Dagmap : exploration interactive de relations d'héritage

Pierre-Yves Koenig, Guy Melançon

DagMap : exploration interactive de relations d'héritage

Pierre-Yves Koenig* *** — **Guy Melançon**** ***

** Laboratoire d'Informatique de Robotique et de Microélectronique*
UMR CNRS 5506
161 rue Ada, F-34392 Montpellier Cedex 5

Pierre-Yves.Koenig@lirmm.fr

*** Laboratoire Bordelais de Recherche en Informatique - UMR CNRS 5800*
351, cours de la Libération F-33405 Talence Cedex

Guy.Melancon@labri.fr

**** Institut National de Recherche en Informatique et en Automatique*
Bordeaux

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) fournissent 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 détail de la visualisation est obtenue en reprenant les idées originales de Furnas et van Ham & van Wijk, étendues aux DagMap.*

ABSTRACT. *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.*

MOTS-CLÉS : *Treemap, DAG, hiérarchies.*

KEYWORDS: *Treemap, DAG, Hierarchies.*

DOI:10.3166/RIA.22.353-368

RSTI - RIA – 22/2008. Visualisation et extraction des connaissances, pages 353 à 368

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 "mères".

Les relations d'héritage entre entités (objets, 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 conséquent, le graphe résultant est un graphe orienté acyclique (DAG). Le dessin et la visualisation interactive de DAGs est un défi en soi. 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 al.*, 2001)). Ces algorithmes dessinent le plus souvent les sommets sur différents niveaux (cf figure 1). Les entités les plus générales – sommets *sources* – 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 nœuds-liens sont le plus souvent mesurées en fonction de leur capacité à éviter le croisement d'arête (Gutwenger *et al.*, 2004).

Bien que les diagrammes nœuds-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 tailles différentes. La lisibilité des arêtes impose aussi de garder un éloignement suffisant entre les niveaux afin d'éviter des arêtes trop à l'horizontale. Même avec des DAGs de taille moyenne, éviter le chevauchement de sommets voisins se fait au détriment de la comparaison de leur tailles.

Il est à noter que c'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 (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 l'aide du dessin mais aussi d'une interaction adaptée.

La représentation du DAG par le DagMap requiert dans un premier temps de le déployer en arbre. Cela nécessite de concevoir des interactions spécifiques afin de retrouver 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.

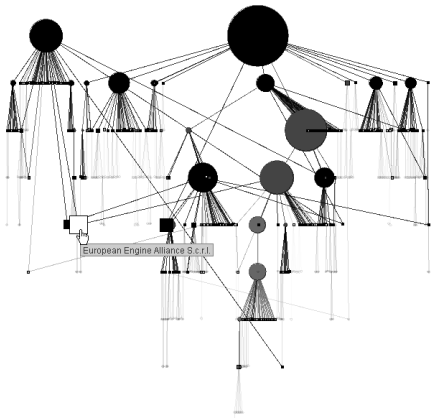


Figure 1. *Dessin nœud-lien classique de DAG, avec un attribut mis en évidence par la taille et la couleur des sommets*

2. 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 nœuds-liens des arbres (voir (Battista *et al.*, 1998; Kaufmann *et al.*, 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 ses applications commerciales, apportent une preuve de son utilité 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 systèmes de visualisation de données hiérarchiques (arborescentes dans le cas de (Vliegen *et al.*, 2006)).

Figure 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 blanches 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 montrent les tailles (similaires) 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 et 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 société 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 à la tête de la hiérarchie ? Y a t-il un niveau de la hiérarchie dans lequel se concentre le contrôle ?

– Les sociétés mères d'une filiale donnée ont-elles une stratégie de développement similaire (sont-elles présentes 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 attestent 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 (*cf.* section 4, 5).

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 al.*, 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 (*cf.* section 4.1). Différentes interactions sur le DagMap vont être décrites, basées sur le mécanisme du DOA (*cf.* section 4.1) ou sur la structure du DAG (*cf.* section 5). Le DagMap et les interactions ont été implémentés en utilisant la boîte à outils *prefuse* (Heer *et al.*, 2005). Les interactions ont été réalisées et testées 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.

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

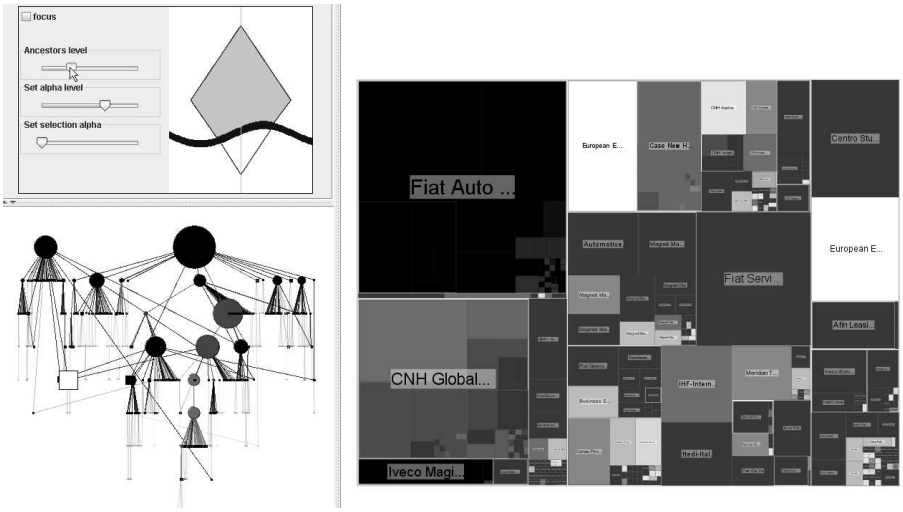


Figure 3. *L'application DagMap. La taille relative des fenêtres de droite et de gauche peut être ajustée. Les vues des différentes fenêtres sont synchronisées. Lorsqu'on sélectionne un sommet, celui-ci est animé et voit sa saturation varier pour faciliter sa localisation. Les attributs des sommets peuvent 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 (cf. section 4)*

3. 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 ait un seul père (*cf.* figure 4). Il est à noter qu'on n'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 présentée 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 réparti dans la hiérarchie. Quand on sélectionne un élément – une cellule – du DagMap, il est mis en évidence ainsi que toutes les cellules du DagMap correspondantes.

Dans notre exemple (voir figure 2), une filiale révèle sa présence dans différentes régions du DagMap quand le contrôle est distribué par différentes sociétés mères distinctes – ceci est particulièrement utile quand les filiales voisines appartiennent à différentes régions géographiques. Le code couleur utilisé dans la figure 2 a été établi par

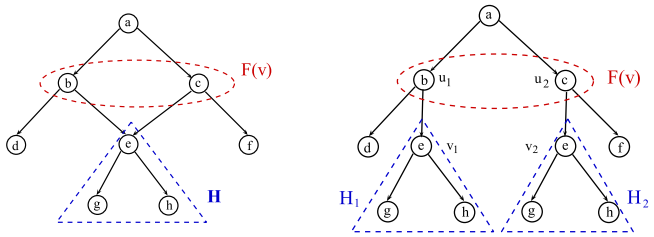


Figure 4. *Un arbre étiqueté est obtenu du DAG par duplication des sommets (étiquetés) avec des ancêtres multiples*

les géographes (du plus foncé au plus pâle : l'Europe, paradis fiscaux, l'Amérique du nord, l'Asie, l'Amérique du sud et l'Afrique)[2].

La taille de l'arbre obtenu du DAG peut potentiellement être 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 al.*, 2002), que la plupart des techniques traitant des DAGs souffrent de cette limitation (voir aussi (Melançon *et al.*, 2000; Gutwenger *et al.*, 2004)). Cette limitation est contournée par le fait que l'arbre n'est calculé qu'à partir d'une partie du DAG (*cf.* section 4.1), permettant de garder la taille de l'arbre sous contrôle.

4. 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, *cf.* figure 3). La forme de cet interacteur reflète la forme générale d'un DAG. La ligne incurvée couvrant l'interacteur peut être déplacée de haut en bas, faisant varier la coupe (niveau de détail) auquel le DagMap sera construit (voir figure 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

2. Une version couleur des figures de cet article est disponible à l'url : http ://www.lirmm.fr/∼koenig/dagmap/egc08/

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 et l'ensemble des sommets au dessus de celle-ci, jusqu'aux sociétés de plus haut niveau (figure 5 seconde image). La distance ici n'a pas été mesurée uniquement en termes 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 exercé 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.

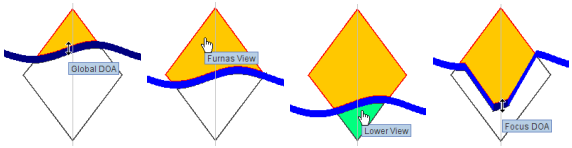


Figure 5. *Un interacteur en forme de losange est utilisé pour induire une coupe du DAG, en le déplaçant de haut en bas (image (a) à (c)). (La forme de l'interacteur reflète la forme général d'un DAG.) Le DagMap est alors calculé en partant des éléments de la coupe jusqu'aux éléments au niveau le plus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

(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 sommets en dessous de celle-ci (figure 5 troisième image)).

Maintenant, supposons que l'utilisateur porte une attention particulière sur un élément et en fasse 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 (figure 5 droite). L'application se met en mode "focus" en cochant un bouton de contrôle dans l'interface (voir figure 3), l'utilisateur peut jouer avec la ligne incurvée et définir combien de détails sont requis autour de l'élément focus.

4.1. *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 al.*, 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 implémentée requiert une hiérarchisation des données. Dans la plupart 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 blocs d'instructions.

La situation étudiée par (van Wijk *et al.*, 2004), de même que par (Quigley *et al.*, 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-arbre dont il est la racine.

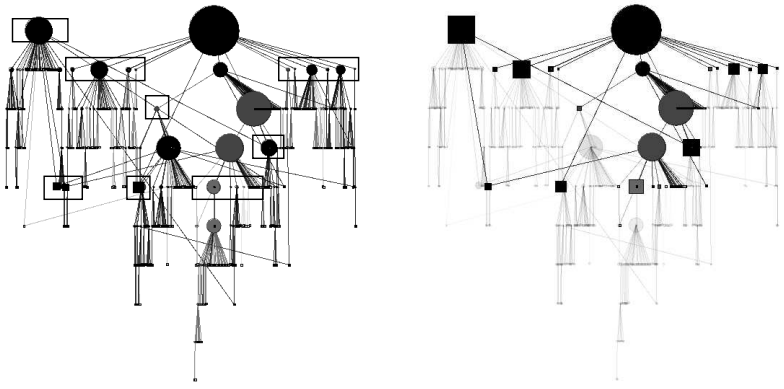
Nous allons montrer comment ces mécanismes développés pour les arbres fonctionnent aussi lorsque la hiérarchie est un DAG. Un exemple de coupe est illustrée dans la figure 6. Revenons sur le cas des arbres.

La structure d'arbre joue un rôle fondamental dans le processus de suppression dans le but d'obtenir une vue abstraite des données, et repose sur le calcul de statistiques (indices) pour les sommets de l'arbre. Dans le cas de Furnas, l'indice associé à un sommet v, est appelé le degré *à priori* d'intérêt (*API*) et correspond a son niveau $\ell(v)$ dans l'arbre (exprimé par des valeurs négatives 0, -1, -2, ...). Ce degré reflète 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 [3] 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 utilisée pour la visualisation *coïncide* avec la structure de données à visualiser.

Cette situation est différente dans le cas de (van Wijk *et al.*, 2004) qui produisent une vue abstraite pour un graphe basé sur une structure d'arbre distinct du graphe visualisé (à l'exception des sommets feuilles qui correspondent aux sommets du graphe d'origine). Ils utilisent une valeur DOA, avec $DOA \in [0, 1]$, afin de selectionner les clusters formant ainsi une abstraction du graphe. 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 graphe constituée des clusters les plus hauts (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 :

$$d_{C'} \leq d_{root} \cdot DOA < d_C \qquad [1]$$

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



(a) (b)

Figure 6. *Les rectangles (image (a)) entourent les sommets identifiés comme membres de la coupe (ici avec un $DOA = 0.3$). L'image (b) montre le DAG après avoir pris la partie* supérieure *du DAG "à la Furnas" pour construire le DagMap*

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' sélectionnés sont ceux se trouvant immédiatement au-dessous de cette valeur ($d_{C'} \leq d_{root} \cdot DOA$), tandis que leurs pères 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 (i.e. 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 [1] peut être directement appliquée aux DAGs pour déterminer une coupe à un certain degré d'abstraction (DOA). Étant donné 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 (figure 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 montrent comment les inéquations [1] et [2] peuvent être adaptées pour tenir compte d'un sommet *focus* dans l'arbre. Supposons que l'utilisateur a sélectionné un élément f, qui peut être un élément visible sélectionné directe-

ment à l'écran ou à travers une requête sur les données. L'inégalité peut être adaptée comme suit :

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

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 al.*, 2004) pour une discussion 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.

4.2. *Variation sur les statistiques des sommets*

Différentes 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 (égal 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.

5. 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, il peut définir une "bande" rassemblant tous les éléments à une certaine distance au-dessus de la coupe. Ce faisant, il visualise comment les éléments dépendent de leurs ancêtres.

En faisant varier la hauteur de la bande, l'utilisateur obtient des informations sur les attributs des cellules d'ancêtres 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, comme le montre l'exemple de la figure 8. Dans d'autres situations, les géographes ont pu constater 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 2, 3 et 8.

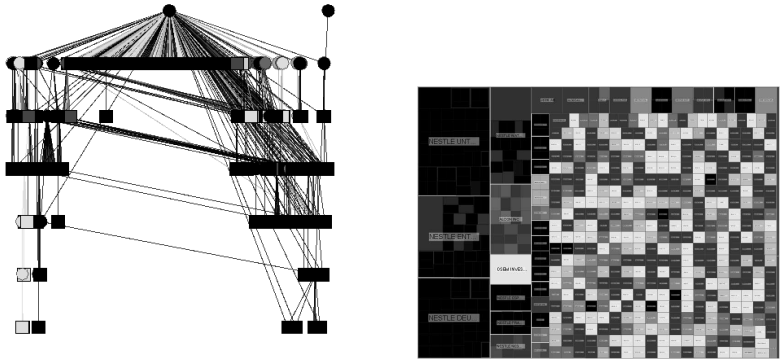


Figure 7. *Le DAG décrit les liens entre Nestlé et ses filiales. Même si le DAG n'est pas dense, le diagramme nœud-lien ne permet pas de savoir comment les filiales sont distribuées dans les différentes régions du monde, ou comment les sociétés de haut niveau exercent leur contrôle sur les filiales de bas niveau. Le DagMap facilite la détection de motifs (formes particulières) formés par les attributs de l'ensemble 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*

6. Conclusion et travaux futurs

Nous avons présenté le DagMap généralisant les TreeMaps aux graphes orientés acycliques, vu en tant que structure d'héritage générale. 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. En effet, l'ensemble de données est intrinsèquement codé en DAG.

De plus, les besoins des utilisateurs ont requis le développement d'une technique combinant astucieusement la visualisation et la comparaison, au sein d'une même visualisation, des attributs des données et de la structure hiérarchisée.

Nous avons également appliqué le DagMap pour visualiser les dépendances entre les modules dans la distribution Ubuntu (Linux). Les modules sont structurés de manière hiérarchique, avec les modules de base et essentiels au bas de la hiérarchie (comme les bibliothèques C) – voir la figure 10. Cet exemple diffère de notre cas d'étude principal parce que le DAG d'Ubuntu est plus dense, rendant une visualisation nœud-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 packages incontournables et ceux qui sont facultatifs, aidant potentiellement à faire un choix entre les environnements de bureau possibles.

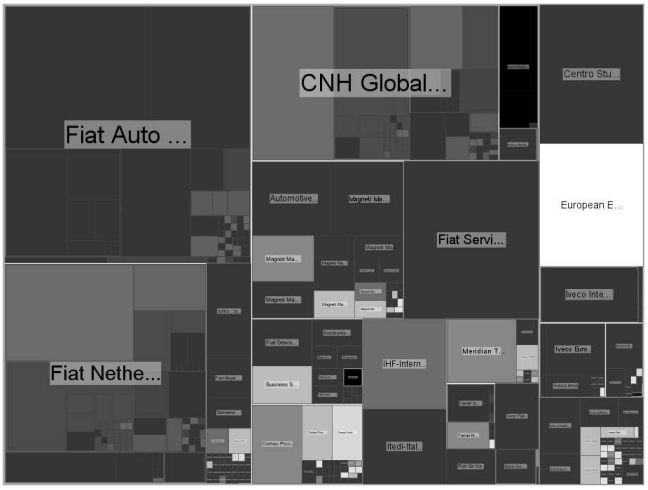


Figure 8. *Le DagMap montré ici est obtenu à partir de celui de la figure 2 auquel on a rajouté une bande (figure 9). L'emboîtement des filiales devient clair quand on peut visualiser les sociétés mères se trouvant au dessus d'elles. Le code couleur indique que les sociétés sont contrôlées par des sociétés mères suspectées de jouer le rôle de paradis fiscaux (Fiat Netherland, en bas à gauche ; CNH Global, en haut au milieu)*

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 nœuds-liens pour des graphes (voir (Fekete *et al.*, 2003)).

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 al.*, 2004), a été conçu et employé 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 aient 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 d'effectuer une expérimentation contrôlée pour établir objectivement les avantages et inconvénients de notre méthode. La représentation sous forme de diagramme nœud-lien, de dagmap et la combinaison des deux représentations seront comparées. Différentes questions seront soumises à un panel de sujets (géographes et non géographes) associées à une des représentations possibles. Le temps, les réponses, le nombre de clics et de "drag

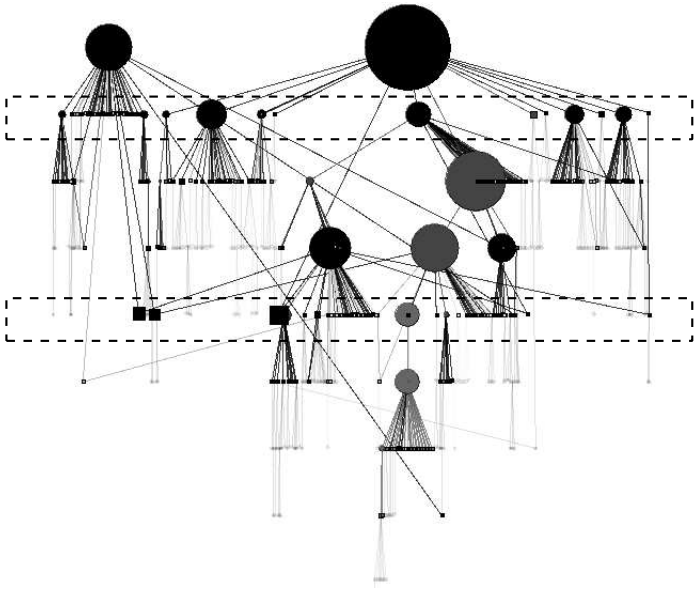


Figure 9. *Après sélection d'un coupe, l'utilisateur peut définir une bande située au dessus de la coupe et visualisée sur le DagMap en tant que cellules colorées superposées. La variation de la hauteur de cette bande fournit un retour visuel direct sur le DagMap*

and drop" seront comptabilisés afin de déterminer la méthode la plus efficace suivant les questions posées.

Une prochaine étape sera l'étude de situations où des données sont groupées avec des recouvrements qui s'imbriquent et se chevauchent. En effet, une telle hiérarchie 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 mot-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.

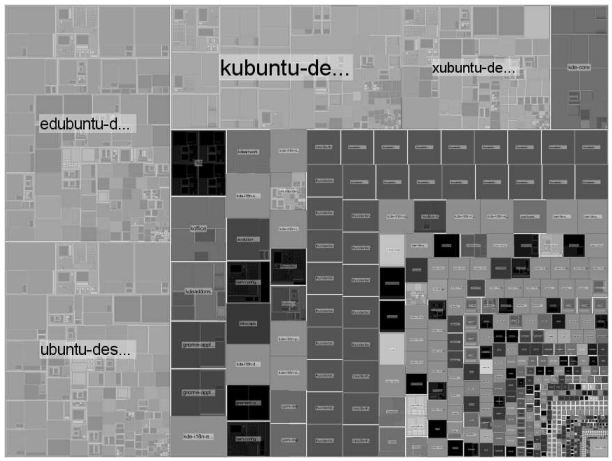


Figure 10. *Après sélection d'une coupe, l'utilisateur peut définir une bande se trouvant au-dessus de la coupe et visualisée sur le DagMap en tant que cellules colorées superposées*

7. Bibliographie

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

Bohan C., Gautier B., Rozenblat C., Auber D., Koenig P.-Y., « Cities Networks through Multinational Firms Networks : A Multi-Level Graph Approach », *15th European Colloquium on Theoretical and Quantitative Geography*, Zurich, Switzerland, 2007.

Bruls M., Huizing K., van Wijk J. J., « Squarified Treemaps », *in* W. d. Leeuw, R. v. Liere (eds), *Joint Eurographics and IEEE TCVG Symposium on Visualization (Data Visualization '00)*, Springer-Verlag, Amsterdam, p. 33-43, 2000.

Chintalapani G., Plaisant C., Shneiderman B., « Extending the Utility of Treemaps with Flexible Hierarchy », *Eighth International Conference on Information Visualisation (IV'04)*, IEEE Computer Society, p. 335-344, 2004.

Fekete J., Wang D., Dang N., Aris A., Plaisant C., « Overlaying Graph Links on Treemaps », *IEEE Information Visualization 2003 Symposium Poster Compendium*, IEEE Press, p. 82-83, 2003.

Furnas G. W., « Generalized Fisheye Views », *Human Factors in Computing Systems CHI '86*, ACM Press, p. 16-23, 1986.

Gautier B., « Integration of the Moroccan cities by the multinational firms of agro-alimentary sector », *IGU Commission of "Monitoring cities of tomorrow"*, Sun Yat Sen University Press, Guangzhou, China, p. 193-209, 2007.

Gutwenger C., Mutzel. P., « An experimental study of crossing minimization heuristics », *in* B. Liotta (ed.), *Graph Drawing 2003*, vol. 2912 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 13-24, 2004.

Heer J., Card S. K., Landay J. A., « prefuse : a toolkit for interactive information visualization », *SIGCHI conference on Human factors in computing systems*, ACM Press, Portland, Oregon, USA, p. 421-430, 2005.

Herman I., Marshall M. S., Melançon G., Duke D. J., Delest M., Domenger J.-P., « Skeletal Images as Visual Cues in Graph Visualization », *in* E. Gröller, H. Löffelmann, W. Ribarsky (eds), *Data Visualization '99, Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, SpringerVerlag, p. 13-22, 1999.

Herman I., Melançon G., Delest M., « Tree Visualisation and Navigation Clues for Information Visualisation », *Computer Graphics Forum*, vol. 17, n° 2, p. 153-165, 1998.

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

Koenig P.-Y., Melançon G., Bohan C., Gautier B., « Combining DagMaps and Sugiyama Layout for the Navigation of Hierarchical Data », *IV '07 : Proceedings of the 11th International Conference Information Visualization*, IEEE Computer Society, p. 447-452, 2007.

Melançon G., Herman I., « DAG Drawing from an Information Visualization Perspective », *in* W. d. Leeuw, R. v. Liere (eds), *Joint Eurographics and IEEE TCVG Symposium on Visualization (Data Visualization '00)*, Springer-Verlag, Amsterdam, p. 3-13, 2000.

North S. C., Woodhull G., « Online Hierarchical Graph Drawing », *Graph Drawing : 9th International Symposium, GD 2001*, Vienna, Austria, p. 77-81, 2002.

Quigley A., Eades P., « FADE : Graph Drawing, Clustering, and Visual Abstraction », *in* J. Marks (ed.), *Symposium on Graph Drawing (GD 2000)*, LNCS 1984, Springer, p. 197-210, 2000.

Schaffer D., Zuo Z., Greenberg S., Bartram L., Dill J., Dubs S., Roseman M., « Navigating Hierarchically Clustered Networks through Fisheye and Full-Zoom Methods », *ACM Transactions on Computer-Human Interaction*, vol. 3, n° 2, p. 162-188, 1996.

Shneiderman B., « Tree visualization with tree-maps : 2-d space-filling approach », *ACM Transactions on Graphics*, vol. 11, n° 1, p. 92-99, 1992.

van Wijk J. J., van Ham F., « Interactive Visualization of Small World Graphs », *in* T. Munzner, M. Ward (eds), *IEEE Symposium on Information Visualisation*, IEEE Computer Science press, Austin, TX, USA, 2004.

Vliegen R., van Wijk J. J., van der Linden E.-J., « Visualizing Business Data with Generalized Treemaps », *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, n° 5, p. 789-796, 2006.