. et I. Herman (2000). Dag drawing from an information visualization perspective. In W. d. Leeuw et R. v. Liere (Eds.), *Joint Eurographics and IEEE TCVG Symposium on Visualization (Data Visualization '00)*, Amsterdam, pp. 3–13. Springer-Verlag.

North, S. C. et G. Woodhull (2002). Online hierarchical graph drawing. In *Graph Drawing : 9th International Symposium, GD 2001*, Vienna, Austria, pp. 77–81.

Quigley, A. et P. Eades (2000). Fade : Graph drawing, clustering, and visual abstraction. In J. Marks (Ed.), *Symposium on Graph Drawing (GD 2000)*, LNCS 1984, pp. 197–210. Springer.

Schaffer, D., Z. Zuo, S. Greenberg, L. Bartram, J. Dill, S. Dubs, et M. Roseman (1996). Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interaction 3*(2), 162–188.

Shneiderman, B. (1992). Tree visualization with tree-maps : 2-d space-filling approach. *ACM Transactions on Graphics 11*(1), 92–99.

van Wijk, J. J. et F. van Ham (2004). Interactive visualization of small world graphs. In T. Munzner et M. Ward (Eds.), *IEEE Symposium on Information Visualisation*, Austin, TX, USA. IEEE Computer Science press.

Vliegen, R., J. J. van Wijk, et E.-J. van der Linden (2006). Visualizing business data with generalized treemaps. *IEEE Transactions on Visualization and Computer Graphics 12*(5), 789–796.

Summary

This paper introduces DagMaps, a variation on TreeMaps usefully adapted to interactively explore hierarchies describing inheritance relations. Inheritance relations naturally occur when describing object oriented software architecture, for instance. Another example we studied with geographers describes how world companies relate to their subsidiaries. In all these cases the graph underlying the inheritance relations is a directed acyclic graph (DAG). The exploration of inheritance relations is conducted with the help of a TreeMap built from the DAG. We extend ideas from from Furnas and van Ham & van Wijk in order to interactively select

a cut acting as level of details view on the DAG based on node attributes. Useful interaction enables the user to vary the level of detail, visually identify multiple inheritance as duplicates in the DagMap and locate common inheritance/genealogy in the DAG.

Visualisation interactive de grands ensembles de règles d'association

Cheikh T. Diop*, Arnaud Giacometti*
Patrick Marcel* Marie Ndiaye*,**

*Université Gaston Berger de Saint-Louis, BP 234 Saint-Louis Sénégal
cdiop@ugb.sn,
**Université François-Rabelais de Tours, 3 place Jean Jaures 41000 Blois France
{arnaud.giacometti, patrick.marcel, marie.ndiaye}@univ-tours.fr

Résumé. L'étude des principales techniques de visualisation de règles d'association nous a permis d'identifier un problème majeur des techniques usuelles qui représentent chaque règle par une figure. Ce problème est la difficulté à visualiser un grand nombre de figures, et donc de règles. Dans cet article, nous proposons deux nouvelles techniques de visualisation où une figure représente un groupe de règles. Ces techniques reposent sur les tableaux croisés et les points d'intérêt. Elles permettent de visualiser de grands ensembles de règles en utilisant des mesures d'intérêt globales qui donnent de l'information sur des groupes de règles.

1 Introduction

Les techniques de Data Mining permettent d'extraire des informations ou des modèles intéressants à partir de grandes quantités de données. Les algorithmes particulièrement populaires concernent l'extraction de règles d'association. Ces algorithmes produisent des centaines, voire des milliers de règles. Les règles sont ensuite présentées à l'utilisateur afin que celui-ci puisse analyser la connaissance découverte.

De nos jours, de nombreuses techniques de visualisation de règles d'association sont présentées dans la littérature. Cependant, une étude des techniques de visualisation usuelles nous a permis de déceler un problème lié au volume important des résultats d'extraction. En effet, avec les techniques de visualisation actuelles, l'utilisateur est très vite submergé par le nombre important de figures qui lui sont présentées. Par conséquent, il rencontre des difficultés pour effectuer convenablement l'analyse des connaissances extraites.

1.1 Contribution

L'objectif des techniques de visualisation n'est pas seulement de présenter les règles extraites mais aussi et surtout de visualiser les mesures d'intérêt qui permettent d'évaluer la per-

tinence des règles. La plupart des mesures usuelles sont locales dans le sens où elles donnent de l'information sur une seule règle (voir les travaux de Bruzzese et Buono (2004); Li Yang (2005); Blanchard et al. (2003); Chakravarthy et Zhang (2003); Hofmann et al. (2000)). Le besoin de présenter les règles en leur associant leurs mesures locales conduit à visualiser les règles individuellement en représentant chacune d'elles par une figure, ce qui pose un réel problème lorsque le nombre de règles est très grand. Pour remédier à ce problème, nous proposons, dans ce papier, deux nouvelles techniques de visualisation qui permettent de visualiser les règles par groupe. Nous utilisons des mesures dites globales pour permettre à l'utilisateur d'évaluer la pertinence des groupes de règles. Une mesure globale est une mesure qui donne de l'information sur un groupe de règles ayant des propriétés communes. Les mesures globales que nous utilisons sont des mesures agrégées car elles sont calculées à partir de mesures locales de plusieurs règles. La première technique repose sur les tableaux croisés et la deuxième est basée sur les points d'intérêt. Avec les tableaux croisés, on représente de manière absolue les valeurs de mesure globale des groupes de règles construits. Avec les points d'intérêt, on représente des ratios entre les valeurs de mesure globale des groupes de règles construits.

1.2 Organisation de l'article

Le reste de l'article est organisé comme suit : nous effectuons un survol des techniques de visualisation usuelles dans la section 2. Ensuite nous présentons la description générale d'une fonction de visualisation, dans la section 3. Puis, nous présentons deux nouvelles techniques de visualisation dans la section 4. Enfin, nous concluons et présentons nos perspectives dans la section 5.

2 Etat de l'art des techniques de visualisation usuelles

Il existe plusieurs techniques de visualisation de règles d'association dans la littérature. Ces techniques utilisent des repères très variés tels que les coordonnées parallèles (voir Bruzzese et Davino (2003); Inselberg (1985); Bruzzese et Buono (2004)), les matrices (voir Yahia et Nguifo (2004); Chakravarthy et Zhang (2003); Zhao et al. (2005)), les graphes (Bruzzese et Buono (2004); Klemettinen et al. (1994)) ou encore des outils géométriques (voir Hofmann et al. (2000); Blanchard et al. (2003)). Dans cette section, nous présentons un état de l'art rapide des techniques usuelles permettant de visualiser des règles individuellement. Nous nous intéressons aux techniques qui visualisent des ensembles de règles. Les règles visualisées sont constituées de deux itemsets qui sont la tête et le corps et de mesures d'intérêt locales. Les mesures d'intérêt les plus utilisées sont le support et la confiance. Une mesure moins connue appelée utilité d'item est aussi utilisée (voir Bruzzese et Davino (2003)).

2.1 Les coordonnées parallèles

Elles permettent de représenter des données multidimensionnelles en utilisant des axes parallèles (voir Inselberg (1985)). Dans les travaux de Bruzzese et Davino (2003), un axe représente un attribut. Il est gradué par des items (figure 1). Dans les travaux de Li Yang (2005) un axe représente un item (figure 2). Dans ces deux types de visualisation, les règles sont représentées par des lignes qui rencontrent les axes. Pour la visualisation présentée dans

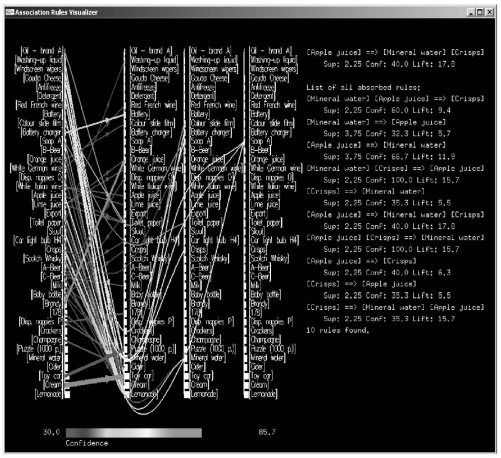


FIG. 1 – *Copordonnées parallèles présentées par Bruzzese et Davino (2003)*

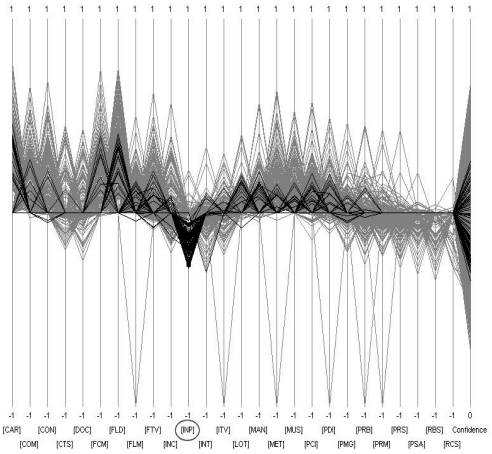


FIG. 2 – *Copordonnées parallèles présentées par Li Yang (2005)*

les travaux de Bruzzese et Davino (2003), une ligne relie les items qui interviennent dans la règle qu'elle représente alors que pour celle qui est proposée dans les travaux de Li Yang (2005), une ligne coupe un axe en fonction de la valeur d'utilité de l'item correspondant à l'axe pour la règle représentée par la ligne. Dans les travaux de Bruzzese et Davino (2003), seule l'utilité des items pour la règle est présentée sur la visualisation. Elle correspond à l'ordonnée du point d'intersection entre l'axe et la ligne de la règle. Dans les travaux de Li Yang (2005), l'épaisseur et la couleur de la ligne représentent respectivement le support et la confiance de la règle correspondante.

2.2 Les graphes

Pour les graphes, la tête et le corps des règles correspondent aux nœuds et les règles sont matérialisées par des arcs entre les nœuds (voir figure 3). Bruzzese et Buono (2004), représentent les itemsets par des nœuds, alors que dans les travaux de Hao et al.; Appice et Buono (2005) les nœuds représentent des items. Le support et la confiance des règles sont représentés par la longueur, l'épaisseur ou la couleur des arcs, ou ils sont directement inscrits sur l'arc reliant deux nœuds.

2.3 Les matrices

Wong et al. (1999) utilisent une matrice 3D pour visualiser les règles (voir figure 4). Des items sont placés en ligne et des barres représentant les mesures d'intérêt sont placées à l'extrémité des colonnes. Une règle est représentée sur une colonne entière. Pour spécifier qu'un item intervient dans la règle, un cube est placé sur la cellule correspondant à l'intersection

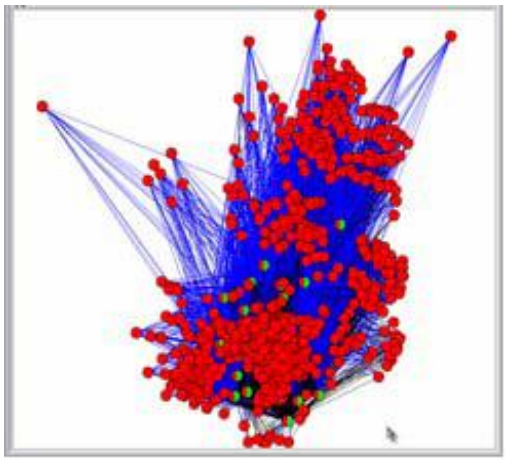


FIG. 3 – *Graphes*

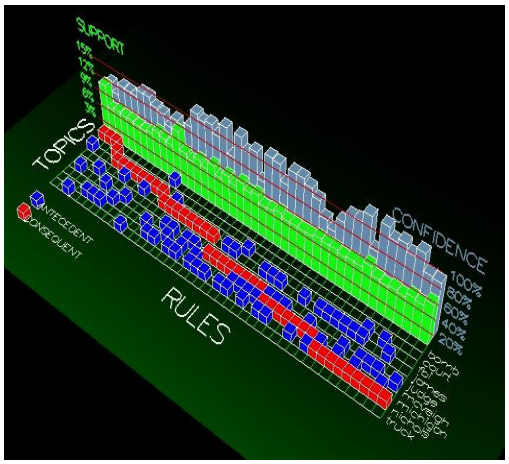


FIG. 4 – *Matrice*

de la ligne de l'item et de la colonne de la règle. La couleur du cube permet de déterminer l'appartenance de l'item à la tête ou au corps de la règle.

La technique proposée par Chakravarthy et Zhang (2003) est proche de la technique proposée par Wong et al. (1999) à la différence que les items présents dans les têtes et les corps des règles sont placés respectivement en ligne et en colonne, et les mesures d'intérêt sont représentées par des figures différentes.

2.4 Les mosaic plots

Etant donnés des attributs avec leurs valeurs, les mosaic plots proposés par Hofmann et al. (2000) représentent toutes les règles d'association impliquant les valeurs de ces attributs. Chaque règle correspond à un rectangle dont la surface totale est fonction du support et la couleur à l'intérieur du rectangle est fonction de la confiance de la règle (voir figure 5).

2.5 Problème posé

Toutes les techniques exposées ci-dessus présentent un inconvénient qui est la difficulté à visualiser un grand ensemble de règles. Ce problème s'explique par le fait que les règles sont représentées individuellement dans les visualisations. En effet, lorsqu'il y a beaucoup de règles alors il y a beaucoup d'informations à faire apparaître sur la visualisation. Ce qui se traduit sur la visualisation, soit par un nombre important de figures, soit par l'absence de certaines figures. Du coté de l'utilisateur, dans le premier cas, il a une vue globale de l'ensemble de règles mais il dispose de trop d'informations (chaque figure fait apparaître au moins une caractéristique de la règle qu'elle représente), ce qui rend l'analyse des règles difficile. Dans le second cas, il n'a pas de vue globale de l'ensemble de règles. Pour résoudre ce problème, nous proposons dans ce papier deux nouvelles techniques de visualisation qui permettent de présenter les règles par

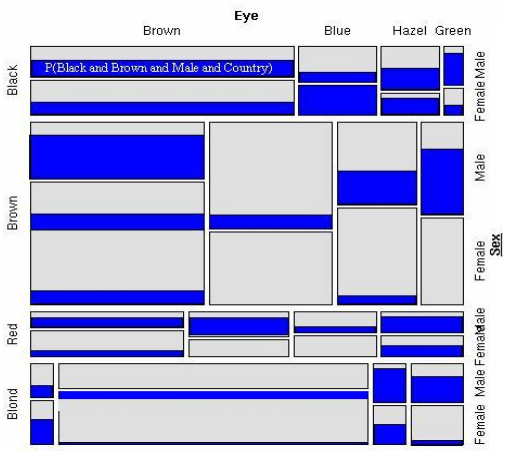


FIG. 5 – *Mosaic Plots avec les valeurs des attributs Hair, Eye et Sex pour le corps et la valeur France de l'attribut Country pour la tête des règles*

groupe. Le fait de visualiser les règles par groupe nous permet d'avoir une vue globale de l'ensemble de règles avec une quantité d'information réduite. Dans la section 3, nous proposons une description générale d'une fonction de visualisation que nous utilisons pour présenter ces nouvelles techniques.

3 Fonction de visualisation

Dans cette section, nous nous intéressons aux fonctions qui permettent de passer d'un ensemble de règles et de paramètres de visualisation à une visualisation. Mais avant de donner une définition de ces fonctions, nous précisons d'abord à la section 3.2 la forme des règles à visualiser. Puis, nous introduisons une définition formelle d'une visualisation à la section 3.3 et une description des paramètres de visualisation à la section 3.4. Ensuite, nous présentons à la section 3.5 la description formelle d'une fonction de visualisation et un algorithme générique qui utilise cette fonction pour construire une visualisation.

3.1 Définitions de base

Dans ce papier, étant donné un ensemble d'attributs $\mathcal{A}$ et un attribut A de $\mathcal{A}$, on note $dom(A)$ le domaine de l'attribut A et $dom(\mathcal{A})$ le produit cartésien des domaines des attributs de $\mathcal{A}$, i.e. $dom(\mathcal{A}) = \times_{A \in \mathcal{A}} dom(A)$. Pour tout tuple $t \in dom(\mathcal{A})$, on appelle schéma de t l'ensemble des attributs $\mathcal{A}$.

Par ailleurs, soit All une constante n'appartenant à aucun des domaines $dom(A)$ ($A \in \mathcal{A}$). On note $dom(A)^+$ le domaine de l'attribut A auquel nous ajoutons la constante All, i.e. $dom(A)^+ = dom(A) \cup \{All\}$. Enfin, on note $dom(\mathcal{A})^+$ le produit cartésien des ensembles

$dom(A)^+$ ($A \in \mathcal{A}$), i.e. $dom(\mathcal{A})^+ = \times_{A \in \mathcal{A}} dom(A)^+$. Le rôle de la constante All est précisé à la section suivante.

3.2 Données et motifs

Dans ce travail, les motifs auxquels nous nous intéressons sont des règles d'association. Soit $\mathcal{A}$ un ensemble d'attributs. Nous considérons que $\mathcal{A}$ est le schéma de la table à partir de laquelle les règles sont extraites. Un item x défini sur $\mathcal{A}$ est un couple (attribut, valeur) noté (A, a) avec $A \in \mathcal{A}$ et $a \in dom(A)$. Par la suite, un item (A, a) est aussi noté $(A = a)$ ou plus simplement a si aucune ambiguïté n'est possible, i.e. si les domaines des différents attributs sont disjoints. Enfin, un itemset X défini sur $\mathcal{A}$ est un ensemble d'items définis sur $\mathcal{A}$.

Par la suite, nous notons $Items$ l'ensemble des items, i.e. $Items = \{(A = a)/A \in \mathcal{A}, a \in dom(A)^+\}$, et $\mathcal{P}(Items)$ l'ensemble des parties de $Items$.

Exemple 3.1. *Considérons un ensemble d'attributs* $\mathcal{A} = \{Fruit, Ville, Année\}$ *avec :*
$dom(Fruit) = \{Pomme, Orange\}$, $dom(Ville) = \{Blois, Tours\}$ *et* $dom(Année) = \{2000, 2005\}$.
$\{Blois, 2000\}$ *et* $\{Orange, Tours\}$ *sont des itemsets qui appartiennent à* $\mathcal{P}(Items)$.

Soit un k-itemset $X = \{(A_{i_1} = a_{i_1}), ..., (A_{i_k} = a_{i_k})\}$ défini sur $\mathcal{A}$, on appelle schéma de X noté $sch(X)$ l'ensemble d'attributs $\{A_{i_1}, ..., A_{i_k}\} \subseteq \mathcal{A}$. De plus, on appelle extension de X noté X^+ l'ensemble d'items défini par : $X^+ = X \cup \{(A = All) \mid A \in \mathcal{A} \setminus sch(X)\}$. Intuitivement, on ajoute à X tous les items $(A = All)$ tels que A est un attribut de $\mathcal{A}$ n'apparaissant pas à $sch(X)$. Cela permet de représenter tous les itemsets selon le même ensemble d'attributs $\mathcal{A}$.

La notion d'itemset étant maintenant définie, un ensemble de règles R est vu comme une instance de relation de schéma $Schéma\_Règle = \{Identifiant, Corps, Tête, Support, Confiance\}$ où $dom(Corps) = \mathcal{P}(Items)$, $dom(Tête) = \mathcal{P}(Items)$, $dom(Support) = dom(Confiance) = [0, 1]$ et $dom(Identifiant)$ est un ensemble de valeurs numériques ou de chaînes de caractères qui identifient de manière unique les règles de R. Dans la suite, l'ensemble de toutes les règles possibles est noté $Règles$ et l'ensemble des parties de $Règles$ est noté $\mathcal{P}(Règles)$.

Dans notre approche, nous considérons qu'une règle d'association $r \in Règles$ est un tuple constitué d'un identifiant $id(r)$, de deux itemsets qui sont respectivement le corps $corps(r)$ et la tête $tête(r)$ de la règle, de la valeur de support $sup(r)$ et de la valeur de confiance $conf(r)$ de la règle. Une telle règle sera notée $r =< id(r), corps(r), tête(r), sup(r), conf(r) >$.

Exemple 3.2. $R = \{< r_1, \{Orange\}, \{Blois\}, 0.6,\ 0.2 >, < r_2, \{Pomme\}, \{Blois\}, 0.5, 0.3 >\}$ *est un ensemble constitué de deux règles.*

Le tableau 1 présente quelques notations que nous utilisons tout au long de ce papier.

3.3 Visualisation

Dans notre approche, nous appelons visualisation une représentation formelle d'un dessin fait sur un écran.

Notation	Signification
r	une règle
rep	un repère
u	des paramètres de visualisation
f	une figure
$Items$	Ensemble de tous les items
$\mathcal{P}(Items) = 2^{Items}$	Ensemble de tous les itemsets
$\mathcal{P}^2(Items)$	Ensemble de toutes les parties de $\mathcal{P}(Items)$
$Règles$	Ensemble de toutes les règles
$R \subseteq Règles$	Ensemble de règles à représenter
$\mathcal{P}(Règles) = 2^{Règles}$	Ensemble de toutes les parties de $Règles$
$\mathcal{P}^2(Règles)$	Ensemble de toutes les parties de $\mathcal{P}(Règles)$
$\mathcal{R} \subseteq \mathcal{P}(Règles)$	Ensemble de parties de $\mathcal{P}(Règles)$, $\mathcal{R} \in \mathcal{P}^2(Règles)$

TAB. 1 – *Notations*

Définition 3.1. ***Visualisation***
Etant donné un ensemble d'attributs $Schéma\_Visualisation$ appelé schéma de visualisation, une visualisation définie sur $Schéma\_Visualisation$ est un couple $V =< rep, F >$ où :

- *rep est une instance de relation de schéma $Schéma\_Visualisation$, appelée* repère *de V. Les tuples de rep sont appelés les* références *du repère rep. Elles permettent d'interpréter et/ou de comparer les figures de V.*
- *F est une instance de relation de schéma $Schéma\_Visualisation$. Les tuples de F sont les figures de V. Dans notre approche, toute figure $f \in F$ représente un sous-ensemble de règles à visualiser, ce sous-ensemble pouvant être un singleton lorsque les règles sont visualisées au niveau le plus fin (c'est à dire individuellement).*

Exemple 3.3. *Etant donné l'ensemble d'items introduit à l'exemple 3.2, la figure 6 représente une visualisation $V =< rep, F >$ de schéma $Schéma\_Visualisation = \{Corps, Tete, Hauteur, Couleur\}$ avec : $dom(Corps) = dom(Tete) = \mathcal{P}^2(Items)$, $dom(Couleur) = \{Blanche, Grise\}$, et $dom(Hauteur) = [0, 1]$.*

Plus précisément, le repère de V est défini par : $rep = \{\{Orange\}, \{Pomme\}, \{Blois\}\} \times \{\{Orange\}, \{Pomme\}, \{Blois\}\} \times \{null\} \times \{null\}$. Par exemple, la référence $< Orange, Blois, null, null >$ représente les coordonnées de la barre blanche en haut à droite de la figure 6. Quant à l'ensemble des figures de V, il est défini sur cet exemple par $F = \{f_1, f_2\}$ avec $f_1 =< \{Orange\}, \{Blois\}, 0.6, Blanche >$ et $f_2 =< \{Pomme\}, \{Blois\}, 0.9, Grise >$. Intuitivement, la figure f_1 représente sous la forme d'une barre blanche de hauteur 0.6 une règle d'association de corps $\{Orange\}$ et de tête $\{Blois\}$, la couleur (resp. sa hauteur) de la barre ayant été déterminée à partir du support (resp. de la confiance) de la règle en question.

3.4 Paramètres de visualisation

Dans notre approche, le calcul d'une visualisation dépend de paramètres de visualisation permettant d'ajuster certaines des propriétés de la visualisation, à savoir :

- le repère : par le biais de valeurs permettant de calculer des composantes du repère, et/ou

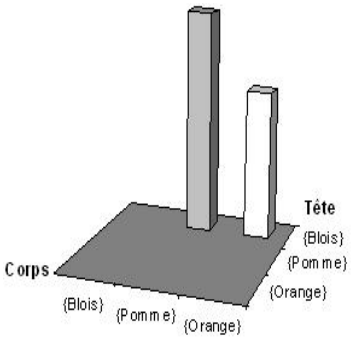


FIG. 6 – *Matrice*

- les règles à représenter : par le biais de critères d'agrégation et/ou
- les figures à visualiser : par le biais de valeurs permettant de calculer les position, forme ou couleur des figures.

Définition 3.2. ***Paramètres de visualisation***
Les paramètres de visualisation constituent un tuple de valeurs qui interviennent dans le calcul du repère et/ou de l'ensemble de règles à visualiser et /ou des figures d'une visualisation.

Exemple 3.4. *Pour les barres de la figure 6, l'apparence des figures est fonction d'un paramètre exprimant un seuil de support (0.4). Les barres correspondant aux règles dont le support est supérieur ou égal à ce seuil portent une même couleur (blanche) et les autres barres portent une autre couleur (grise). De même, le repère est fonction des itemsets à mettre en abscisse et en ordonnée, fournis comme paramètres de visualisation.*

3.5 Fonctions de visualisation

Dans notre approche, une **fonction de visualisation** est une fonction qui construit une visualisation.

Définition 3.3. ***Fonction de visualisation***
Une fonction de visualisation est une fonction composée de trois sous fonctions, $Calcul\_Repère$, $Prétraitement\_Règles$ *et* $Calcul\_Figure$*, qui calcule une visualisation à partir d'un ensemble de règles et de paramètres de visualisation.* $Calcul\_Repère$ *construit le repère de la visualisation,* $Prétraitement\_Règles$ *construit une partition des règles à visualiser, et* $Calcul\_Figure$ *construit pour chaque groupe de règles de la partition calculée une figure de la visualisation.*

$Calcul\_Repère$ associe à un ensemble de règles R et des paramètres de visualisation u, un repère $rep = Calcul\_Repère(R, u)$. Elle a pour signature :

$$\mathcal{P}(Règles) \times dom(Schéma\_Paramètres) \rightarrow \mathcal{P}(dom(Schéma\_Visualisation))$$

$Prétraitement\_Règles$ construit les groupes de règles en fonction du repère et des paramètres de visualisation. $Prétraitement\_Règles$ fait correspondre à un ensemble de règle R,

Algorithme 1 : Construction d'une visualisation

ENTRÉES : un ensemble de règles R et des paramètres de visualisation u
SORTIES : une visualisation V
UTILISE : une fonction de visualisation composée de $Calcul\_Repère$, $Prétraitement\_Règles$ et $Calcul\_Figure$

1: $F = \emptyset$ //Initialise l'ensemble des figures
2: $rep = Calcul\_Repère(R, u)$ //Calcule le repère
3: $\mathcal{R} = Prétraitement\_Règles(R, rep, u)$ //Prétraite les règles
4: **pour tout** $R' \in \mathcal{R}$ **faire**
5: $f = Calcul\_Figure(R', rep, u)$ //Calcule la figure f qui correspond à R'
6: $F = F \cup \{f\}$ //Ajoute la figure f à F
7: **fin pour**
8: $V = (rep, F)$
9: Retourne la visualisation V

FIG. 7 – *Algorithme de construction d'une visualisation*

un repère rep et des paramètres de visualisation u, une partition de R notée $\mathcal{R} = Prétraitement\_Règles(R, rep, u)$. Sa signature est :

$$\mathcal{P}(Règles) \times \mathcal{P}(dom(Schéma\_Visualisation)) \times dom(Schéma\_Paramètres) \rightarrow \mathcal{P}^2(Règles)$$

$Calcul\_Figure$ calcule une figure pour un groupe de règles. Elle fait correspondre à un groupe de règles R', un repère rep et des paramètres de visualisation u, une figure $f = Calcul\_Figure(R', rep, u)$. $Calcul\_Figure$ a pour signature :

$$\mathcal{P}(Règles) \times \mathcal{P}(dom(Schéma\_Visualisation)) \times dom(Schéma\_Paramètres) \rightarrow dom(Schéma\_Visualisation)$$

L'algorithme générique de la figure 7 présente une méthode de construction de visualisation à partir d'une fonction visualisation. L'algorithme prend en entrée un ensemble de règles R et un tuple u représentant les paramètres de visualisation et fournit une visualisation V composée d'un repère rep et d'un ensemble de figures F.

Notons que les fonctions de visualisation que nous décrivons ne dessinent pas les visualisations à l'écran mais servent à calculer les informations nécessaires à l'affichage des dessins.

4 Deux nouvelles techniques de visualisation

Nous distinguons deux types de mesures permettant d'évaluer la pertinence des règles d'association : des mesures locales qui donnent de l'information sur une seule règle et des mesures

globales qui donnent de l'information sur un groupe de règles. Le fait d'utiliser des mesures d'intérêt locales dans la construction d'une visualisation conduit à associer une figure à une règle. Si l'ensemble des règles à visualiser est très grand, l'utilisateur se retrouve avec des informations difficiles à analyser à cause de la quantité importante de figures. L'utilisation de mesures globales permet d'associer un groupe de règles à une seule figure, ce qui permet de réduire la quantité de figures présentée à l'utilisateur.
Dans cette section, nous présentons deux nouvelles techniques de visualisation : la première est une technique basée sur les tableaux croisés et la deuxième est à base de points d'intérêt. La particularité de ces techniques de visualisation est qu'une figure représente un groupe de règles d'association. Nous utilisons des mesures d'intérêt globales agrégées pour construire les figures. Après avoir présenté les nouvelles techniques de visualisation, nous proposons une implémentation de ces techniques.

4.1 Principe de base

Ces deux techniques reposent essentiellement sur deux aspects :
– la construction de groupes de règles à partir de l'ensemble des règles à visualiser et
– le calcul de mesures globales pour les groupes de règles construits.

Dans cette section, nous montrons comment les groupes de règles sont construits et nous présentons les types de mesures globales employées.

4.1.1 Construction des groupes de règles

A partir d'un ensemble de règles d'association, on peut former des groupes de règles en se basant soit sur les valeurs de mesures locales des règles soit sur la tête et/ou le corps des règles. Pour les deux techniques proposées, nous construisons les groupes en nous basant sur les items qui forment la tête et le corps des règles.

Considérons un ensemble de règles R et un ensemble d'attributs $\mathcal{A}$ qui est le schéma de la relation des données de départ. Dans notre approche, les groupes de règles sont formés à partir de R et de deux ensembles d'attributs inclus dans $\mathcal{A}$ et disjoints. Notons $\mathcal{C}$ et $\mathcal{T}$ ces deux ensembles d'attributs. Les groupes de règles sont construits de sorte que l'extension de leur corps contienne un itemset de $dom(\mathcal{C})^+$ et l'extension de leur tête contienne un itemset de $dom(\mathcal{T})^+$. Les règles d'un groupe contiennent le même itemset de $dom(\mathcal{C})^+$ dans leur corps et le même itemset de $dom(\mathcal{T})^+$ dans leur tête. En d'autres termes, si X est un itemset de $dom(\mathcal{C})^+$ et Y un itemset de $dom(\mathcal{T})^+$, un groupe de règles noté R_X^Y est obtenu en sélectionnant les règles de R dont l'extension du corps contient l'itemset X et dont l'extension de la tête contient l'itemset Y. Formellement, R_X^Y est défini par : $R_X^Y = \{r \in R | X \subseteq corps(r)^+ \wedge Y \subseteq tête(r)^+\}$ avec $X \in dom(\mathcal{C})^+$ et $Y \in dom(\mathcal{T})^+$. Dans la suite, nous notons $\mathcal{R}_{\mathcal{C}}^{\mathcal{T}}$ l'ensemble des groupes de règles défini par : $\mathcal{R}_{\mathcal{C}}^{\mathcal{T}} = \{R_X^Y \neq \emptyset | X \in dom(\mathcal{C})^+, Y \in dom(\mathcal{T})^+\}$. $\mathcal{R}_{\mathcal{C}}^{\mathcal{T}}$ est une partition de R.

Exemple 4.1. *Le tableau 2 montre un extrait des données à partir desquelles les règles sont obtenues. Il s'agit de fruits produits en quantité variable dans les villes de Blois et Tours pour les années 2000 et 2005. Dans le tableau 3, nous avons des règles extraites à partir de ces données.*

Si nous considérons l'ensemble de règles $R = \{ r_1, ..., r_{13}\}$ *et les ensembles d'attributs* $\mathcal{C} = \{Fruit\}$ *et* $\mathcal{T} = \{Quantité\}$*, on obtient les groupes de règles suivants :*
$R_{\{Pomme\}}^{\{Petite\}} = \{r_3\}$, $R_{\{Pomme\}}^{\{Moyenne\}} = \{r_1, r_4, r_8\}$, $R_{\{Pomme\}}^{\{Grande\}} = \{r_7, r_{11}\}$, $R_{\{Pomme\}}^{\{All\}} = \{r_{12}\}$,

Producteur	Fruit	Ville	Année	Quantité
0001	Orange	Tours	2000	Grande
0002	Raisin	Tours	2000	Moyenne
0003	Orange	Tours	2005	Moyenne
0004	Pomme	Blois	2000	Grande
0005	Raisin	Blois	2005	Grande
0006	Pomme	Blois	2005	Petite
0007	Poire	Blois	2000	Petite
0008	Poire	Blois	2005	Grande
...	...	...	...	...

TAB. 2 – *Extrait des données*

id(r)	corps(r)	tête(r)	sup(r)	conf(r)
r_1	{Pomme,2000}	{Tours,Moyenne}	0.4	0.5
r_2	{Orange,2000}	{Tours}	0.3	0.5
r_3	{Pomme,2005}	{Tours,Petite}	0.3	0.5
r_4	{Pomme,2005}	{Tours,Moyenne}	0.4	0.7
r_5	{Orange,2005}	{Tours,Grande}	0.3	0.4
r_6	{Orange,2005}	{Tours,Petite}	0.4	0.6
r_7	{Pomme,2000}	{Blois,Grande}	0.5	0.3
r_8	{Pomme,2000}	{Blois,Moyenne}	0.3	0.8
r_9	{Orange,2000}	{Blois}	0.6	0.2
r_{10}	{Orange,2000}	{Blois,Moyenne}	0.4	0.15
r_{11}	{Pomme,2005}	{Blois,Grande}	0.3	0.1
r_{12}	{Pomme,2005}	{Blois}	0.5	0.3
r_{13}	{Orange,2005}	{Blois,Moyenne}	0.4	0.3

TAB. 3 – *Règles d'association*

$R^{\{Petite\}}_{\{Orange\}} = \{r_6\}$, $R^{\{Moyenne\}}_{\{Orange\}} = \{r_{10}, r_{13}\}$, $R^{\{Grande\}}_{\{Orange\}} = \{r_5\}$, $R^{\{All\}}_{\{Orange\}} = \{r_2, r_9\}$, $R^{\{Petite\}}_{\{All\}} = \emptyset$, $R^{\{Moyenne\}}_{\{All\}} = \emptyset$, $R^{\{Grande\}}_{\{All\}} = \emptyset$ *et* $R^{\{All\}}_{\{All\}} = \emptyset$.

Dans un groupe de règles, chaque règle possède ses propres mesures d'intérêts. Pour évaluer la pertinence d'un groupe de règles, on a besoin de mesures d'intérêts non pas pour chaque règle mais pour le groupe de règles, d'où l'utilisation de mesure globale.

4.1.2 Mesures globales

Contrairement aux techniques de visualisation existantes, les techniques de visualisation que nous proposons dans cette section 4 reposent sur des mesures globales. Nous utilisons des mesures globales particulières que nous appelons des mesures globales agrégées car elles sont construites en appliquant une fonction d'agrégation sur des mesures locales.

Définition 4.1. ***Mesure globale agrégée***
Une mesure globale agrégée est une mesure dont les valeurs sont obtenues en appliquant une fonction d'agrégation à des valeurs de mesures locales d'un ensemble de règles.

Exemple 4.2. *La moyenne des confiances et le maximum des supports sont des exemples de mesures globales agrégées. La moyenne des confiances est obtenue en appliquant la fonction d'agrégation moyenne sur la confiance des règles d'un ensemble.*

Soient une mesure locale m et une fonction d'agrégation F_{ag}. Pour un groupe de règles R^Y_X, nous notons v^Y_X sa valeur de mesure globale, i.e. la valeur obtenue en appliquant la fonction d'agrégation F_{ag} aux valeurs de mesures locale $m(r)$ ($r \in R^Y_X$) de ses règles, i.e. $v^Y_X = F_{ag}(\{m(r)|r \in R^Y_X\})$.

4.1.3 Agrégation et interactivité

Dans ce travail, nous nous limitons à la description de l'interactivité qui se fait à travers les paramètres de visualisation u. Concernant l'interactivité pour les deux techniques de visualisation présentées aux sections 4.2 et 4.3, l'utilisateur propose une fonction d'agrégation, une mesure locale à laquelle la fonction d'agrégation sera appliquée et deux ensembles d'attributs. Ces ensembles sont des sous-ensembles du schéma de la relation des données à partir desquelles les règles sont extraites. On peut ainsi agréger n'importe quelle mesure locale en appliquant les fonctions d'agrégation usuelles telle que la moyenne, le maximum ... Les ensembles fournis permettent non seulement de fixer les attributs suivant lesquels les règles sont

$v^{\{Petite\}}_{\{Pomme\}}$	0.5	$v^{\{Petite\}}_{\{Orange\}}$	0.6
$v^{\{Moyenne\}}_{\{Pomme\}}$	0.8	$v^{\{Moyenne\}}_{\{Orange\}}$	0.3
$v^{\{Grande\}}_{\{Pomme\}}$	0.3	$v^{\{Grande\}}_{\{Orange\}}$	0.4
$v^{\{All\}}_{\{Pomme\}}$	0.3	$v^{\{All\}}_{\{Orange\}}$	0.5

TAB. 4 – *Valeur de mesure globale des Groupes de règles*

	Petite	Moyenne	Grande	All
Pomme	0.5	0.8	0.3	0.3
Orange	0.6	0.3	0.4	0.5

TAB. 5 – *Tableau croisé*

visualisées mais aussi la granularité de la visualisation. Plus il y a d'attributs dans les ensembles, plus le niveau de granularité est fin. En d'autres termes, en fonction des attributs qui composent les deux ensembles, on peut visualiser des figures représentant des groupes de règle ou des règles individuelles.

Définition 4.2. ***Paramètres de visualisation (tableaux croisés et points d'intérêt)***
Les paramètres de visualisation constituent un tuple $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$ dont la première et la deuxième composante sont des ensembles d'attributs, la troisième composante est une fonction d'agrégation et la dernière composante est une mesure locale.

Exemple 4.3. *En considérant le schéma de relation $\mathcal{A} = \{Fruit, Ville, Quantité, Année\}$ et les règles du tableau 3, l'utilisateur peut proposer comme paramètres de visualisation le tuple $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$ où : $\mathcal{C} = \{Fruit\}$, $\mathcal{T} = \{Quantité\}$, $F_{ag} = Max$ et $m = conf$.*

Ces paramètres permettent de construire les groupes de règles de l'exemple 4.1 et de calculer les valeurs de mesure globale agrégée pour les groupes de règles non vides. Les valeurs sont présentées dans le tableau 4.

Nous montrons dans la section 4.4.1 comment, à partir de règles stockées dans une base de données, on peut construire les groupes de règles et calculer leur valeur de mesure globale.

4.2 Les tableaux croisés

Les tableaux croisés ont déjà été utilisés pour visualiser des données. Dans cette section, nous présentons une visualisation de groupes de règles sous forme de tableau croisé basée sur des mesures globales agrégées.

4.2.1 Visualisation

Rappelons qu'une visualisation est un couple dont la première composante est un repère et la seconde composante est un ensemble de figures. Dans cette section, nous présentons une description détaillée de ces composantes pour les tableaux croisés en nous appuyant sur le tableau croisé 5.

Ce tableau croisé est construit à partir des groupes de règles de l'exemple 4.1 et leur valeur de mesure globale présentée dans le tableau 4. En ligne, nous avons les itemsets {Pomme} et {Orange} qui sont des valeurs appartenant à $dom(\mathcal{C})^+$. En colonne, nous avons les itemsets {Petite}, {Moyenne}, {Grande} et {All} qui sont des valeurs appartenant à $dom(\mathcal{T})^+$. Une cellule contient la mesure globale associée au groupe constitué des règles dont le corps contient l'itemset en ligne et dont la tête contient l'itemset en colonne.

Définition 4.3. ***Schéma de visualisation (tableaux croisés)***
Le schéma de visualisation des tableaux croisés est l'ensemble $\{Corps, Tête, Mesure\}$ *tel que* $dom(Corps) = dom(Tête) = \mathcal{P}^2(Items)$ *et* $dom(Mesure) = \mathbb{R}$.

Définition 4.4. ***Repère (tableaux croisés)***
Un repère est un ensemble de tuples de schéma $\{Corps, Tête, Mesure\}$ *représentant l'ensemble des références des cellules du tableau croisé. Les deux premières composantes des tuples sont respectivement des sous-ensembles de corps de règles et des sous-ensembles de têtes de règles. La dernière composante est la valeur nulle.*

Exemple 4.4. *Le tableau croisé 5 a pour repère* $rep = \{t_1, ..., t_8\}$ *avec :*
$t_1 =< \{Pomme\}, \{Petite\}, null >$, $t_2 =< \{Pomme\}, \{Moyenne\}, null >$, $t_3 =< \{Pomme\}, \{Grande\}, null >$, $t_4 =< \{Pomme\}, \{All\}, null >$, $t_5 =< \{Orange\}, \{Petite\}, null >$, $t_6 =< \{Orange\}, \{Moyenne\}, null >$, $t_7 =< \{Orange\}, \{Grande\}, null >$ *et* $t_8 =< \{Orange\}, \{All\}, null >$.

Définition 4.5. ***Figure (tableaux croisés)***
Une figure f *est un tuple* $< X, Y, v_X^Y >$ *de schéma* $\{Corps, Tête, Mesure\}$ *tel que* X *et* Y *sont respectivement des sous-ensembles de corps de règles et des sous-ensembles de têtes de règles et* v_X^Y *est une valeur de mesure globale.*

Exemple 4.5. *Considérons le tableau croisé* 5. *Les figures de la visualisation sont :*
$f_1 =< \{Pomme\}, \{Petite\}, 0.5 >$, $f_2 =< \{Pomme\}, \{Moyenne\}, 0.8 >$, $f_3 =< \{Pomme\}, \{Grande\}, 0.3 >$, $f_4 =< \{Pomme\}, \{All\}, 0.3 >$, $f_5 =< \{Orange\}, \{Petite\}, 0.6 >$, $f_6 =< \{Orange\}, \{Moyenne\}, 0.3 >$, $f_7 =< \{Orange\}, \{Grande\}, 0.4 >$ *et* $f_8 =< \{Orange\}, \{All\}, 0.5 >$.
f_2 *est la figure qui représente le groupe de règles* $R_{\{Pomme\}}^{\{Moyenne\}}$ *dont la valeur de mesure agrégée est :* $v_{\{Pomme\}}^{\{Moyenne\}} = Max(\{conf(r_1), conf(r_4), conf(r_8)\}) = 0.8$.
La visualisation du tableau 5 est finalement représentée par le couple $V =< rep, F >$ *où* rep *est le repère présenté à l'exemple 4.4 et* $F = \{f_1, ..., f_8\}$.

4.2.2 Fonction de visualisation

$Calcul\_Repère$ permet de calculer les tuples du repère. Soient un ensemble de règles $R \subseteq Règles$ et des paramètres de visualisation $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$. A l'ensemble de règles R et aux paramètres de visualisation u, $Calcul\_Repère$ associe un ensemble de tuples. Chaque tuple est composé d'un itemset Y de $dom(\mathcal{T})^+$ contenu dans la tête d'au moins une règle de R, d'un itemset X de $dom(\mathcal{C})^+$ contenu dans le corps d'au moins une règle de R et de la valeur nulle. Soient $\mathcal{X} = \{X \in dom(\mathcal{C})^+ | (\exists r \in R), (X \subseteq corps(r)^+)\}$ et $\mathcal{Y} = \{Y \in dom(\mathcal{T})^+ | (\exists r \in R), (Y \subseteq tête(r)^+)\}$.

$$Calcul\_Repère(R, u) = \{< X, Y, null > | (X \in \mathcal{X}), (Y \in \mathcal{Y})\}$$

$Prétraitement\_Règles$ permet de construire les groupes de règles correspondant aux cellules du tableau croisé. Soient un ensemble de règles $R \subseteq Règles$, un repère rep et des paramètres de visualisation $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$. A l'ensemble de règles R, au repère rep et aux paramètres de visualisation u, $Prétraitement\_Règles$ associe un ensemble de groupes de règles $\mathcal{R}_{\mathcal{C}}^{\mathcal{T}}$ qui est l'ensemble des R_X^Y tels que le tuple $< X, Y, null >$ appartient à rep.

$$Prétraitement\_Règles(R, rep, u) = \{R_X^Y \neq \emptyset | < X, Y, null > \in rep\}$$

Enfin, $Calcul\_Figure$ permet de calculer les figures à associer aux groupes de règles calculés avec $Prétraitement\_Règles$. Soient un repère rep et des paramètres de visualisation $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$. Au groupe de règles $R_X^Y \in Prétraitement\_Règles(R, rep, u)$, au repère rep et aux paramètres de visualisation u, $Calcul\_Figure$ associe une figure $f =< X, Y, v_X^Y >$ telle que v_X^Y est la mesure agrégée pour le groupe de règles R_X^Y.

$$Calcul\_Figure(R_X^Y, rep, u) =< X, Y, v_X^Y > \text{ où } v_X^Y = F_{ag}(\{m(r)|r \in R_X^Y\})$$

4.2.3 Agrégation et interactivité

Le ratio entre le nombre de figures affichées avec la technique des tableaux croisés proposée et le nombre de figures affichées dans le cas où une règle est représentée par une figure est plus petit ou égal à $\frac{|dom(\mathcal{C})^+|*|dom(\mathcal{T})^+|}{|R|}$. Ce ratio est maximal lorsque $\mathcal{C} \cup \mathcal{T} = \mathcal{A}$, i.e. lorsque les règles sont visualisées au niveau le plus fin. Il est d'autant plus petit que le nombre d'attributs dans $\mathcal{C}$ et $\mathcal{T}$ est petit. En modifiant les paramètres de visualisation $\mathcal{C}$ et $\mathcal{T}$, l'utilisateur peut visualiser les règles extraites de manière plus ou moins agrégée.

Exemple 4.6. *Si nous posons $\mathcal{C} = \{Fruit, Année\}$ et $\mathcal{T} = \{Quantité\}$, nous obtenons la visualisation du tableau 6 qui présente le même ensemble de règles que la visualisation du tableau 5 mais à un niveau de granularité plus fin.*

		Petite	Moyenne	Grande	All
Pomme	2000		0.8	0.3	
	2005	0.5	0.7	0.1	0.3
Orange	2000		0.15		0.5
	2005	0.6	0.3	0.4	

TAB. 6 – *Tableau croisé avec plus d'attributs dans $\mathcal{C}$*

4.3 La méthode à base de points d'intérêt

La seconde méthode de visualisation que nous proposons consiste à adapter aux règles d'association une technique de visualisation de données utilisant les points d'intérêt.

Dans les travaux de Costa et Venturini (2006), les points d'intérêt sont des données correspondant à des points placés sur un cercle. Ces données sont considérées comme les données de référence. D'autres données sont représentées par des points disposés à l'intérieur du cercle. La position d'un point dans le cercle dépend de l'attraction exercée par les points d'intérêt sur lui. Cette attraction représente la similarité entre la donnée associée à ce point et les données de référence. Elle est matérialisée par la distance entre le point et les points d'intérêt.

Dans notre approche, à la différence de la technique exposée dans Costa et Venturini (2006), un point d'intérêt est un itemset qui représente un ensemble de corps de règles. Un point dans le cercle est aussi un itemset, mais il représente un ensemble de têtes de règles.

4.3.1 Visualisation

La figure 8 montre une visualisation de règles qui utilise les points d'intérêt et qui est construite à partir des groupes de règles de l'exemple 4.1 et de leur valeur de mesure globale

présentée dans le tableau 4. Sur le cercle, nous avons un point d'intérêt pour chaque itemset X tel qu'il existe un groupe de règles $R_X^Y \in \mathcal{R}_C^{\mathcal{T}}$ non vide. Sur la figure 8, les itemsets correspondant aux points d'intérêt sont $\{Pomme\}$ et $\{Orange\}$. Dans le cercle, un point est affiché pour chaque itemset Y tel qu'il existe au moins un point d'intérêt X tel que $R_X^Y \in \mathcal{R}_C^{\mathcal{T}}$ et $R_X^Y \neq \emptyset$. Ce point est affiché avec la légende Y. Sa position est calculée à partir de l'ensemble des valeurs de mesure globale des groupes de règles $R_X^Y \in \mathcal{R}_C^{\mathcal{T}}$ non vides et des points d'intérêt. Par exemple, la position de l'itemset $Y = \{Moyenne\}$ est calculée à partir des valeurs de mesure globale $v_{\{Pomme\}}^{\{Moyenne\}}$ et $v_{\{Orange\}}^{\{Moyenne\}}$ et des points d'intérêt $\{Pomme\}$ et $\{Orange\}$.

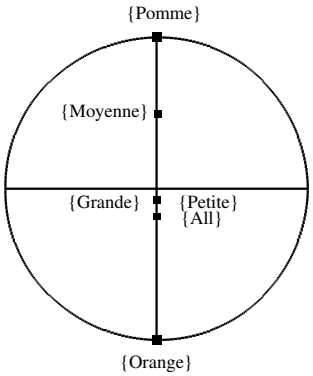


FIG. 8 – *Visualisation de règles avec les points d'intérêt*

Un point dans le cercle de légende Y est d'autant plus proche d'un point d'intérêt de légende X que la valeur v_X^Y est élevée. Par exemple, dans le cercle représenté sur la figure 8, le point de légende $\{Moyenne\}$ est plus proche du point d'intérêt $\{Pomme\}$ que du point d'intérêt $\{Orange\}$ car $R_{\{Pomme\}}^{\{Moyenne\}}$ possède une valeur plus grande que la valeur associée à $\{Orange\}$. Ce qui signifie qu'il est plus probable qu'une règle possédant $\{Pomme\}$ dans son corps ait {Moyenne} dans sa tête qu'une règle possédant $\{Orange\}$ dans son corps ait {Moyenne} dans sa tête.

Définition 4.6. ***Schéma de visualisation (points d'intérêt)***
Le schéma de visualisation des points d'intérêt est l'ensemble $\{Itemset, Angle, Rayon\}$ *tel que* $dom(Itemset) = \mathcal{P}^2(Items)$, $dom(Angle) =]0, 2\pi]$ *et* $dom(Rayon) = [0, 1]$.

Définition 4.7. ***Repère (points d'intérêt)***
Un repère est un ensemble de tuples de schéma $\{Itemset, Angle, Rayon\}$ *appelés points d'intérêt. Un point d'intérêt est constitué d'un itemset qui est un sous-ensemble de corps de règles, d'un angle et d'un rayon de valeur fixée à 1.*

Exemple 4.7. *Le repère de la visualisation sur la figure 8 est* $rep = \{pt_1, pt_2\}$ *où les points d'intérêt sont :* $pt_1 =< \{Pomme\}, \frac{\pi}{2}, 1 >$ *et* $pt_2 =< \{Orange\}, \frac{3\pi}{2}, 1 >$.

Définition 4.8. ***Figure (points d'intérêt)***
Une figure est un tuple de schéma $\{Itemset, Angle, Rayon\}$. *L'itemset de la figure est un*

sous-ensemble de tête de règles. L'angle et le rayon de la figure donnent la position de la figure.

Exemple 4.8. *Les figures de la visualisation représentée à la figure 8 sont :* $f_1 =< Petite, -\frac{\pi}{2}, 0.1 >$, $f_2 =< Moyenne, \frac{\pi}{2}, 0.5 >$, $f_3 =< Grande, -\frac{\pi}{2}, 0.1 >$ *et* $f_4 =< All, -\frac{\pi}{2}, 0.2 >$.
La visualisation de la figure 8 est finalement représentée par le couple $V =< rep, F >$ *où* rep *est le repère présenté à l'exemple 4.7 et* $F = \{f_1, ..., f_4\}$.

4.3.2 Fonction de visualisation

$Calcul\_Repère$ permet de déterminer les points d'intérêt et leur angle dans le plan.
Soient un ensemble de règles d'association R et des paramètres de visualisation $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$. Notons $\mathcal{X}$ l'ensemble des itemsets de $dom(\mathcal{C})^+$ contenus dans au moins un corps de règle de R, i.e. $\mathcal{X} = \{X \in dom(\mathcal{C})^+ | (\exists r \in \text{R}), (X \subseteq Corps(r)^+)\}$. Notons N le cardinal de $\mathcal{X}$ et n le rang d'un itemset X dans $\mathcal{X}$. A l'ensemble de règles R et aux paramètres de visualisation u, $Calcul\_Repère$ associe un repère rep qui est l'ensemble des points d'intérêt $< X, \theta, ray >$ tels que $X \in \mathcal{X}$, $\theta = \frac{2\pi n}{N}$ et $ray = 1$.

$$Calcul\_Repère(R, u) = \{< X, \theta, ray > | X \in \mathcal{X} \wedge \theta = \frac{2\pi n}{N} \wedge ray = 1\}$$

$Prétraitement\_Règles$ permet de former des groupes de règles de même tête en fonction d'un ensemble de règles et de paramètres de visualisation.
Soient un ensemble de règles d'association R, un repère $rep = \{pt_1, ..., pt_N\}$ et des paramètres de visualisation $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$. Soit $\mathcal{Y} = \{Y \in dom(\mathcal{T})^+ | (\exists r \in \text{R}), (Y \subseteq Tête(r)^+)\}$. Notons R^Y l'union des R^Y_X tels que $< X, \theta, 1 >$ appartient au repère rep et Y appartient à $\mathcal{Y}$. A l'ensemble de règles R, au repère rep et aux paramètres de visualisation u, $Prétraitement\_Règles$ associe l'ensemble des R^Y.

$$Prétraitement\_Règles(R, rep, u) = \{R^Y | Y \in \mathcal{Y}\} \; o \; R^Y = \bigcup_{<X,\theta,1> \in rep} R^Y_X$$

$Calcul\_Figure$ calcule les coordonnées des points à placer dans le cercle.
Soient un repère $rep = \{pt_1, ..., pt_N\}$ avec $pt_n =< X_n, \theta_n, 1 >, n \in \{1, ..., N\}$ et des paramètres de visualisation $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$. A l'ensemble de règles $R^Y \in Prétraitement\_Règles(R, rep, u)$, au repère rep et aux paramètres de visualisation u, $Calcul\_Figure$ associe une figure $f = Calcul\_Figure(R^Y, rep, u) =< Y, \theta, ray >$ telle que $\theta = \arctan \frac{b}{a}$ et $ray = \sqrt{a^2 + b^2}$ où $a = \sum_1^N (cos(\theta_n) * v^Y_{X_n})$ et $b = \sum_1^N (sin(\theta_n) * v^Y_{X_n})$ en supposant que $v^Y_X = 0$ si $R^Y_X = \emptyset$.

4.3.3 Agrégation et interactivité

Le ratio entre le nombre de figures affichées avec la technique des points d'intérêt proposée et le nombre de figures affichées dans le cas où une règle est représentée par une figure est égal à $\frac{|\{R^Y \neq \emptyset | Y \in dom(\mathcal{T})^+\}|}{|R|} \leq \frac{|dom(\mathcal{T})^+|}{|R|}$. Ce ratio est plus petit que celui calculé à la section 4.2.3

pour les tableaux croisés. Ceci montre que la technique de visualisation à base de points d'intérêt est plus synthétique que la technique de visualisation basée sur les tableaux croisés. Néanmoins, la technique de visualisation à base de points d'intérêt ne permet pas de visualiser les mesures globales agrégées en absolu, mais seulement de manière relative. Comme les tableaux croisés, les points d'intérêt permettent de visualiser les règles à des niveaux plus ou moins fin.

Exemple 4.9. *En considérant les mêmes attributs dans $\mathcal{C}$ et $\mathcal{T}$ de l'exemple 4.6, nous obtenons la figure suivante.*

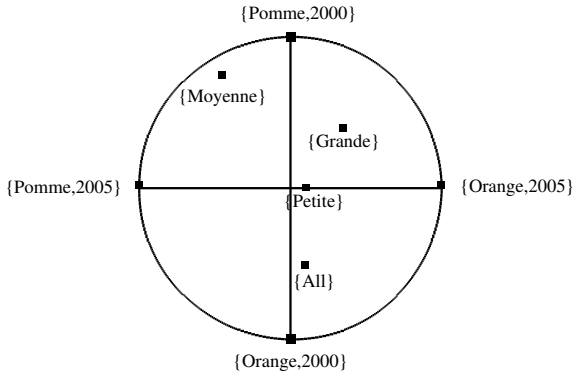


FIG. 9 – *Visualisation de règles avec les points d'intérêt*

4.4 Implémentation

Les visualisations proposées peuvent être calculées facilement si les règles sont stockées dans une base de données relationnelle. Dans cette section, nous proposons une implémentation des fonctions de visualisation des tableaux croisés et des points d'intérêt présentées dans les sections 4.2.2 et 4.3.2. Pour chaque fonction, nous formulons les requêtes employées pour calculer le repère et les figures. Nous utilisons comme entrées un ensemble de règles R et des paramètres de visualisation $u =< \mathcal{C}, \mathcal{T}, F_{ag}, m >$ où $\mathcal{C} = \{AC_1, ..., AC_I\}$ et $\mathcal{T} = \{AT_1, \cdots, AT_J\}$.

4.4.1 Calcul des valeurs de mesure d'intérêt pour les groupes de règles

Dans notre approche, les règles sont stockées dans une base de données relationnelle sous forme de trois tables CORPS, TÊTE et RÈGLE. Comme le montrent les tableaux 7, 8 et 9, CORPS contient le corps des règles, TÊTE contient la tête des règles et RÈGLE contient les règles (identifiant de leur tête et de leur corps) avec leurs mesures d'intérêts. Les règles sont mises à plat dans les tables, i.e. nous n'utilisons pas d'attributs imbriqués pour stocker la tête et le corps des règles.

Cid	CFruit	CAnnée	CQuantité	CVille
c1	Pomme	2000	All	All
c2	Orange	2005	All	All
c3	Pomme	2005	All	All
...	...	...	...	...

TAB. 7 – CORPS

Tid	TFruit	TAnnée	TQuantité	TVille
t1	All	All	Moyenne	Tours
t2	All	All	Moyenne	Blois
t3	All	All	Grande	Tours
...	...	...	...	...

TAB. 8 – TÊTE

A partir des tables CORPS, TÊTE et RÈGLE, la requête suivante permet d'obtenir la vue Ensemble_Itemsets contenant les items de X et de Y ainsi que la valeur de mesure globale agrégée v_X^Y des groupes de règles $R_X^Y \in \mathcal{R}_{\mathcal{C}}^{\mathcal{T}}$.

CREATE VIEW $Ensemble\_Itemsets$ **AS**
SELECT $C.AC_1, \cdots, C.AC_I, T.AT_1, \cdots, T.AT_J, F_{ag}(m)$
FROM CORPS C, TÊTE T, RÈGLE R
WHERE C.Cid=R.Cid
AND T.Tid=R.Tid
GROUP BY $C.AC_1, \cdots, C.AC_I, T.AT_1, \cdots, T.AT_J$;

Exemple 4.10. *La table 10 présente la vue contenant les tuples qui correspondent aux groupes de règles formés avec les règles du tableau 3.*

Rid	Cid	Tid	Sup	Conf
r1	c1	t1	0.4	0.5
r2	c2	t3	0.3	0.4
r3	c3	t1	0.4	0.7
...	...	...	...	...

TAB. 9 – RÈGLE

CFruit	CQuantité	Max(conf)
Pomme	Petite	0.5
Pomme	Moyenne	0.8
Pomme	Grande	0.3
Pomme	All	0.3
Orange	Petite	0.6
Orange	Moyenne	0.3
Orange	Grande	0.4
Orange	All	0.5

TAB. 10 – $Ensemble\_Itemsets$

Fruit	Année	Angle
Pomme	2000	$\frac{\pi}{2}$
Orange	2005	$\frac{3\pi}{2}$

TAB. 11 – *Table* $Point\_Intérêt$

4.4.2 Implémentation pour les tableaux croisés

Pour les tableaux croisés, le repère est la projection sur $\mathcal{C} \cup \mathcal{T}$ des tuples de la vue $Ensemble\_Itemsets$ et l'ensemble des figures est l'ensemble des tuples de $Ensemble\_Itemsets$.

4.4.3 Implémentation pour les points d'intérêt

Les tuples de $Ensemble\_Itemsets$ ne permettent pas d'avoir directement la visualisation pour les points d'intérêt. Afin d'obtenir le repère et les figures, nous avons besoin de données supplémentaires : les items et les coordonnées des points d'intérêt.

Supposons que les angles des points d'intérêt sont calculés et stockés dans une table $Point\_Intérêt$ de schéma $\mathcal{C} \cup \{Angle\}$ où $Angle$ est un attribut pour les angles. Le rayon des points d'intérêt étant constant, il n'est pas stocké dans la table. Les tuples de la table $Point\_Intérêt$ constituent l'ensemble des points d'intérêt du repère.

Exemple 4.11. *La table 11 contient les points d'intérêt de la visualisation présentée à la figure 8.*

Nous utilisons la vue $Ensemble\_Itemsets$ et la table $Point\_Intérêt$ pour calculer les figures. Les coordonnées cartésiennes (et non polaires comme dans le modèle) des figures peuvent être calculées par la requête suivante.

CREATE VIEW $Point$ **AS**
SELECT $E.AT_1, \cdots, E.AT_J$,
$avg(rayon * cos(PT.position) * E.m\_ag)$,
$avg(rayon * sin(PT.position) * E.m\_ag)$
FROM $Ensemble\_Itemsets\ E$,$Point\_Intérêt\ PT$
WHERE $E.AC_1 = PT.AC_1$ **AND**
$\cdots$
AND $E.AC_I = PT.AC_I$
GROUP BY $E.AT_1, \cdots, E.AT_J$;

Exemple 4.12. *Les figures du tableau 12 sont obtenues en appliquant cette requête à $Ensemble\_Itemsets$ et $Point\_Intérêt$ présentés respectivement sur les tableaux 10 et 11.*

TQuantité	*a*	*b*
Petite	0	0.1
Moyenne	0	−0.5
Grande	−0.1	−0.1
All	0	0.2

TAB. 12 – *Table $Figures$*

5 Conclusion et perspectives

Dans ce papier, nous nous sommes intéressés au problème de visualisation de grands ensembles de règles. Les techniques usuelles que nous avons étudiées et qui représentent une règle par une figure rencontrent des difficultés pour visualiser de grands ensembles de règles. Nous définissons une visualisation comme un couple, constitué d'un repère et d'un ensemble de figures, tel que les références du repère et les figures ont le même schéma. Pour résoudre le problème posé, nous avons proposé deux nouvelles techniques de visualisation. L'une est basée sur les tableaux croisés et l'autre sur les points d'intérêt. Ces techniques permettent de visualiser de grands ensembles de règles d'association en formant des groupes de règles et en utilisant des mesures globales agrégées calculées à partir des mesures locales des règles de chaque groupe. Nous avons aussi introduit une piste d'implémentation en montrant une manière de calculer les visualisations lorsque les règles sont stockées dans une base de données relationnelle.

Les perspectives de ce travail sont détaillées dans les trois points suivant :

- Notre objectif à court terme est d'implémenter dans une même plate-forme les techniques des tableaux croisés et des points d'intérêt en utilisant les fonctions de visualisation présentées dans les sections 4.2.2 et 4.3.
- Pour les deux techniques proposées, nous distinguons des interactions qui ne sont possibles qu'après la construction de la visualisation : par exemple, détailler un groupe de règles i.e. visualiser individuellement les règles du groupes, supprimer des références du repère, sélectionner des figures... Nous projetons d'introduire d'autres paramètres permettant de pendre en compte ce type d'interaction.
- Dans les deux nouvelles techniques de visualisation proposées, les mesures globales utilisées sont des mesures agrégées. En général, leur calcul nécessite la connaissance de toutes les mesures locales des règles qui forment les groupes de règles, ce qui suppose que toutes les règles aient été extraites auparavant. Un de nos objectifs est maintenant d'examiner comment définir des techniques de regroupement de règles et de nouvelles mesures globales permettant le calcul des mesures globales sans avoir à connaître toutes les règles intéressantes des groupes formés. Cela permettrait d'utiliser des critères de regroupement dès la phase d'extraction des règles pour ne calculer que les informations nécessaires au calcul des groupes et des mesures globales associées.
- Les paramètres de visualisation ne permettent d'effectuer que quelques interactions telles que changer le niveau de granularité de la visualisation, changer la fonction d'agrégation, ... Nous envisageons d'intégrer à notre approche d'autres modes d'interaction permettant à l'utilisateur de modifier l'ensemble des règles visualisées, le repère ou l'ensemble des figures. Dans ce cadre, nous travaillons à la définition d'un langage de requêtes permettant la manipulation de visualisations.

Références

Appice, A. et P. Buono (2005). Analyzing multi-level spatial association rules through a graph-based visualization. In *IEA/AIE'2005*, London, UK, pp. 448–458. Springer-Verlag.

Blanchard, J., F. Guillet, et H. Briand (2003). Exploratory visualization for association rule rummaging. In *MDM/KDD2003*, pp. 107–114.

Bruzzese, D. et P. Buono (2004). Combining visual techniques for association rules exploration. In *AVI '04*, New York, NY, USA, pp. 381–384. ACM Press.

Bruzzese, D. et C. Davino (2003). Visual post-analysis of association rules. Volume 14, pp. 621–635.

Chakravarthy, S. et H. Zhang (2003). Visualization of association rules over relational dbmss. In *SAC '03*, New York, NY, USA, pp. 922–926. ACM Press.

Costa, D. D. et G. Venturini (2006). Visualisation interactive de données avec des méthodes à base de points d'intérêt. *RNTI-E-6 2*(2), 335–346.

Hao, M. C., U. Dayal, M. Hsu, T. Sprenger, et M. H. Gross. Visualization of directed associations in E-Commerce transaction data. pp. 185–192.

Hofmann, H., A. P. J. M. Siebes, et A. F. X. Wilhelm (2000). Visualizing association rules with interactive mosaic plots. In *KDD '00*, New York, NY, USA, pp. 227–235. ACM Press.

Inselberg, A. (1985). The plane with parallel coordinates. In S. B. . Heidelberg (Ed.), *The Visual Computer*, Volume 1, pp. 69–91.

Klemettinen, M., H. Mannila, P. Ronkainen, H. Toivonen, et A. I. Verkamo (1994). Finding interesting rules from large sets of discovered association rules. In *CIKM '94*, New York, NY, USA, pp. 401–407. ACM Press.

Li Yang, S. M. (2005). Pruning and visualizing generalized association rules in parallel coordinates. *IEEE Transactions on Knowledge and Data Engineering 17*(1), 60–70.

Wong, P. C., P. Whitney, et J. Thomas (1999). Visualizing association rules for text mining. In *INFOVIS '99*, Washington, DC, USA, pp. 120. IEEE Computer Society.

Yahia, S. B. et E. M. Nguifo (2004). Emulating a cooperative behavior in a generic association rule visualization tool. In *ICTAI '04*, Washington, DC, USA, pp. 148–155. IEEE Computer Society.

Zhao, K., B. Liu, T. M. Tirpak, et W. Xiao (2005). Opportunity map : a visualization framework for fast identification of actionable knowledge. In *CIKM '05*, New York, NY, USA, pp. 60–67. ACM Press.

Summary

The study of the main visualization techniques of the association rules allows us to identify the major problem of the usual techniques that represent each rule by a figure. The problem of these techniques is the browsing of a large amount of the rules. In this paper, we propose two new visualization techniques of association rules. Our techniques represent a group of rules by one figure and visualize a large set of rules by using the measures of global interest which provide the information in the rules groups. The first technique is based on the cross tables and the second one is based on the points of interest.

Analyse et visualisation interactive de sessions Web

Marie-Jeanne Lesot*, Nicolas Labroche*, Lionel Yaffi**

*Université Pierre et Marie Curie, Paris 6
Laboratoire d'Informatique de Paris 6, UMR CNRS 7606
Case 169, 4 Place Jussieu 75252 Paris cedex 5
{nicolas.labroche, marie-jeanne.lesot}@lip6.fr
**Intelligent Learning Objects
lionel.yaffi@ilobjects.com

Résumé. Cet article présente un outil pour l'étude de l'usage de sites Internet qui repose sur une analyse automatique de l'activité des internautes et sur un mécanisme interactif de visualisation des navigations. La phase d'analyse résume l'information avant la phase de visualisation ; elle catégorise les traces d'activité des internautes à l'aide d'un algorithme de clustering relationnel biomimétique, et fournit un ensemble de profils représentatifs pour chacun des clusters, définis à partir de degrés de typicalité. L'outil proposé a été appliqué et évalué sur des fichiers log issus du muséum de Bourges et a montré sa capacité à produire des visualisations pertinentes et interprétables des navigations des internautes.

1 Introduction

L'analyse des usages sur Internet a pour objectif de mettre en évidence les différents comportements de navigation des internautes, ainsi que leurs motivations et besoins d'information. La très grande quantité de données disponible nécessite le développement d'outils performants qui soient à la fois capables d'extraire automatiquement des informations sur les comportements de navigation des internautes, et de faciliter leur interprétation. Les connaissances résultant de ces analyses peuvent alors permettre l'adaptation de la structure des sites (Masseglia et al., 1999) ou le développement de systèmes de personnalisation (Nasraoui et al., 2002).

Cette analyse d'usages passe fréquemment par la visualisation de l'activité des internautes, basée sur le paradigme "web-paths" introduit par Pitkow et Bharat (1994) : un site web est représenté par un graphe orienté et l'activité d'un internaute par un chemin dans ce graphe. Une telle visualisation offre une grande interprétabilité, mais se heurte au problème général de la représentation de graphes de grande taille (Herman et al., 2000), du fait de la taille des sites Internet. Deux approches principales, détaillées dans la Section 2 de cet article, sont classiquement employées pour pallier ces difficultés : la première vise à améliorer les capacités de représentation des outils de visualisation, la seconde à sélectionner l'information pertinente à afficher pour en réduire la quantité.

Nous proposons dans ce cadre un outil complet et modulaire pour l'analyse et la visualisation de l'activité des internautes, qui, comme d'autres logiciels existants (Hong et al., 2001; Youssefi et al., 2004), combine les deux approches précédentes. L'outil proposé couvre, dans

une architecture modulaire, l'ensemble des étapes du processus : il contient un module d'acquisition des traces d'usages, un module d'analyse et d'extraction d'informations des sessions, et un module de visualisation des informations les plus pertinentes.

Le module d'acquisition, qui effectue un prétraitement et la reconstruction des sessions de navigation, autorise un suivi des internautes avec une précision supérieure à celle des fichiers log classiques.

Le module d'analyse propose un mécanisme de filtrage de l'information à représenter, par classification non supervisée. Celle-ci est effectuée à l'aide d'un algorithme biomimétique relationel dont la comparaison à d'autres méthodes a montré qu'il était le plus adapté dans le cadre de cet outil. La particularité de l'approche mise en œuvre est d'opérer au niveau des sessions des internautes plutôt qu'au niveau des pages du site (bien que ces deux méthodes puissent être combinées) : l'objectif est de trouver des sessions complètes représentatives plutôt que des portions de sessions associées à un besoin d'information précis, mais déconnectées de tout contexte. L'outil proposé détermine ainsi des groupes d'internautes qui possèdent le même comportement et qui correspondent donc à une information redondante qu'il n'est pas opportun d'afficher. En outre, pour réduire plus encore les informations à représenter, nous proposons de définir des individus représentatifs des clusters identifiés, en utilisant une méthode de construction de prototypes basés sur la notion de typicalité.

Après cette phase d'analyse qui résume l'information à afficher, la visualisation peut être appliquée en se focalisant sur les sessions les plus représentatives de l'activité du site étudié. Pour cette étape, l'outil proposé recourt à des représentations adaptées telles que les arbres agrégés ou des graphes d'usages interactifs.

Après une section consacrée à la description de quelques travaux similaires à l'outil proposé (section 2), l'article détaille successivement les trois modules d'acquisition des données d'usage (section 3), d'analyse et d'extraction d'informations (section 4) et de visualisation (section 5). La section 6 illustre les résultats obtenus sur des données réelles provenant du site du muséum de Bourges, enfin la section 7 conclut et donne les perspectives de ce travail.

2 Travaux similaires

La visualisation de graphes de sites web pose le problème de la visualisation de grands graphes, qui est classiquement résolu suivant deux types d'approches : la première consiste à améliorer les capacités de représentation des modèles utilisés, et la seconde à sélectionner les informations les plus significatives à afficher, par filtrage et analyse statistique.

Plus précisément, la première approche vise au développement de méthodes pour la représentation elle-même : Herman et al. (2000) rappellent par exemple les propriétés principales que le graphe doit vérifier pour en assurer une bonne visualisation, comme la minimisation du nombre d'arcs qui se croisent. Eelco et Weinreich (2005) proposent d'utiliser des codes couleurs ou des techniques de zooms localisés de type fisheye. Mukherjea et Foley (1995) lient de manière interactive les informations et leurs propriétés visuelles et utilisent une représentation hiérarchique sous forme d'arbre (plus facilement interprétable que les graphes) qui autorise un mécanisme de zoom sémantique : les nœuds représentant des ensembles de pages au contenu similaire peuvent être développés à la demande. Munzner et Burchard (1995) proposent d'utiliser des espaces hyperboliques pour augmenter la quantité d'information qui peut être affichée par rapport à l'espace euclidien traditionnel. Chi et al. (1998) proposent une représentation de

type “disk tree” qui prend en compte la topologie, le contenu et les usages d’un site web pour aider à comprendre les relations entre production d’informations (nouvelles pages ajoutées ou mises à jour) et consommation lors des visites des internautes.

Une seconde catégorie d’approches de visualisation de grands graphes est basée sur des méthodes de filtrage et d’analyse pour diminuer et sélectionner l’information à afficher. L’outil Webviz (Pitkow et Bharat, 1994), par exemple, permet à un expert de mettre en lumière des formes de navigation cachées en appliquant des filtres aux utilisateurs, aux périodes, aux sites ou à certains sous-répertoires. Herman et al. (2000) classifient les nœuds et les arcs des grands graphes pour obtenir de plus petits graphes qui peuvent être représentés efficacement. Mukherjea et Foley (1995) classifient les pages qui partagent certaines propriétés (même répertoire, auteur ou thème) ou en fonction de la structure du graphe avant d’appliquer des filtres sur les nœuds et/ou les arcs.

Enfin une troisième catégorie de méthodes combine les deux approches précédentes : l’outil Webquilt (Hong et al., 2001) par exemple décrit un canevas complet pour l’analyse et la visualisation des sessions. Son objectif est de conduire des tests d’utilisabilité à distance sur de nombreux utilisateurs et de faciliter l’analyse des résultats en ne représentant que quelques sessions ainsi que les chemins optimaux (au sens de l’utilisabilité de l’interface). Youssefi et al. (2004) définissent le “Visual Web Mining” comme l’application de méthodes de visualisation sur des résultats issus de fouille de données du Web (ou “web mining”) : par exemple l’identification de sous-séquences communes dans les sessions.

L’outil présenté dans cet article s’inscrit dans la lignée des travaux précédents. Il s’en distingue cependant sur trois points principaux : (1) il résume l’information d’usage en classifiant les sessions des internautes (i.e. la “vraie” information d’usage) et non pas les documents accédés (ou les nœuds du graphe correspondant), (2) il utilise un algorithme de classification non supervisée biomimétique qui possède des qualités intéressantes pour ce type d’application et (3) il construit des prototypes représentatifs pour chaque cluster à partir du calcul de la typicalité des objets qui y sont regroupés, de façon à améliorer l’interprétabilité des résultats.

3 Module d’acquisition des données d’usage

Le premier module de l’outil proposé est consacré à la collecte et au prétraitement des données d’usage sur Internet qui seront ensuite manipulées par les modules d’analyse et de visualisation. Il existe dans ce domaine de nombreuses méthodes, nous présentons ici celles qui ont été développées spécifiquement pour notre outil.

3.1 Enregistrement de l’activité des internautes

La première fonctionnalité permet l’enregistrement de l’activité des internautes avec une précision plus importante que les fichiers log. Ces derniers sont généralement bruités du fait de l’utilisation de proxy ou de systèmes de caches. Notre module prend la forme d’un script installé sur le serveur Web et agit comme un serveur proxy intermédiaire qui modifie à la volée les documents HTML renvoyés en y ajoutant un programme javascript. Celui-ci associe un identifiant unique à chaque utilisateur et capture ensuite toute son activité, à un niveau plus bas que les fichiers log traditionnels : il enregistre par exemple les événements de type clic de souris, défilement de page ainsi que le chargement de documents. Pour chaque action réalisée

par l'internaute, le programme javascript génère une entrée de log et l'envoie à un programme Java qui les stocke.

3.2 Reconstruction des sessions utilisateurs

La seconde fonctionnalité de traitement des données prend en entrée un fichier log traditionnel ou tel que décrit ci-dessus et produit la liste des sessions des internautes.

Les logs peuvent être filtrés par adresse IP, date, type MIME (pour filtrer les images et les scripts notamment), répertoire (pour ne s'intéresser qu'à une partie du site) et statut HTTP (pour ne conserver que les requêtes de statut 200, que le serveur a correctement traitées).

Après ce premier filtrage, on définit un internaute comme un triplet : adresse IP, identifiant (s'il est connu) et user-agent (champ informant sur le navigateur et le système d'exploitation de l'internaute) pour limiter les problèmes dus aux serveurs proxy. Les requêtes sont alors triées par utilisateur et par date. Les sessions utilisateurs sont finalement reconstruites selon la méthode suivante : si la période d'inactivité entre 2 requêtes du même internaute excède une certaine durée (fixée généralement à 30 minutes), on considère que la session est terminée. Sinon, on regarde si le champ référant de la requête (qui indique la page d'où provient l'internaute) apparaît parmi les pages de la session déjà reconstruite. Dans le cas où il est indéfini, on cherche si la page de la nouvelle requête est accessible depuis une des pages déjà visitées. Cette heuristique de reconstruction des sessions utilise également la structure du site pour compléter les chemins des sessions et ainsi pallier les problèmes inhérents aux systèmes de cache.

4 Module d'analyse et d'extraction d'information

Le second module analyse les sessions pour améliorer la lisibilité et la compréhension du graphe d'usage du site Web en (1) construisant des vues ciblées de l'activité à l'aide d'un algorithme de classification non supervisée qui produit des groupes de sessions homogènes, et (2) en résumant l'information ainsi extraite sous la forme de profils d'usage représentatifs. Cette approche permet également de traiter des sites de grande taille puisque les groupes de sessions homogènes n'accèdent chacun qu'à une petite portion des pages du site.

Nous décrivons la mesure de similarité employée pour comparer les sessions utilisateurs puis nous détaillons l'algorithme de clustering et la méthode de construction des prototypes.

4.1 Mesure de similarité entre sessions Web

La définition de la similarité entre sessions peut se faire selon différents critères : par exemple on peut considérer que deux sessions sont proches si elles ont accédé le même nombre de fois les pages du site, ou pendant la même durée (Labroche, 2003) ou si elles forment les mêmes séquences de pages. Dans l'outil proposé, nous avons fait le choix de considérer qu'une session correspond à un ensemble de pages, sans prendre en compte l'ordre dans lequel elles ont été visitées, et de définir que deux sessions sont similaires si elles accèdent à des pages similaires.

La similarité entre pages est définie en fonction de la similarité du chemin de leurs urls comme dans (Nasraoui et al., 1999) et non à partir du contenu des documents qui nécessite une phase coûteuse d'extraction et d'indexation. Cette approche présuppose que la structure

des répertoires du site reflète son contenu. Plus précisément, la similarité $S_{url}(u_1, u_2)$ entre 2 urls u_1 et u_2 est calculée comme la somme pondérée des éléments identiques des chemins des 2 urls (les répertoires et les fichiers accédés) en donnant un poids plus important aux éléments les plus proches de la racine du site.

La similarité normalisée entre deux sessions s_1 and s_2 est ensuite calculée comme :

$$S(s_1, s_2) = \frac{\tilde{S}(s_1, s_2)}{\sqrt{\tilde{S}(s_1, s_1)\tilde{S}(s_2, s_2)}} \quad \text{où} \quad \tilde{S}(s_1, s_2) = \sum_{u_1 \in s_1} \sum_{u_2 \in s_2} S_{url}(u_1, u_2) \qquad (1)$$

Cette mesure compare les sessions en tenant compte du nombre de visites de chaque url, ainsi que de la similarité entre urls. De ce fait, elle porte plus d'information que les mesures basées sur les représentations plus simples mentionnées ci-dessus. De plus, elle peut aisément être améliorée en modifiant la mesure de similarité entre urls, par exemple en considérant le contenu des pages ou une description sémantique liée à une ontologie. Par ailleurs, cette mesure a été comparée à une mesure dérivée d'une distance d'édition qui considère les sessions comme des séquences d'urls (Labroche, 2007), et a montré qu'elle obtenait des clusters plus compacts en un temps plus court.

4.2 Classification non supervisée

4.2.1 Objectif

Comme rappelé par Mobasher (2006), les nombreux travaux conduits dans les dix dernières années pour l'extraction, l'analyse ou la prédiction des besoins d'information des utilisateurs suivent différentes approches parmi lesquelles la recherche de règles d'association ou séquentielles, et l'utilisation d'algorithmes de clustering. Dans ce dernier cas (Labroche et al., 2003; Baraglia et Palmerini, 2002; Heer et Chi, 2001; Fu et al., 1999), l'objectif peut être d'identifier les groupes d'utilisateurs qui accèdent aux mêmes ressources ou les groupes de pages qui apparaissent fréquemment dans les mêmes sessions. Ces approches permettent de définir, dans une seconde phase, des profils typiques de l'accès sur le site et donc de mettre en lumière des informations qui ne sont pas directement accessibles par une simple étude des fichiers log.

Pour effectuer cette étape de clustering, nous proposons d'utiliser l'algorithme relationnel biomimétique Leader Ant (Labroche, 2007) résumé dans le tableau 1 et détaillé ci-dessous. Il est particulièrement adapté à l'analyse des usages du web car il estime automatiquement le nombre de clusters et peut manipuler tout type de représentation de sessions. En effet, il ne dépend que de la définition d'une mesure de similarité entre sessions et ne fait pas l'hypothèse que les données prennent la forme de vecteurs numériques. De plus, sa complexité en $O(kn)$ où k désigne le nombre de clusters et n le nombre de sessions, le rend adapté au traitement de grands jeux de données. Enfin, il a été montré (Labroche, 2007) que son efficacité à produire des clusters interprétables est comparable à celle d'algorithmes de référence comme les c-médoïdes flous (Krishnapuram et al., 2001).

4.2.2 L'algorithme Leader Ant

L'algorithme Leader Ant (LA) (Labroche, 2006, 2007) est inspiré du système de reconnaissance chimique des fourmis. Dans le modèle biologique, la reconnaissance entre fourmis

TAB. 1 – *Description de l'algorithme Leader Ant (Labroche, 2006, 2007)*

Entrée: Un jeu de données avec n objets
Sortie: Une partition des n objets

(1)	Initialisation des fourmis artificielles
(2)	Association d'un objet à chaque fourmi
(3)	Calcul du seuil de similarité
(4)	Construction itérative des nids
(5)	Choix aléatoire d'une fourmi artificielle sans nid
(6)	Rencontres aléatoires avec des fourmis de chacun des nids
(7)	Estimation de la similarité avec chacun des nids
(8)	Affectation de la fourmi à un nid ou création d'un nouveau nid, en fonction du seuil de similarité
(9)	Suppression des nids les plus petits (optionnel)

dépend principalement de l'odeur émise par chaque fourmi et de leur modèle de l'odeur de leur nid. Dans le modèle LA qui s'en inspire, un nid de fourmis artificielles représente un cluster, et une fourmi est représentée par son génome qui correspond à un objet du jeu de données, et son label qui modélise son odeur, et indique son nid.

LA, résumé dans le tableau 1 et présenté en détail dans (Labroche, 2006), est un algorithme relationnel agglomératif qui produit la partition des objets en une seule passe sur les données. A chaque itération, une fourmi sans nid est choisie aléatoirement et détermine son label, en simulant des rencontres aléatoires avec des fourmis de chacun des nids existants. Pendant ces rencontres, la fourmi estime la similarité de son génome avec ceux des fourmis du nid évalué. Lorsque tous les nids ont été évalués, la fourmi rejoint le nid avec lequel sa similarité est maximale, ou construit son propre nid si cette valeur maximale n'excède pas la valeur d'un seuil. Ce seuil est appris durant une phase d'initialisation et est égal à la moyenne des valeurs de similarité observées entre les génomes de fourmis choisies aléatoirement, pendant un nombre fixé de rencontres. Lorsque toutes les fourmis ont été affectées, les plus petits nids peuvent éventuellement être supprimés et leurs fourmis être réaffectées aux nids restants les plus similaires.

L'algorithme LA est similaire aux algorithmes de type k-moyennes mais il remplace le calcul du centre des clusters par des rencontres aléatoires entre fourmis. Cela lui permet de traiter des données non nécessairement numériques, et d'être efficace sur de grands jeux de données en évitant les calculs de moyennes ou de médoïdes : comme déjà mentionné, LA s'exécute en temps linéaire avec le nombre de données et le nombre de clusters. Ses paramètres sont le nombre de rencontres prises en compte pour le calcul du seuil de similarité, le nombre de rencontres entre une fourmi et les membres d'un nid pour calculer sa similarité au nid, et éventuellement la taille des nids à supprimer à la fin de la procédure. L'algorithme possède l'avantage de ne pas nécessiter la détermination préalable du nombre de clusters recherchés, information rarement disponible a priori, et en particulier absente dans le cadre de l'analyse des usages du web.

4.3 Construction de prototypes

L'étape de clustering précédente fournit une décomposition de l'ensemble de données en sous-groupes homogènes et distincts qui résument la base initiale. Elle permet par là de réduire la quantité d'information à présenter à un utilisateur. Pour la réduire plus encore, on peut définir pour chaque cluster un représentant significatif qui résume ses caractéristiques.

Il existe de nombreuses approches pour résumer des données ou construire des représentants de groupes de données comme par exemple la moyenne, la médiane ou leurs équivalents pondérés. Nous considérons ici la définition de prototype basé sur la notion de typicalité proposée par Rifqi (1996); Lesot et al. (2007) en accord avec les résultats des études cognitives sur les représentants de catégories (Rosch, 1978). Sa différence principale par rapport aux représentants classiques précédents vient du rôle joué par les autres groupes de données : pour les méthodes classiques, le représentant d'un groupe de données dépend exclusivement des données du groupe, alors que le prototype basé sur la notion de typicalité dépend également des membres des autres groupes. En effet, son objectif est de fournir une caractérisation plus complète, en soulignant à la fois les points communs des membres du groupe, comme le font les approches classiques, mais aussi en mettant en évidence leurs traits distinctifs ou discriminants par opposition aux autres groupes. Ainsi, le prototype souligne la spécificité d'un cluster par rapport aux autres, ce qui le rend plus représentatif que d'autres approches (voir Lesot et al. (2005, 2007) pour une comparaison plus complète entre les différentes méthodes).

La formalisation de ce principe conduit à une méthode de construction de prototype en trois étapes détaillées ci-dessous avec les notations suivantes : s est une donnée (par exemple une session utilisateur) et C le cluster auquel elle appartient, S la mesure de similarité utilisée, et δ la mesure de dissimilarité qu'elle induit, définie par $\delta(s_1, s_2) = 1 - S(s_1, s_2)$.

1. Calcul de la ressemblance interne $R(s, C)$ et de la dissimilarité externe $D(s, C)$: elles sont respectivement définies comme la ressemblance moyenne de s aux autres membres de son groupe, et comme sa dissimilarité moyenne aux membres des autres clusters :

$$R(s, C) = avg(S(s, s_2), s_2 \in C) \qquad D(s, C) = avg(\delta(s, s_2), s_2 \notin C)$$

2. Calcul du degré de typicalité qui quantifie à quel point une donnée est typique du cluster auquel elle appartient : il est défini comme l'agrégation de la ressemblance interne et de la dissimilarité externe

$$T(x, C) = \varphi(R(x, C), D(x, C))$$

   φ est l'opérateur d'agrégation qui exprime comment la typicalité dépend de R et D. Dans cet article, nous considérons l'opérateur MICA (Kelman et Yager, 1995), qui a un comportement conjonctif, disjonctif ou de compromis, suivant les valeurs à agréger. Il possède de plus la propriété de renforcement total : si à la fois R et D sont élevées, elles se renforcent mutuellement pour donner une valeur de typicalité encore plus élevée ; si au contraire elles sont faibles tous les deux, elles se pénalisent mutuellement pour donner une valeur plus faible encore.

3. Construction du prototype : le prototype est alors défini comme l'agrégation des membres les plus typiques du groupe (Lesot et al., 2005), ou simplement comme le membre le plus typique. C'est ce dernier choix qui est appliqué dans la suite de l'article.

5 Module de visualisation

Le dernier module de l'outil proposé, qui a été réalisé en Java en s'appuyant sur le moteur de rendu de graphes de la bibliothèque Jung (O'Madadhain et al., 2005), réalise la visualisation des graphes d'usage, dans lesquels les nœuds correspondent aux documents du site et les arcs aux liens suivis par les utilisateurs. Comme énoncé dans la section 1, plusieurs critères doivent être pris en compte pour garantir l'interprétabilité des graphes : le nombre de nœuds et d'arcs doit être limité, le nombre de croisements entre arcs doit être minimal, différentes formes et couleurs doivent être utilisées pour retranscrire plus d'information que la structure du site et des parcours, l'outil doit être interactif pour que l'utilisateur puisse adapter la vue à ses besoins.

L'outil proposé repose sur la classification des sessions et la recherche de navigations typiques pour réduire la quantité de nœuds et d'arcs à représenter. De façon à réduire encore le nombre d'éléments à afficher, l'outil intègre le mécanisme de généralisation d'urls proposé par Fu et al. (2000) : la généralisation à un niveau n coupe l'arborescence du site à une profondeur n et rattache toutes les requêtes sur les pages élaguées à leurs pages parentes encore présentes dans le graphe. Ainsi l'url "$/a/b/c/d.html$" est généralisée au niveau 3 en "$/a/b/c$". Cette généralisation peut être effectuée avant la phase de clustering (pour réduire la quantité de calculs) ou bien après pour simplifier la représentation.

L'outil proposé emploie plusieurs codes couleurs, formes et indices de transparence pour les nœuds et les arcs en fonction de la fréquence des visites des utilisateurs. Chaque couleur est associée à une plage de valeurs : vert $[0-20\%]$, bleu $[20-60\%]$ et rouge $[60-100\%]$. Deux types d'informations sont véhiculés grâce à la forme et la couleur des nœuds et des arcs : la fréquence moyenne de visite depuis le nœud parent et le pourcentage d'utilisateurs qui suivent le lien au moins une fois durant leur session.

Deux nœuds particuliers sont ajoutés aux graphes : le cercle noir est le point de départ virtuel et le carré rouge est le point d'arrivée virtuel, pour toutes les sessions. Ceci permet de mieux comprendre le comportement des personnes qui arrivent sur le site ou qui le quittent.

L'interface de l'outil est décomposée en trois parties (cf figure 1) : la liste des clusters identifiés en haut à gauche, la liste des sessions de chaque cluster en bas à gauche et l'affichage des graphes au centre, en fonction des choix faits par l'utilisateur. L'outil permet notamment de visualiser l'ensemble des sessions sur le même graphe ou bien de ne représenter qu'un seul cluster ou encore un sous-ensemble des sessions d'un cluster, sélectionnées depuis la liste.

6 Résultats préliminaires

6.1 Données

L'outil proposé a été testé sur des données fournies par le muséum d'histoire naturelle de Bourges concernant l'utilisation de son site web. Elles correspondent à 9913 requêtes représentant 1512 sessions et ont été enregistrées sur 164 pages web distinctes.

Bien que le muséum soit consacré de façon générale à l'histoire naturelle, son site web est spécialisé sur l'étude et la protection des chauves-souris, et son index est principalement consacré à ce sujet. L'autre particularité du site web est qu'il ne contient que deux niveaux de profondeur, ce qui ne facilite pas le processus de discrimination par la mesure de similarité utilisée, qui est basée sur le chemin des pages web accédées.

6.2 Résultats expérimentaux

Pour les expériences, une généralisation au niveau 1 des urls a été effectuée lors de la phase d'affichage, ce qui permet de ne conserver, lors de la visualisation, que l'information des catégories de pages (les répertoires).

Le temps de calcul moyen sur 10 tests est de 1.57 ± 0.17 secondes (processeur Core2Duo 2GHz). La qualité moyenne de la partition obtenue, calculée comme le quotient de la distance intra-cluster (la moyenne des distances entre sessions d'un même cluster) et la distance inter-clusters (la moyenne des distances entre sessions affectées à des clusters différents), est de 0.31 ± 0.01. Les clusters sont très compacts ($d_{intra} = 0.19 \pm 0.01$), et relativement bien séparés ($d_{inter} = 0.63 \pm 0.04$).

En ce qui concerne les clusters eux-mêmes, les tests conduisent à l'identification de 5 clusters parmi lesquels 3 profils principaux peuvent être identifiés, illustrés sur la figure 1. La figure montre les parcours des internautes affectés à chacun de ces trois clusters ; les nœuds entourés en bleu clair sont les nœuds appartenant au représentant typique de chaque cluster. Les clusters peuvent être interprétés de la façon suivante : le premier, représenté en haut correspond aux internautes qui consultent uniquement les pages du site en anglais. Aussi l'utilisateur typique de ce comportement accède directement au contenu anglais, sans même passer par la page d'index française, avant de quitter le site.

Le cluster au centre peut être interprété comme les internautes intéressés par les actualités du muséum : ils y accèdent soit directement, soit en passant par la racine du site. Le parcours typique associé correspond aux utilisateurs qui entrent directement sur la page des Actualités : on peut constater que le passage par la racine du site est en effet partagé par des internautes d'autres clusters, et donc moins représentatif de ce comportement particulier.

Enfin, le cluster représenté en bas peut être interprété comme les utilisateurs qui naviguent de façon équitable entre l'index du site ("/") et le contenu html ("html") : le parcours typique associé consiste à accéder à la racine du site puis au contenu html.

Deux fonctionnalités de visualisation supplémentaires disponibles dans l'outil proposé ont été employées pour mieux comprendre les parcours des internautes : (1) l'affichage simultané de plusieurs sessions choisies par l'expert pour voir en quoi elles sont similaires (notamment en comparaison du parcours typique) et (2) la limitation du graphe des usages au voisinage d'une page (dont la taille est paramétrée interactivement) qui permet de voir dans quel contexte cette page est visitée et quittée.

7 Conclusions et perspectives

Nous avons présenté dans cet article un nouvel outil pour l'analyse et la visualisation de sessions web, qui repose sur un algorithme de clustering biomimétique relationnel, Leader Ant, et sur la définition de prototypes basés sur des degrés de typicalité, pour produire une visualisation pertinente de l'usage d'un site web. Le problème de la représentation des sessions web n'est pas trivial, car la complexité du graphe associé à l'activité des internautes croît avec la taille du site et le nombre d'utilisateurs. L'outil proposé permet de visualiser et de manipuler les clusters identifiés en sélectionnant les clusters que l'on souhaite examiner. Il permet donc à un webmaster de localiser aisément les parties du site web que différents groupes d'internautes consultent. Il permet de plus d'étudier les raisons pour lesquelles un groupe d'utilisa-

teurs quitte le site, ce qui peut être utile par exemple pour les sites web commerciaux. Les expériences réalisées sur le site du muséum de Bourges montrent que l'implémentation de l'outil est prometteuse, et illustrent ses capacités à traiter des logs utilisateurs réels.

L'outil a été développé dans un cadre modulaire dont l'avantage est de simplifier l'évolution, puisque chaque composant peut être amélioré indépendamment des autres. Les améliorations envisagées comprennent en particulier l'introduction de nouvelles mesures de similarité entre sessions utilisant le contenu des pages web accédées, la modification de la représentation du graphe par la génération d'imagettes des sites web remplaçant les nœuds textuels ou encore la possibilité d'effectuer des zooms sémantiques ou structurels, par exemple par le biais d'effet fisheye pour permettre à l'utilisateur de se concentrer sur certaines parties du site web.

Références

Baraglia, R. et P. Palmerini (2002). Suggest : A web usage mining system. In *Proc. of IEEE Int. Conf. on Information Technology : Coding and Computing*, pp. 282–287.

Chi, E., J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, et S. K. Card (1998). Visualizing the evolution of web ecologies. In *Proc. of the Conf. of CHI'98*, pp. 400–407.

Eelco, H. et H. Weinreich (2005). Interactive web usage mining with the navigation visualizer. In *Proc. of CHI'05, Extended Abstracts*, pp. 1451–1454.

Fu, Y., K. Sandhu, et M. Shih (1999). Clustering of web users based on access patterns. In Springer (Ed.), *Proc. of the 1999 KDD Workshop on Web Mining*, pp. 21–38.

Fu, Y., K. Sandhu, et M. Shih (2000). A generalization-based approach to clustering of web usage sessions. In *Web Usage Analysis and User Profiling*, pp. 21–38. Springer.

Heer, J. et E. Chi (2001). Identification of web user traffic composition using multi-modal clustering and information scent. In *Proc. of the Workshop on Web Mining, SIAM Conference on Data Mining*, pp. 51–58.

Herman, I., G. Melançon, et M. S. Marshall (2000). Graph visualization and navigation in information visualization : A survey. *IEEE Trans. on Visualization and Computer Graphics 6*(1), 24–43.

Hong, J. I., J. Heer, S. Waterson, et J. A. Landay (2001). Webquilt : a framework for capturing and visualizing the web experience. In *World Wide Web*, pp. 717–724.

Kelman, A. et R. Yager (1995). On the application of a class of MICA operators. *Int. Journal of Uncertainty, Fuzzyness and Knowledge-Based Systems 7*, 113–126.

Krishnapuram, R., A. Joshi, O. Nasraoui, et L. Yi (2001). Low-complexity fuzzy relational clustering algorithms for web mining. *IEEE Transactions on Fuzzy Systems 9*, 595–607.

Labroche, N. (2003). *Modélisation du système de reconnaissance chimique des fourmis pour le problème de la classification non supervisée : application à la mesure d'audience sur Internet*. Ph. D. thesis, Laboratoire d'Informatique de l'Université de Tours.

Labroche, N. (2006). Fast ant-inspired clustering algorithm for web usage mining. In *Proc. of IPMU'06*, pp. 2668–2675.

Labroche, N. (2007). Learning web users profiles with relational clustering algorithms. In *Workshop on Intelligent Techniques for Web Personalization AAAI Conference*, pp. 54–64.

Labroche, N., N. Monmarché, et G. Venturini (2003). Antclust : Ant clustering and web usage mining. In *Proc. of the Genetic and Evolutionary Computation Conference, Gecco'03*, pp. 25–36.

Lesot, M.-J., L. Mouillet, et B. Bouchon-Meunier (2005). Fuzzy prototypes based on typicality degrees. In *Proc. of the 8th Fuzzy Days 2004*, pp. 125–138.

Lesot, M.-J., M. Rifqi, et B. Bouchon-Meunier (2007). Fuzzy prototypes : From a cognitive view to a machine learning principle. In *Fuzzy Sets and Their Extensions : Representation, Aggregation and Models*, pp. 431–452. Springer.

Masseglia, F., P. Poncelet, et M. Teisseire (1999). Using data mining techniques on web access logs to dynamically improve hypertext structure. *ACM SigWeb Letters 8*(3), 1–19.

Mobasher, B. (2006). Data mining for personalization. In *The Adaptive Web : Methods and Strategies of Web Personalization*, Volume 4321 of *LNCS*, pp. 90–135. Springer.

Mukherjea, S. et J. Foley (1995). Visualizing the worl-wide web with the navigational view builder. Technical report, Graphics, Visualization and Usability Center, College of Computing, Georgia Institute of Technology.

Munzner, T. et P. Burchard (1995). Visualizing the structure of the World Wide Web in 3D hyperbolic space. In *Proc. of the VRML 1995 Symposium*, pp. 33–38.

Nasraoui, O., A. Joshi, et R. Krishnapuram (1999). Relational clustering based on a new robust estimator with application to web mining. In *Proc. of NAFIPS'99*, pp. 705–709.

Nasraoui, O., R. Krishnapuram, A. Joshi, et T. Kamdar (2002). Automatic web user profiling and personalization using robust fuzzy relational clustering. In *E-Commerce and Intelligent Methods*, pp. 233–261. Springer.

O'Madadhain, J., D. Fisher, P. Smyth, S. White, et Y. Boey (2005). Analysis and visualization of network data using jung. Available online at http ://jung.sourceforge.net/doc/index.html.

Pitkow, J. et K. Bharat (1994). Webviz : A tool for world wide web access log analysis. In *Proc. of the First Int. World-Wide Web Conference*, pp. 271–277.

Rifqi, M. (1996). Constructing prototypes from large databases. In *Proc. of IPMU'96*, pp. 301–306.

Rosch, E. (1978). Principles of categorization. In E. Rosch et B. Lloyd (Eds.), *Cognition and categorization*, pp. 27–48. Lawrence Erlbaum associates.

Youssefi, A., D. Duke, et M. J. Zaki (2004). Visual web mining. In *Poster Proc. of WWW'04 Conference*, pp. 394–395.

Summary

This paper introduces a new tool for web usage mining and visualization that relies on the bio-mimetic relational clustering algorithm Leader Ant and the definition of prototypes based on typicality computation to produce an efficient visualization of the activity of users on a website. The tool is tested and evaluated on real web log files from the museum of Bourges website and shows that it can easily produce meaningful visualizations of typical user navigation.

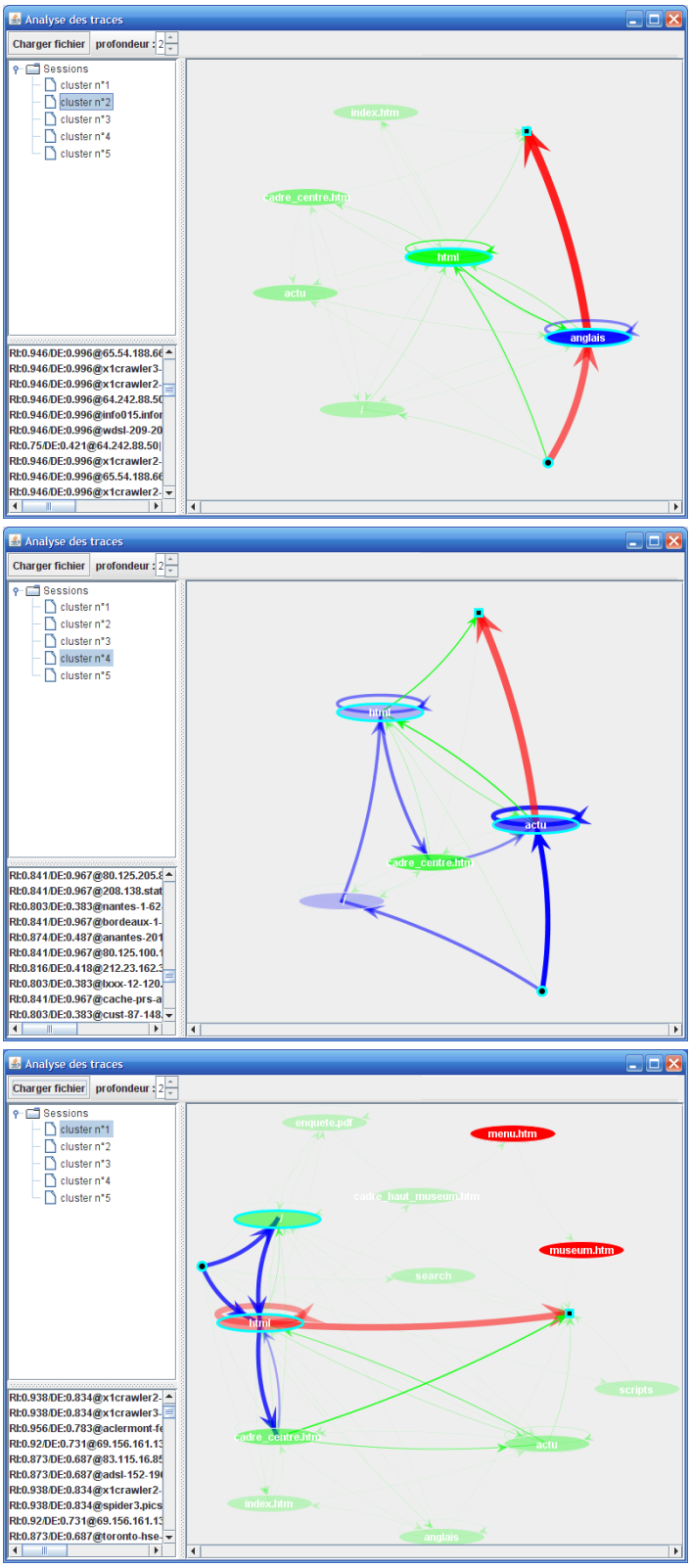


FIG. 1 – *Captures d'écran des résultats obtenus avec les données du muséum de Bourges pour les trois clusters principaux.*

CAViz : Exploration interactive des résultats de l'analyse factorielle des correspondances pour des images

Nguyen-Khang Pham*,**, Annie Morin*, Patrick Gros*

* IRISA, Campus de Beaulieu F-35042, Rennes Cedex
{pnguyenk,amorin,pgros}@irisa.fr
http://www.irisa.fr
** Université de Cantho, Campus III, 1 Ly Tu Trong, Ville de Cantho, Vietnam
pnkhang@cit.ctu.edu.vn
http://www.cit.ctu.edu.vn

Résumé. Notre investigation a pour but d'explorer interactivement les résultats de l'Analyse factorielle des correspondances (AFC) (Benzécri, 1973) appliquée sur des images afin de pouvoir extraire des connaissances et de mieux interpréter les résultats obtenus à l'issue d'une AFC. Nous proposons un outil graphique interactif, CAViz, qui permet de visualiser et d'extraire des connaissances à partir des résultats de l'analyse factorielle des correspondances (AFC) sur les images. Originellement l'AFC est destinée à l'analyse d'un tableau de contingence. Une application est l'analyse des données textuelles. En Analyse des données textuelles, le tableau de contingence croise mots et documents. Pour l'adaptation de l'AFC aux images, la première étape consiste donc à définir des « mots visuels » dans les images (analogue des mots dans les textes). Ces mots sont construits à partir de descripteurs locaux (SIFT, Scale Invariant Feature Transform) des images.
CAViz projette le nuage des points dans des plans factoriels et permet d'extraire visuellement des informations intéressantes comme : des mots caractérisant, des facteurs importants en utilisant des indicateurs pertinents de l'AFC (qualité de représentation, et contribution à l'inertie). Une application à la base Caltech4 (Sivic et al., 2005) démontre l'intérêt de CAViz pour l'analyse des résultats de l'AFC.

1 Introduction

La fouille de données (Fayyad and et al., 1996) vise à extraire des connaissances utiles cachées à partir de grandes bases de données. Cette utilité est très liée au but des utilisateurs, c'est-à-dire, que c'est l'utilisateur qui peut déterminer si les connaissances résultantes répondent à son but. Il est donc préférable que les outils pour la fouille de données soient interactifs et permette la participation des utilisateurs. L'idée ici est d'augmenter la participation humaine à travers des techniques de visualisation interactive dans l'environnement de fouille de données. Au cours de la dernière décennie, un grand nombre de méthodes de visualisation développées dans différents domaines ont été utilisés dans l'exploration des données et le processus d'extraction des connaissances (Fayyad et al., 2001), (Keim, 2002). Les méthodes de visualisation sont utilisées pour la sélection des données (phase de prétraitement) et la visualisation des résultats (phase de poste-traitement). Certaines méthodes d'extraction vi-

suelle de données récentes (Ankerst et al., 2000), (Do et Poulet, 2004) essaient de faire participer plus intensivement l'utilisateur à la fouille de données par l'utilisation de la visualisation. Cette coopération peut apporter des avantages comme l'utilisation des connaissances du domaine au cours de la construction du modèle, l'amélioration de la confiance et de la compréhensibilité des modèles obtenus en utilisant la capacité de reconnaissance des formes de l'humain dans la phase de construction et exploration du modèle.

En ADT, l'outil Bi-Qnomis (Morin, 2004) développé par M. Kerbaol permet de visualiser les résultats et de trouver des thèmes relevant dans un corpus textuel traité par une AFC. Les nuages des points (documents et mots) sont projetés sur un plan factoriel. Puis pour chaque axe, on fait une liste des mots ayant une forte contribution (correspondant à une méta-clé). Ces mots sont affichés à gauche et le titre des documents bien représentés sur ce plan sont affichés en haut. L'utilisateur peut cliquer sur un titre pour voir en détail un document. En examinant des métaclés et des documents un expert aura ensuite le résumé du contenu de ces documents. Bi-Qnomis fournit aussi une visualisation des métaclés et leurs mots en utilisation un arbre hyperbolique.

Par contre, pour les images on rencontre des difficultés car il n'y a pas de « vrais » mots au sens littéral dans les images. On utilise donc des « mots visuels » (analogue des mots textuels) à la place des mots textuels et les images comme les documents. Récemment, certaines méthodes développées originellement pour l'analyse des données textuelles (ADT) comme pLSA (probabilistic Latent Semantic Analysis) (Hofmann, 1999), LDA (Latent Dirichlet Allocation) (Blei, 2003) sont appliquées en analyse d'images, par exemple pour la classification des images (Willamowski, 2004), la découverte des thèmes dans l'image (Sivic et al., 2005), la classifications des scènes (Bosch et al., 2006), et la recherche d'images (Lienhart et Slaney (2007)). Ces méthodes essaient de modéliser le corpus et de réduire la dimension du problème. Un des inconvénients de ces méthodes est l'utilisation d'un modèle ad-hoc et d'un algorithme EM pour trouver une optimisation locale. Par ailleurs, il est difficile d'interpréter les résultats de ces méthodes. La plupart des travaux utilisent ces méthodes comme boîtes noires.

Nous nous intéressons ici à l'adaptation de l'AFC pour les images et à l'interprétation de ses résultats à l'aide des indicateurs pertinents à travers un outil de visualisation dans lequel l'utilisateur peut explorer les résultats de façon interactive pour mieux les comprendre. L'article est organisé de la façon suivante : nous décrivons brièvement la méthode de l'AFC dans la section 2. La section 3 présente l'extraction interactive des connaissances à partir des résultats de l'AFC. Dans la conclusion, nous présentons les perspectives de ce travail.

2 Adaptation de l'AFC aux images

2.1 AFC

L'AFC est une méthode exploratoire classique pour l'analyse des tableaux de contingence. Elle a été proposée par J. P. Benzécri (1973) dans le contexte de la linguistique, c'est-à-dire pour l'analyse de données textuelles. La première étude a été réalisée sur les tragédies de Racine. L'AFC sur un tableau croisant des mots et des documents permet de répondre aux questions suivantes : y a-t-il des proximités entre certains mots ? Y a-t-il des proximités entre certains documents ? Y a-t-il des liens entre certains mots et certains documents ? L'AFC comme la plupart des méthodes factorielles utilise une décomposition en valeurs singulières

d'une matrice particulière et permet la visualisation des mots et des documents dans un espace de dimension réduit. Cet espace de dimension réduit a la particularité d'avoir un nuage de points projetés (mots et/ou documents) d'inertie maximale. Par ailleurs, l'AFC fournit des indicateurs pertinents pour l'interprétation des axes comme la contribution d'un mot ou d'un document à l'inertie de l'axe ou la qualité d'un mot et/ou d'un document sur un axe (Morin, 2004).

FIG. 1 – *Les points d'intérêt détectés par le détecteur Hessian-Affine.*

2.2 Construction des mots visuels et représentation des images

Afin d'adapter l'AFC aux images, nous devons représenter le corpus d'images sous forme d'un tableau de contingence. Ici on traite les images comme les documents et les « mots visuels » (à définir) comme des mots.

Les mots dans les images, appelés mots visuels, doivent être calculés pour constituer un vocabulaire de $\boldsymbol{N}$ mots. Chaque image sera donc représentée enfin par un histogramme de mots. La construction des mots visuels se fait en deux étapes : (i) calcul des descripteurs locaux pour un ensemble d'images, (ii) classification (clustering) des descripteurs obtenus. Chaque cluster correspondra à un mot visuel. Il y aura donc autant de mots que de clusters obtenus à l'issue de l'étape (ii). Le calcul des descripteurs locaux dans une image se fait aussi en deux étapes : il faut d'abord détecter des points d'intérêt dans l'image. Ces points d'intérêt sont, soit des maximums du Laplacien de Gaussien (Lindeberg, 1998), soit des extremums locaux 3D de la différence de Gaussien (Lowe, 1999), soit des points extraits par un détecteur Hessian-affine (Mikolajczyk, 2004). Ensuite, le descripteur de ce point d'intérêt est calculé sur le gradient des niveaux de gris dans la région autour du point. On a sélectionné des descripteurs invariants à la rotation et au changement d'échelle, les descripteurs SIFT (Lowe, 2004). Chaque descripteur SIFT est un vecteur à 128 dimensions. La seconde étape consiste à former des mots visuels à partir des descripteurs locaux calculés à l'étape précédente. La plupart des travaux effectue un ***k-means*** sur les descripteurs locaux et prend les moyennes de chaque cluster comme mots visuels (Willamowski, 2004, Sivic, 2005, Bosch et al., 2006). Après avoir construit le vocabulaire visuel, chaque descripteur est affecté au cluster le plus proche. Pour cela, on calcule dans $\boldsymbol{R^{128}}$ les distances de chaque descripteur aux représentants des clusters définis précédemment. Une image est ensuite caractérisée par la fréquence de ses descripteurs dans chaque cluster. On obtient ainsi un tableau de contingence croisant les images et les clusters.

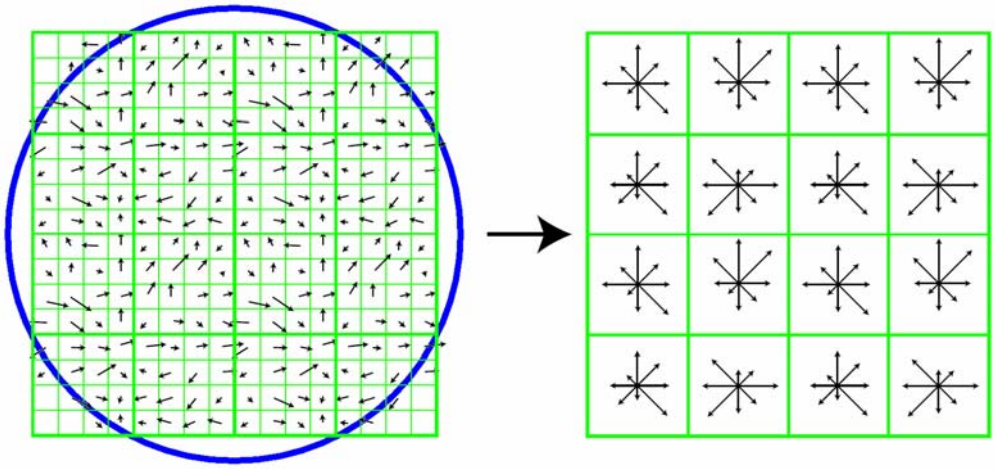


FIG. 2 – *Un descripteur SIFT calculé à partir de la région d'autour du point d'intérêt. A gauche : gradient de l'image, à droite : descripteur du point d'intérêt.*

3 Exploration interactive des résultats de l'AFC

3.1 Projection sur le plan factoriel

L'écran est divisé en deux parties : à gauche, nous projetons les nuages de points (images et/ou des mots visuels) et la partie droite est réservée pour afficher les images sélectionnées. Un point « image » s'affiche sous forme d'un carré rouge, un point « mot » se représente par un carré bleu. L'utilisateur peut sélectionner une ou un groupe d'images en pointant sur l'image intéressante. Toutes les images se trouvant dans un voisinage de rayon r de l'image intéressante sont également affichées. Les points correspondant aux images sélectionnées seront changés de couleur (de rouge en vert). Les mots visuels (sous forme d'une ellipse) seront dessinés sur les images. Les images sélectionnées sont affichées dans la partie droite. Cela nous donne tout de suite un résumé général du contenu de ces images.

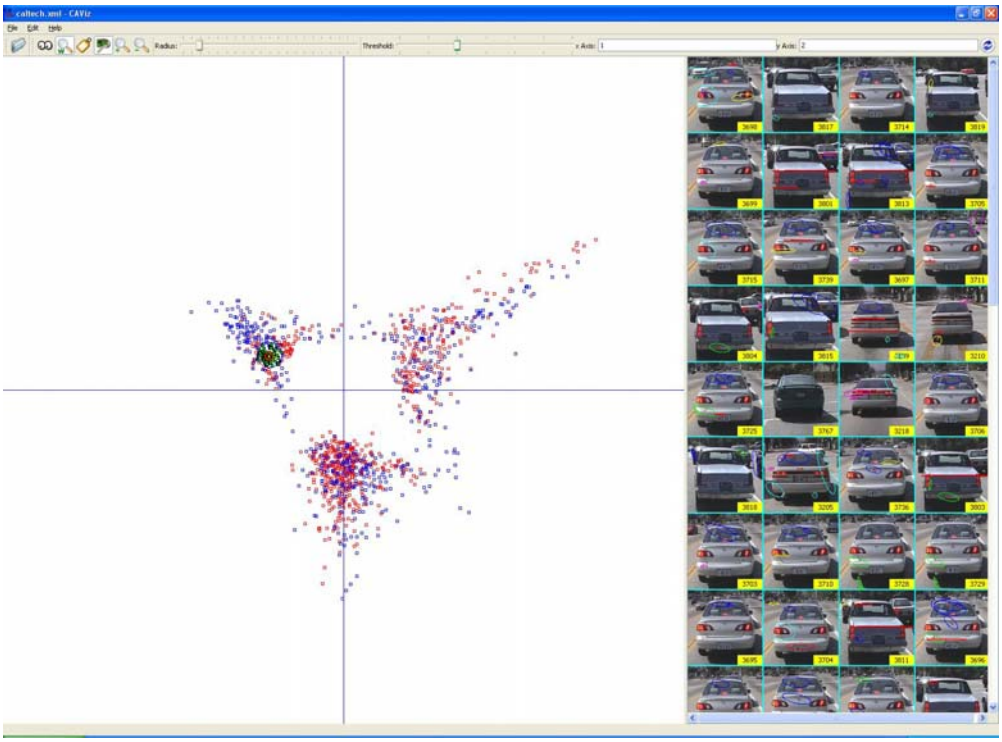


FIG. 3 – *Projection de la base Caltech sur les axes 1 et 2.*

Pour se concentrer sur des images et/ou des mots intéressants, nous n'affichons dans le plan que les images et/ou les mots dont la contribution à l'inerte est grande, généralement 2 ou 3 fois la contribution moyenne. L'inertie totale sur un axe est égale à la valeur propre associée à cet axe. Le seuil est donc facile à déterminer. La figure 3 affiche la projection de la base Caltech4 (Sivic et al., 2005) sur les axes 1 et 2 avec le seuil égal à 2 fois la contribution moyenne. M. Kerbaol appelle métaclés les groupes des mots dont la contribution est très élevée sur un axe. Nous avons donc 2 métaclés par axe, une positive et une négative. Les mots visuels appartenant à chaque méta-clé seront affiché sur l'image correspondante. Dans la figure 4, nous disposons les métaclés et leurs mots visuels. L'image se trouvant à chaque bout des axes correspond à une métaclé. L'image en haut à gauche est superposée par des mots visuels proches d'elle. Il est facile de vérifier que la plupart de ces mots se trouve à la fois dans la méta-clé à gauche et la métaclé en bas.

L'information concernant une image est également extraite de façon interactive en sélectionnant l'image en question. Il y a 2 indicateurs pertinents pour interpréter le résultat de l'AFC : la qualité de représentation d'une part et la contribution à l'inertie d'autre part. Ces informations nous aideront à certaines tâches ultérieures. La figure 5 montre un exemple de l'extraction interactive des informations. L'image dans cet exemple est bien représentée par les axes 2 (négative), 3 (négative), 18 (négative), 12 (positive), … (les premiers axes) et contribue beaucoup à l'inertie des axes 2, 18, 3, 12, … (les premiers axes).

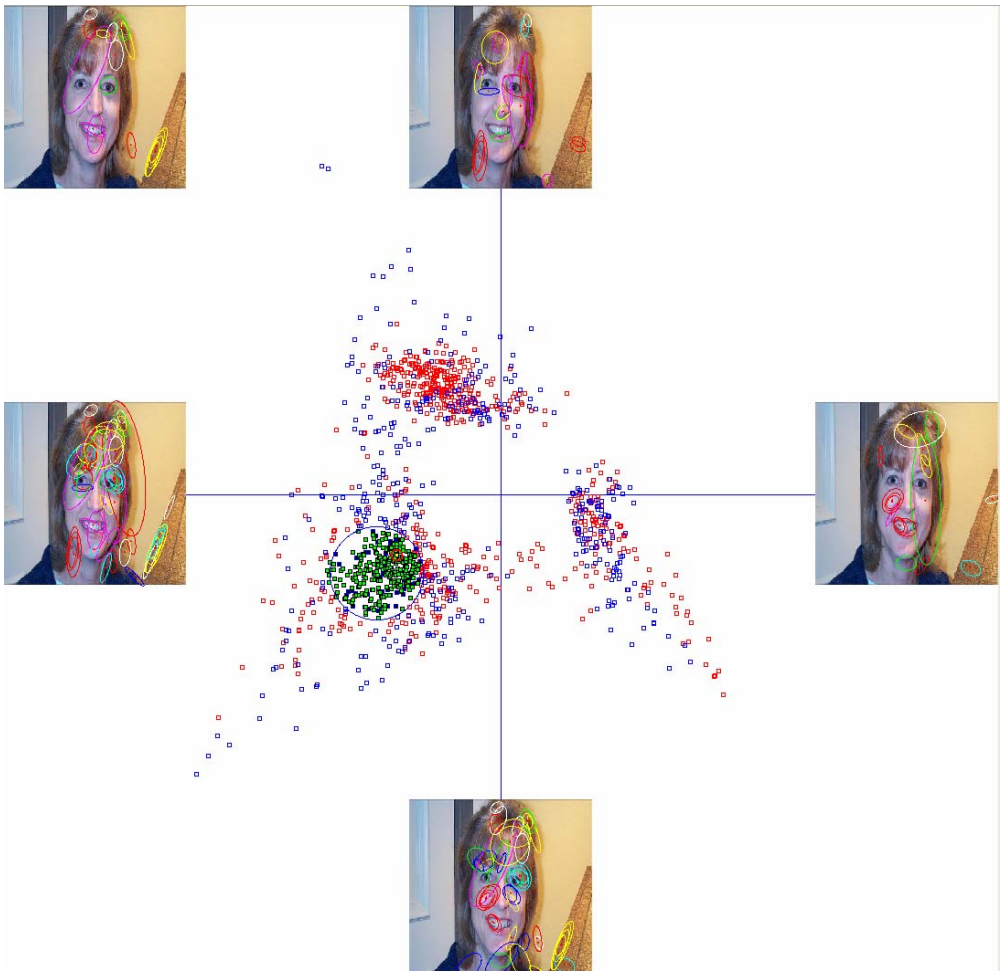


FIG. 4 – *Visualisation des métaclés.*

Un des avantages de l'AFC est que l'on peut afficher des mots et des documents (images) dans un même plan. Les mots proches d'une image caractérisent bien cette image. Il est aisé de visualiser les mots caractérisant bien une ou un groupe d'images (correspondant à un thème) en sélectionnant l'image et affichant les mots proches de cette image. La figure 6 montre les mots qui représentent bien le thème « visage ».

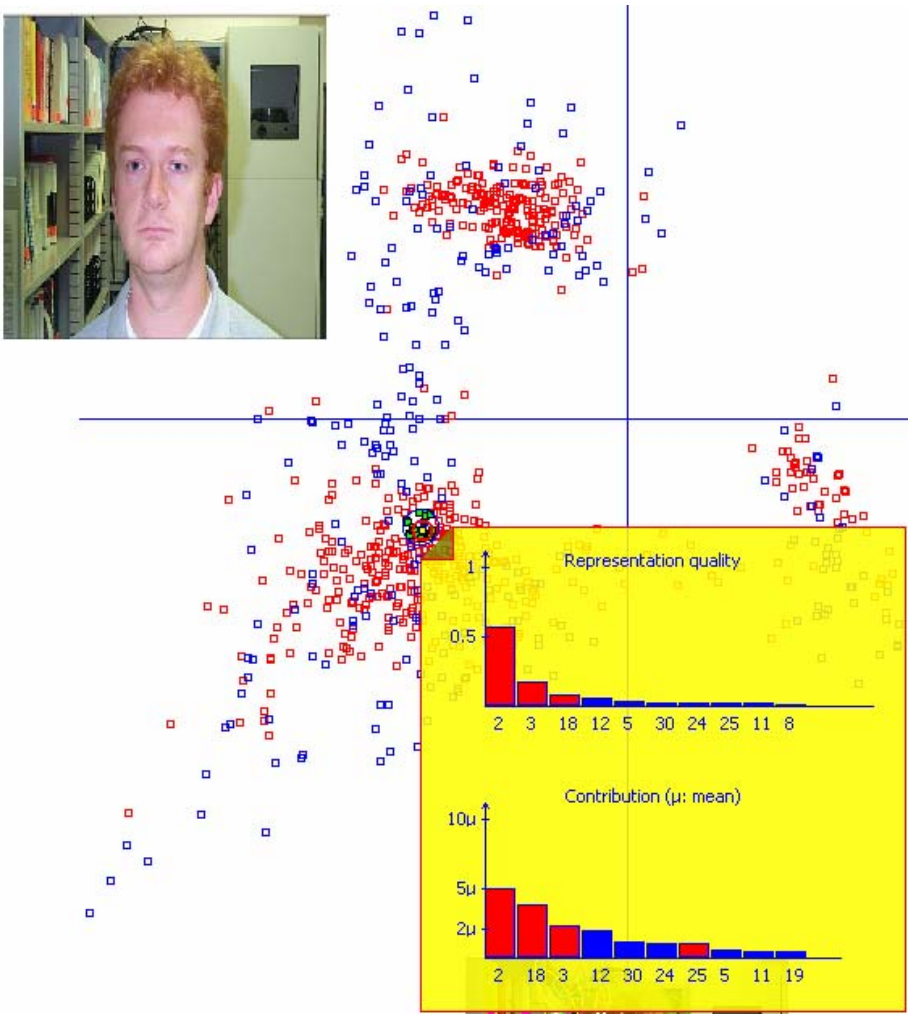


FIG. 5 – *Extraction des informations concernant l'image : histogramme des qualités de représentation sur les axes et histogramme des contributions à l'inertie des axes d'une image.*

3.2 Découverte de thèmes d'images

Après avoir affiché le nuage des points sur un plan factoriel, un groupe de points proches l'un de l'autre peut définir un thème dans ce plan factoriel. Néanmoins, pour pouvoir découvrir les thèmes plus fins, il nous faut chercher les axes (i.e le plan) qui représentent ces thèmes. CAViz permet d'afficher les informations sur la qualité de représentation des axes et fournit un couplage des vues qui permet de voir la qualité de représentation d'un groupe d'images.

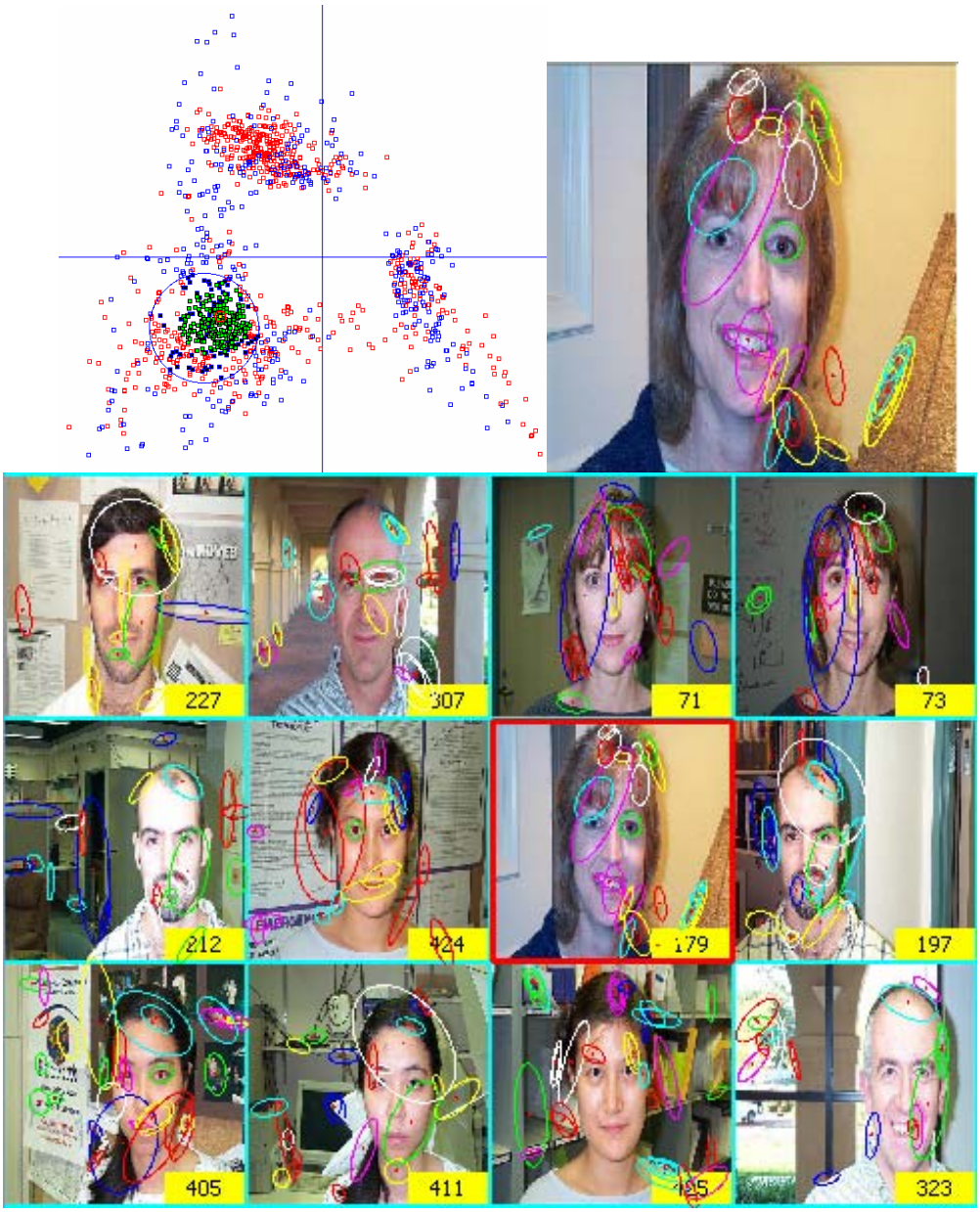


FIG. 6 –*Les mots caractérisant le thème « visage ».*

3.2.1 Qualité de représentation des images

La qualité de représentation d'un point ***i*** (correspond à l'image ***i***) sur l'axe ***j*** est le cosinus carré de l'angle entre l'axe ***j*** et le vecteur joignant le centre du nuage au point ***i***. Plus le cosinus carré est proche de 1, plus la position du point observé dans la projection est proche de la position réelle du point dans l'espace. On utilise ce critère pour chercher les axes représentants bien les images et les mots.

Le nuage de points est projeté sur le premier plan factoriel (i.e sur les axes 1 et 2). On extrait la qualité de représentation des images sur les axes en regardant les premiers facteurs dans l'histogramme de qualité de représentation. Ces informations nous donnent un guide pour chercher le bon plan factoriel qui représente bien un groupe d'images.

3.2.2 Couplage des vues

CAViz affiche les points dans la partie gauche et les images sélectionnées dans la partie droite. En cliquant une image à droite, le point correspondant à gauche est sélectionné et ses informations sont affichés. Ce couplage des vues nous permet de sélectionner facilement les images intéressants La perception visuelle humaine est un bon outil pour reconnaissance des formes. Nous sélectionnons les images similaires à droites et regardons la qualité de représentation de ces images. Si on trouve qu'il y a de mêmes axes qui représentent bien les images sélectionnées à droite, on prend alors ces axes et projette les points dans les axes.

3.2.3 Découverte de thèmes

Nous donnons ici une étude de cas sur la découverte de thèmes dans la base Caltech4 (Sivic et *al.*, 2005) tirée da la base Caltech101 (Fergus et *al.*, 2003). Cette base contient 4090 images répartie en 5 catégories. Le tableau 1 décrit cette base.

Catégories	nombre d'images
Faces	435
Motorbikes	800
Airplanes	800
Backgrounds	900
cars (rear)	1155

TAB. 1 – *Description de la base caltech4*

FIG. 7 – *Les images tirées de la base Caltech4.*

On applique une AFC et projette les points dans le premier plan et pointe sur un groupe à gauche (correspondant aux voitures). Les images sont affichées donc à droite. On commence à sélectionner des images à droite en cliquant et regardant la qualité de représentation. On a trouvé qu'il y a des images qui sont bien représentées par les axes 6 (partie négative) et 7 (partie positive) et qu'il y a des images bien représentés par les axes 11 (partie négative) et 12 (partie positive) (figures 6, 7) . On projette ensuite les points selon ces axes. On trouve alors des thèmes plus fins qui contiennent des images très similaires.

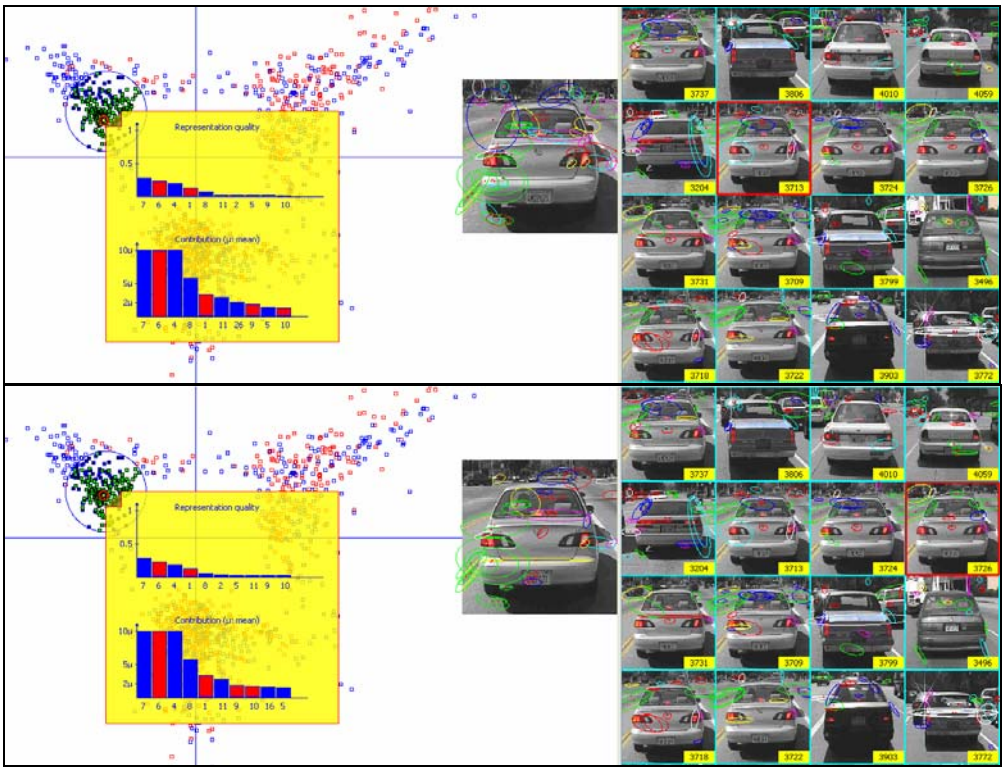


FIG. 8 –*Les images bien représentées par les 2 axes 6 (rouge : négative) et 7 (bleu : positive).*

En projetant sur les 2 axes 6 et 7, on trouve qu'il y a un groupe de points en haut (axe 6-négative) à gauche (axe 7-positive). On choisit ce groupe et voit les images affichées à droite : ces images sont très similaires (ce sont les voitures blanches, figure 10)

Par ailleurs, on trouve aussi un autre groupe de points en bas à gauche. On fait la même chose comme le groupe précédant, on trouve un autre thème pour les voitures rouges ! De même façon, sur les 2 axes 11 et 12, on trouve encore un autre thème pour des voitures (figure 11).

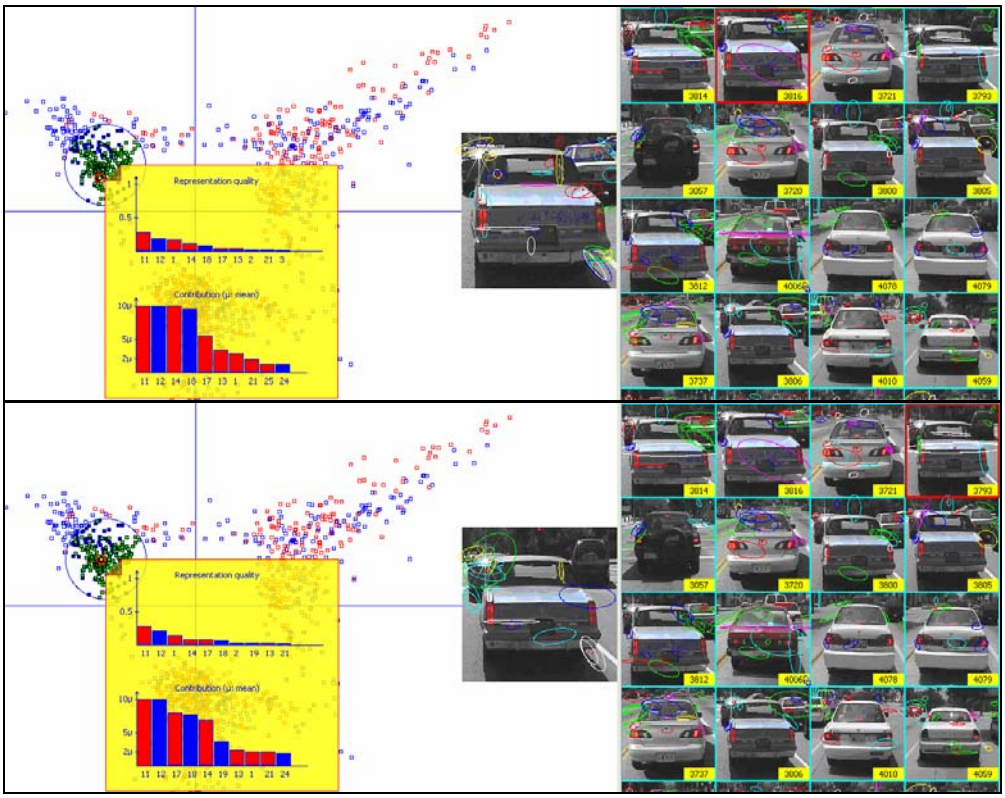


FIG. 9 –*Les images bien représentées par les 2 axes 11 et 12.*

4 Conclusion et perspectives

Nous avons présenté dans cet article un outil, appelé CAViz, qui visualise les résultats de l'AFC sur les images. Cet outil aide l'utilisateur à extraire des connaissances, à interpréter les résultats de l'AFC. Nous avons présenté une application sur la découverte des thèmes d'images dans la base Caltech4. Les thèmes obtenus dans cette étude ont démontré l'intérêt de CaViz pour l'interprétation de l'AFC en utilisant des indicateurs comme la qualité de représentation et la contribution à l'inertie.

Quelques améliorations seront utiles comme la recherche rapide d'axes importants pour un groupe des points et la flexibilité dans la sélection des points. Pour cela on peut extraire des informations du centre de gravité du groupe ou utiliser un rectangle, une ellipse ou une forme arbitraire au lieu d'un cercle. Un des développements possibles de cette méthode est la construction d'un système qui facilite l'exploration, la navigation, et l'étiquetage des images en intégrant les informations visuelles et d'autres informations associées aux images.

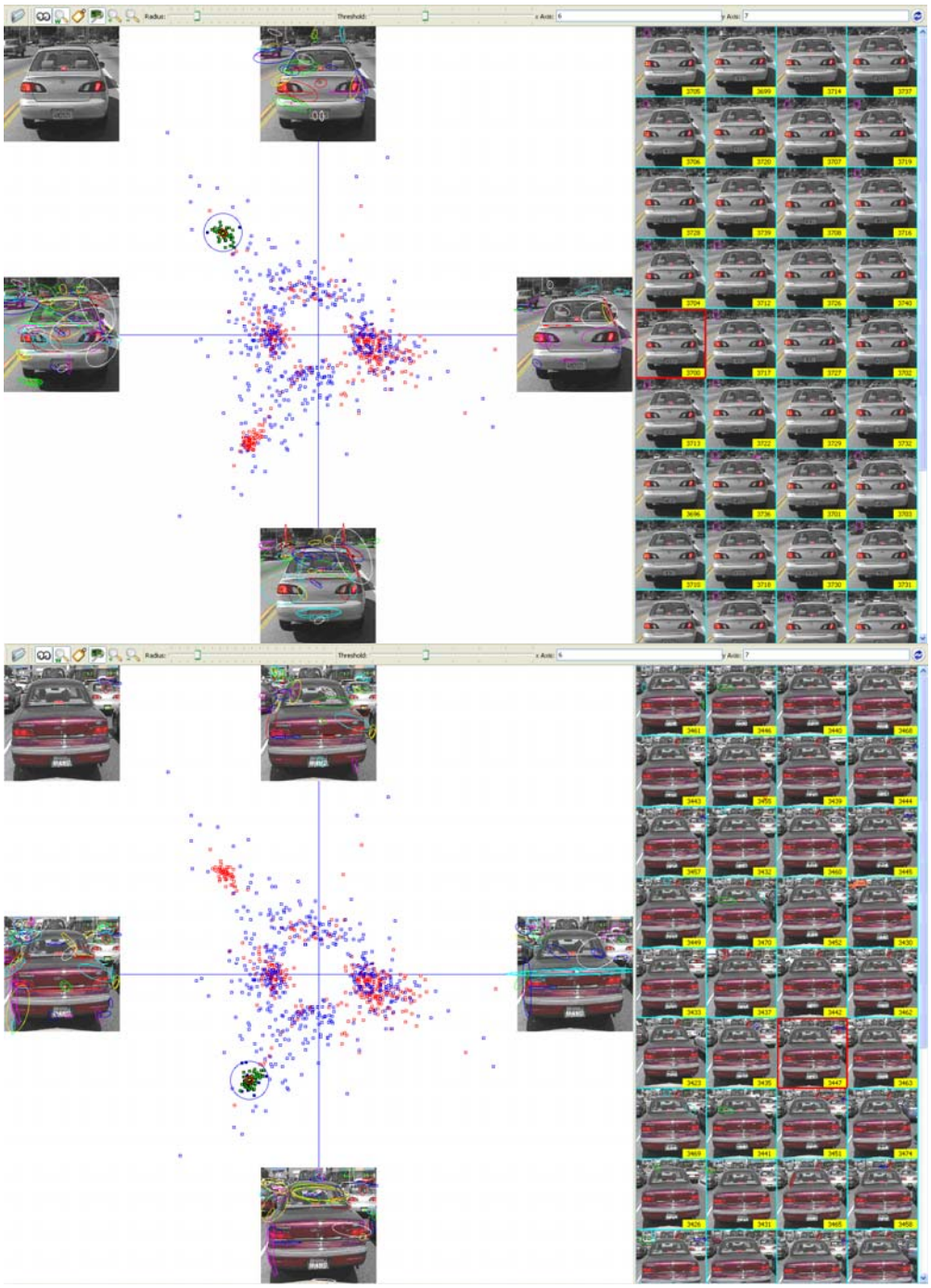


FIG. 10 – *Découvert du thème en projetant sur le bon plan factoriel : on trouve 2 différents thèmes « voitures » sur le plan 6 – 7.*

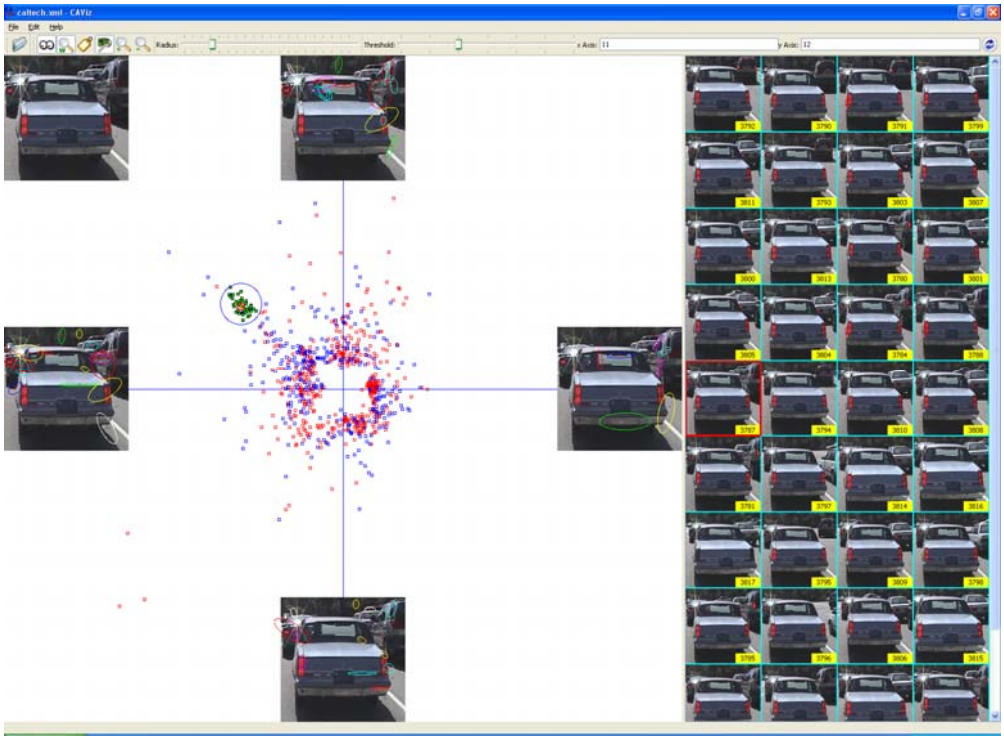


FIG. 11 – *Un autre thème « voiture » sur le plan 11 – 12.*

Remerciements

Nous remercions A. Zisserman pour préparer les données de Caltech4 et F. Poulet pour ses commentaires.

Références

Ankerst, M., M. Ester, and H-P Kriegel (2000). Towards an effective cooperation of the computer and the user for classification. In *Proceeding of KDD'00, 6th ACM SIGKDD*, 179 – 188.

Benzécri, J.-P. (1973). *L'analyse des correspondances*. Paris: Dunod

Bosch, A., A. Zisserman, and X. Munoz (2006). Scene classification via plsa. In *Proceedings of the European Conference on Computer Vision*.

Do, T-N., and F. Poulet (2004). Enhancing svm with visualization. In *Discovery Science 2004, E. Suzuki et S. Arikawa Eds.*, 183 – 194.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37 – 54.

Fayyad, U., G. Grinstein, and A. Wierse (2001). Information Visualization in Data Mining and Knowledge Discovery. Morgan Kaufmann Publishers.

Fergus, R., P. Perona, and A. Zisserman (2003). Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 264 – 271.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI'99)*.

Keim, D. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1 – 8.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, 60(2): 91 – 110.

Mikolajczyk, K. and C. Schmid (2004). Scale and affine invariant interest point detectors. In *Proc. IJC V*, 60(1): 63 – 86.

Morin, A. (2004). Intensive Use of Correspondence Analysis for Information Retrieval, In *Proceedings of the 26th International Conference on Information Technology Interfaces*, 255 – 258.

Sivic, J., B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman (2005). Discovering object categories in image collections. In *Proceedings of the International Conference on Computer Vision*.

Willamowski, J., D. Arregui, G. Csurka, C. Dance, and L. Fan (2004). Categorizing nine visual classes using local appearance descriptors. *Workshop Learning for Adaptable Visual Systems (ICPR 2004) Cambridge, United Kingdom*.

Summary

Our investigation aims at interactively exploring, interpreting the results of Correspondence Analysis (CA). We have proposed an interactive graphical tool, CAViz for exploring and interpreting CA results. A case study on topic discovery shows that CAViz has given an insight into the CA results.

Fouille de données à l'aide d'un environnement de programmation visuelle

Thanh-Nghi Do*, Jean-Daniel Fekete**

Equipe Aviz, INRIA Futurs/LRI, Bât.490, Université Paris Sud 91405 Orsay Cedex
*dtnghi@lri.fr
http://www.lri.fr/~dtnghi
**Jean-Daniel.Fekete@inria.fr
http://www.lri.fr/~fekete

Résumé. Nous présentons un nouvel environnement de programmation visuelle (V4Miner) pour l'extraction interactive de connaissances à partir de données. La plate-forme utilise simultanément un ensemble de méthodes de visualisation et d'apprentissage automatique permettant de découvrir de manière intuitive, interactive et rétroactive des connaissances. La tâche de fouille est exprimée sous la forme d'un flot graphique de données où une méthode est représentée par un composant graphique de JavaBeans, la communication entre les méthodes se base sur un bus de données. L'utilisateur peut créer et contrôler totalement le processus d'extraction de connaissances sans aucune programmation. Notre plate-forme de programmation visuelle permet d'une part de réduire la complexité d'une tâche de fouille de données et d'autre part d'améliorer la qualité et la compréhensibilité des résultats obtenus. Des premiers résultats sont présentés sur des ensembles de données du Kent Ridge Bio Medical Dataset Repository.

1 Introduction

Le volume de données stockées double actuellement tous les 9 mois (Fayyad et al, 2004) et donc le besoin d'extraction de connaissances dans les grandes bases de données est de plus en plus important. La fouille de données (Fayyad et al, 1996) vise à traiter des ensembles de données pour identifier des connaissances nouvelles, valides, potentiellement utilisables et compréhensibles. Seul l'utilisateur peut déterminer la pertinence des résultats par rapport à ses attentes. Les outils de fouille de données doivent donc être interactifs et anthropocentrés.

(Fayyad et al, 2001) et (Keim, 2002) ont mentionné l'utilisation des méthodes graphiques interactives pour augmenter le rôle de l'utilisateur dans l'analyse exploratoire et la fouille de données. Les méthodes de visualisation peuvent être utilisées pour le pré-traitement de données (par exemple la sélection de données), en fouille de données (utilisation de la capacité humaine en reconnaissance des formes) et en post-traitement (par exemple pour voir les résultats pour améliorer la confiance et la compréhensibilité du modèle).

Dans cet article, nous proposons un environnement de programmation visuelle (V4Miner) permettant à l'utilisateur d'extraire de manière intuitive et interactive des connaissances à partir de données en utilisant simultanément un ensemble de méthodes de visualisation et d'apprentissage automatique. La rétroaction est prise en compte dans le

processus de fouille de données parce que l'environnement permet à l'utilisateur de créer, contrôler et interagir avec le processus d'extraction de connaissances sans aucune programmation. Une tâche de fouille est exprimée par un flot visuel de données où une méthode est représentée par un composant en JavaBeans (Sun Microsystems Inc. 1994-2007), la communication entre les méthodes se base sur un bus de données. L'utilisateur peut facilement concevoir lui-même un flot de données et paramétrer visuellement les composants. Ensuite le système peut générer le programme correspondant ou l'utilisateur peut lancer la tâche d'analyse de données à partir de l'environnement graphique. Dans ces deux cas, l'utilisateur peut contrôler de manière interactive et intuitive le processus d'extraction de connaissances, il peut faire des retours en arrière à n'importe quelle étape ou modifier des paramètres en considérant des résultats intermédiaires pour améliorer la qualité des résultats. Nous avons fait l'expérimentation de fouille des données bio-médicales (Jinyan & Huiqing, 2002) et obtenu des résultats intéressants. Une tâche d'analyse des données bio-médicale est difficile parce qu'on traite souvent un grand nombre de dimensions (des milliers) et un petit nombre d'individus (des centaines). A l'aide de V4Miner, cette tâche est rapidement exprimée par un flot visuel de données qui utilise simultanément des méthodes de sélection de dimensions, de classification et de visualisation. On peut interactivement lancer de manière complète ou partielle le flot pour analyser ces données bio-médicales. Les résultats obtenus indiquent que notre plate-forme V4Miner d'une part réduit la complexité d'une tâche de fouille de données et d'autre part permet d'améliorer la qualité et la compréhensibilité des résultats obtenus.

Le paragraphe 2 présente brièvement l'état de l'art sur les plate-formes existantes d'expérimentation pour la fouille de données. Le paragraphe 3 présente ensuite la conception générale, les fonctionnalités et les outils dans l'environnement de programmation visuelle pour la fouille de données. Le paragraphe 4 présente les études de cas des tâches de fouille de données bio-médicales à l'aide de la plate-forme de programmation visuelle avant la conclusion et les travaux futurs dans le paragraphe 5.

2 Etat de l'art

On trouve quelques plate-formes d'expérimentation pour la fouille de données dans le répertoire des logiciels du site KDnuggets (http://www.kdnuggets.com/software/index.html).

Weka (Witten & Frank, 2000) est une bibliothèque gratuite qui implémente en Java des algorithmes d'apprentissage et une plate-forme de programmation visuelle. Il donne à l'utilisateur la possibilité d'ajouter facilement ses propres algorithmes, mais il n'y a pas beaucoup de méthodes de visualisation et de techniques interactives.

HIVE (Ross & Chalmers, 2003) est un environnement de programmation visuelle pour la réduction de dimensions. Il n'y a pas beaucoup de méthodes d'analyse exploratoire de données. Il est difficile de l'étendre parce que HIVE est très spécifique.

YALE (Mierswa et al., 2006) est un environnement gratuit libre d'expérimentation pour l'extraction de connaissances. Il intègre plusieurs méthodes de visualisation et d'apprentissage à l'environnement. YALE ré-utilise la bibliothèque WEKA. Il est souple et extensible. Cependant YALE n'est pas convivial pour la programmation visuelle.

TANAGRA (Rakotomalala, 2005) est une plate-forme gratuite d'expérimentation pour la fouille de données. Il implémente en Delphi des méthodes de fouilles de données issues du domaine de la statistique exploratoire, de l'analyse de données, de l'apprentissage

automatique, mais il ne comporte pas beaucoup de méthodes de visualisation et de techniques interactives.

Orange (Demsar et al., 2004) est une plate-forme d'expérimentation pour la fouille de données. Il implémente en C++/Qt des méthodes d'apprentissage automatique et visualisation.

GeoVista Studio (Takatsuka & Gahegan, 2002) est un environnement de programmation visuelle pour l'analyse en géospatiale. Il est gratuit, libre, souple, extensible et implémente en Java des algorithmes d'apprentissage et des méthodes de visualisation et techniques interactives. GeoVista Studio n'est pas encore convivial pour l'utilisateur.

Notre environnement de programmation visuelle V4Miner permet à l'utilisateur d'extraire de manière intuitive et interactive des connaissances en utilisant simultanément un ensemble de méthodes de visualisation et d'apprentissage automatique. Il est libre, gratuit, souple, général et extensible en Java.

3 Environnement de programmation visuelle (V4Miner)

La conception de l'environnement graphique interactif de programmation visuelle V4Miner décrit figure 1 vise à réduire la complexité et à impliquer plus significativement l'utilisateur dans le processus d'extraction de connaissances.

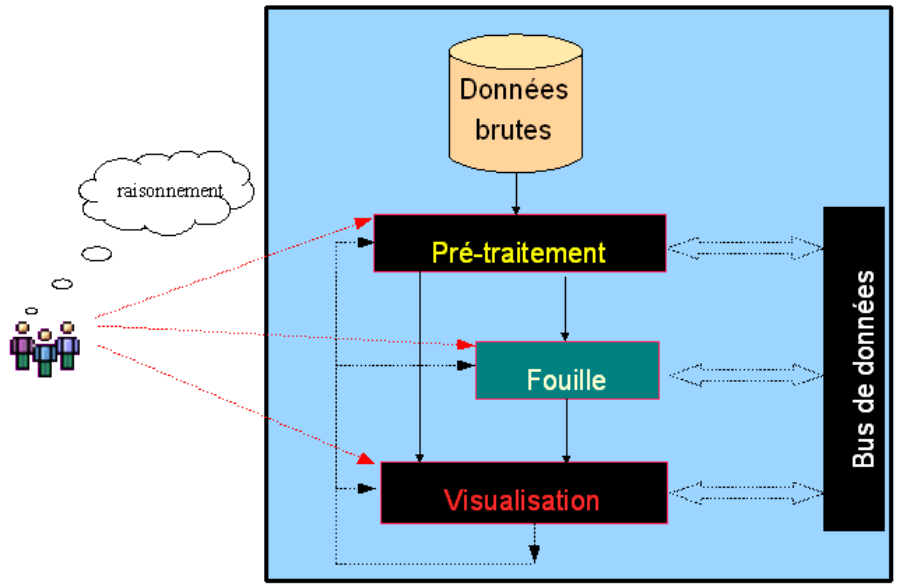


FIG. 1 – *Conception de V4Miner pour l'extraction de connaissances*

Nous proposons de rassembler un ensemble de méthodes existantes de visualisation et d'apprentissage automatique dans une plate-forme unique permettant de donner à l'utilisateur l'opportunité d'intervenir de différentes manières dans le processus de fouille.

V4Miner pour la fouille de données

L'environnement graphique interactif V4Miner est basé sur le concept de composant natif de JavaBeans standard et un Beanbuilder JBeanStudio (Takatsuka, 2002-2007). On peut réutiliser un ensemble de méthodes existantes de visualisation et d'apprentissage automatique à moindre coût. Il est souple, général et extensible. Dans la plate-forme V4Miner, une méthode est encapsulée par un JavaBean et le bus de données est utilisé pour la communication entre les méthodes d'analyse. Aucune connaissance préalable de la programmation n'est requise. L'utilisateur effectue une tâche de fouille en créant lui-même un flot visuel de données. Il a la possibilité de choisir les méthodes à utiliser pour sa tâche et relier les entrées et sorties des méthodes. Tout fonctionne de manière graphique, intuitive et interactive (figure 2). L'utilisateur peut contrôler totalement le processus d'extraction de connaissances. Il peut lancer ensuite de manière complète ou partielle l'exécution des méthodes du processus. La rétroaction est prise en compte, c'est-à-dire qu'il est possible de retourner en arrière à n'importe quelle étape et que l'on peut paramétrer visuellement les méthodes en considérant des résultats intermédiaires pour améliorer la qualité des résultats. Le système peut générer un programme exécutable de Java à partir du flot visuel de données. La plate-forme V4Miner réduit donc la complexité et le coût d'une tâche de fouille de données. La compréhensibilité des résultats peut-être améliorée parce que l'utilisateur participe activement au processus d'extraction de connaissances et il peut utiliser un ensemble de méthodes de visualisation pour interpréter les résultats.

Nous avons intégré dans l'environnement V4Miner plusieurs méthodes de visualisation, des algorithmes d'apprentissage automatique et des méthodes statistiques. Nous avons ré-utilisé des outils existants de la bibliothèque Infovis Toolkit (Fekete, 2004) et la bibliothèque Weka (Witten & Frank, 2000).

L'utilisateur peut exprimer une tâche de fouille de données en utilisant simultanément :

- des méthodes de visualisation comme les coordonnées parallèles (Inselberg, 1985), les matrice de scatterplot 2D, 3D (Carr et al, 1983), le fisheye, l'excentric labeling (Fekete & Plaisant, 1999), la treemap (Johnson, 1992), l'icicle-tree, le node-link-tree, le node-link-graph ou la matrix-graph (Henry & Fekete, 2006),
- des méthodes de réduction et de sélection de dimensions : l'analyse en composantes principales PCA (Pearson, 1901), les méthodes de noyaux PCA (Schölkopf et al., 1998), l'analyse factorielle discriminante FDA (Fisher, 1936), les méthodes de noyaux FDA (Mika et al., 1999), l'analyse factorielle des correspondances AFC (Greenacre, 1984), le gain informationnel ou la mesure du Khi-2,
- des méthodes d'apprentissage automatique à l'aide de séparateurs à vaste marge SVM (Vapnik, 1995) : SMO (Platt, 1999), LS-SVM (Suykens & Vandewalle, 1999), LSVM (Mangasarian & Musicant, 2001), PSVM (Fung & Mangasarian, 2001), Newton-SVM (Mangasarian, 2001), SVM-1 (Fung & Mangasarian, 2002), Reduced SVM (Lee & Mangasarian, 2000) et d'autres méthodes de noyaux (Cristianini & Shawe-Taylor, 2000),
- des arbres de décision : C4.5 (Quinlan, 1993), CART (Breiman et al., 1984) et les forêts aléatoires RF (Breiman, 2001),
- des algorithmes de clustering : les k moyennes KM (McQueen, 1967), EM (Dempster, 1977) et OPTICS (Ankerst et al., 1999),
- des méthodes de régression : la régression logistique LR (Bliss, 1934) et l'arbre de régression logistique LMT (Landwehr et al., 2003),
- des méthodes d'ensemble : le Bagging (Breiman, 1996) et le Boosting (Freund & Schapire, 1996)

- d'autres méthodes : les plus proches voisins kNN (Fix & Hodges, 1952), le Naive Bayes (Good, 1965), Apriori (Agrawal et al., 1993), la visualisation de la marge (Frank & Hall, 2003), les courbes ROC ou les réseaux de neurones.

Bien entendu il serait aussi intéressant d'étudier les possibilités de mixage des différentes approches de visualisation et d'apprentissage en espérant améliorer ainsi le processus et ou les résultats. L'utilsateur peut bénéficier des avantages de chaque méthode pour améliorer les performances de l'utilisation d'une seule méthode. Par exemple, les méthodes automatiques peuvent traiter de grands ensembles de données avec de bons taux de précision et des méthodes de visualisation sont simples et compréhensibles. L'approche coopérative permet de bénéficier des avantages de chaque méthode pour obtenir des résultats compréhensibles et de qualité.

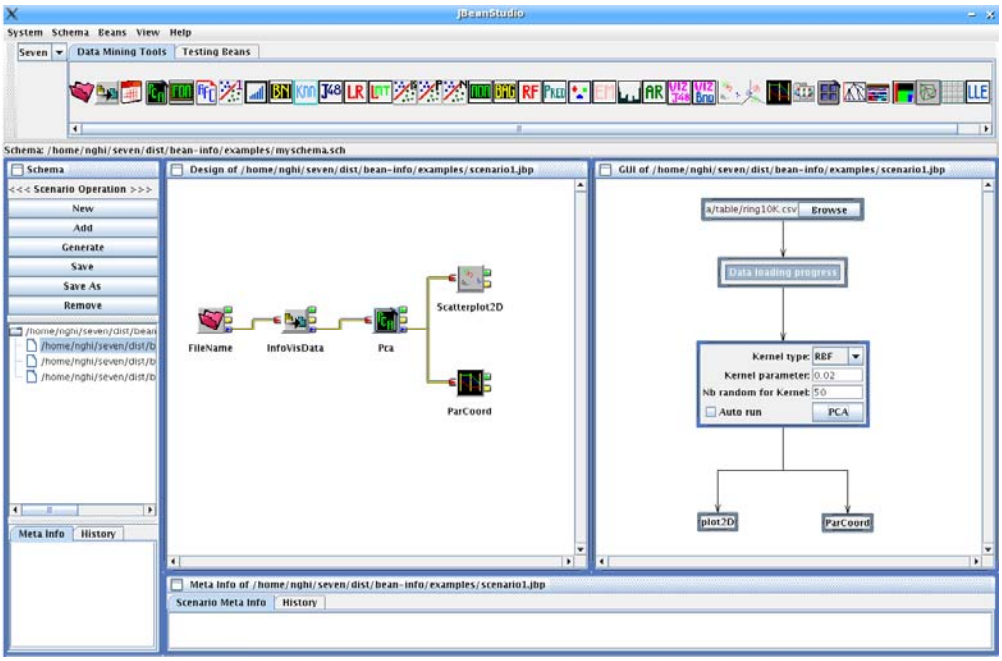


FIG. 2 – *Tâche de réduction de dimensions des données avec l'algorithme PCA et une fonction de noyau RBF à l'aide de l'environnement V4Miner*

Par exemple, nous considérons une tâche de réduction de dimensions à l'aide de l'environnement V4Miner. L'utilisateur crée facilement lui-même un flot visuel avec les composants du chargement de données FileName, InfoVisData, de l'algorithme d'analyse en composantes principales Pca et les visualisation des ensembles de données sur les projections avec Scatterplot2D et ParCoord comme sur l'exemple de la figure 2. L'utilisateur peut paramétrer les méthodes. S'il souhaite lancer de manière complète le processus il doit alors cocher "Auto run". L'utilisateur peut interagir avec les Scatterplot-2D pour l'analyse exploratoire des données.

4 Fouille de données bio-médicales à l'aide de V4Miner

Nous étudions la classification des données bio-médicales (Jinyan & Huiqing, 2002) décrites dans le tableau 1.

	Classes	Individus	Dimensions	Protocole de test
Colon Tumor	2	62	2 000	leave-1-out
Ovarian Cancer	2	253	15 154	10-fold
Lung Cancer	2	181	12 533	32 Trn – 149 Tst
Leukemia	2	72	7 129	38 Trn – 34 Tst

TAB. 1 – *Description des ensembles de données bio-médicales*

Cette tâche est difficile parce qu'on traite souvent un grand nombre de dimensions (des milliers) et un petit nombre d'individus (des centaines). Donc, la première étape est de sélectionner un sous-ensemble de dimensions à l'aide de l'algorithme SVM-1 pour que les coordonnées parallèles, les scatterplot-2D et l'algorithme d'arbre de décision C4.5 puissent traiter efficacement les données dans le sous-ensemble de dimensions obtenu. L'utilisateur l'effectue facilement par l'intermédiaire du flot visuel à l'aide de l'environnement V4Miner comme représenté sur la figure 3.

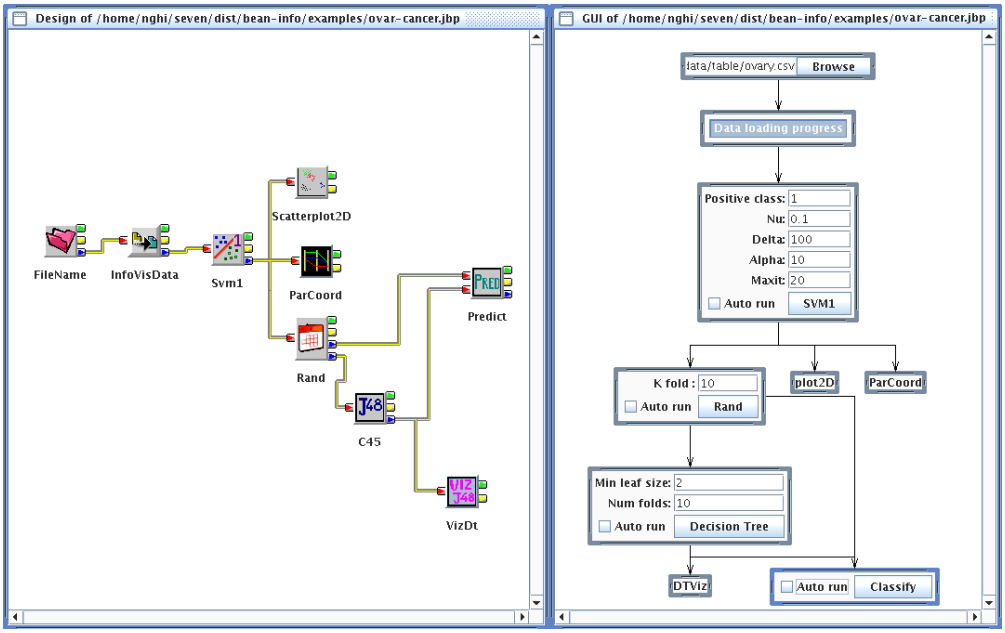


FIG. 3 – *Tâche de la fouille des données bio-midécales avec l'algorithme de SVM-1, C4.5 et les méthodes de visualisation Scatterplot-2D, les coordonnées parallèles*

Pour la classification des données Ovarian Cancer ayant 253 individus en 15154 dimensions et 2 classes, le déroulement du processus est de charger d'abord l'ensemble de données, puis sélectionner un sous-ensemble de dimensions avec l'algorithme SVM-1 et

finalement visualiser l'ensemble de données avec les Scatterplot-2D, les coordonnées parallèles ou classifier avec l'algorithme d'arbre de décision C4.5. L'algorithme SVM-1 ne garde que 9 dimensions importantes et la visualisation est ensuite possible (figure 4 et 5).

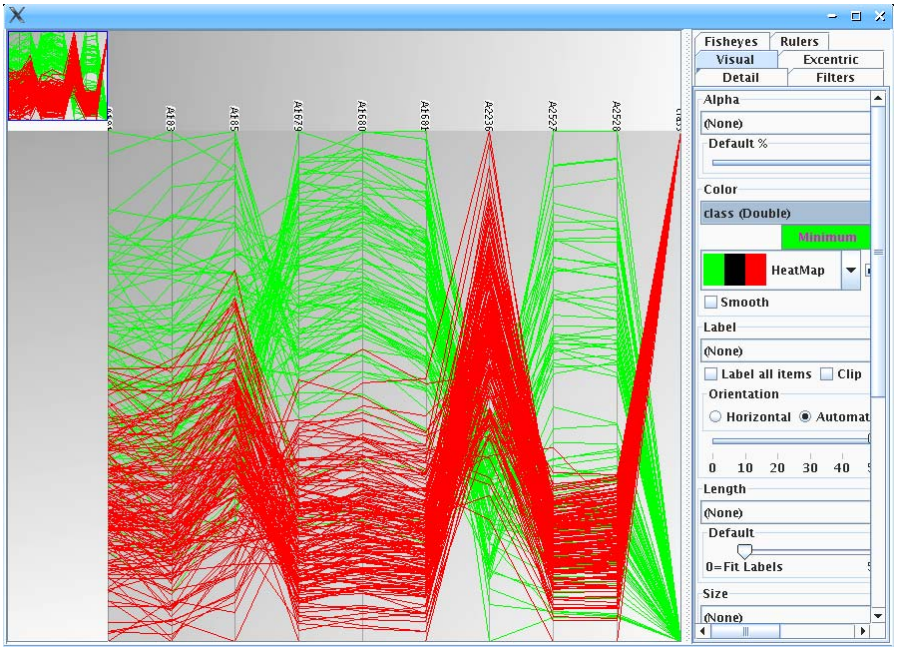


FIG. 4 – *Visualisation des données Ovarian Cancer avec les coordonnées parallèles*

Il est intéressant de remarquer que l'approche coopérative entre la visualisation et l'apprentissage automatique permet d'obtenir des résultats qu'il serait impossible d'obtenir avec un seul outil. Le SVM-1 sélectionne un sous-ensemble de dimensions significatives à partir du très grand nombre de dimensions, puis la visualisation sert à interpréter les résultats obtenus. Les visualisations des coordonnées parallèles et des Scatterplot-2D permettent à l'utilisateur de mettre en évidence les dimensions importantes (par exemple les dimensions 181 et 1679 dans la figure 5). L'algorithme de l'arbre de décision C4.5 peut aussi classifier efficacement ces données dans le sous-ensemble de dimensions. L'arbre résultant contient 7 nœuds et est représenté sur la figure 6.

A partir du petit arbre de décision obtenu, on extrait 4 règles d'induction sous forme si-alors qui sont compréhensibles par tout utilisateur.

1. SI ((A1679 <= 0,43383) & (A181 <= 0,615384)) ALORS classe = Cancer
2. SI ((A1679 <= 0,43383) & (A181 > 0,615384)) ALORS classe = Normal
3. SI ((A1679 > 0,43383) & (A185 <= 0,209486)) ALORS classe = Cancer
4. SI ((A1679 > 0,43383) & (A185 > 0,209486)) ALORS classe = Normal

Nous avons utilisé le même flot visuel pour classifier les autres ensembles de données. Les résultats obtenus après avoir sélectionné les dimensions sont comparés avec ceux obtenus par une classification sur l'ensemble des dimensions par les algorithmes de SVM.

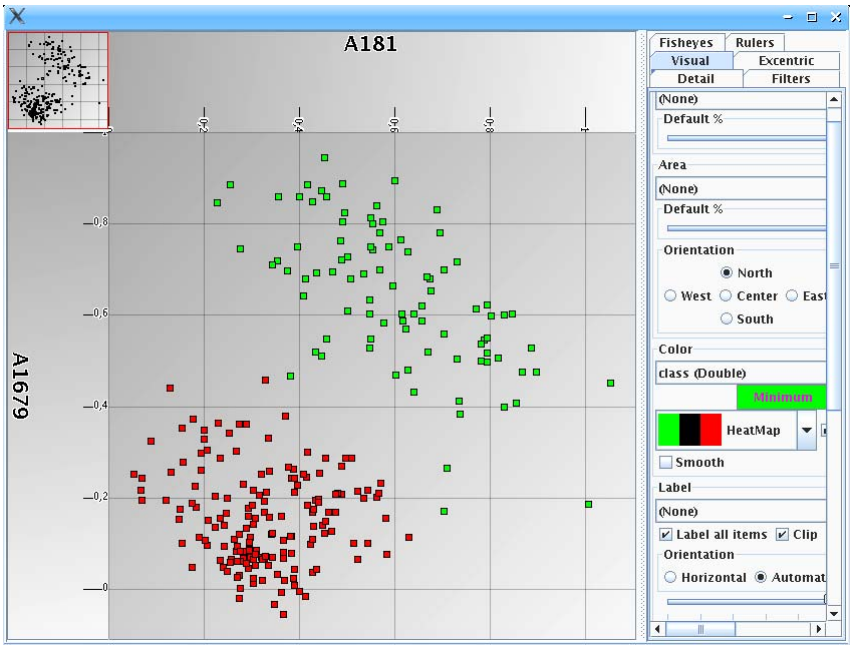


FIG. 5 – *Visualisation des données Ovarian Cancer avec le scatterplot-2D*

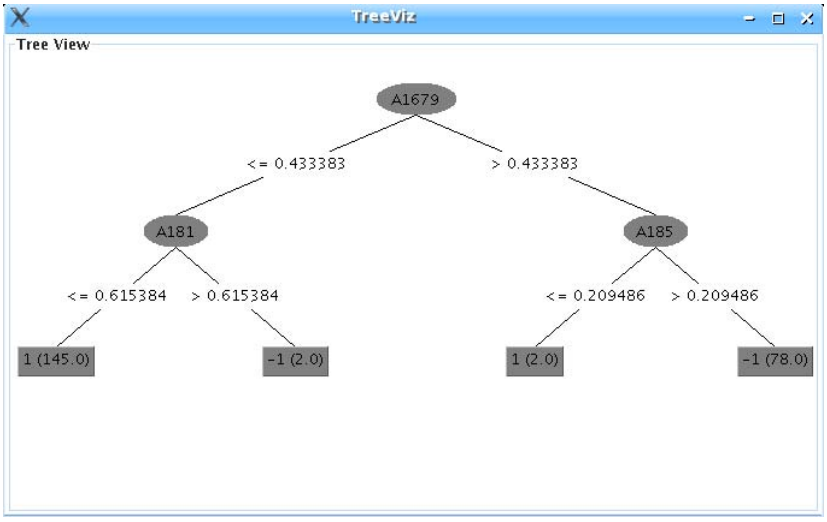


FIG. 6 – *Arbre de décision pour la classification des données Ovarian Cancer*

Les résultats concernant le taux de précision sont donnés dans le tableau 2 (les meilleurs résultats sont en caractères gras). On remarque que pour tous les ensembles de données traités sauf un, les résultats sont meilleurs lorsque l'on utilise un sous-ensemble de dimensions plutôt que l'ensemble complet de dimensions.

	SVM Dim., erreurs	SVM-1 + Vis dim., erreurs	SVM-1 + C4.5 dim., erreurs	SVM-1 + SVM dim., erreurs
Colon Tumor	2000, 6	8, 4	8, 5	**8, 2**
Ovarian Cancer	15154, 0	**9, 0**	**9, 0**	**9, 0**
Lung Cancer	**12533, 2**	6, 5	6, 5	6, 5
Leukemia	7129, 2	**9, 2**	**9, 2**	**9, 2**

TAB. 2 – *Résultats de la classification des données bio-médicales*

Il est intéressant aussi de constater que le nombre de dimensions utilisées est réduit de manière très significative : par exemple sur Colon Tumor, on passe de 2000 à 8 dimensions (soit une diminution d'un facteur 250) mais le taux de précision est amélioré, sur Ovarian Cancer on passe de 15154 à 9 dimensions (soit une diminution d'un facteur 1684) sans perte de précision et sur Leukemia, on passe de 7129 à 9 dimensions (soit une diminution d'un facteur 792) en conservant exactement le même taux de précision. Pour Lung Cancer on passe de 12533 à 6 dimensions (soit une diminution d'un facteur 2089) mais on perd un petit peu d'information (1,34 % de taux de précision) par rapport l'utilisation de l'ensemble complet de dimensions. On ne présente pas dans le tableau 2 les autres facteurs importants qui sont d'une part la réduction de la complexité de la tâche de fouille de données et d'autre part l'amélioration de la compréhensibilité des résultats obtenus.

5 Conclusion et perspectives

Nous avons présenté un nouvel environnement de programmation visuelle (V4Miner) pour l'extraction interactive de connaissances à partir de données. La plate-forme utilise simultanément un ensemble de méthodes de visualisation et d'apprentissage automatique pour la fouille de données de manière intuitive, interactive et rétroactive. L'environnement permet d'impliquer plus significativement l'utilisateur dans le processus d'extraction de connaissances. Aucune connaissance préalable de la programmation n'est requise. Une tâche de fouille est exprimée par un flot visuel de données où une méthode est représentée par un composant graphique, la communication entre les méthodes se base sur un bus de données. L'utilisateur conçoit lui-même un flot visuel de données en choisissant les méthodes qui lui permettent d'atteindre son but. Tout fonctionne de manière graphique, intuitive et interactive. L'utilisateur peut contrôler totalement le processus d'extraction. Il peut lancer de manière complète ou partielle le flot visuel pour la fouille de données. La rétroaction est prise en compte. L'utilisateur a la possibilité de retourner en arrière à n'importe quelle étape et peut paramétrer visuellement les méthodes pour améliorer la qualité des résultats. Le système peut générer un programme exécutable en Java à partir du flot de données. Notre plate-forme V4Miner d'une part réduit donc la complexité et le coût d'une tâche de fouille de données et d'autre part améliore la qualité et la compréhensibilité des résultats obtenus. Nous avons obtenu des résultats expérimentaux intéressants en fouille de données bio-médicales.

Il serait intéressant de continuer à intégrer d'autres méthodes de fouille de données des librairies Weka, Yale ou des algorithmes d'analyse de données symboliques de SODAS (Diday, 2000) dans l'environnement de programmation visuelle pour la fouille de données.

Références

Agrawal, R., T. Imielinski, and A. Swami (1993). Mining Association Rules between Sets of Items in Large Databases. in *SIGMOD* Conference on Management of Data, pp. 207-216.

Ankerst, M., M-M. Breunig, H-P. Kriegel, and J. Sander (1999). OPTICS: Ordering Points To Identify the Clustering Structure. in *SIGMOD* Conference on Management of Data, pp. 49-60.

Bliss, C-I. (1934). The Method of Probits. in *Science*, 79(2037): 38-39.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees*. Chapman & Hall, New York.

Breiman, L. (1996). Bagging Predictors. in *Machine Learning*, 24(2):123-140.

Breiman, L. (2001). Random Forests. in *Machine Learning*, 45(1):5-32.

Cleveland, W-S. (1993). *Visualizing Data*. Hobart Press, Summit NJ.

Cristianini, N. and J. Shawe-Taylor (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Demsar J., B. Zupan, and G. Leban (2004). *Orange: From Experimental Machine Learning to Interactive Data Mining*. White Paper, Faculty of Computer and Information Science, University of Ljubljana.

Dempster, A., N. Laird, and D. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B, 39(1):1–38.

Diday, E. (2000). Symbolic Data Analysis and the SODAS project : purpose, history, perspective. in *Analysis of Symbolic Data Exploratory methods for extracting statistical information from complex data*, Springer-Verlag, pp. 1-22.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37-54.

Fayyad, U., G. Piatetsky-Shapiro, and R. Uthurusamy (2004). Summary from the KDD-03 Panel – Data Mining: The Next 10 Years. in *SIGKDD Explorations*, 5(2):191-196.

Fayyad, U., G. Grinstein, and A. Wierse (2001). *Information Visualization in Data Mining and Knowledge Discovery*. Morgan Kaufmann Publishers.

Fekete, J-D. (2004). The InfoVis Toolkit. in *IEEE InfoVis*, 167-174.

Fekete, J-D. and C. Plaisant (1999). Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization. Proceeding of the *SIGCHI* conference on Human factors in computing systems, pp. 512-519.

Fisher, R.A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics* (7):179-188.

Fix, E. and J. Hodges (1952). Discriminatoiry Analysis: Small Sample Performance. Technical Report 21-49-004, USAF School of Aviation Medicine, Randolph Field, USA, 1952.

Frank, E. and M. Hall (2003). Visualizing Class Probability Estimators. in *PKDD*, Lecture Notes in Computer Science 2838, Springer, pp. 168-179.

Freund, Y. and R. Schapire (1996). Game Theory, On-line Prediction and Boosting. in Proceedings of the Ninth *Annual Conference on Computational Learning Theory*, pp. 325–332.

Fung, G. and O. Mangasarian (2001). Proximal Support Vector Machine Classifiers. Proceedings of the 7th *ACM SIGKDD*, International Conference on Knowledge Discovery and Data Mining, San Francisco, pp. 77-86.

Fung, G., and O. Mangasarian (2002). A Feature Selection Newton Method for Support Vector Machine Classification, Data Mining Institute Technical Report 02-03, Computer Sciences Department, University of Wisconsin, Madison, USA.

Good, I. (1965). *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. MIT Press.

Greenacre, M-J. (1984). *Theory and Applications of Correspondence Analysis*. Academic Press, London.

Henry, N. and J-D. Fekete (2006). MatrixExplorer: Un système pour l'analyse exploratoire de réseaux sociaux. Proceedings of IHM, *ACM Press*, Canada, pp. 67-74.

Inselberg, A. (1985). The Plane with Parallel Coordinates. in *Special Issue on the Computational Geometry of The Visual Computer*, 1(2):69-97.

Jinyan L., and L. Huiqing (2002). Kent Ridge Bio-medical Data Set Repository, http://sdmc-.lit.org.sg/GEDatasets.

Johnson, B. (1992). TreeViz: Treemap visualization of hierarchically structured information Demonstration summary appears. Proceeding of *ACM-CHI'92*, pp.369-370.

Keim, D. (2002). Information Visualization and Visual Data Mining. in *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1-8.

Landwehr, N., M. Hall, and E. Frank (2003). Logistic Model Trees. in *ECML*, Lecture Notes in Computer Science 2837, Springer, pp. 241-252.

Lee Y-J, and O. Mangasarian (2000). RSVM: Reduced Support Vector Machines. Data Mining Institute Technical Report 00-07, Computer Sciences Department, University of Wisconsin, Madison, USA, 2000.

MacQueen J. (1967). Some methods for classification and analysis of multivariate observations, Proceedings of 5th *Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, (1):281-297.

Mangasarian, O. (2001). A Finite Newton Method for Classification Problems. Data Mining Institute Technical Report 01-11, Computer Sciences Department, University of Wisconsin, Madison, USA, 2001.

Mangasarian, O. and D. Musicant (2001). Lagrangian Support Vector Machines. in *Journal of Machine Learning Research* Vol. 1, pp. 161-177.

Mierswa, I., M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler (2006). YALE: Rapid Prototyping for Complex Data Mining Tasks. Proceedings of the 12th ACM *SIGKDD*, International Conference on Knowledge Discovery and Data Mining, pp. 935-940.

Mika, S., G. Rätsch, J. Weston, B. Schölkopf, and K-R. Müller (1999). Fisher Discriminant Analysis with Kernels. in *Neural Networks for Signal Processing* Vol. 9, pp. 41-48.

Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine* 2(6):559–572.

Platt, J. (1999). Fast Training of Support Vector Machines Using Sequential Minimal Optimization. in *Advances in Kernel Methods -- Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola Eds., pp. 185-208.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Rakotomalala, R. (2005). TANAGRA : un logiciel gratuit pour l'enseignement et la recherche. in *RNTI-E-3* Vol. 2, pp. 697-702.

Ross G., and M. Chalmers (2003). A visual workspace for constructing hybrid multidimensional scaling algorithms and coordinating multiple views. in *IEEE InfoVis*, 247-257.

Schölkopf, B., A. Smola, and K-R. Müller (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. in *Neural Computation* Vol. 10, pp. 1299-1319.

Suykens, J. and J. Vandewalle (1999). Least Squares Support Vector Machines Classifiers. in *Neural Processing Letters*, 9(3):293-300.

Takatsuka, M. (2002-2007). JBeanStudio: A Component-Oriented Visual Software Authoring System for a Problem Solving Environment. http://jbeanstudio.sourceforge.net.

Takatsuka, M. and M. Gahegan (2002). GeoVISTA Studio: A Codeless Visual Programming Environment for Geoscientific Data Analysis and Visualization. in *Computers & Geosciences*, 28(10):1131-1144.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

Witten I-H., and E. Frank (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco.

Summary

The visual data flow environment called V4Miner simultaneously uses a set of interactive techniques, visualization methods and machine learning algorithms to discover knowledge in an intuitive, interactive and retroactive way. A data mining task is easily expressed by a graphical visual data flow where a method is represented by a graphic JavaBeans component, the communication between methods is based on a data bus. The user can create and fully control the discovery process without any programming. The visual data flow makes it possible on the one hand to reduce the complexity of an analysis task and on the other hand to improve quality and users' comprehensibility of the obtained results.