Visualisation d'information : paradigmes de navigation multi-echelle et approches "focus+contexte"

Pierre-Yves Koenig

Université Montpellier II
— Sciences et Techniques du Languedoc —

Thèse

pour obtenir le grade de
Docteur de l'Université Montpellier II

Discipline : Informatique
Spécialité Doctorale : *Informatique*
Ecole Doctorale : *Information, Structure, Systèmes*

présentée et soutenue publiquement par

Pierre-Yves Koenig
le 12 novembre 2009

Visualisation d'information : paradigmes de navigation multi-échelle et approches "focus+contexte"

Jury

Laurence Nigay, Professeur, Universite Joseph Fourier Grenoble 1, LIG, ... Rapporteur
Pascale Kuntz-Cosperec, Professeur, Polytech'Nantes, LINA, ... Rapporteur
Pascal Poncelet, Professeur, Université Montpellier II, ... Président
Maguelonne Teisseire, Directrice de recherche, CEMAGREF, TETIS, ... Examinateur
Céline Rozenblat, Professeur, Université de Lausanne, ... Examinateur
Fabien Jourdan, Chercheur, INRA Toulouse, ... Examinateur
David Auber, Maitre de conférences, Université Bordeaux I, LaBRI, ... Examinateur
Guy Melançon, Professeur, Université Bordeaux I, LaBRI, ... Directeur de Thèse

A mes parents Alain et Monique Koenig

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Guy Melançon, qui a permis que cette thèse aboutisse. Merci pour sa confiance, son écoute, sa patience et ses encouragements tout au long de cette thèse. Merci de m'avoir permis d'explorer des voies tout en me guidant quotidiennement.

Je remercie les membres de mon Jury de m'avoir fait l'honneur de relire mon manuscrit. Ils m'ont permis, par leurs remarques et corrections, d'améliorer la qualité de ce document.

Je souhaite remercier l'université de Montpellier II ainsi que le Lirmm pour m'avoir permis de réaliser cette thèse dans de bonnes conditions. Je remercie Maylis Delest qui m'a accueilli chaleureusement au LaBRI.

Je tiens aussi à remercier l'ensemble des membres du groupe Gravité, Tatoo et Spangeo qui m'ont apporté un cadre de recherche de qualité ainsi qu'une opportunité de présenter, d'améliorer et d'évaluer mes travaux.

Je remercie ma compagne, Laetitia Lefebvre, qui a réussi à me supporter durant toutes ces années de thèse qui ne sont pas toujours faciles et d'avoir réussi à lire cette thèse en entier. Je remercie aussi mes parents pour le temps passé à corriger les nombreuses fautes présentes dans cette thèse et pour les différentes traductions effectuées durant ces dernières années. Je remercie aussi mon frère Louis-Benoît et mes amis qui ont su m'écouter et essayer de comprendre mes idées quand elles étaient encore naissantes et mal formulées.

Je conclurais en remerciant tous ceux qui, de prés comme de loin, ont contribué à mon travail de thèse.

Visualisation d'information : paradigmes de navigation multi-échelle et approches "focus+contexte"

Résumé La visualisation d'information est une approche des plus prometteuses pour l'exploration, l'analyse et la compréhension de données. Dans le cadre de cette thèse, les différentes collaborations avec les utilisateurs finaux de nos méthodes (biologistes et géographes) ont permis d'élaborer de nouvelles visualisations et interactions adaptées aux besoins spécifiques de leur domaine.

La notion d'exploration "focus+contexte" est ici centrale. Nous l'avons abordée tant au travers de la visualisation de données multidimensionnelles que de graphes hiérarchiques. Une méthode permettant de visualiser et d'interagir sur des données multidimensionnelles est proposée. Cette méthode permet d'identifier visuellement des corrélations non linéaires à l'aide d'une matrice de graphes de corrélation. Nous présentons alors une technique permettant de visualiser de façon interactive les structures hiérarchiques sous forme de graphes orientés acycliques (DAGs). Les résultats d'une évaluation formelle de la méthode sont alors présentés. Une extension à des données plus massive utilisant des techniques de visualisation "focus+context" est alors proposée.

Mots-clés Visualisation d'information, focus+contexte, navigation, hiérarchie, dag, graphe de corrélation

Information Visualization : paradigms of multi-scale navigation and "focus+context" approachs

Abstract Visualization of information is a promising approach to the exploration, analysis and understanding of data. For the purpose of this thesis, the various collaborations with the end user of our method (Biologists and Geographers) have enabled us to elaborate new visualizations and interactions adapted to their specific needs. The notion of "focus+context" exploration is central in our case. We have approached the analysis through both the visualization of multidimensional data and hierarchical graphs.

A method which allows one to visualize and interact with multidimentional data is proposed. This method enables visual identification of nonlinear correlations through the help of a matrix of graph correlation. We hereby present a technique which allows the visualization in an interactive manner of hierarchical structures in the form of directed acyclic graphs (DAGs). The results of the formal valuation of the method are presented. An extension to more massive date through the use of "focus+context" visualization is thus proposed.

Keywords Information visualization, focus+context, naviguation, hierarchie, dag, scatterplot

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Motivation

> A picture is worth more than a thousand of words ... if it is the right picture.[1]

Il est communément admis qu'une image est parfois plus efficace qu'un millier de mots (si c'est la bonne image). Ce dicton suggère combien la visualisation d'information peut être utile pour expliquer et comprendre des informations.

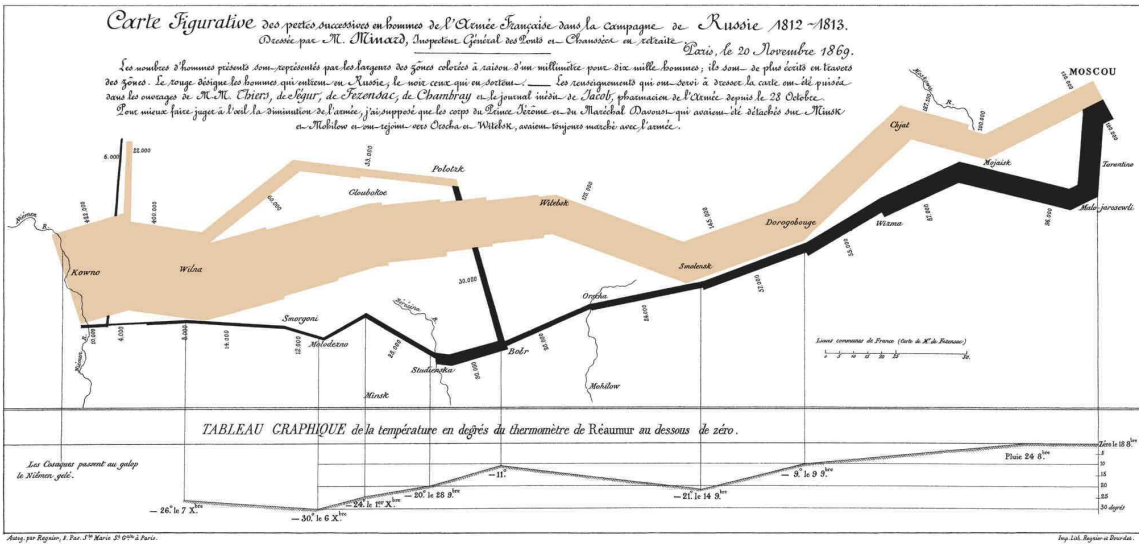


FIG. 1.1 – M. Minard 1869 - Carte figurative des pertes successives en hommes de l'armée française pendant la campagne de Russie (1812-1813).

L'image a une force qu'on veut exploiter. Le graphique de Minard en 1869 est un exemple célèbre qui a été mis en avant par Tufte (voir figure 1.1). Ce graphique est une carte figurative des pertes successives en hommes de l'armée française pendant

[1]Citation de Fred R. Barnard dans un journal de commerce. Cette citation vient dans le cadre de la publicité sur les bus "One Look is Worth A Thousand Words". Cette citation est aussi attribuée à Napoleon Bonaparte : "Un bon croquis vaut mieux qu'un long discours" (voir http ://en.wikipedia.org/wiki/A_picture_is_worth_a_thousand_words)

la campagne de Russie (1812-1813). Différents artifices visuels ont été utilisés pour rendre compte des facteurs importants de cette campagne. Les chiffres d'hommes présents sont représentés par les largeurs des zones colorées à raison d'un millimètre par dix mille hommes ; ils sont de plus écrits en travers des zones. Le rouge désigne les hommes qui entrent en Russie, le noir ceux qui en sortent. Ce graphique regroupe une échelle de temps (progression de l'armée de gauche à droite (en rouge) puis de droite à gauche (en noir)), de température (en bas du graphique), le nombre d'effectif (épaisseur du trait) ainsi qu'une localisation géographique (partant de "Kowno" à gauche jusqu'à "Moscou" à droite).

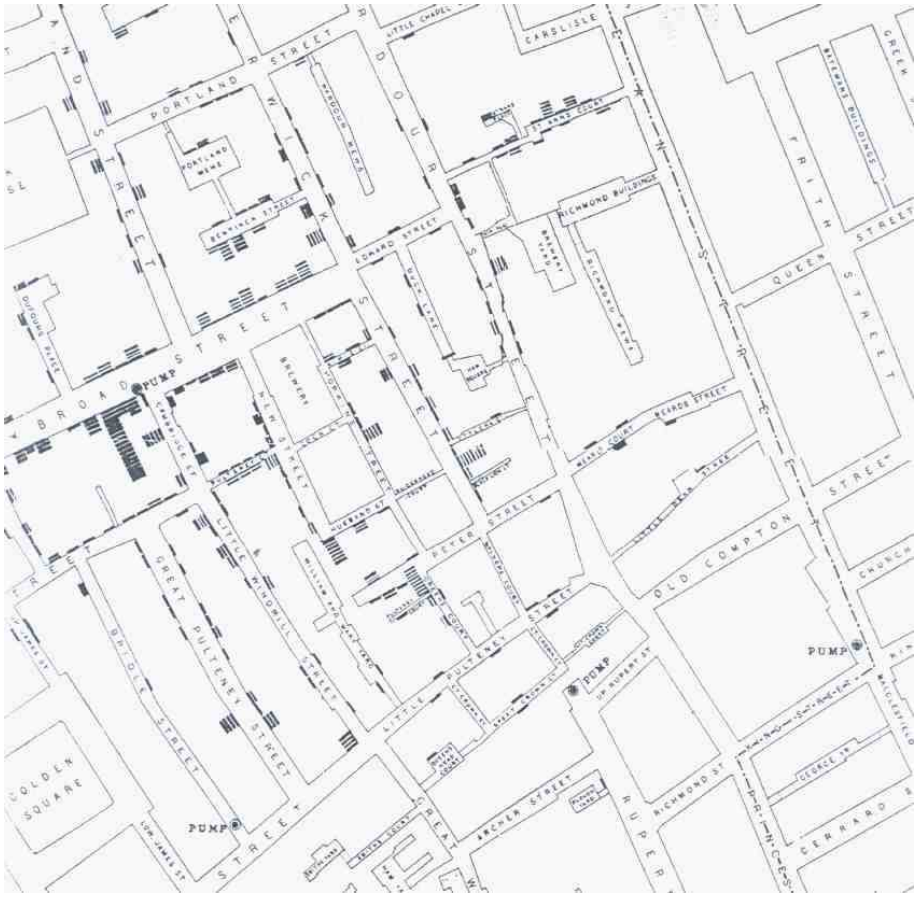


FIG. 1.2 – Le plan de Soho annoté par John Snow en 1854. Les traits noirs représentent les décès dûs au choléra. Les points noirs symbolisent les pompes à eau du quartier. Snow remarqua que les décès étaient localisés aux abords de la rue Broad Street.

En 1854 le district de Soho à Londres est frappé par une épidémie de choléra. John Snow reporte sur une carte du district de Soho (voir figure 1.2) les cas de décès du choléra (traits noirs) et les différentes pompes à eau (points noirs). La carte ainsi annotée par Snow révèle une corrélation forte entre la densité de cas de choléra et une pompe à eau située sur Broad Street. La mise hors service de cette pompe facilita l'endiguement de l'épidémie. La visualisation d'information permet ici une interprétation des données et apporte une aide à la décision.

Un des plus grands bénéfices de la visualisation de données est le nombre conséquent d'informations qui peuvent être rapidement interprétées (voir figures 1.1 et 1.2). L'information importante dans un nombre conséquent de mesures est immédiatement perceptible par l'utilisateur et devient accessible à travers l'interaction. Dans le chapitre suivant (2), nous montrons un système de visualisation affichant simultanément des informations sur 751 molécules, suivant 10 modes opératoires. L'utilisateur va en quelques secondes identifier l'information sous-jacente qui l'intéresse au moyen d'interactions spécifiques.

La visualisation permet souvent de mettre en évidence des problèmes sur les données elles-mêmes (problèmes lors de la collecte des données, par exemple les mesures biaisées d'un capteur). Avec une visualisation appropriée, les erreurs et artefacts dans les données sautent souvent aux yeux. Par exemple, les mesures d'un capteur défectueux produiraient une information non cohérente repérable sur une représentation. Pour cette raison, la visualisation peut être un outil dans le contrôle de qualité [4].

La visualisation est également utile aussi bien quand on veut traiter des données massives que quand on veut traiter des données de petites tailles. La visualisation facilite la compréhension de données de grande comme de petite échelle.

La visualisation permet la perception de phénomènes (ou propriétés) émergents qui n'auraient pas été anticipés. Nous donnons un exemple de telle propriété dans le chapitre 4. La visualisation facilite l'élaboration d'hypothèses. On verra au chapitre 4 comment une visualisation permet de formuler des hypothèses sur la stratégie des entreprises dans le contexte de la mondialisation.

L'analyse de données est une tâche commune à différents domaines de recherche. La visualisation d'information est une approche utile pour l'analyse et l'exploration de données. Une des principales contributions de la visualisation (comparée aux approches non visuelles) est de permettre une interaction directe avec les données et de fournir un retour immédiat. La visualisation peut alimenter le processus d'analyse et guider l'utilisateur dans l'exploration et le raffinement d'hypothèse scientifique.

1.2 Système de visualisation

La visualisation d'information peut être considérée comme un système, un processus composé de différentes étapes. De nombreux diagrammes illustrant ces étapes ont été proposés. La figure 1.3 proposée par Colin Ware [91] montre les différentes interactions entre les étapes que nous allons détailler. Partant des données ("Data"), on applique différents processus et transformations, par exemple le calcul d'indices, des regroupements ou encore des filtres ... ("Processing & Transformation"). L'étape suivante est la construction d'un objet graphique, une vue ("Graphics Engine") afin d'obtenir un objet graphique (visuel). Cette vue des données est soumise à l'analyse de l'utilisateur ("Human Information Analyst"). L'utilisateur perçoit l'information par le biais de processus cognitif ("Visual and Cognitive Processing"). Ce processus est lui-même modélisé de manière détaillée par van Wijk [88] (voir figure 1.4). Il peut alors manipuler l'objet graphique, par exemple à travers des zooms (géométriques) avant ou arrière ("Data Manipulation") influant sur l'objet graphique. Il peut aussi

changer les différents paramètres permettant d'obtenir l'objet graphique ("Data Exploration"). Grâce à son expertise, il peut rajouter de l'information à l'espace de données ("Data Gathering"). Par exemple, dans l'analyse financière, l'utilisateur peut vouloir inclure une nouvelle dimension suivant le contexte géopolitique tel que le prix du pétrole.

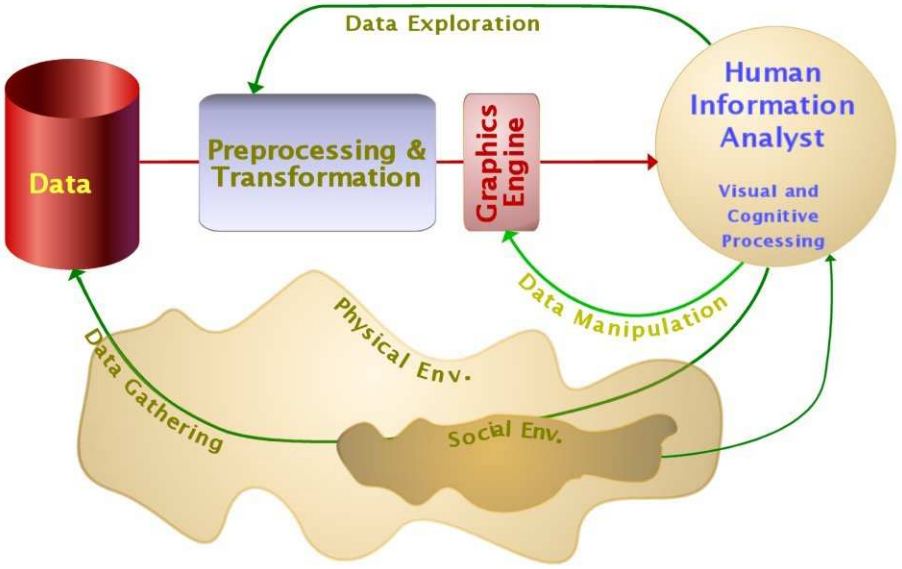


FIG. 1.3 – Diagramme proposé par Colin Ware [91] montrant les différentes étapes d'un système de visualisation.

Le diagramme 1.4 pose un autre regard sur le processus de fouille visuelle [88]. Il se décompose en entités : les données ("data"), la visualisation ("visualisation") et enfin l'utilisateur ("user"). Dans ce diagramme, la partie intéressante est la partie concernant l'utilisateur. Partant des données ("D"), on produit une première visualisation ("V") par le biais d'une "spécification" ("S"). Le terme de spécification dans le diagramme est volontairement vague et englobe les différentes techniques de visualisation. Cette visualisation est perçue comme une image ("I") par l'utilisateur ("user"). La perception ("P") de l'image est fonction des capacités cognitives de l'utilisateur . Cette perception produit une connaissance sur les données ("K"). L'utilisateur ayant une connaissance sur les données, il peut vouloir obtenir une nouvelle image des données en faisant varier les paramètres de la spécification ("S") ou voire en changer. Cette étape est l'exploration ("E") des données. Une nouvelle image apporte ainsi de nouvelles connaissances et le processus se répète jusqu'à la découverte d'un phénomène.

Les diagrammes 1.5 et 1.6 donnent une vue du processus de visualisation très proche de la vision "ingénieur" (de l'implémentation). Ces pipelines sont présents dans la majorité des systèmes de visualisations implémentés [19, 27]. Chaque entité ou paire d'entité du diagramme correspond à un module d'une application. Partant de la donnée brute, on filtre cette information pour se concentrer sur un sous-ensemble qui sera utilisé pour produire une visualisation.

[2] `http://www.infovis-wiki.net/`

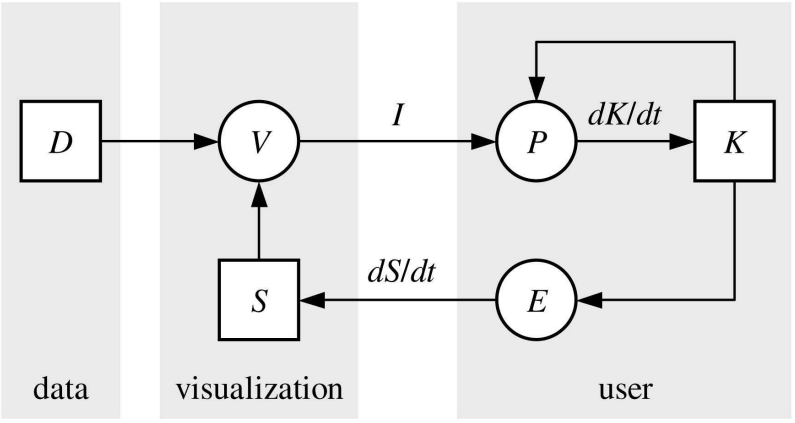


FIG. 1.4 – Diagramme proposé dans "View on Visualization" [88] montrant les processus cognitifs de l'utilisateur lors de la visualisation.

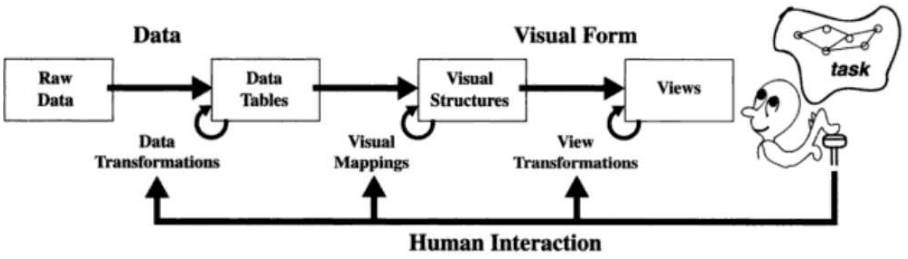


FIG. 1.5 – Description d'un système de visualisation mettant en évidence les différentes étapes internes à la transformation [19].

1.3 Conception d'un système de visualisation

La visualisation d'information apparaît donc comme une stratégie gagnante pour l'analyse de données. Le défi qui se pose pour un chercheur en visualisation est de concevoir un système qui soit utile. Produire une bonne visualisation implique le choix d'une métaphore visuelle qui reflète bien ce qui se passe dans les données. Le processus de visualisation étant itératif, le choix des interactions doit être adapté aux problèmes, aux données ainsi que la représentation des données elles-mêmes. Différents articles font état des 10 principaux problèmes non résolus [46, 61, 30, 22, 88]. Ces problèmes se posent en différents termes. L'utilisabilité des méthodes doit être évaluée. Une bonne visualisation doit limiter les tâches cognitives lourdes nécessitant un travail sur la perception de l'information. Les méthodes doivent

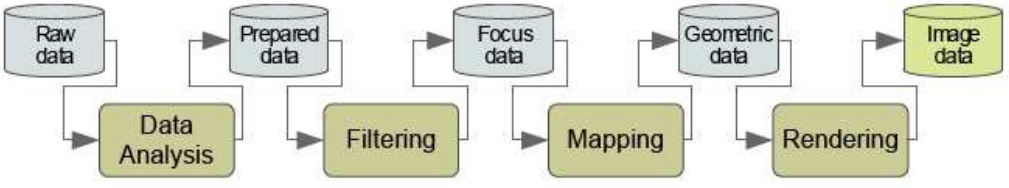


FIG. 1.6 – Processus de visualisation proposé par le site InfoVis Wiki[2].

pouvoir s'adapter à différents domaines. Le langage de la visualisation d'information doit être compris par les scientifiques d'autres domaines. On doit établir des critères quantifiables pour juger une visualisation en terme de lisibilité, d'efficacité et d'esthétisme. Le passage à l'échelle des visualisations doit être étudié ainsi que l'étude des graphes dynamiques. Dans cette thèse, nous avons abordé quelques-uns de ces points au travers de cas d'études spécifiques.

1.4 Contribution et structure du document

Les questions étudiées dans cette thèse portent sur la notion d'exploration "focus+contexte". Celle où l'utilisateur définit implicitement ou explicitement un centre d'intérêt. Cette notion n'est pas attachée à un type de visualisation particulier et nous l'avons abordée tant au travers de la visualisation de données multidimensionnelles que de graphes hiérarchiques. De plus, cette notion peut prendre corps à différents moments du processus décrit en figures 1.3 et 1.4. En effet, soit le centre d'intérêt peut influer sur le filtrage des données (Data Exploration), soit sur la vue des données (Data Manipulaiton). Nos travaux se sont développés autour de deux cas d'études qui nous ont permis d'explorer cette notion et de proposer des nouveaux modes de visualisation.

La thèse aborde les deux cas d'étude dans des chapitres distincts. Le premier concerne la visualisation de données multidimensionnelles et exploite le phénomène de perception humaine multi-niveaux d'une image par l'oeil de l'utilisateur. Le second cas d'étude décrit la relation d'héritage modélisée par un graphe (orienté acyclique), pour lequel nous avons conçu une visualisation originale.

Chaque cas d'étude a fait l'objet d'une collaboration avec les utilisateurs finaux pour guider nos choix de conception des métaphores visuelles et des interactions. La visualisation proposée pour les relations d'héritage a par ailleurs fait l'objet d'une validation expérimentale.

Plutôt que de proposer un état de l'art dans un chapitre d'introduction, nous avons préféré reporter dans chacun des chapitres une discussion de l'état de l'art pertinent pour le problème étudié. Dans le chapitre suivant, la perception d'une image par l'homme sera abordée. Après un état de l'art des techniques utilisant cette perception, notre technique sera détaillée suivie d'un exemple d'application constituant notre cas d'étude. Le chapitre 3 dresse un état de l'art des techniques de visualisation de structures arborescentes et de structures plus complexes. Dans le chapitre 4, nous présentons une technique permettant de visualiser de façon interactive les structures hiérarchiques sous forme de graphes orientés acycliques (DAG). Nous présenterons alors les résultats d'une évaluation de notre méthode. Une extension de cette méthode afin de traiter des données plus massives nécessite l'utilisation de techniques de visualisations "focus+contexte". Un état de l'art de ces techniques fera l'objet du chapitre 5. Notre adaptation de ces techniques au DAG constituera le chapitre suivant. Enfin, différentes perspectives seront formulées, puis nous conclurons.

La première année de cette thèse s'est déroulée dans le cadre d'un projet en collaboration avec des bio-informaticiens qui nous ont amenés à explorer les données

multidimensionnelles. Mon travail s'est ensuite inscrit dans le cadre du projet Spangeo[3] en relation étroite avec des géographes. Le projet Spangeo est un projet ANR qui regroupait des informaticiens du LIRMM et du LaBRI et des géographes de Montpellier, Lille, Paris. Notre travail a donné lieu à des publications nationales et internationales dans le domaine de la géographie comme dans le domaine informatique (voir section Publications).

[3]http ://s4.parisgeo.cnrs.fr/spangeo/spangeo.htm

Chapitre 2

Scatterplot

L'homme est doté de facultés surprenantes. Dans de nombreux domaines, les chercheurs essaient de reproduire, d'égaler ces facultés. En visualisation d'information, on s'intéressera tout naturellement à la perception visuelle de l'information. Dans ce chapitre, nous allons voir dans un premier temps comment l'homme perçoit l'information, les images et comment cela peut être modélisé. Nous allons voir comment utiliser ces résultats pour construire une nouvelle visualisation interactive permettant l'étude de données 2D (graphe de corrélation). Une application concrète sera alors présentée s'intéressant aux données multidimensionnelles.

2.1 Perception organisationnelle d'une image

2.1.1 Perception multi-échelle d'une image

L'oeil humain a une grande capacité à percevoir des motifs dans des structures complexes. De plus, il a naturellement une perception multi-échelle de son environnement. Par exemple, la perception d'une salle de musée commence par les murs, quatre grands rectangles, puis sur un rectangle (un mur) on distingue d'autres rectangles plus petits (des tableaux), un tableau est une image qu'on décompose en zones. De même, quand on regarde un poster (voir figure 2.1), on identifie instantanément les différentes zones le composant (zones de texte, images, contacts ...).

L'information contenue dans un document texte peut être perçue à différentes granularités [93] : un document est d'abord perçu comme un bloc ; quand le regard s'attarde, on peut alors distinguer des blocs plus petits représentant les paragraphes puis sur un paragraphe on distingue alors des phrases, des mots et enfin des lettres (voir figure 2.2).

Ce phénomène a été modélisé par Fisher et Wattenberg [93]. Ils considèrent une page de texte (document) comme une simple image, puis en appliquant des techniques de graphisme classique tels que le lissage et la segmentation, ils calculent sa structure multi-échelle. Nous allons décrire de manière détaillée ce modèle dans les sections qui suivent. Nous verrons ensuite comment leurs résultats ont été exploités pour visualiser des données multidimensionnelles de manière multi-échelle.

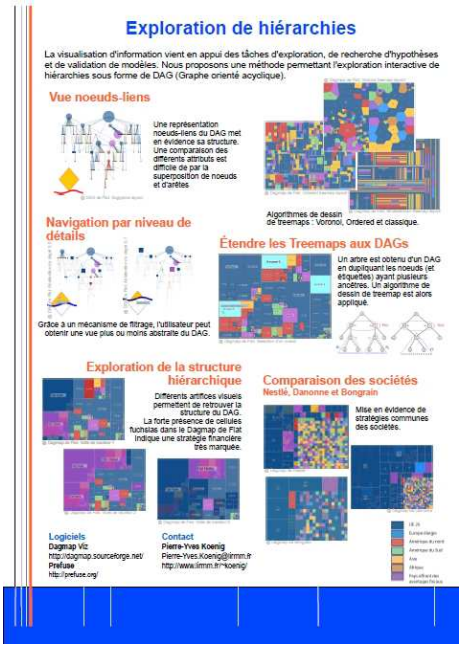


FIG. 2.1 – Image d'un poster.

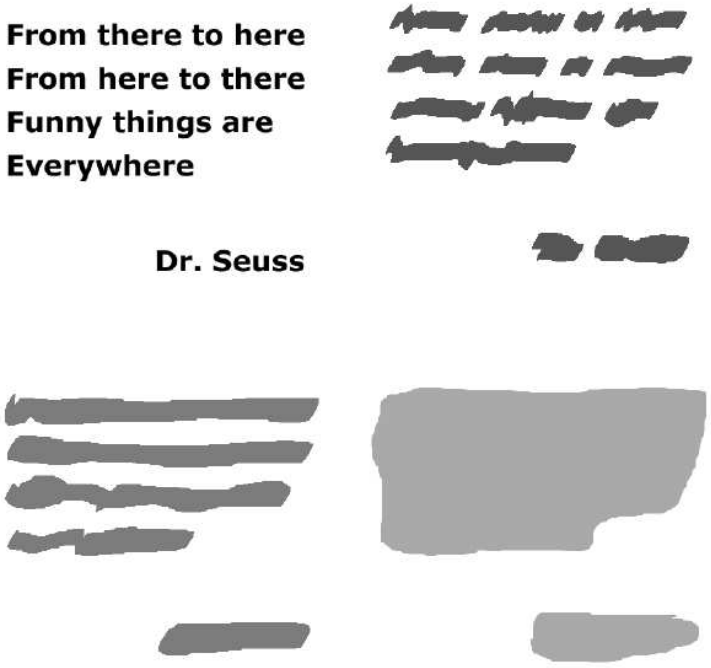


FIG. 2.2 – Décomposition d'une image par blocs de différentes granularités.

2.1.2 Lissage et filtres gaussiens

Le calcul des différents blocs sur la page ('paragraphes', lignes de texte, mots ou lettres) nécessite de définir leur contour. L'approche utilisée par Fisher et Wattenberg utilise le lissage de l'image à l'aide de filtres gaussiens. Ainsi, les lettres des mots se fondent en un seul bloc uniforme et peuvent être extraites de l'image à l'aide de techniques de segmentation. Les paramètres de lissage peuvent être poussés jusqu'à produire des blocs uniformes pour les lignes de texte et les paragraphes.

Le lissage est une technique qui consiste à réduire les irrégularités et singularités d'une courbe en mathématiques. Pour plus de détails, le lecteur peut se référer aux ouvrages [80, 82]. Pour chaque point de la courbe, une nouvelle valeur lui est attribuée en fonction de son voisinage. L'importance donnée aux voisins est définie par un noyau de lissage. Cette technique peut s'appliquer aux signaux sonores ou aux images par exemple. Prenons l'exemple d'un signal en une dimension décrit par une fonction $f(x)$.

Un des noyaux les plus utilisés est le noyau gaussien

$$g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2},$$

qui décrit la distribution de probabilité d'une loi normale centrée ($\mu = 0$) réduite ($\sigma = 1$) $N(0, 1)$ (voir figure 2.4). En d'autres mots, la nouvelle valeur qui est calculée prend en compte le voisinage du point x et attribue un poids plus fort aux voisins les plus proches, en fonction de cette distribution. Ce noyau se généralise au cas $N(\mu, \sigma)$ et prend alors la forme

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

La convolution de ces deux fonctions ($z = f * g$) est donnée par une intégrale et résulte en une autre fonction qui décrit un nouveau signal.

$$z(x) = (f * g)(x) = \int_{-\infty}^{+\infty} f(x - t).g(t)dt$$

Le même principe peut être appliqué pour lisser des données discrètes, comme par exemple le signal de la figure 2.3. On recalcule pour chaque valeur du domaine une nouvelle valeur en fonction du voisinage accordant un poids plus important aux voisins qui sont proches du point considéré. Les paragraphes qui suivent décrivent de façon informelle comment se fait le calcul algorithmiquement.

Le noyau gaussien peut aussi s'exprimer comme une fonction discrète (comme un histogramme - voir figure 2.5)

Le processus de lissage basique est très simple. On procède à travers les données point par point. Pour chaque point, on calcule une nouvelle valeur qui est fonction de la valeur d'origine du point et des points voisins. Avec un lissage gaussien, la fonction utilisée est notre courbe de Gauss discrétisée. La fonction $z(x)$ se calcule en déplaçant une fenêtre d'une largeur bornée n le long de la fonction en faisant agir les poids du noyau, à l'image d'une moyenne glissante pondérée.

$$z(x) = \sum_{m=-\infty}^{+\infty} f(n - m).g(m)$$

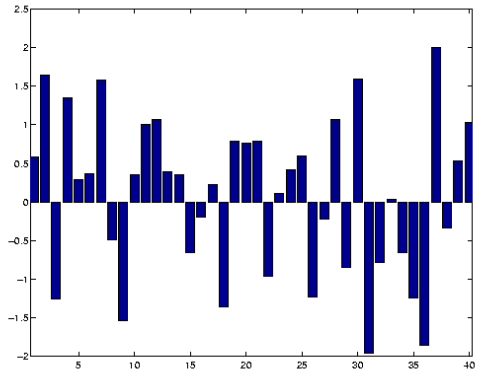


FIG. 2.3 – Signal aléatoire à une dimension.

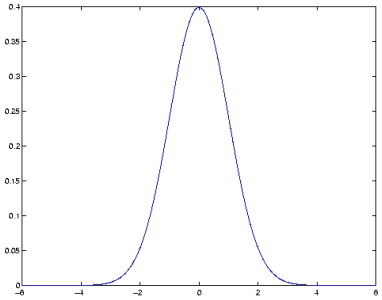


FIG. 2.4 – Densité de probabilité de la gaussienne centrée ($\mu = 0$) et réduite ($\sigma = 1$).

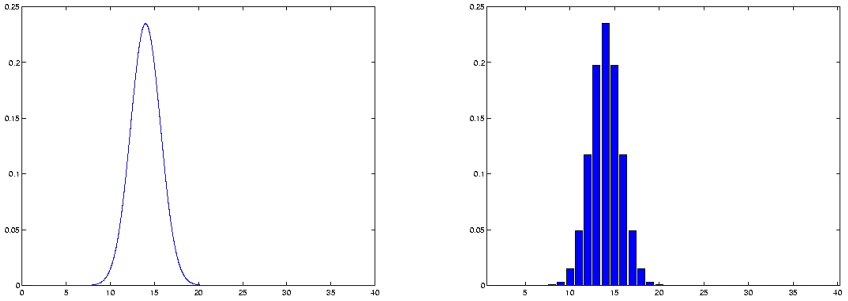


FIG. 2.5 – Densité de probabilité d'une gaussienne de moyenne $\mu = 14$ normalisée et discrétisée.

A la fin du processus, on obtient le lissage des données d'origine (voir figure 2.6). Un paramètre essentiel du lissage est, comme on l'a vu, la courbe gaussienne utilisée. Cette courbe est plus ou moins large suivant la valeur de son écart type σ. La figure 2.7 montre une courbe ayant subi un lissage avec différentes valeurs de σ. Le choix du paramètre σ est capital. Si σ est trop petit, le signal de départ sera très peu modifié. Si σ est trop grand, on réduit la signal à une seule information (la moyenne des valeurs).

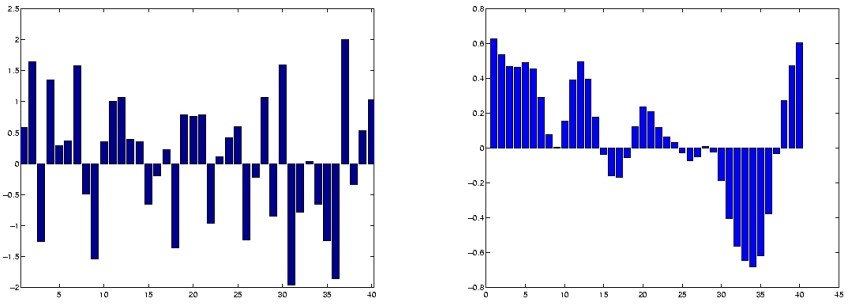
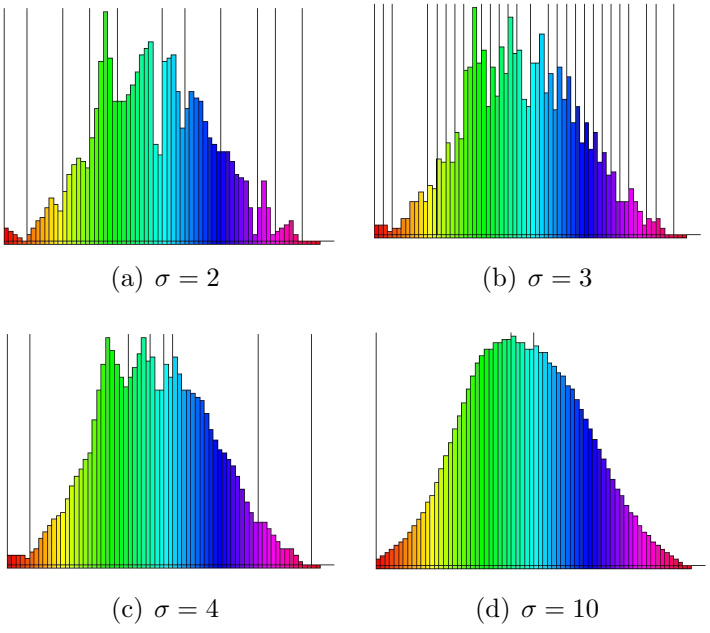


FIG. 2.6 – Lissage de la courbe avec un noyau gaussien.



(a) $\sigma = 2$ (b) $\sigma = 3$

(c) $\sigma = 4$ (d) $\sigma = 10$

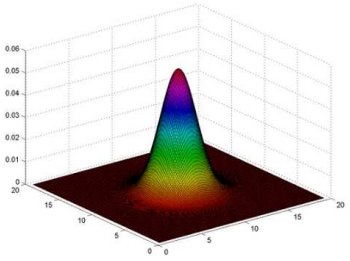
FIG. 2.7 – Lissage d'un signal suivant différents noyaux de lissage correspondant à plusieurs valeurs d'écart type de la courbe de gauss.

De la même manière, on peut définir un noyau gaussien à deux variables qui permet de lisser un signal bidimensionnel.

$$g(x,y) = Ae^{-(\frac{(x-\mu_x)^2}{2\sigma_x^2}+\frac{(y-\mu_y)^2}{2\sigma_y^2})}$$

La discrétisation du noyau s'exprime de manière matricielle. On peut tout aussi bien lisser un signal discret en deux dimensions. C'est le point de vue qu'on pose

sur une image. Nous avons vu qu'une image pouvait avoir différents niveaux de granularités. Le signal est ici en deux dimensions. En appliquant un filtre (un lissage) sur une image, on peut obtenir une abstraction de celle-ci. Un lissage permet, pour chaque pixel de la zone à laquelle il s'applique, de modifier sa valeur en fonction des valeurs des pixels avoisinants, affectées de coefficients. Le filtre est représenté par une matrice qui discrétise la fonction gaussienne (le noyau gaussien), caractérisée par ses dimensions (3×3, 5×5, ...) et ses coefficients. Le centre de cette matrice correspond au pixel concerné. Les coefficients de la matrice déterminent les propriétés du filtre. La figure 2.8 présente le dessin d'une fonction gaussienne et des filtres associés de différentes tailles. L'image à traiter est considérée comme une matrice de pixels (matrice image). À chaque élément de la matrice est associée une valeur reflétant la couleur du pixel. Ainsi, le produit de la matrice image par le filtre donne une matrice correspondant à l'image traitée.



(a) Fonction gaussienne

1	2	1
2	4	2
1	2	1

(b) Filtre 3×3

1	2	3	2	1
2	6	8	6	2
3	8	10	8	3
2	6	8	6	2
1	2	3	2	1

(c) Filtre 5×5

FIG. 2.8 – Fonction gaussienne et filtres gaussiens.

2.1.3 Hiérarchisation visuelle de l'information

Dans "A Model of Multi-Scale Perceptual Organisation in information Graphics" [93], Martin Watterberg et Danyel Fisher proposent une méthode permettant d'extraire la structure de l'information contenue dans une image. Les auteurs formulent l'hypothèse que "la structure d'une image doit refléter les données qu'elle contient (représente)". Par exemple, l'image 2.2 représente un message avec une signature. Nous voyons que la décomposition de l'image suit bien son contenu séparant dans un premier temps le message de la signature, puis dans le message les différentes phrases dont il est formé, les mots, enfin les lettres. Leur méthode utilise les filtres précédemment décrits.

Partant de l'image (ici en niveau de gris), on lui applique un filtre gaussien avec des noyaux ayant différents écarts types (σ). Les différentes valeurs de σ correspondent aux différentes échelles. La figure 2.9(a) montre l'effet de ces filtres sur une image. La valeur de σ est de plus en plus forte de gauche à droite.

Un processus de détection de contour peut alors révéler les différents éléments de l'image. On obtient ainsi des "Geshalts cartoons" représentant une abstraction des données à une certaine échelle (voir figure 2.9(b)). Suivant les différents noyaux utilisés, on va obtenir une abstraction plus ou moins forte. Dans la figure 2.9(b),

(a) Images lissées (b) Images segmentées

FIG. 2.9 – Lissage suivant différentes courbes gaussiennes (valeurs d'écart type différentes) et segmentation d'image.

on voit de gauche à droite différentes abstractions : mots, phrases, paragraphes. On peut alors construire une hiérarchie de l'information entre ces abstractions (voir figure 2.10).

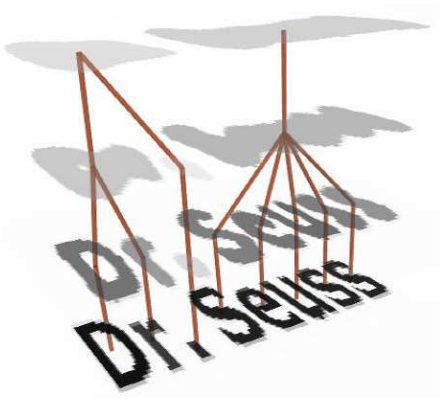


FIG. 2.10 – Hiérarchie extraite d'une image.

Le résultat de la segmentation donne une perception à plusieurs niveaux du document qui est illustré par la figure 2.10. Cette idée d'utiliser le flou pour extraire une hiérarchie est pertinente. Dans ses travaux, Kosara s'est beaucoup intéressé au flou et à ses applications en visualisation notamment dans [56], où il l'utilise afin de mettre en évidence le résultat d'une recherche dans un éditeur de texte (voir figure 2.11).

2.2 Analyse de données bidimensionnelles à l'aide de graphes de corrélation

En statistique, on cherche souvent à voir si deux variables sont corrélées, si une variable apporte plus d'informations que l'autre, si elles sont complémentaires et combien elles sont complémentaires. Il est fondamental en statistique de mesurer la corrélation. On peut utiliser le coefficient de corrélation, mais on peut aussi fabriquer

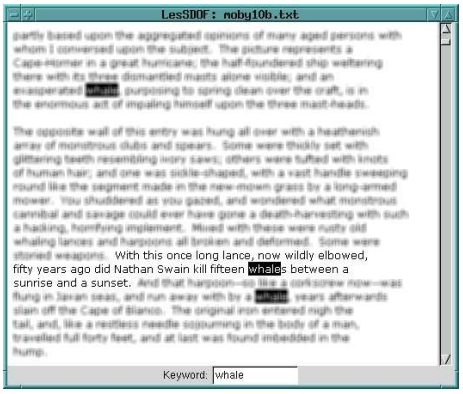


FIG. 2.11 – Utilisation du flou pour mettre en évidence une partie d'un texte.

une visualisation 2D des valeurs des variables pour visualiser cette corrélation : les valeurs donnent l'abscisse et l'ordonnée dans le plan, la corrélation parfaite étant une ligne sur la diagonale. Tout ce qui s'éloigne de la diagonale permet de mesurer à quel point les valeurs sont corrélées. Notez comme dans la figure 2.12 les graphes de corrélation ayant un coefficient de 1 peuvent être très différents.

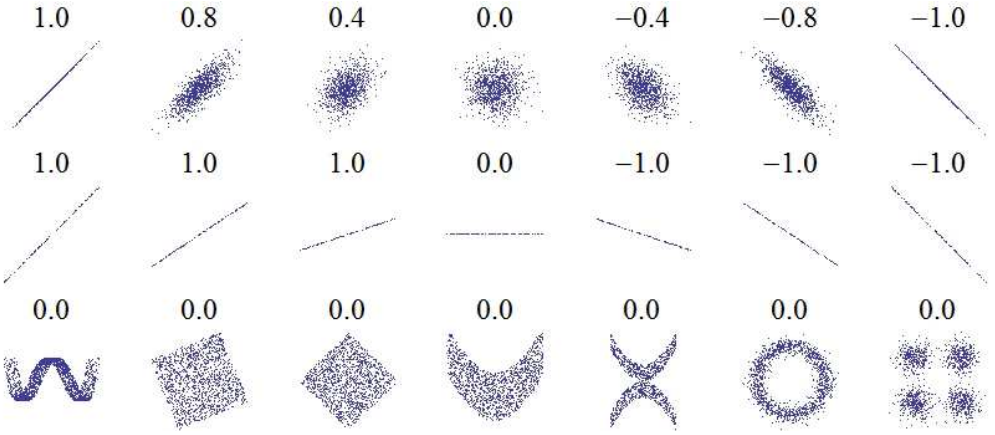
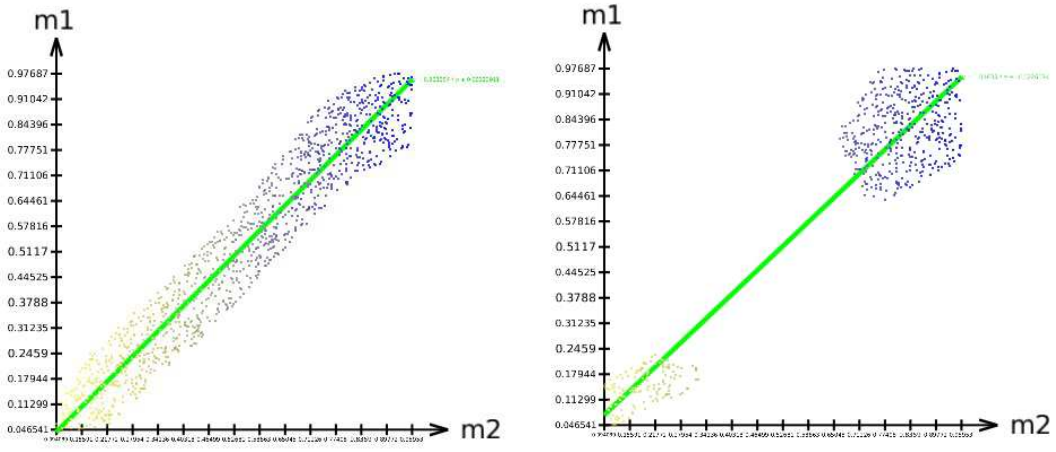


FIG. 2.12 – Exemple de diagramme 2D représentant des graphes de corrélation ayant différentes valeurs de coefficient de corrélation.

Les images de la figure 2.12 pourraient laisser croire que le coefficient de corrélation suffit pour déterminer le type de corrélation. Il n'en est rien car les situations rencontrées sont souvent plus complexes et composent différentes corrélations dans différentes régions (voir figure 2.13), d'où l'intérêt de pouvoir sélectionner des portions du diagramme ou d'interagir sur l'image.

Notre étude a été réalisée en collaboration avec les utilisateurs finaux qui souhaitaient non seulement visualiser la corrélation entre les variables mais aussi interagir avec le graphe de corrélation. Nous voulons donc permettre à l'utilisateur d'aller sélectionner des éléments du graphe de corrélation pour avoir des informations sous-jacentes. On reprend l'idée de Wattenberg et Fisher pour créer notre visualisation. Ainsi, nous avons élaboré une méthode de visualisation permettant d'explorer ces données. Différents paramètres permettent d'influer de façon interactive sur la visualisation permettant ainsi une recherche rapide de propriétés ou spécificités des données.



(a) (b)

FIG. 2.13 – Exemple de diagramme 2D représentant des graphes de corrélation ayant le même coefficient de corrélation. Notez comme la corrélation peut être localisée sur des régions différentes.

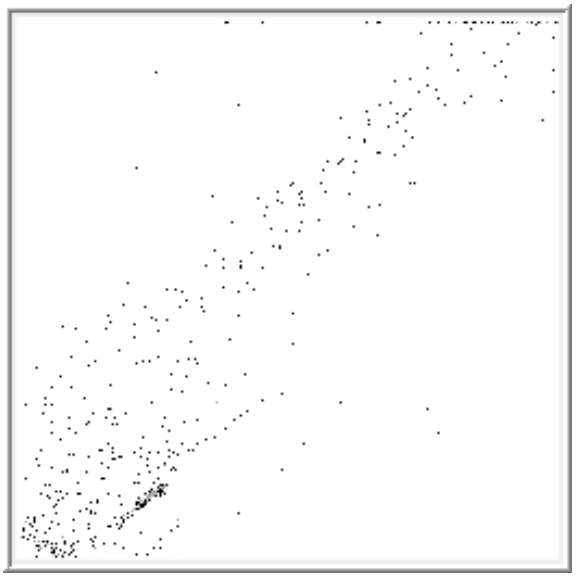


FIG. 2.14 – Représentation des données sous forme de scatterplot.

Dans un graphe de corrélation ou scatterplot 2D, à chaque élément e est attribué des coordonnées en abscisse (x) et en ordonnée (y) suivant les valeurs val_1 et val_2 des variables considérées. Il est à noter que plusieurs éléments peuvent être représentés par le même point dans le graphe de corrélation si ils ont les mêmes valeurs val_1 et val_2. Un niveau de gris permet alors de retranscrire cette information dans le scatterplot. Le nuage de points ainsi obtenu donne une bonne représentation de la relation entre les deux variables (voir figure 2.14).

Notre méthode considère un scatterplot comme étant une image. Chaque pixel $p_{(x,y)}$ représente l'ensemble des éléments tel que : $x = val_1, y = val_2$. On attribue alors un niveau de gris au pixel suivant le cardinal de l'ensemble qu'il représente. La représentation ainsi obtenue reste difficile à exploiter. De par la taille d'un pixel à l'écran, la sélection de ces éléments est difficile. On va exploiter l'idée de structurer l'image pour faciliter la sélection. On applique donc un filtre gaussien à l'image, ce qui a pour conséquence de la rendre floue (voir figure 2.15).

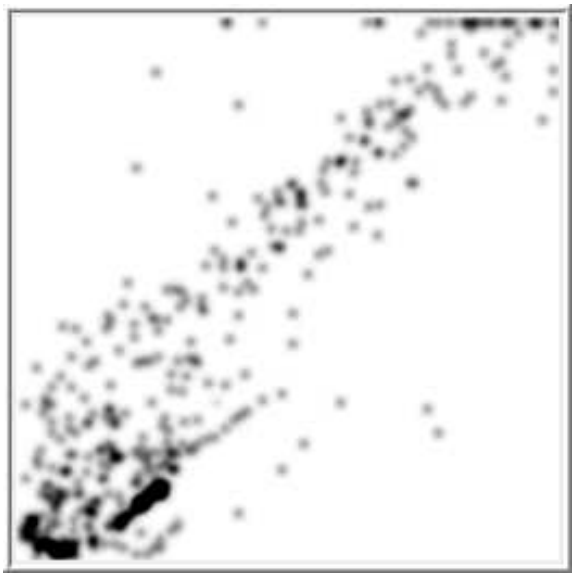


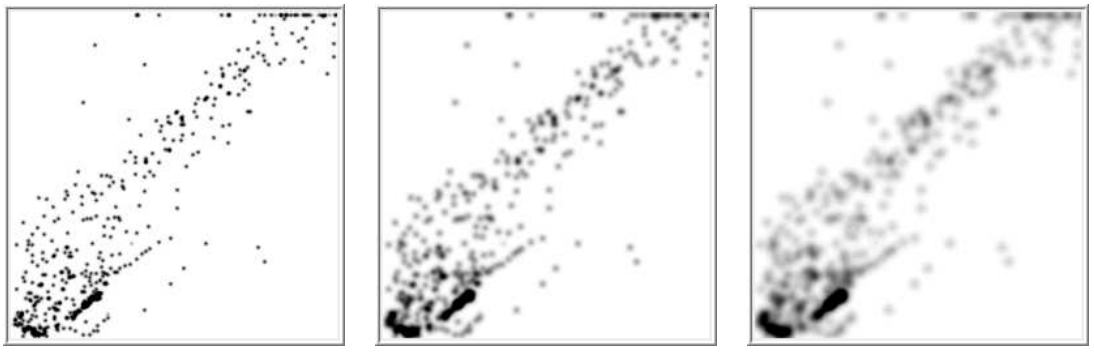
FIG. 2.15 – Filtre gaussien appliqué au scatterplot de la figure 2.14.

Les pixels ayant un niveau de gris traduisant la fréquence d'éléments présents, les zones les plus concentrées sont alors plus visibles. Les zones isolées sont plus faciles à sélectionner.

On peut faire varier le paramètre σ du filtre gaussien afin d'obtenir des zones plus ou moins fines. Les images de la figure 2.16 montrent le résultat obtenu suivant différentes valeurs de σ.

Dans cette image ainsi floue, les pixels voisins ayant un niveau de gris proche correspondent à des éléments similaires. On peut alors vouloir les regrouper en une même zone. On segmente donc l'image en zones de différentes intensités de gris en fixant le nombre k de niveaux de segmentation souhaité correspondant à k intervalles de valeurs pour les niveaux de gris.

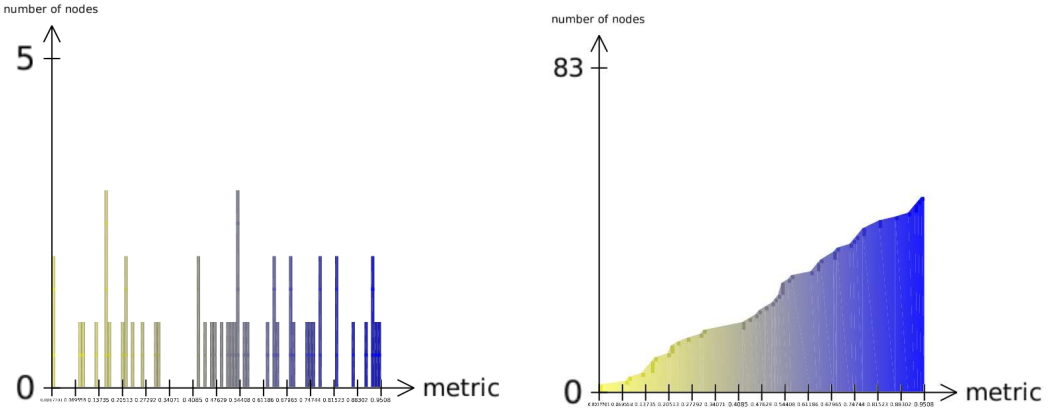
On peut déterminer ces intervalles en découpant l'histogramme des valeurs de gris en k intervalles. Mais on peut alors avoir des intervalles contenant un trop grand nombre d'éléments. On préférera utiliser un histogramme cumulé des valeurs



(a) $\sigma = 1$ (b) $\sigma = 2$ (c) $\sigma = 3$

FIG. 2.16 – Différentes vues d'un même scatterplot (figure 2.14 suivant la force du filtre gaussien appliqué ($\sigma = 1$, 2 et 3)).

de gris, qui tient compte de la distribution des valeurs. La figure 2.17 montre un histogramme et sa version cumulée.



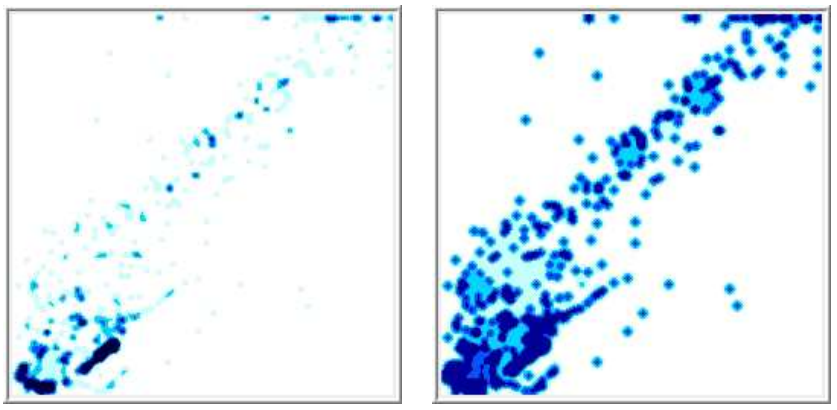
(a) Histogramme (b) Histogramme cumulé

FIG. 2.17 – Histogramme et histogramme cumulé d'un même jeu de données.

La figure 2.18 montre la segmentation de l'image 2.15 selon $k = 5$ niveaux d'intensité. A gauche, la segmentation est effectuée selon un histogramme normal et à droite selon un histogramme cumulé.

Les différentes interactions sur le σ du filtre gaussien et le nombre de segmentations permettent de paramétrer et changer la vue sur les données (voir figures 2.19 et 2.20). La figure 2.21 illustre la méthode de la construction d'un scatterplot à la segmentation. La figure 2.22 décrit le processus de visualisation de notre méthode. Voici le détail des différents éléments impliqués dans le processus.

- Données Brutes : Les données brutes sont le résultat des expérimentations et sont sous forme de spectrographes.
- Analyse de données : recherche de la concentration des éléments, normalisation des valeurs ou encore filtre de données incohérentes.
- Données structurées : données utilisables sous forme de table de données (csv).
- Transcription visuelle : plongement des données dans le plan (suivant deux attributs des données). Création d'un scatterplot.



(a) Segmentation avec histogramme normal

(b) Segmentation avec histogramme cumulé

FIG. 2.18 – Segmentation de la figure 2.15 selon $k = 5$ niveaux d'intensité. A gauche avec un histogramme normal et à droite avec un histogramme cumulé.

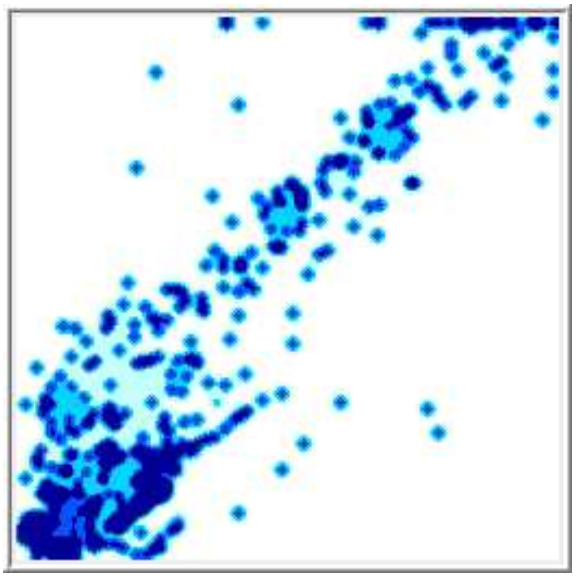
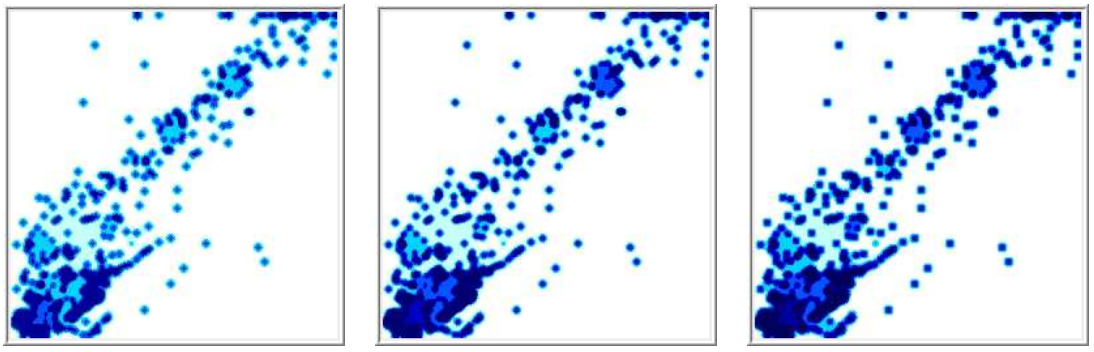
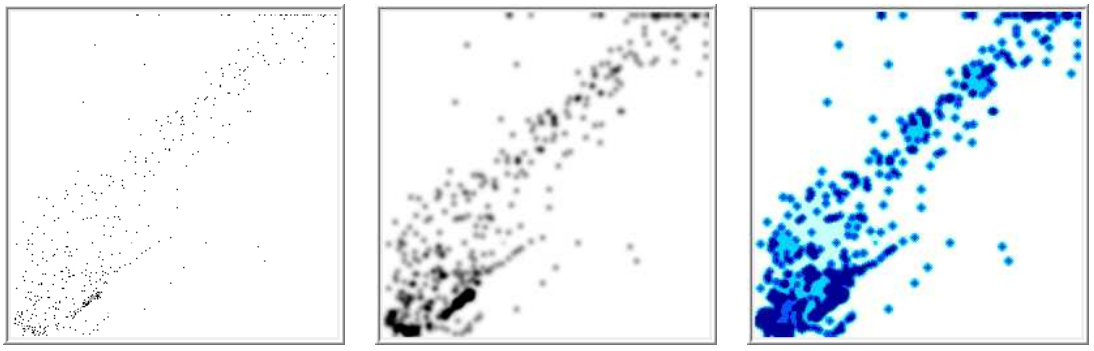


FIG. 2.19 – Segmentation de la figure 2.15 selon $k = 5$ niveaux d'intensité. Les différents niveaux de bleu traduisent cette intensité.



(a) Segmentation $k = 5$ (b) Segmentation $k = 6$ (c) Segmentation $k = 7$

FIG. 2.20 – Différentes vues segmentées d'un même scatterplot suivant le nombre de niveaux k choisi (voir figure 2.16).



(a) Scatterplot (b) Filtre gaussien (c) Segmentation

FIG. 2.21 – Processus complet du scatterplot à l'image segmentée pouvant être exploitée par l'utilisateur.

– Rendu : structuration de l'image par une suite de traitements d'images (filtre gaussien puis segmentation).

L'idée d'utiliser cette technique de flou avait été exposée dans [24] pour explorer des données relationnelles. Une paire d'indices est calculée pour les sommets d'un graphe. On génère alors un scatterplot afin de comparer ces deux indices. Avec la technique du flou, il est alors possible de sélectionner simultanément les sommets du graphe ayant un comportement similaire suivant la paire d'indices. La sélection se fait en cliquant sur une région définie dans le scatterplot (voir la figure 2.23, image de droite). Cette sélection est répercutée sur une représentation noeud lien du graphe (voir figure 2.23, image de gauche). Notre méthode étend cette idée en permettant à l'utilisateur d'interagir sur la construction du scatterplot. En effet les différentes étapes de la construction du scatterplot sont paramétrables.

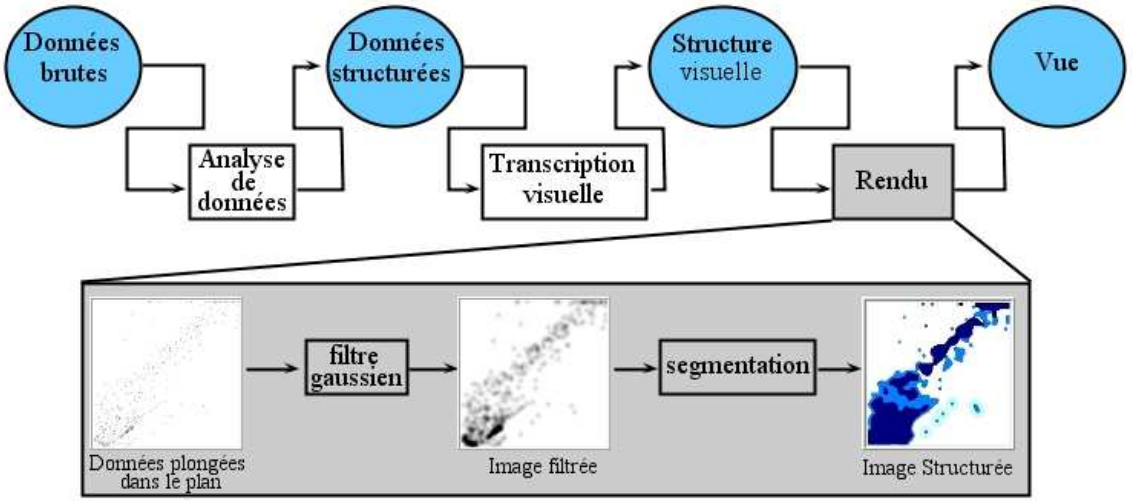


FIG. 2.22 – Processus de visualisation de l'application. Les éléments cerclés sont des états et les rectangles sont des opérations.

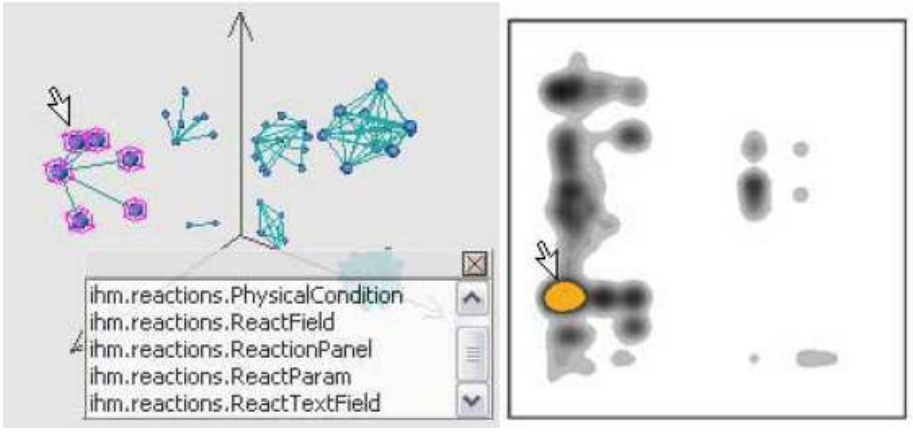


FIG. 2.23 – Dans [24], le scatterplot est utilisé pour la sélection des éléments dans le graphe : la sélection d'une zone sur le scatterplot induit la sélection des sommets correspondants dans la vue noeud lien du graphe.

2.3 Analyse de données multidimensionnelles : matrice de scatterplots

De nombreuses techniques permettent de visualiser des données multidimensionnelles . Dans [7], Becker et Cleveland utilisent une matrice de scatterplots. La sélection d'un ensemble d'éléments dans un scatterplot permet de voir alors leur positionnement dans les autres scatterplots de la matrice. Becker et Cleveland ont défini une technique qu'ils nomment *brushing* qui permet de sélectionner une zone rectangulaire de l'un des diagrammes et d'observer comment les points de cette zone se distribuent dans les autres diagrammes. Cette interaction a été étendue et enrichie dans [64].

La technique que nous avons développée est différente puisqu'elle n'utilise pas le brushing pour déterminer la région d'intérêt. Dans notre méthode, les régions d'intérêts sont définies lors de la construction de la visualisation : puisque notre diagramme est structuré, que des régions ont été identifiées par le lissage et la segmentation, elles peuvent être sélectionnées comme telles sans effort.

La technique développée pour une paire d'observation x et y peut être généralisée à l'exploration et la visualisation de données multidimensionnelles où chaque dimension correspond à une observation. On fabrique alors une matrice de scatterplots 2D, chacune des paires d'observation donnant lieu à un scatterplot 2D. On plonge dans une même visualisation plusieurs de ces scatterplots pour, simultanément, visualiser toutes les corrélations possibles (voir figure 2.24).

Lors de la sélection d'un élément dans un scatterplot (point vert), la zone le contenant est mise en évidence (la zone est colorée en rouge). Cette zone regroupe l'ensemble des points ayant un comportement similaire à l'élément sélectionné. Afin de comparer le comportement de ces éléments suivant d'autres conditions (d'autres variables), les zones contenant ces éléments sont, elles aussi, mises en évidence dans les autres scatterplots de la matrice.

2.4 Cas d'étude

La métabolomique s'intéressent aux systèmes cellulaires simples et, du moins pour ce qui est des données publiées, principalement aux concentrations des métabolites intracellulaires.

L'expérience concerne une population de souris qui ont été normalement nourrie, ou qui ont subi un changement dans leur régime alimentaire habituel. Un autre facteur étudié est l'administration de drogues ou de substances toxiques (une molécule ou un groupe de molécules), en ajout du régime alimentaire. Le but est de comprendre comment leur organisme réagit à ces perturbations (changement de régime alimentaire et/ou prise de drogue). L'expérience suit un protocole breveté [65] que nous allons décrire brièvement ici.

L'effet du régime alimentaire, avec ou sans prise de drogue, est observé à travers des échantillons de sang ou d'urine. Des échantillons sont prélevés sur les souris à plusieurs reprises (une fois par mois les deux premiers mois puis plus régulièrement). Chaque échantillon est transformé et suit une série d'étapes pour mesurer la présence

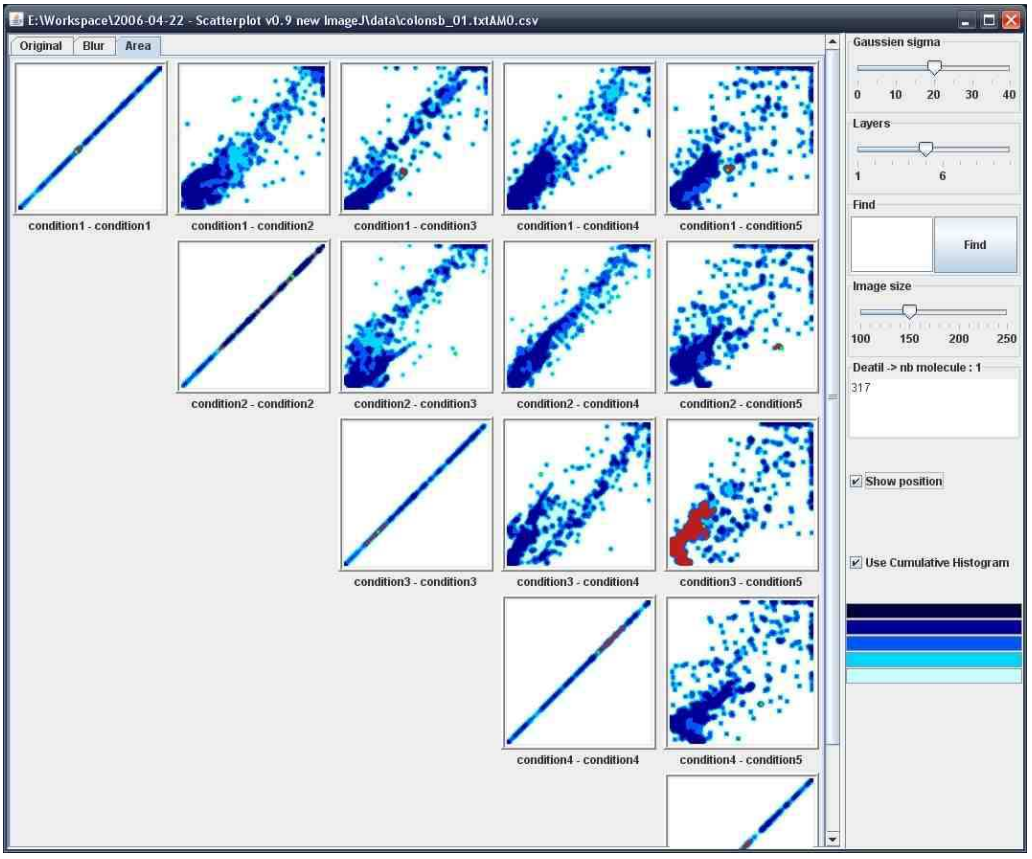


FIG. 2.24 – Capture d'écran de l'application montrant les différentes interactions possibles. La représentation est sous forme de matrice de scatterplots. Chacun des scatterplots est construit suivant une paire de situations.

de certaines molécules (exprimée en concentration) en réaction au stress physiologique de la souris.

Les échantillons sont alors analysés afin de produire un spectrographe mesurant combien des molécules données (les molécules qu'on recherche) sont présentes dans l'organisme de l'animal (un pour chaque condition expérimentale). A cette étape, seules des mesures indirectes révèlent la concentration des molécules présentes dans l'organisme. Un travail supplémentaire est nécessaire afin d'identifier les molécules présentes.

Après avoir collecté ces données numériques, nous revenons à des questions biologiques. Comment a réagi l'organisme du cobaye au stress ? La présence et la concentration de molécules dans les échantillons mettent en évidence des réactions spécifiques du métabolisme que nous voulons découvrir et étudier. Comme les concentrations varient d'une situation à une autre, le biologiste peut formuler des hypothèses sur les modes de combat de l'organisme contre le stress (prise de drogue / changement de régime alimentaire).

Le protocole qu'on a décrit produit un nombre important de cas de figure. Toutes les populations de souris (régime alimentaire normal ou anormal, avec ou sans prise de drogue) subissent 6 prises d'échantillons et 750 molécules sont alors recherchées dans chacune de ces situations. La complexité ne vient pas du volume (finalement assez modéré) des données, mais plutôt de la nécessité de comparer simultanément différentes paires de situations. En effet, nous voulons étudier des molécules présentant un profil qui diffère suivant des situations distinctes, nous construisons un scatterplot rassemblant les informations suivant deux situations. Une forte concentration de molécules dans les deux situations est bien évidemment intéressante. La question toutefois est de savoir si ce fort profil d'expression est observé dans toutes les situations. Un autre scénario intéressant est quand deux molécules ont systématiquement un profil inverse (forte concentration pour l'un et faible concentration pour l'autre, ou vice-versa, dans toutes les situations).

L'expérimentation nécessite donc que nous construisions une représentation incluant une série de scatterplots permettant à l'utilisateur de faire des comparaisons entre les différentes conditions d'un côté et d'observer l'effet de drogue au cours du temps. Pour ce faire, l'utilisateur voit une matrice de scatterplots. La figure 2.24 présente une telle matrice. Chacun des scatterplots est construit suivant une paire de situations. Chaque ligne correspond à un temps donné en abscisse (le temps après le début de l'expérimentation). L'ordonnée, quant à elle, croît de gauche à droite. Ainsi, pour la première ligne, les différents scatterplots sont construits avec en abscisse la concentration au temps $t = 0$. Seule l'ordonnée varie utilisant la concentration à $t = 32$, $t = 88$ ...

Chaque ligne/colonne correspond à différents intervalles de temps après le début de l'expérimentation - le temps augmente de gauche à droite. Ainsi, pour la première ligne, les différents scatterplots sont construits avec en abscisse la concentration au temps $t = 0$. Seule l'ordonnée varie utilisant la concentration à $t = 32$, $t = 88$ ... L'utilisateur repère une zone d'intérêt correspondant à des molécules dont le profil diffère des autres molécules (par exemple la zone s'éloignant de la diagonale dans le scatterplot "condition2 - condition3"). L'utilisateur peut alors sélectionner cette zone et voir le profil de ces molécules dans d'autres situations (dans les autres

scatterplots). La liste des molécules sélectionnées est affichée sur le côté droit de la matrice. Dans notre exemple, une seule molécule a été sélectionnée (la molécule 317).

L'application développée permet à l'utilisateur de sélectionner des molécules jugées intéressantes et d'observer la répartition de celles-ci suivant les différentes situations. Typiquement, l'utilisateur s'intéressant aux molécules ayant un profil étrange va sélectionner des zones qui s'éloignent de la diagonale (les molécules sur la diagonale étant des molécules qui n'ont pas de comportement différent suivant les situations) et observer leur comportement dans les autres situations. Une liste des molécules sélectionnées est fournie comme indication à l'utilisateur. De plus, l'utilisateur peut aussi intervenir sur le processus de création des scatterplots au moyen de glissières (slider) afin d'avoir plus ou moins de détails. L'utilisateur peut aussi rechercher une molécule particulière grâce à une fonction de recherche.

2.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode permettant de visualiser et d'interagir sur des données multidimensionnelles. L'objectif premier de cette méthode est de proposer une visualisation permettant de trouver des corrélations non linéaires dans les données par opposition à avoir une image globale de toutes les dimensions. Tirant profit de la perception multi-échelle de l'image, nous avons proposé une sélection facilitée de régions d'intérêts en structurant l'image. Ce travail a fait l'objet d'une publication en conférence internationale [67].

Notre méthode a été implémentée et testée en collaboration avec des biologistes qui l'ont jugée utile et complémentaire des autres techniques de statistiques plus classiques comme l'analyse en composante principale (ACP). En effet, les méthodes statistiques classiques comme l'ACP font ressortir des corrélations linéaires entre les variables. Notre outil permet toutefois de repérer des relations non linéaires si elles sont présentes. De plus, l'analyste peut s'intéresser, non pas aux corrélations, mais à la non corrélation entre les concentrations métabolites.

Cette méthode entre dans le paradigme "focus+contexte". En effet, la structuration de l'image en images de granularités différentes nous permet d'avoir différentes abstractions des données. Dans cette vue structurée, la sélection d'une zone donne plus de détails sur celle-ci et son rôle dans la matrice de graphes de corrélations. Notre approche vient s'ajouter aux travaux fondateurs de Becker et Cleveland [7].

Chapitre 3

Visualisation de données hiérarchisées

Un arbre ou une arborescence est une structure mathématique qui permet d'organiser les données de manière logique et hiérarchisée. Un dessin d'arbre permet une représentation de cette information. On a tous en tête la représentation de l'arbre généalogique de notre famille (voir figure 3.1). Dans cet exemple, il existe typiquement deux façons d'utiliser l'arbre : en plaçant une personne à la racine de l'arbre et en développant en direction des feuilles ses ancêtres ou au contraire à la racine l'aïeul et en direction des feuilles les descendants. L'organigramme d'une compagnie est souvent représenté par un diagramme où l'arbre décrit la hiérarchie formés par les différents collaborateurs. Un autre exemple commun d'utilisation d'arbre est la représentation des systèmes de fichiers, par exemple Windows Explorer. Un système de fichiers est composé de répertoires qui contiennent des sous-répertoires ... et enfin des fichiers. Cette organisation rend plus efficace la consultation et la manipulation des données. En informatique, cette structure a pris une très grande importance. Ainsi, on dispose d'une batterie d'algorithmes permettant de parcourir et de trier l'information à l'aide des arbres.

FIG. 3.1 – Représentation d'un arbre généalogique.

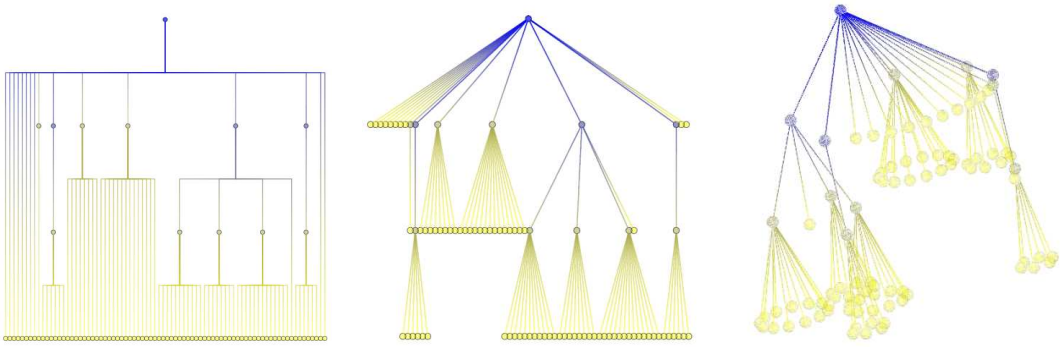
Dans ce chapitre, nous allons faire un état de l'art de différentes techniques per-

mettant la représentation et la manipulation de ces structures arborescentes. Deux approches sensiblement différentes seront présentées. La première mettant l'accent sur les relations entre les objets (Qui est le père de X dans la figure 3.1 ?) et l'autre donnant plus d'importance aux attributs des données (Combien d'homme y a t il dans la famille de la figure 3.1 ?). Enfin, la dernière section fera un état de l'art des techniques permettant la représentation de hiérarchies plus complexes.

3.1 Visualisation de hiérarchies arborescentes

3.1.1 Diagramme noeud lien

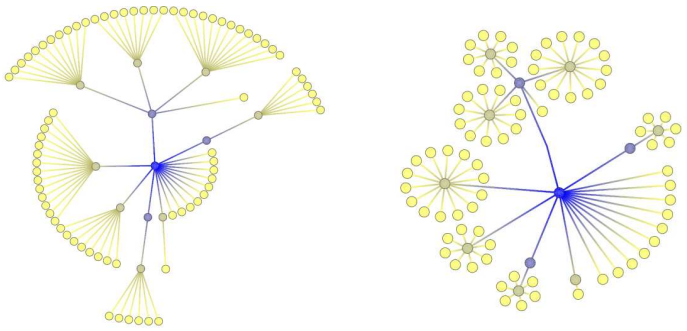
Une première représentation très intuitive de données relationnelles est le diagramme noeud lien. Les entités sont représentées par des noeuds, des sommets (le plus souvent représentés par des cercles, sphères, carrés ou encore cubes) et les relations entre ces entités sont représentées par des arêtes, des traits reliant les deux sommets. Pour des données hiérarchiques, le dessin de ce diagramme peut être orienté. En effet, si il existe une relation entre A et B où A est le père de B dans la hiérarchie alors on peut dessiner A au dessous de B comme c'est le cas dans l'arbre généalogique (voir figure 3.1), ou au dessus comme c'est le cas dans un organigramme, ou encore de gauche à droite comme c'est le cas dans la représentation de fichier sous Windows Explorer.



(a) Dendogramme (b) Walker [75, 50] (c) Cone Tree [20]

FIG. 3.2 – Représentation noeud lien de la structure de fichiers. Arbre comprenant 90 sommets. La couleur des sommets rend compte de la profondeur dans l'arbre (du bleu pour la racine au jaune pour les feuilles).

Les images de la figure 3.2 sont différentes représentations d'un même système de fichiers. Les différents algorithmes utilisés pour le dessin permettent de définir la position des éléments dans le plan (x, y, voire z) et le tracé des arêtes. Ces algorithmes tentent de respecter un certain nombre de contraintes d'ordre esthétique [5, 55]. Ainsi, les noeuds et les arêtes doivent être uniformément distribués. Les arêtes devraient avoir la même longueur et leur tracé devrait être en ligne droite, des structures isomorphes devraient être représentées de la même manière ou encore le croisement d'arêtes devrait être minimisé.



(a) Arbre Radial [28] (b) Arbre Ballon [38]

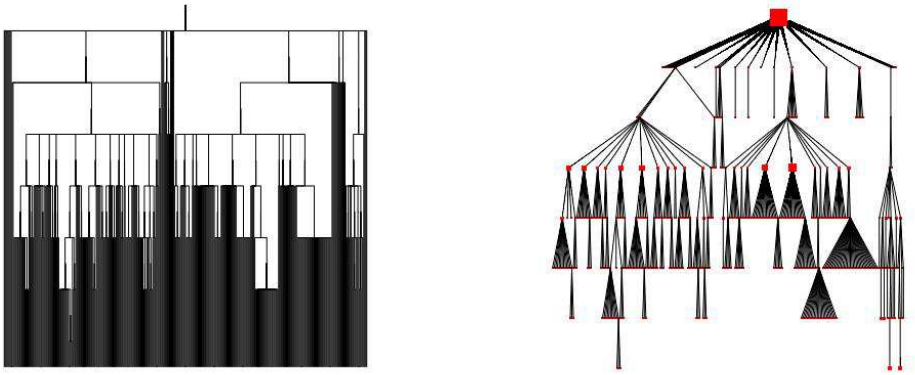
FIG. 3.3 – Autre représentation de la hiérarchie de la figure 3.2 avec des algorithmes circulaires.

La figure 3.2(a) montre une représentation sous forme de dendrogramme, toutes les feuilles de l'arbre sont placées sur un même niveau (en bas), les pères sont alors dessinés vers le haut. Dans cette représentation beaucoup d'espace est consacré à la hiérarchie, les sommets sont alors peu lisibles.

L'algorithme de Reingold et Tilford [75, 50] (voir figure 3.2(b)) est un bon exemple d'algorithme respectant ces critères. En effet, les structures isomorphes sont représentées de la même manière, la distance entre les sommets est quant à elle paramétrable par l'algorithme. Cet algorithme peut être modifié afin de produire un dessin de bas en haut ou encore de gauche à droite. Les sommets sont dessinés par niveaux suivant leur position dans la hiérarchie. Au premier niveau, la racine de l'arbre est dessinée. Les fils de la racine constituent le niveau 2, les fils des fils le niveau 3 ... . Dans la figure 3.2(b), la racine est en haut du dessin, les différents niveaux sont alors sur des lignes horizontales espacées verticalement de la racine. La figure 3.3(a) montre une variation des algorithmes précédents, le dessin radial [28]. Les sommets sont positionnés sur des cercles concentriques suivant leur profondeur dans l'arbre. La racine se trouve alors au centre du dessin. Un sous-arbre est dessiné dans un secteur du cercle, les secteurs ne peuvent se chevaucher. La figure 3.3(b) montre une autre variante possible, les fils d'un sommet sont dessinés sur un cercle centré sur le sommet formant ainsi un "ballon" autour du sommet [38].

Afin de représenter les relations entre les entités (sommets) de la hiérarchie, un espace important est nécessaire entre les différents niveaux. Des attributs des données peuvent être représentés par la taille, la couleur, la forme ou encore la texture des sommets ou des arêtes. Dans de tels diagrammes, la lecture de ces attributs peut être difficile même pour de petites hiérarchies. La hiérarchie utilisée pour les images de la figure 3.2 est un arbre de taille modeste. La figure 3.4 montre un graphe de plus grande importance. Notez comme il devient difficile de distinguer les sommets et à fortiori leurs attributs.

La section suivante introduit des méthodes visant à résoudre ce problème.



(a) Dendogramme (b) Walker

FIG. 3.4 – Différents algorithmes de dessin de hiérarchie avec un graphe plus important : 456 sommets et 455 arêtes. Hiérarchie représentant la structure de fichier.

3.1.2 Approche par pavage de l'espace : TreeMap

Les approches par pavage de l'espace ("Space filling") sont des techniques de visualisation qui mettent l'accent sur les attributs des données. Les éléments sont représentés par une forme géométrique (souvent le rectangle) et sont placés les uns à côté des autres (comme quand on pose des carreaux sur le sol d'une cuisine) dans l'espace attribué à la visualisation.

La mosaic display [33] est l'une de ces approches. Elle tente d'utiliser l'aire des rectangles pour représenter des quantités statistiques. L'information est alors représentée par des rectangles dont la hauteur et la largeur rendent compte d'attributs des données. L'aire de ces rectangles permet de visualiser l'importance d'une propriété. La figure 3.5 montre un exemple illustrant la distribution de la population en fonction de la couleur des cheveux et de la couleur des yeux.

La TreeMap, introduite par Shneiderman [81], est l'une des premières techniques utilisant le pavage du plan afin de représenter des hiérarchies d'information sur un espace 2D. Cette méthode représente 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.

L'algorithme de construction d'une TreeMap est un algorithme récursif. Chaque sommet de l'arbre est représenté par un rectangle qui est subdivisé soit horizontalement soit verticalement pour y représenter le sous-arbre de ses fils. A chaque appel de l'algorithme, un niveau de la hiérarchie est alors dessinée.

L'algorithme de Schneiderman donne typiquement une image comme celle de la figure 3.7 où on perçoit bien l'alternance des divisions successivement horizontales et verticales.

Les noeuds internes à la hiérarchie sont apparents à travers l'emboîtement des cellules et peuvent être eux aussi "équipés" de différents attributs comme la taille

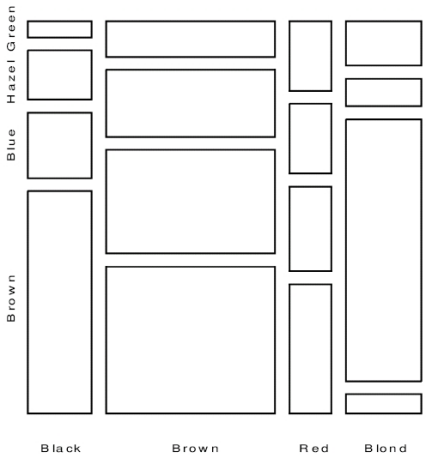


FIG. 3.5 – Mosaic display [33].

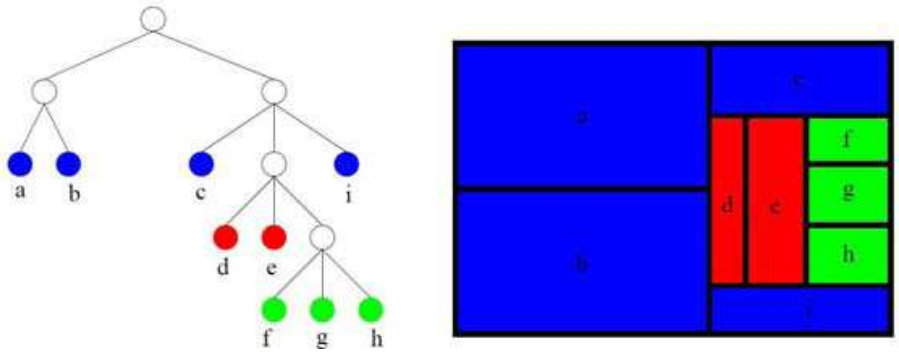


FIG. 3.6 – Construction d'une TreeMap.

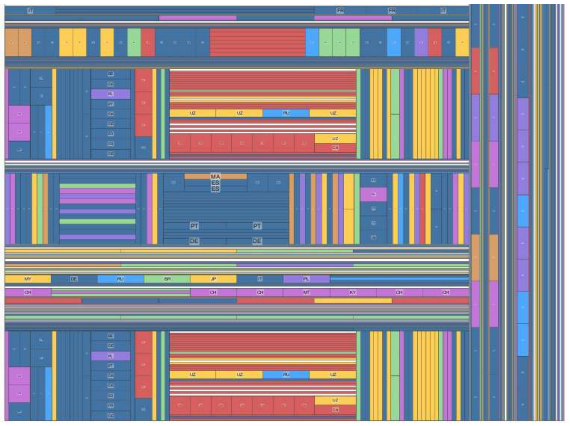


FIG. 3.7 – Représentation sous forme de TreeMap avec l'algorithme de dessin de Shneiderman [81].

ou la couleur (comparer les images de la figure 3.6).

Cette approche diffère radicalement des représentations classiques noeuds liens des arbres (voir [5, 55]) 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 sa facilité d'utilisation. Une meilleure interactivité [23] et une plus grande versatilité [89] en font un bon choix pour la mise en place de systèmes de visualisation de données hiérarchiques (arborescentes dans le cas de [89]). Les TreeMaps sont utilisées pour visualiser un large éventail de hiérarchie incluant des portefeuilles d'actions [54], des informations d'actualité [94], des blogs [90], des données économiques [89], des matches de tennis [51], des collections de photos [10], l'utilisation des systèmes de fichiers [81, 95]. Le lecteur peut consulter le site internet [1] où Schneiderman décrit l'historique de la TreeMap. Ce site fournit un aperçu des différentes applications et extensions proposées à la TreeMap.

Voici l'algorithme proposé par Shneiderman dont la formulation est précisément celle donnée par Schneiderman dans son papier fondateur. (*PaintRectangle est une routine pour dessiner à l'écran un rectangle plein utilisant une certaine couleur, Size retournant la taille d'un sommet. Le premier appel de l'algorithme est le suivant : TreeMap(root, P, Q, 0, color) avec root la racine de l'arbre à dessiner, P les coordonnées du coin haut gauche, Q les coordonnées du coin bas droit et color une couleur*).

Algorithme 1 : TreeMap

Input : root : la racine de l'arbre ou du sous-arbre
P, Q : tableaux contenant les coordonnées des deux coins opposés d'un rectangle
axis : variable entre O et 1 indiquant le sens de la division (verticale ou horizontale)
color : indique la couleur à utiliser pour le rectangle courant
Output : Un dessin de l'arbre
PaintRectangle(P, Q, color) – paint full area
width := Q[axis] - P[axis]
for *i=0 to numChildren* **do**
    Q[axis] := P[axis] + (Size(child[i])/Size(root)) × width
    TreeMap(child[i], P, Q, 1-axis, color)
    P[axis] := Q[axis]

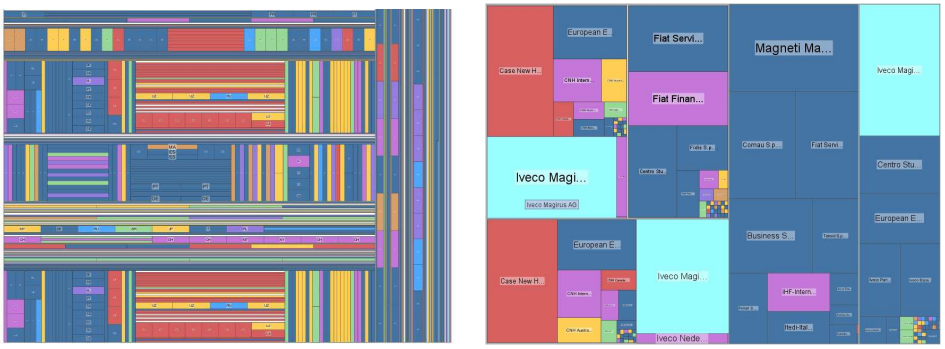
L'algorithme original de Schneiderman peut donner des visualisations qui souffrent de certains défauts rendant difficilement perceptible la différence d'aire entre les cellules ou encore la structure d'arbre. En effet, la comparaison de deux cellules dans la TreeMap peut être difficile. Cela est dû à l'aspect effilé des cellules : le ratio largeur / hauteur du rectangle qui représente une cellule était très éloigné de 1. Une cellule dont le ratio est de 1 est un carré. Sachant qu'il est plus facile de comparer

[1] http ://www.cs.umd.edu/hcil/treemap-history/

deux carrés entre eux, on souhaite que le ratio moyen des cellules dans la TreeMap s'approche de 1.

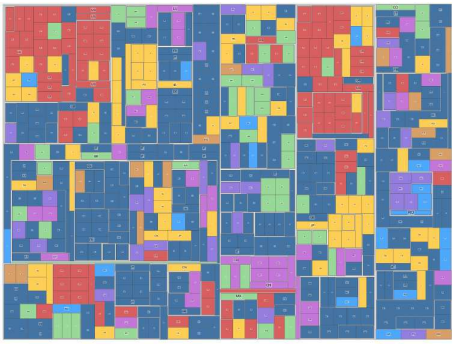
De nombreux auteurs ont cherché à améliorer l'algorithme de départ proposant des modifications sur le calcul des cellules ou encore sur l'apparence de celles-ci. Une première amélioration porte sur le ratio hauteur / largeur des cellules. Il est utile de rechercher à produire des cellules dont le ratio est proche de 1, permettant ainsi de comparer l'aire des rectangles [17] (voir figure 3.8). Le bénéfice de l'algorithme "Squarified" TreeMap [17] est de faciliter la comparaison des aires des cellules.

D'autres algorithmes interviennent sur l'ordre dans lequel les cellules sont placées dans la TreeMap. Notez que dans la figure 3.8, l'algorithme de "Squarified" place les cellules de grande taille en haut à gauche et les cellules de petite taille se retrouve en bas à droite. Dans [10], les auteurs proposent une technique permettant de préserver certaines contraintes d'ordre tout en fournissant un ratio acceptable (voir figure 3.8).



(a) Schneiderman [81]

(b) Squarified [17]



(c) Ordered [10]

FIG. 3.8 – Dessin d'un arbre suivant différents algorithmes de TreeMap. Comparez le ratio hauteur / largeur des cellules. La comparaison des cellules entre elles est facilitée dans les TreeMaps (b) et (c).

Il est cependant plus facile de comparer deux polygones entre eux et d'autant plus facile s'ils s'approchent des cercles (deux cercles sont plus faciles à comparer entre eux que deux carrés) [3]. Ainsi, Balzer propose les voronoi TreeMaps, [3] où les cellules ont des formes variées qui permettent d'approcher la forme des cercles (voir figure 3.9).

Une autre extension des TreeMap avec des cellules non rectangulaires est la

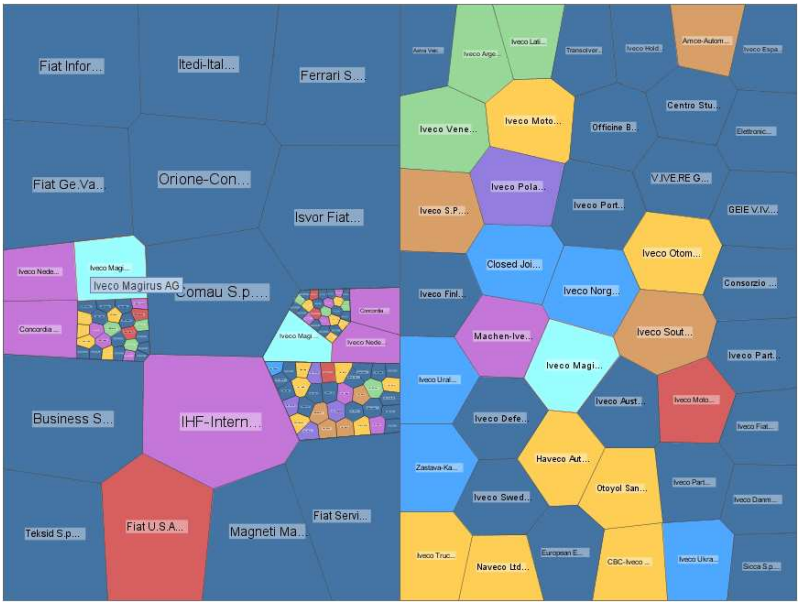


FIG. 3.9 – Voronoi TreeMap [3]. Les cellules de la TreeMap ne sont pas forcément des rectangles mais plutôt des polygones qui s'approchent de cercles. Ces formes permettent une meilleure comparaison des cellules entre elles.

TreeMap circulaire [95]. La figure 3.10 en montre un exemple. D'autres extensions permettent une représentation en trois dimensions de la TreeMap [12, 13, 76].

Dans Grokker [2], les éléments sont représentés par des cercles offrant ainsi une meilleure comparaison (voir figure 3.10). Il est à noter que dans cette dernière proposition, un espace important est laissé entre les cercles et au centre du dessin.

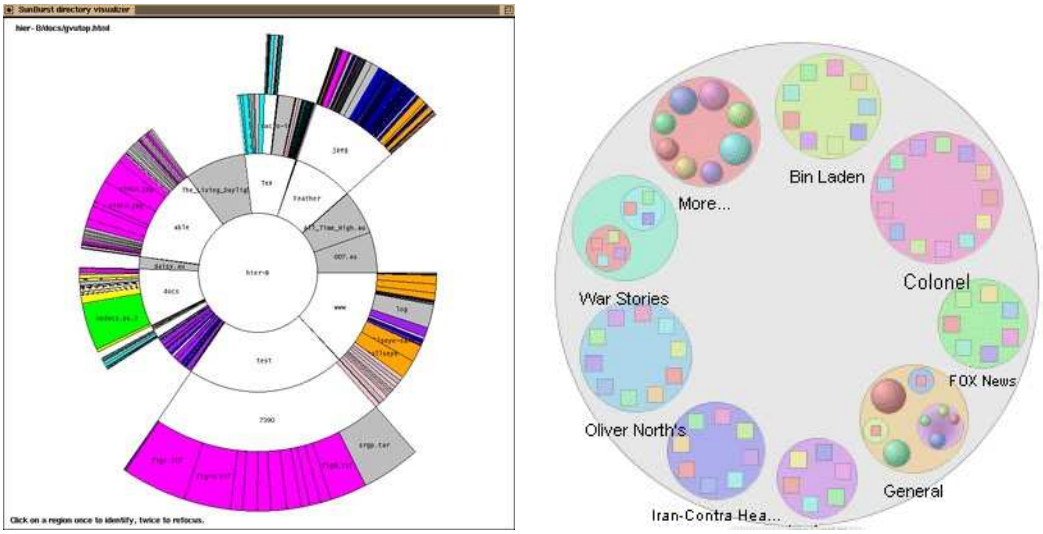
D'autres améliorations portent non plus sur la forme des cellules mais plutôt sur la perception de la structure hiérarchique, c'est-à-dire sur l'emboîtement des cellules. Afin de permettre une meilleure interprétation de la structure même des données, mettre en évidence la notion de contenant et contenu, on peut augmenter l'espace entre les cellules (voir figure 3.11). L'idée de donner à la TreeMap une meilleure perception des bordures des cellules a été suggérée par Watterberg et Fisher [93] et cadre avec leur étude de la perception multi-échelle d'une image. Cette méthode utilise les principes de discrétisation des Gesalt Carton [93].

Dans [86], les auteurs, par un artifice graphique, augmentent la lisibilité de la structure. Ils ajoutent comme des coussinets sur les cellules de la TreeMap (voir figure 3.12). La mise en évidence de la structure peut aussi être faite par un décalage, les cascades TreeMap [62]. La figure 3.13 nous en montre un exemple. Cette méthode permet notamment de prendre moins d'espace. En effet, la place qui est nécessaire pour montrer la notion de contenu est plus importante qu'un simple décalage.

L'espace laissé entre les cellules afin de faire ressortir la structure peut être coloré pour accentuer l'effet (voir figure 3.14).

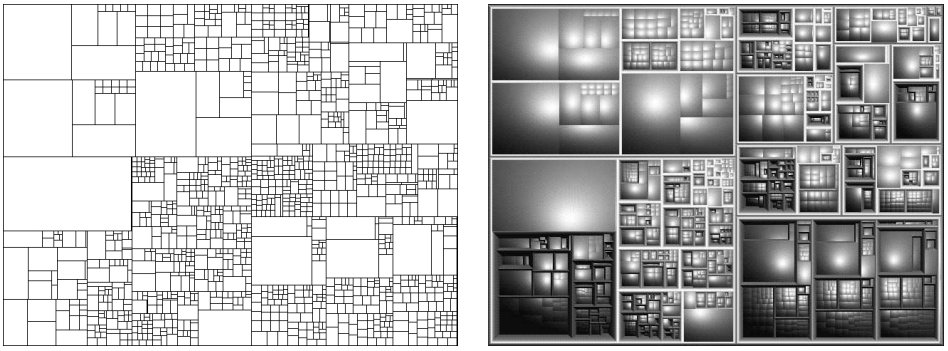
Toutes ces améliorations interviennent à différents niveaux dans l'algorithme de construction de la TreeMap. Voici une reformulation plus générique de l'algorithme

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



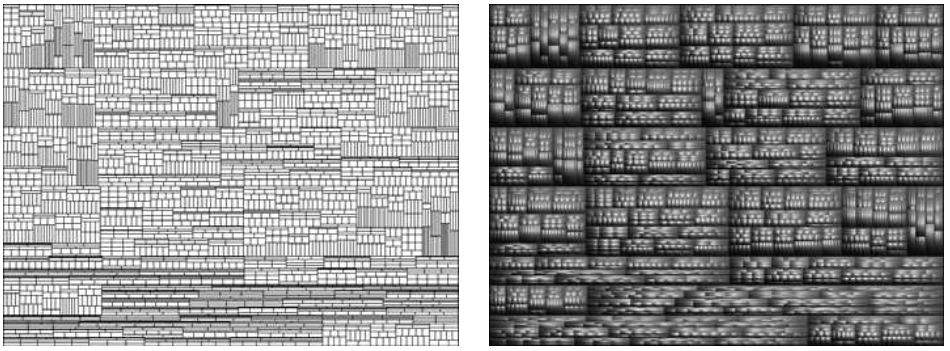
(a) SunBurst (b) Grokker

FIG. 3.10 – (a) SunBurst, (b) Grokker (application commerciale) : technique de représentation par pavage de cercles. La comparaison des éléments entre eux est facile mais beaucoup d'espace est perdu entre les cercles.



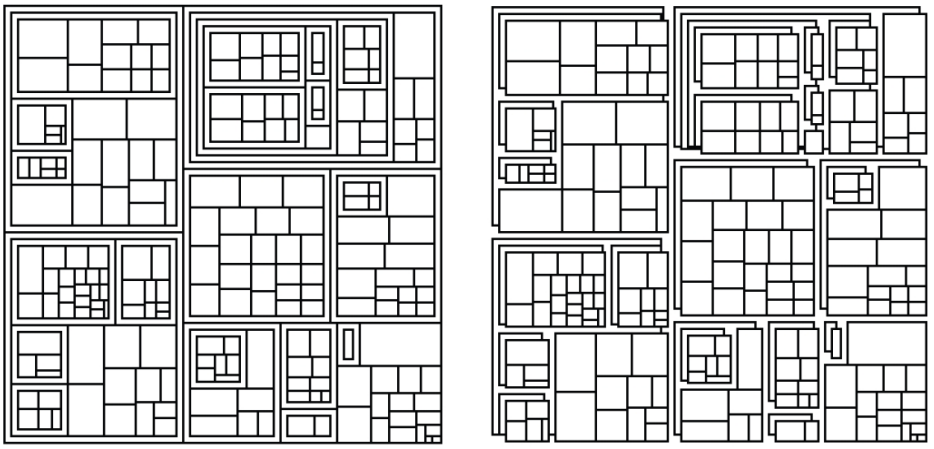
(a) "Squarfied" TreeMap (b) TreeMap avec cushion et espacement entre les niveaux

FIG. 3.11 – Différentes représentations d'une même hiérarchie suivant différents algorithmes de dessin ("Squarified" TreeMap [17], "gap").



(a) TreeMap original (b) Cushion TreeMap

FIG. 3.12 – Différentes représentations d'une même hiérarchie suivant différents algorithmes de dessin ("schneiderman" [81], "cushion" [86]).



(a) Squarified TreeMap (b) Cascade TreeMap

FIG. 3.13 – Représentation d'une même hiérarchie suivant les algorithmes Squarified TreeMap et Cascade TreeMap [62].

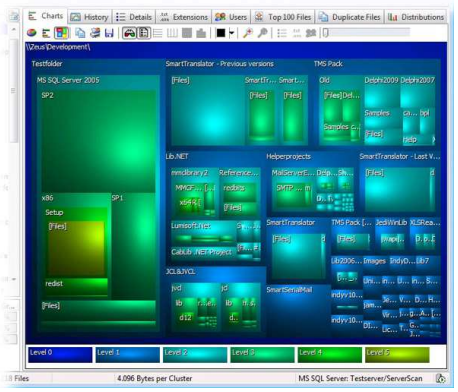


FIG. 3.14 – Coloration des espacements dans la TreeMap.

de construction d'une TreeMap. La généricité de cet algorithme tient à la brièveté de son expression et à la forme associée à un sous-arbre. En effet cette forme n'est pas nécessairement un rectangle calculé comme spécifié par Schneiderman voire n'est pas un rectangle du tout. Elle peut être une forme géométrique quelconque.

Algorithme 2 : TreeMap

Input : un noeud n, une forme géométrique f
Output : Le dessin de n et de ses fils directs
computeTreemapSubdivision(n, f)
foreach *fils c de n* **do**
    updateShape(c)
    TreeMap(c, getShape(c))

Les différentes améliorations exposées précédemment implémentent ou spécifient les fonctions de cet algorithme. Les améliorations portant sur la forme des cellules se situent au niveau de la fonction computeTreeMapSubdivision ("Squarified", "Ordered" et "Voronoi") et les améliorations portant sur la perception de la hiérarchie sont pour leur part contenues dans updateShape (espace entre les cellules ou décalage - cascade).

3.1.3 Vue combinant deux représentations

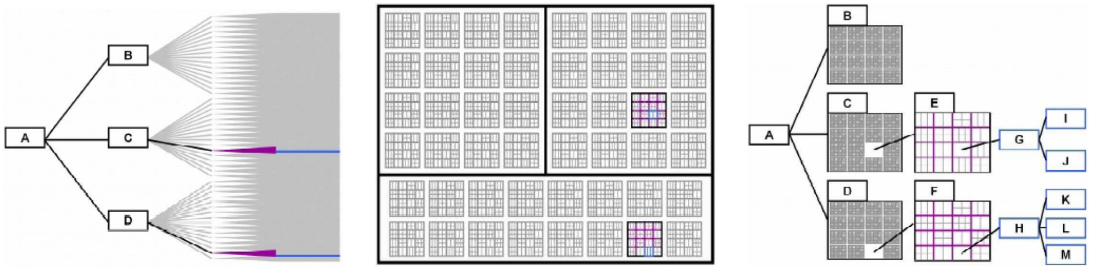
Une représentation noeud lien explicite la structure même de la hiérarchie. La lecture des attributs associés aux données (représentés par la taille, la couleur, la forme ou encore une étiquette (label)) est alors plus difficile. Ceci est à opposer à une représentation par pavage de l'espace (TreeMap) où les attributs sont mis en évidence mais la structure peut alors être moins intuitive. Une solution serait alors de combiner les atouts de ces deux types de visualisation.

Elastic Hierarchies

Dans "Elastic Hierarchies : Combining TreeMaps and Node-Link Diagrams" [96], les auteurs proposent de combiner une représentation sous forme de TreeMap et un diagramme noeud lien (voir figure 3.15). L'utilisateur définit des points d'intérêts. Ainsi, les chemins entre les centres d'intérêts et la racine de la hiérarchie seront explicités avec des sommets et des arêtes. Le reste de la hiérarchie sera présenté sous forme de TreeMap. Dans cette représentation, on minimise la place laissée à la hiérarchie. Nous mentionnons cette technique car elle exploite la notion de TreeMap, mais elle met en jeu aussi la notion de "focus+contexte". Nous reviendrons sur cette visualisation dans le chapitre 6 où nous nous penchons sur l'interaction.

3.2 Visualisation de hiérarchies générales

Les visualisations de hiérarchies arborescentes peuvent être étendues à des structures plus complexes. On va s'intéresser maintenant à des visualisations qui cherchent à représenter des structures plus générales. On souhaite ici mettre l'accent sur des techniques de visualisations où l'arbre est utilisé pour piloter la visualisation. Nous



(a) Diagramme noeud lien (b) TreeMap (c) Hiérarchie élastique

FIG. 3.15 – Différentes représentations d'une même hiérarchie : vue noeud lien, TreeMap et vue élastique [96].

avons été amenés à explorer chacune d'elles avant de converger vers une visualisation pour les graphes orientés acycliques (DAG).

Extension des diagrammes noeud lien aux graphes orientés acycliques

Les graphes orientés acycliques (DAG) peuvent être vus comme la généralisation des arbres où un élément peut avoir plusieurs antécédents. Le diagramme noeud lien peut être étendu à ce type de hiérarchies plus complexes. L'algorithme de dessin procède alors comme suit : au premier niveau (en haut du dessin par exemple), les sommets sources (sommets n'ayant pas de prédécesseur) sont dessinés. Les autres sommets sont alors placés suivant la longueur du plus long chemin les séparant d'une source. Dans la figure 3.16, on voit clairement certains sommets qui possèdent plus d'un parent (par exemple les sommets en bas à gauche).

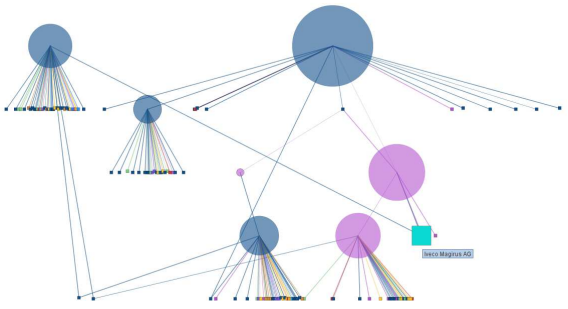


FIG. 3.16 – Diagramme noeud lien d'un DAG [29], avec un attribut mis en évidence par la taille et la couleur des sommets.

La mise par niveau des noeuds est une opération simple. La difficulté dans le dessin des DAG réside dans l'ordre dans lequel les sommets sont placés sur chaque niveau. En effet, il faut minimiser le croisement d'arêtes pour augmenter la lisibilité du diagramme. Une méthode possible repose sur le placement d'un arbre couvrant. Une réorganisation successive des noeuds par couche de haut en bas est alors effectuée (voir [29, 35]). Une autre technique consiste à affecter à chaque noeud le

barycentre des coordonnées de ses voisins [85]. Nous aurons l'occasion de revenir sur le dessin de DAG dans le chapitre suivant.

TreePlus

Dans "TreePlus : Interactive Exploration of Networks with Enhanced Tree Layouts" [59], les auteurs proposent de visualiser un graphe grâce à des algorithmes de dessin d'arbres. Pour cela un arbre couvrant est extrait du graphe : seul cet arbre sera affiché à l'écran. En sélectionnant un noeud, ses voisins sont colorés (bleu pour les fils et rouge pour les pères). Le noeud peut avoir des voisins non visibles à l'écran, un panneau latéral indique alors les voisins visibles et non visibles du noeud (voir figure 3.17).

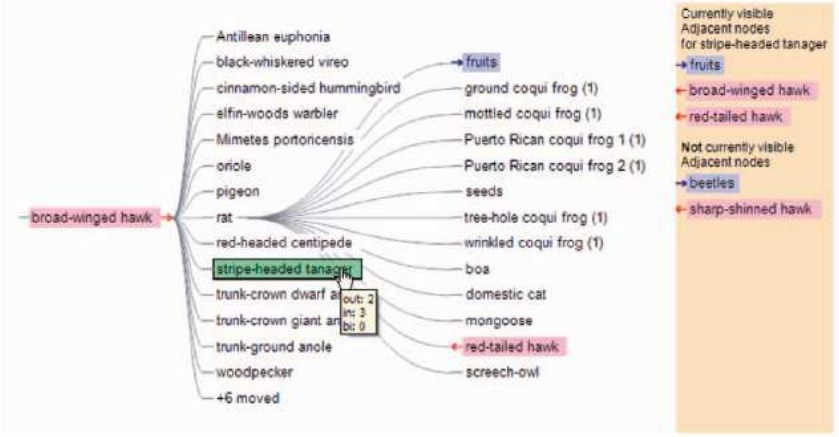


FIG. 3.17 – Représentation d'une hiérarchie avec TreePlus [59].

Overlaying Graph et Bundle Edge

D'autres techniques utilisent la TreeMap afin de visualiser des graphes [31, 47]. La première étape de ces méthodes consiste à extraire un arbre couvrant du graphe. Un arbre couvrant représente alors le squelette du graphe. Le choix (calcul) de cet arbre est très important. En effet, deux arbres couvrant différents donneront des dessins très différents l'un de l'autre. Cet arbre couvrant est alors dessiné à l'écran en utilisant un algorithme de dessin de TreeMap. Les différentes arêtes n'ayant pas été sélectionnées dans l'arbre couvrant sont alors rajoutées sur le dessin. Le dessin de ces arêtes peut se faire par de simples traits [31] ou suivre une technique plus élaborée. Dans [47], les arêtes sont regroupées en faisceaux augmentant ainsi la lisibilité du graphe. Les figures 3.18 et 3.19 montrent différentes variantes de cette technique.

Diagramme Arc et ArcTree

Dans [92], les auteurs proposent une représentation d'un graphe à l'aide d'arc. Tous les sommets sont dessinés sur une même ligne et les arêtes sont dessinées en

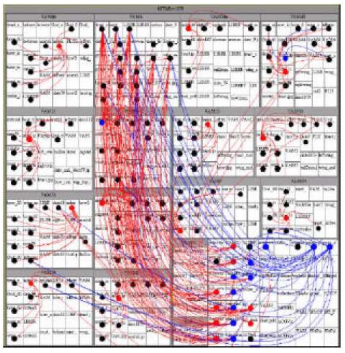


FIG. 3.18 – Dessin de graphe utilisant un arbre couvrant (squelette du graphe). L'arbre est dessiné grâce à un algorithme de TreeMap. Les arêtes supplémentaires sont alors rajoutées au dessin [31].

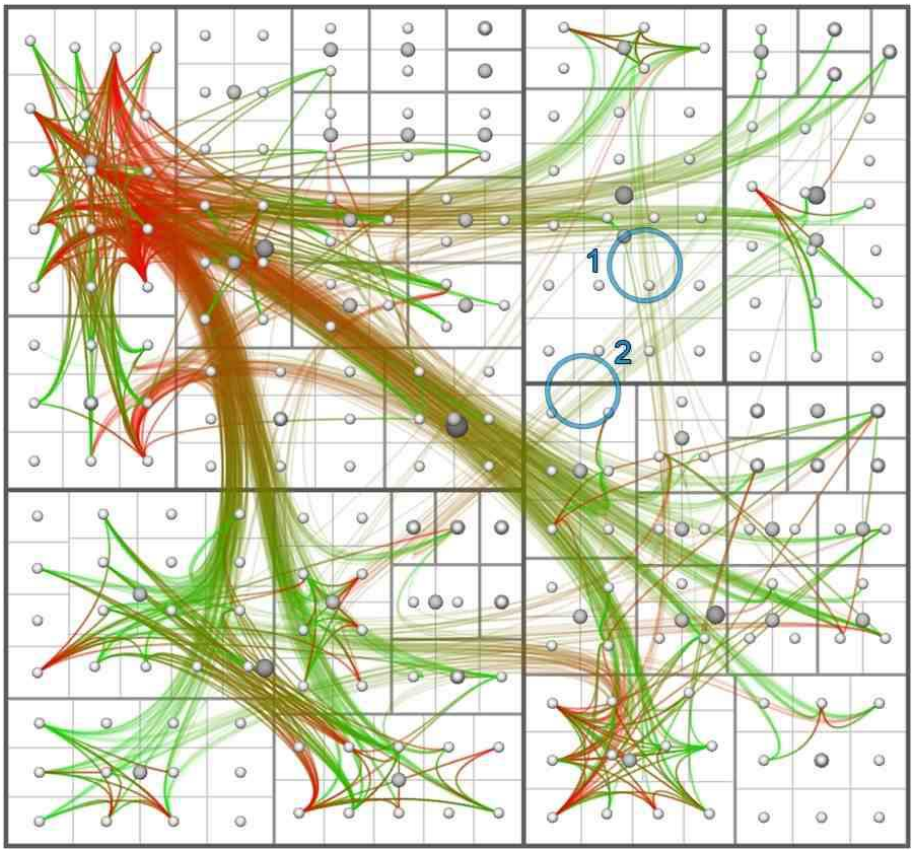


FIG. 3.19 – Dessin de graphe utilisant un arbre couvrant (squelette du graphe). L'arbre est dessiné grâce à un algorithme de TreeMap. Les arêtes supplémentaires sont alors rajoutées au dessin. Les arêtes sont regroupées en faisceaux pour augmenter la lisibilité du dessin [47].

arc au dessus ou en dessous de cette ligne (voir figure 3.20). On peut aussi regrouper des arêtes pour créer des faisceaux.

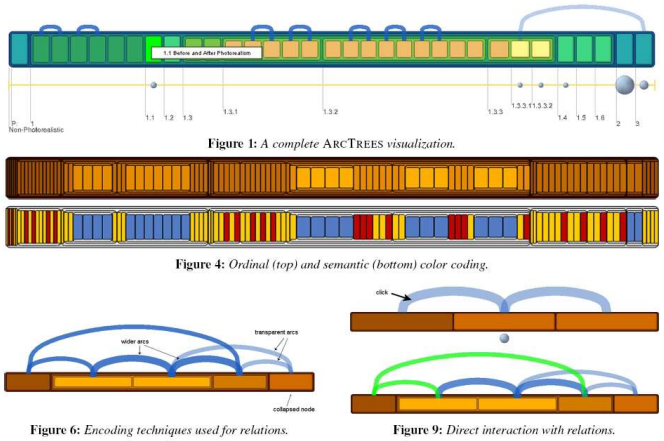


Figure 1: *A complete* ArcTrees *visualization.*

Figure 4: *Ordinal (top) and semantic (bottom) color coding.*

Figure 6: *Encoding techniques used for relations.*

Figure 9: *Direct interaction with relations.*

Fig. 3.20 – Arc Tree [72].

Dans [72], les auteurs reprennent cette idée pour représenter des graphes. Un arbre est dessiné à l'aide d'une TreeMap et les arêtes supplémentaires sont dessinées en arc (voir figure 3.20). La TreeMap, ici, est dessinée sur une seule dimension

3.3 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les techniques de représentation de graphes. Par un soucis de concision, cet état de l'art s'est volontairement concentré sur les visualisations de hiérarchies. Pour cause, le cas d'étude qui a principalement retenu notre attention dans cette thèse s'interresse à la visualisation et la description de hiérarchies.

Les jeux de données étudiés sont issus d'une collaboration avec des géographes. Les géographes s'intéressent aux réseaux de filiation de compagnies multinationales. La structure arborescente dans ce type de réseau est très forte. Cependant cette structure n'est pas forcément un arbre mais peut être un DAG. A travers ces objets, les géographes se posent des questions sur les phénomènes sociaux économiques présents.

Les techniques presentées dans ce chapitre ne répondent que partiellement aux problèmes que le géographe se pose. Dans le chapitre suivant est présentée une nouvelle technique permettant de traiter ce type de structure.

Chapitre 4

Exploration de relations d'héritage multiple : conception du DAGMap et étude d'utilisabilité

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".

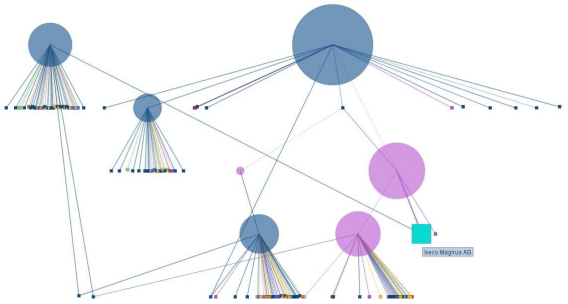


FIG. 4.1 – Dessin noeud lien classique de DAG, avec des attributs mis en évidence par la taille et la couleur des sommets.

Les relations d'héritage entre entités (objets, sociétés, etc) peuvent être décrites formellement par un graphe $G = (V, E)$ où V est l'ensemble des sommets et E l'ensemble des arêtes. 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 sont un défi en soi. Une bonne représentation d'une telle structure permet une meilleure analyse du problème qu'elle modélise, facilitant ainsi la prise de décisions. Par exemple,

un bon diagramme des classes d'un programme permet de visualiser l'architecture mise en place. L'identification et localisation d'éventuels problèmes d'incohérence sont facilitées. De même, la visualisation des relations d'héritage entre une société mère et ses filiales permet de mettre en évidence des stratégies de gouvernance. Comme on l'a déjà mentionné dans le chapitre précédant, ces types de graphes généralisent les structures arborescentes. Contrairement aux arbres pour lesquels il y a énormément d'algorithmes qui donnent des représentations assez lisibles, les DAGs posent un autre problème de part l'existence d'arêtes transversales. En effet, les sommets peuvent avoir plusieurs antécédents dans la hiérarchie. Cela peut impliquer des croisements d'arêtes diminuant la lisibilité du dessin.

Les algorithmes de dessin de DAGs forment une branche importante de la littérature de dessin de graphes (Graph Drawing [5, 55]). Ces algorithmes dessinent le plus souvent les sommets sur différents niveaux (voir figure 4.1). 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* (voir figure 4.1). La qualité et la lisibilité de ces représentations noeuds liens sont le plus souvent mesurées en fonction de leur capacité à éviter le croisement d'arêtes [40]. La section 3.2 du chapitre précédent présentait les principaux algorithmes qui permettent de représenter les DAGs.

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 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 taille. 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 précieux espace vide entre les niveaux. Les approches de pavage [81] 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. Cette technique est illustrée par un cas d'étude. Une section consacrée à l'évaluation de la méthode suit et enfin une conclusion.

4.1 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 (voir figure 4.2). On applique alors l'algorithme de dessin des TreeMaps sur cet arbre. 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).

Plus formellement, soit $G = (V, E)$ un DAG. A chaque noeud $v \in V$, soit $F(v)$ l'ensemble des pères de v dans G. Autrement dit, $F(v) = \{u \in V | (u, v) \in E\}$. Soit $D(v)$ le sous-graphe formé de tous les noeuds qu'on peut atteindre depuis v en allant vers le bas, incluant v lui même. Le sous graphe $D(v)$ est lui même un DAG qui peut être transformé en un arbre $T(v)$ en appliquant récursivement l'algorithme. Pour chaque noeud père $u \in F(v)$, on clone le sous arbre $T(v)$ en un arbre distinct $T_u(v)$ et on le rattache sous u. Ce faisant, les noeuds $u \in F(v)$ ont alors des descendants uniques (en ce qui concernant v).

Réciproquement, soit $T = (V, E)$ un arbre étiqueté tel que $\forall u \in V$, $l(u)$ est son étiquette. T est tel que si deux noeuds $u, v \in V$ ont la même étiquette $l(u) = l(v)$ alors les sous arbres $T_u = (V_u, E_u)$ et $T_v = (V_v, E_v)$ venant de u et v sont isomorphes et les noeuds correspondants ont la même étiquette.

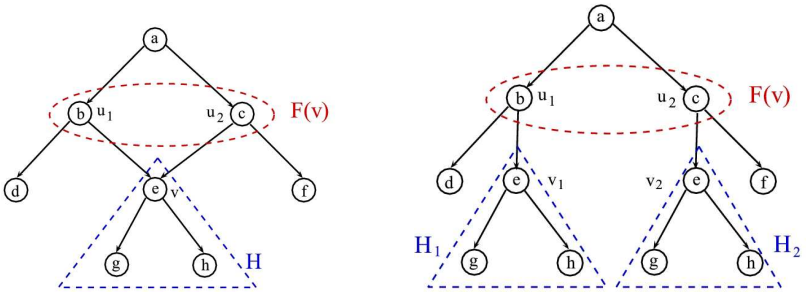


FIG. 4.2 – Un arbre étiqueté est obtenu du DAG par duplication des sommets (étiquetés) avec des ancêtres multiples.

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 (voir figure 4.1)). Il est à noter, comme c'est le cas dans DynaDags [70], que la plupart des techniques traitant des DAGs souffrent de cette limitation (voir aussi [66, 40]). Cette technique va être utilisée dans le cadre d'une exploration interactive "focus+contexte" (voir section 6). L'utilisateur spécifiant une zone d'intérêt du DAG, une partie du DAG peut être filtrée. Le DAGMap est alors construit en dépliant le DAG filtré en un arbre dont la taille est gardée sous contrôle. 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 la figure 4.3, l'utilisateur a sélectionné sur une cellule, la

cellule et les cellules correspondantes sont mises en évidence par une coloration turquoise.

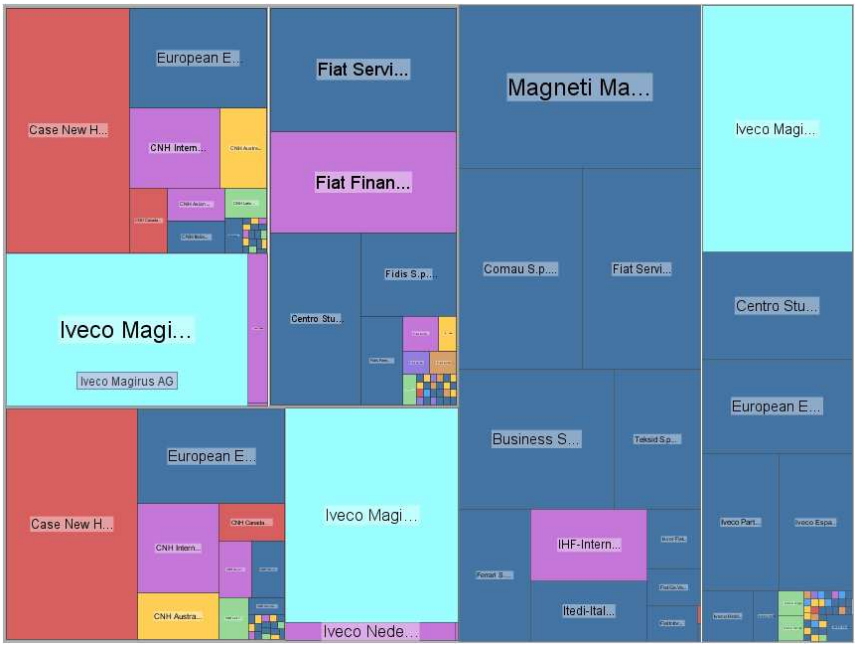


FIG. 4.3 – Le DAGMap permet de visualiser un DAG au travers d'une TreeMap (ici "Squarified" TreeMap [17]). Lors d'un clic sur une cellule, les cellules correspondant à la même entité sont sélectionnées – voir les cellules turquoise avec une aide contextuelle.

Un avantage naturel de la TreeMap par rapport à une vue noeud lien est la lisibilité des attributs des données. La figure 4.4 présente deux représentations d'un même DAG. À gauche la vue noeud lien et à droite le DAGMap. Dans ces représentations, la couleur des sommets est fonction d'un attribut des données. Même si le DAG n'est pas dense, notez comme il est plus facile de distinguer la couleur des cellules dans le DAGMap. En effet, la vue noeud lien souffre ici de chevauchement de sommets, ce qui n'est pas le cas du DAGMap.

Il y a une correspondance formelle entre le DAG et le DAGMap comme le montre la construction algorithmique ci-dessus. Cette correspondance contient de nombreuses informations. Certains motifs structuraux, facilement repérables dans la vue noeud lien, peuvent être repérés aussi facilement dans le DAGMap. Par exemple, dans la figure 4.4, on remarque un ensemble de sommets feuilles directement connectés à une source. Ces sommets de petite taille, sont présents dans le DAGMap et forme la mosaïque de couleurs dans le coin en bas à droite. Ainsi, une partie de la structure est préservée dans la représentation DAGMap.

La figure 4.5 indique la place de la construction du DAGMap dans le processus de visualisation. La vue combinée implique la construction de deux structures visuelles : une pour le DAGMap et une pour la vue noeud lien.

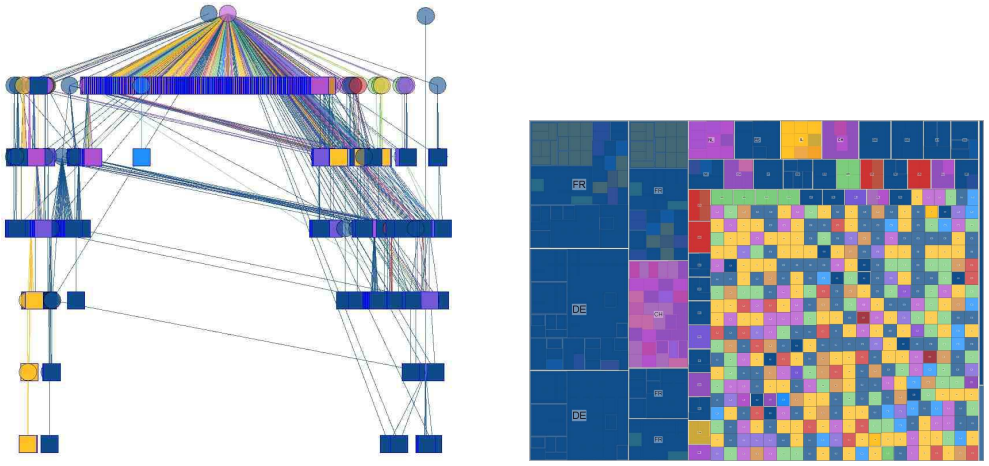


FIG. 4.4 – Vue combinée d'un DAG décrivant les liens entre Nestlé et ses filiales.

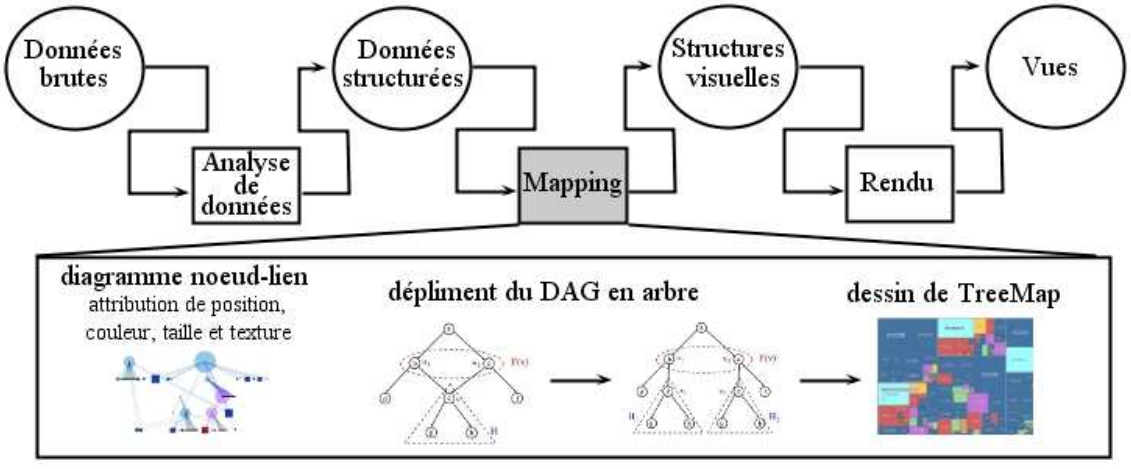


FIG. 4.5 – Processus de visualisation du DAGMap.

4.2 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. Une interaction spécifique est alors développée. Le DAGMap est construit à partir d'un DAG qu'on déplie en un arbre. On définit une "bande" comme étant l'ensemble des sommets se trouvant à un même niveau dans cet arbre. La notion de "couverture" fait référence au lien père fils dans l'arbre. Nous permettons à l'utilisateur de visualiser comment les sommets ancêtres "couvrent" les cellules dans le DAGMap. L'utilisateur peut définir une "bande" rassemblant tous les éléments à un certain niveau dans le DAG déplié. 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 leur voisinage. Cette interaction s'est révélée fondamentale pour nos utilisateurs.

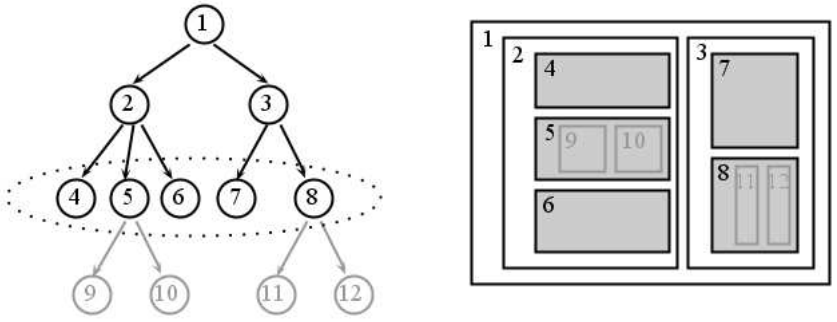


FIG. 4.6 – L'utilisateur peut définir une bande et la visualiser sur le DAGMap en tant que cellules colorées superposées.

Cette notion de "bande" est facile à comprendre dans une représentation noeud lien. En effet, on se déplace par niveau de façon verticale. Ce déplacement vertical dans l'arbre correspond à la notion d'emboîtement dans la TreeMap (voir figure 4.6). La sélection d'une "bande" consiste alors à placer un voile (plus ou moins fort) sur la boîte "couvrant" ainsi les boites imbriquées (le sous-arbre).

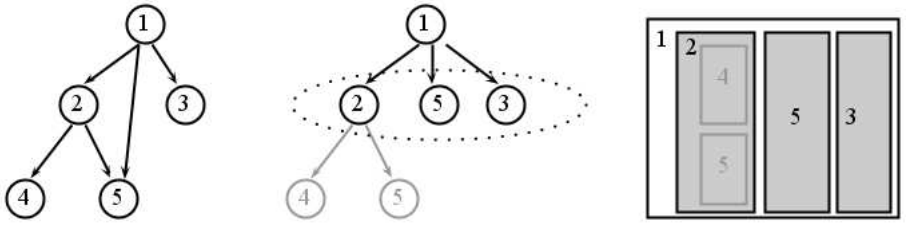


FIG. 4.7 – On peut définir une "bande" et la visualiser sur le DAGMap en tant que cellules colorées superposées.

C'est cette notion de "bande" qu'on va utiliser en l'appliquant à l'arbre calculé

à partir du DAG. Un sommet au niveau k dans le DAG va se retrouver au niveau k dans l'arbre (DAG déplié). Il peut toutefois se retrouver à un niveau inférieur. En effet, le niveau d'un sommet dans le DAG est fonction de la longueur du plus long chemin entre lui et une source, mais il peut exister plusieurs chemins plus courts donnant lieu à une duplication du sommet dans l'arbre déplié. Dans la figure 4.7, le sommet étiqueté "5" apparaît aux niveaux 2 et 3 dans l'arbre déplié. En variant la hauteur de la "bande", on permet à l'utilisateur de voir si un élément se retrouve à différents niveaux dans la hiérarchie. Cela peut être interprété en terme d'importance relative selon les endroits où l'élément est impliqué. La figure 4.8 illustre l'utilisation de la "bande" dans un cas réelle qui sera présenté à la section suivante.

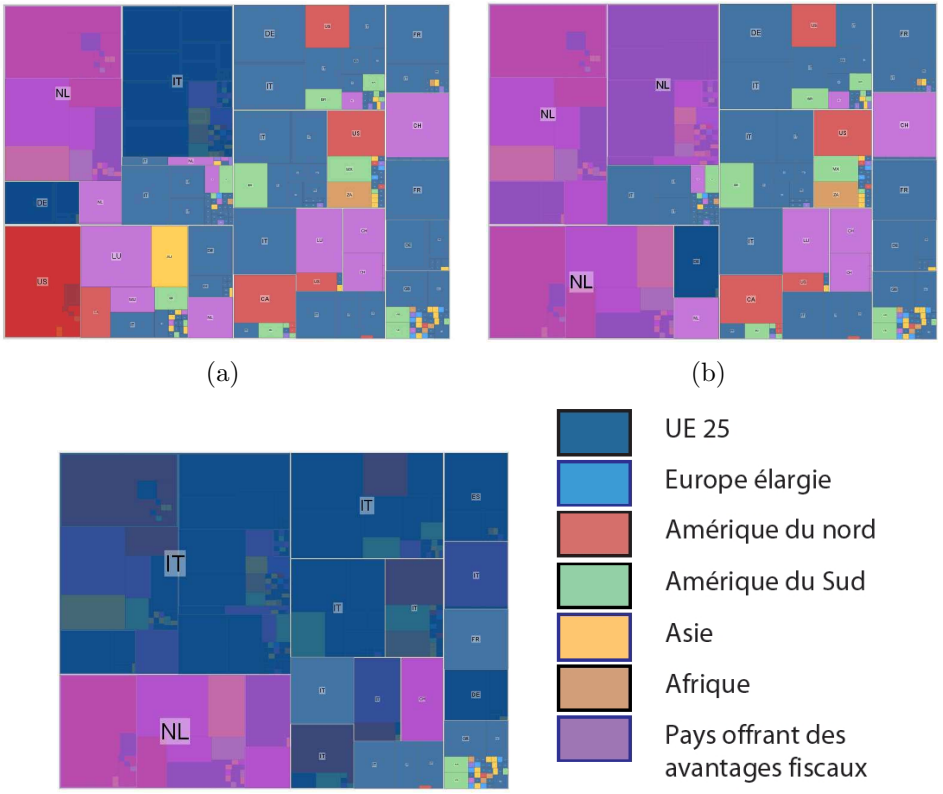


FIG. 4.8 – Même représentation du DAGMap de Fiat pour différentes hauteurs de la "bande". On voit que l'image se recouvre peu à peu d'un voile. Les DAGMaps montrés ici sont obtenus à partir de celui de la figure 4.3 auquel on a appliqué différentes "bandes".

Outre la "bande", le DAGMap est pourvu de nombreuses interactions plus classiques. Le zoom avant et arrière ainsi que le glissement de la vue permet une navigation simple sur le DAGMap. Une interaction astucieuse, empruntée aux travaux de Blanch et Lecolinet [14], permet de sélectionner une cellule par simple trait sur la TreeMap. La cellule sélectionnée correspond alors à l'ancêtre commun des deux cellules aux extrémités du trait (voir figure 4.9).

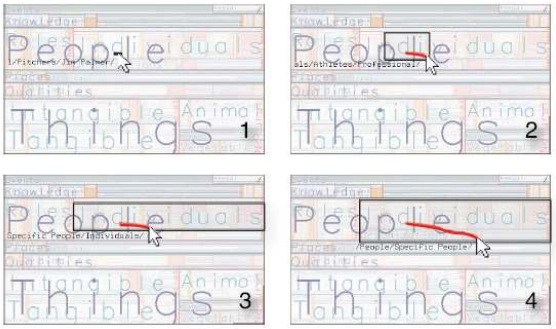


FIG. 4.9 – Treemap Zoomable [14], la sélection d'un sommet dans la TreeMap se fait par un simple trait sur celle-ci.

4.3 Cas d'étude

Notre travail a été initié par la nécessité d'avoir une visualisation centrée sur les attributs (comme les TreeMaps) pour explorer des données économiques. Les géographes ont collecté des données décrivant les liens entre différentes sociétés majeures et leurs filiales (et sous-filiales, etc ...). On a vu que ce type de données peut être modélisé par des DAGs. Ces DAGs sont peu denses et se prêtent bien à la technique exposée précédemment. 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.

Le défi du système de visualisation est donc d'offrir à l'utilisateur un ensemble d'opérations et d'interactions permettant une exploration de la structure (topologie) et des attributs de données.

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 ces jeux de données spécifiques, comme l'attestent les travaux de nos collègues montrant l'utilité et l'utilisabilité des représentations et des applications fournies [37, 15].

Comme le montre la figure 4.8, 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.

Dans l'étude de données décrivant les relations d'héritage entre les compagnies et leurs filiales, les géographes veulent identifier les stratégies territoriales mises en oeuvre par les compagnies mères [37, 15]. L'organisation territoriale des filiales peut

[1]Dans le cadre du projet ANR MDCO SPANGEO. Voir l'URL `http://s4.parisgeo.cnrs.fr/spangeo/spangeo.htm`

suivre une logique de production (des produits spécifiques sont produits dans une partie du monde). Le management et les processus de décision étant, quant à eux, au niveau des maisons mères. D'autres compagnies peuvent déléguer l'ensemble de la chaîne (du management, processus de décision à la production et la commercialisation) par continent. Ainsi, on peut observer une sous-hiérarchie par continent ou partie du monde.

En regardant la structure de la hiérarchie, exploitant les indications de localisation (couleur des sommets) et la découverte de schéma d'organisation, les géographes ont été capables de formuler des hypothèses sur les modèles de dérégulation et management déployés par les différentes compagnies et par rapport à leurs domaines d'activités.

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 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 :

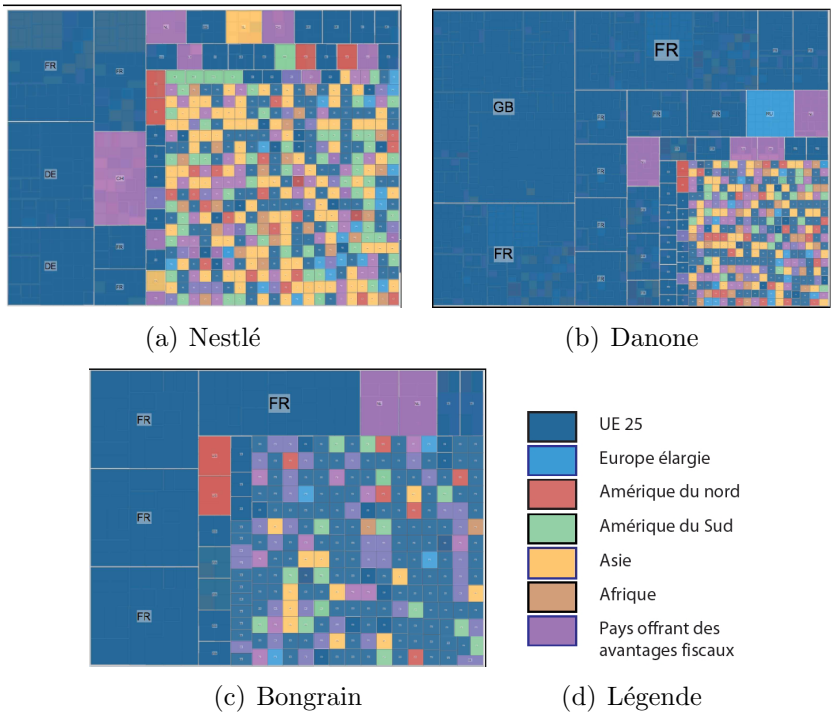
- Les sociétés contrôlant une filiale (ou un ensemble de filiales) sont-elles réparties sur plusieurs régions du monde ou concentrées dans une région spécifique ?
- Le contrôle est-il partagé entre sociétés mères et filiales de moyenne importance, ou est-il concentré à la tête de la hiérarchie ? Y a t-il un niveau de la hiérarchie dans lequel se concentre le contrôle ?
- Les sociétés mères d'une filiale donnée ont-elles des stratégies de développement similaires (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 ?

Comme le montrent les questions précédentes, la 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'interactions spécifiques. Le code couleur utilisé dans les différentes figures de notre cas d'étude a été établi par les géographes (voir figure 4.10(d)).

Distribution du contrôle Dans notre exemple (voir figure 4.3), 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.

Contrôle direct sur les filiales délocalisées Une comparaison des différentes compagnies est possible suivant différents niveaux. Ainsi, les géographes ont pu

déterminer des stratégies communes aux entreprises Nestlé, Danone et Bongrain. Ces sociétés, bien qu'européennes, possèdent des filiales à l'extérieur de l'Europe. Celles-ci sont alors directement contrôlées par la maison mère au plus haut niveau montrant une certaine frilosité des compagnies européennes à déléguer le contrôle des filiales hors de leur territoire. Cela se traduit par la présence d'un nombre important de compagnies non européennes visibles à un niveau élevé (voir figure 4.10).



(a) Nestlé (b) Danone

(c) Bongrain (d) Légende

FIG. 4.10 – DAG de gouvernance de Nestlé, Danone et Bongrain avec une bande au niveau 2.

Mise en évidence d'une stratégie financière : Fiat Cette fonctionnalité 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 représentés par une couleur fushia dans l'exemple des images de la figure 4.8. 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 images de la figure 4.8.

4.4 Évaluation

Le développement du DAGMap a été initié par une collaboration avec les géographes. Les choix de conception et spécifications de l'application sont consécutifs à des discussions avec eux et des réactions envers divers prototypes. De ce fait, on pouvait prétendre de son utilisabilité ou au moins d'une adoption de l'outil dans le cadre de

leur travail. Cela n'apportait toutefois pas une preuve formelle de son utilisabilité. Nous avons donc cherché à l'établir formellement dans le cadre d'une expérience contrôlée.

Malgré l'utilisation du DAGMap, les géographes maintenaient leur retour vers le dessin traditionnel, la vue noeud lien du DAG. On observait, alors, un aller retour constant entre ces deux types de visualisation apportant chacune des avantages. La disposition par niveaux dans le diagramme noeud lien donne une information sur la structure. Le DAGMap, pour sa part, offre une vue claire sur les attributs des données. De fait, leur combinaison pouvait apparaître comme étant la visualisation optimale. Une évaluation formelle confrontant ces trois visualisations (noeud lien, DAGMap et vue combinée) était donc nécessaire.

Le contexte du projet avec les géographes fournit beaucoup d'ingrédients afin de mettre en place l'expérimentation : le choix des jeux de données, des questions, les visualisations à comparer et les interactions.

Dans le cadre de l'expérience, nous nous sommes limités à des interactions assez simples sur les structures pour éviter de biaiser les observations. On décrit ici un premier éventail d'interactions assez basiques qui ont fait l'objet d'utilisation effective par les géographes et d'une évaluation formelle. Au chapitre 6, nous reviendrons sur le DAGMap pour lequel ont été conçues des interactions plus fines et complexes (basées sur des techniques de navigation "focus+contexte") permettant de traiter des données massives.

4.4.1 Méthodologie de l'expérimentation

La mise en place d'une expérimentation formelle nécessite de définir des conditions. Chaque condition correspond à une visualisation qu'on veut comparer aux autres. Il faut ensuite définir pour l'ensemble des conditions des jeux de données. Chaque visualisation va permettre de répondre à des questions sur les jeux de données. Les questions elles-mêmes font l'objet d'un travail minutieux afin que la formulation soit claire et sans ambiguïté.

Nous comparons les trois visualisations de DAG qui ont été décrites précédemment : la vue classique du diagramme noeud lien, le DAGMap et la vue combinée de la représentation noeud lien et du DAGMap. Dans une première phase, nous avons conduit une expérimentation pilote sur un petit ensemble de candidats. Ce pilote permet de déterminer les différents ingrédients de l'expérience afin de rendre l'étude plus robuste. Ceci se traduit par une reformulation des questions, voire à en écarter, le choix des jeux de données et la détermination du temps nécessaire pour effectuer une tâche.

Définition des tâches

Les participants devaient réaliser des tâches utilisant chacune des trois visualisations. Chaque tâche consistait en un type de représentation (noeud lien, DAGMap ou vue combinée), une question (Q1, Q2 ou Q3) et un domaine (jeux de données de Fiat, Nestlé ou Danone). Utilisant un "within-subject design" (conception inter sujet), tous les candidats ont fait toutes les tâches : 36 tâches au total pour chaque candidat. Chaque candidat a pris part dans un premier temps à un tutoriel démonstratif

(3 tâches) durant lequel lui ont été présentés les 3 différentes représentations, questions et les interactions disponibles lui permettant d'effectuer les tâches demandées. Les candidats procédaient alors à une série de 6 tâches où l'examinateur s'assure qu'ils ont bien compris les questions et modalité de réponse - l'examinateur peut intervenir en cas d'erreur du candidat afin d'améliorer sa compréhension des tâches. Il est à noter que ces 9 premières tâches n'ont pas été utilisées dans l'analyse des résultats. Ainsi, chaque question a été répondue 9 fois à l'aide des 3 représentations. Les utilisateurs répondent donc à 27 tâches expérimentales (3 représentations $\times$ 3 questions $\times$ 3 domaines) sans aucune assistance de l'examinateur. L'ordre dans lequel les tâches sont présentées est rendu aléatoire. En effet, on peut s'attendre à ce que l'utilisateur soit plus performant à la fin de l'expérience qu'au début. Si l'ordre des tâches était toujours le même, cet effet d'apprentissage serait bénéfique à l'une des représentations et toujours la même visualisation ce qui biaiserait la comparaison. En utilisant un ordre aléatoire, l'effet d'apprentissage est réparti sur toutes les tâches.

Aucune limite de temps n'a été imposée à l'utilisateur pour effectuer les tâches. Cependant, durant l'expérimentation pilote, une estimation de ce temps était de 1 minute à 1 minute 15 secondes par tâches portant ainsi l'ensemble de l'évaluation à 1 heure (tutoriel, évaluation et questionnaire de fin). Cette durée est jugée acceptable. Une durée plus longue pourrait avoir des conséquences sur les résultats causées par la fatigue des candidats.

La taille de la fenêtre est elle aussi fixée afin de s'assurer que chaque représentation bénéficie du même espace à l'écran (1280 $\times$ 900 pixels). Ainsi, pour la vue combinée, l'écran est divisé en deux parties de même taille.

La formulation des différentes questions a été choisie de telle sorte qu'elle demande une réponse numérique. L'utilisateur a alors à cliquer sur le nombre correspondant à sa réponse (question à choix multiples) (voir figure 4.11). Cette façon de procéder a été choisie afin d'éviter toutes tâches cognitives supplémentaires à l'utilisateur : une réponse saisie au clavier aurait pu induire un temps supplémentaire et on aurait alors mesuré la dextérité du candidat à taper au clavier, ce qui n'est pas le but de l'étude.

Nous allons maintenant détailler les différentes interactions mises à la disposition de l'utilisateur afin d'effectuer les tâches demandées. La visualisation noeud lien permet à l'utilisateur de zoomer en avant/arrière ("in/out"), de faire glisser le dessin ("pan") et de cliquer sur les noeuds et de les déplacer ("drag and drop"). En cliquant sur un noeud, les arêtes incidentes au noeud et les noeuds correspondants sont mis en évidence par une couleur spécifique. Ainsi, les arêtes entrantes et le bord des sommets parents du noeud sont colorés en rose (voir figure 4.12). Les arêtes sortantes et le bord des fils du noeud sont colorés en bleu. Cette interaction est spécifique à la vue noeud lien et a été rajoutée pour l'expérimentation suite aux différentes expérimentations pilotes. La vue DAGMap permet à l'utilisateur de faire un zoom avant et arrière, de faire glisser la vue et de cliquer sur un noeud. Une glissière (un slider) lui permet aussi de piloter un voile au dessus de la visualisation cachant ainsi différents niveaux de la hiérarchie (voir section 4.2). En cliquant sur une cellule (un noeud), l'utilisateur peut voir la couleur de la cellule sans transparence. Cette interaction a été rajoutée afin d'éviter toute confusion de couleur causée par la transparence. Dans la vue

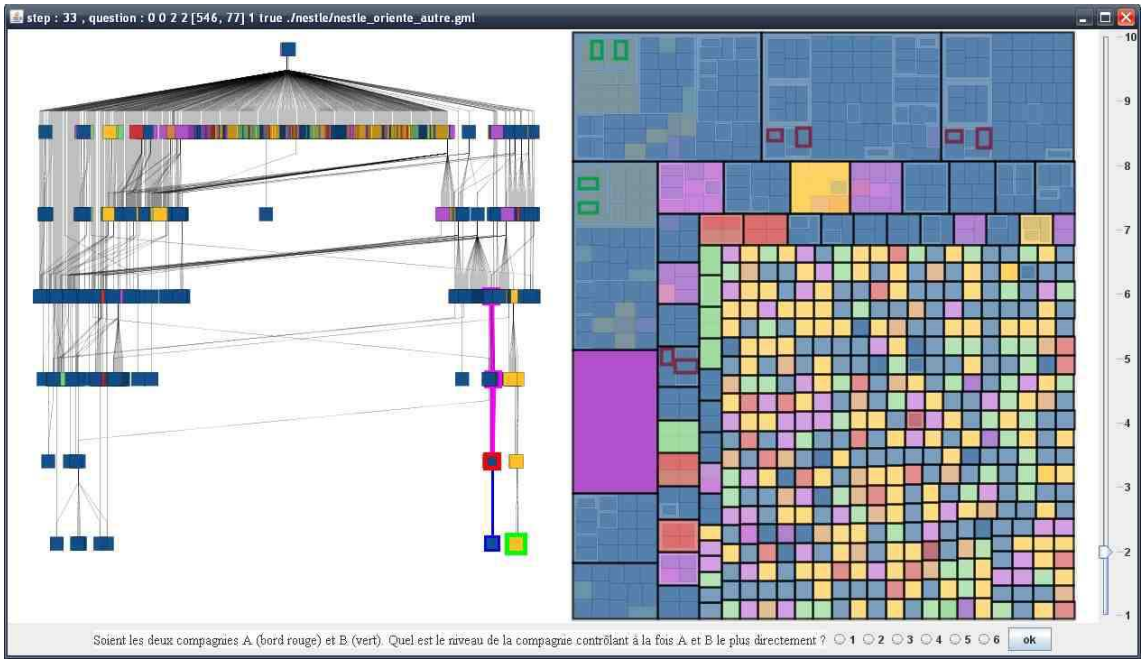


FIG. 4.11 – Description de l'écran classique pour une tâche mettant en scène la vue combinée. A droite de l'écran, une glissière (slider) permet de faire varier la "bande". La question ainsi que les réponses proposées sont en bas de l'écran.

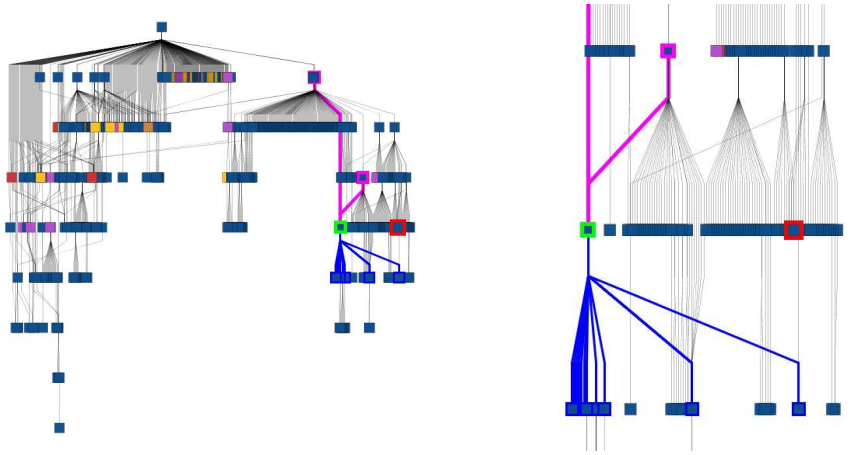
combinée, l'utilisateur dispose simultanément de toutes les interactions de la vue noeud lien et de la vue DAGMap.

Jeux de données

Trois différents jeux de données (domaines) ont été utilisés afin d'éviter le phénomène d'apprentissage sur les données. Ces jeux de données sont issus du projet avec les géographes qui a initié la technique du DAGMap. L'utilisation de ces jeux de données nous a permis d'interviewer non seulement des géographes mais aussi des informaticiens. Chacun de ces jeux de données décrivent comment les compagnies interagissent avec leurs filiales. Trois compagnies majeures (Fiat, Nestlé et Danone) ont permis de constituer ces hiérarchies. Ces hiérarchies (graphes) sont de tailles comparables en terme de nombre de noeuds et d'arêtes (FIAT : 606 noeuds, 627 arêtes, NESTLE : 638 noeuds, 771 arêtes, DANONE : 692 noeuds, 997 arêtes).

Ces jeux de données ont été choisis pour leur caractère peu dense. En effet, pour des graphes plus denses, la vue noeud lien comme la vue DAGMap est plus difficile à interpréter. La vue noeud lien souffre de part le nombre de croisements d'arêtes et chevauchement de sommets. Le DAGMap, lui, souffre de la duplication des sommets.

Dans ces graphes, les noeuds représentent les compagnies. Une arête e_{AB} entre les compagnies A et B indique que la compagnie A contrôle en partie la compagnie B, en d'autres termes la compagnie B est une filiale de la compagnie A. Dans les différentes représentations, la couleur des noeuds indique la localisation géographique des compagnies.



(a) Vue noeud lien original

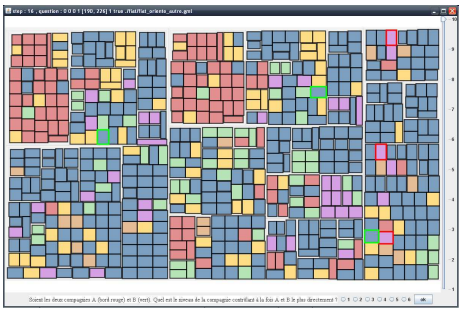
(b) Zoom sur une zone de la vue noeud lien

FIG. 4.12 – (a) Vue noeud lien du DAG de Danone. En cliquant sur un noeud, les arêtes entrantes et le bord des pères du noeud sont colorés en rose, les arêtes sortantes et le bord des fils sont quant à eux colorés en bleu. (b) Même vue du DAG montrant le zoom sur une zone. Le zoom permet de distinguer plus facilement les noeuds.

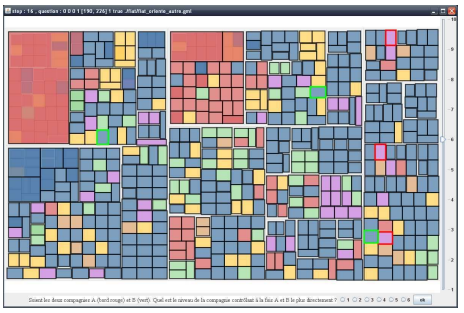
Questions

Pour notre expérience, les questions ont été formulées à partir de notre expérience avec les géographes. Celles-ci sont essentiellement celles de la section 4.2. Les questions ont été par ailleurs simplifiées dans leur contenu et leur formulation. Ainsi, des non-spécialistes peuvent y répondre, aucun pré requis étant nécessaire ouvrant l'expérimentation à un public plus large. Les questions peuvent être décomposées en tâches élémentaires décrites dans [60] de manière à s'assurer qu'elles couvrent l'ensemble des tâches élémentaires applicables à notre visualisation. Nous décrivons maintenant les questions telles qu'elles ont été présentées aux candidats en suivant la taxonomie de [60]. Cette décomposition nous permet de s'assurer que les tâches recouvrent bien toutes les tâches élémentaires sur ces types de visualisation.

Question 1 Q1 : "Soient les deux compagnies A (bord rouge) et B (bord vert). Quel est le niveau de la compagnie contrôlant à la fois A et B le plus directement ?" (la compagnie possédant à la fois A et B ayant le niveau le plus bas dans la hiérarchie.)
Les bords rouges et verts font référence à des attributs visuels présents à l'écran pour aider le candidat à localiser les sociétés concernées. Cette question consiste à trouver le plus petit ancêtre commun dans le DAG. Remis dans le contexte géographique, la question pourrait être reformulée par "A quel niveau dans la hiérarchie les filiales A et B peuvent entrer en relation d'affaires (à travers une compagnie mère commune) le plus directement ?". Pour répondre à cette question, le candidat doit trouver l'ensemble des parents des noeuds concernés. Si il y a un ancêtre commun, la réponse est trouvée, sinon il faut réitérer sur chacun des parents. Ce type de

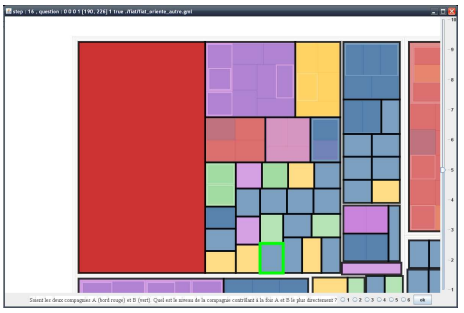


(a) Vue DAGMap original



(b) Vue DAGMap avec changement de la “bande”

(c) Vue DAGMap avec un zoom



(d) Vue DAGMap avec l’effet d’un clic sur un sommet

FIG. 4.13 – Différentes interactions possibles sur la vue DAGMap : (a) Vue DAGMap du DAG de Fiat. (b) Même vue montrant l’effet du slider contrôlant la hauteur de la “bande”. (c) Même vue qu’en (b) avec un zoom sur le coin en haut à gauche. (d) Même vue qu’en (c) montrant l’effet d’un clic sur un sommet : la valeur alpha déterminant la transparence du voile est neutralisée.

tâches peut être décomposées en plusieurs tâches élémentaires : “trouver un noeud”, “trouver les noeuds adjacents à un noeud” [60].

Question 2 Q2 : “Soit A une compagnie (au bord vert). Dans combien de pays différents (couleurs) se trouvent les filiales qu'elle contrôle directement ?”
Cette question consiste à identifier l'ensemble des fils d'un sommet donné et de compter le nombre de couleurs différentes. Cette question peut paraître évidente et simpliste, cependant elle fait partie d'une série de questions que les géographes doivent résoudre pour répondre à certaine problématique telle que : “Etant données deux compagnies, qu'elle est la compagnie qui a un développement spatial correspondant à une stratégie internationale ?”.

Cette question formulée en terme simple sur les graphes revient à compter le nombre d'éléments à distance 1 ayant une caractéristique donnée. Une décomposition en tâches élémentaires serait : “trouver un noeud”, “trouver l'ensemble des noeuds adjacents”, “compter un attribut du noeud” [60].

Question 3 Q3 : “Soit une compagnie (au bord vert). Quelle est la longueur du chemin le plus court qui sépare une filiale d'une maison mère ?”
La longueur du plus court chemin entre une compagnie et une maison mère est une indication intéressante. En effet, plus cette valeur est élevée, plus la compagnie considérée a un rôle spécialisé (par exemple dans la production).

Cette question consiste à suivre différents chemins dans le graphe et retourner la longueur du plus court.

Pour répondre à cette question le candidat doit trouver les plus courts chemins entre un sommet sélectionné et une racine. Ainsi, une décomposition en tâches plus petite serait : “trouver les noeuds adjacents à un noeud”, “choisir un noeud (noeud ayant une profondeur plus petites - on veut remonter vers une racine)” [60].

Candidats

Cette expérimentation a inclu 22 utilisateurs parmi lesquels 15 étaient des géographes et 7 des informaticiens. Tous les utilisateurs étaient familiers avec l'usage d'un ordinateur (qu'ils manipulent tous les jours). Le protocole de l'expérimentation a été mis en place dans une première phase (expérimentation pilote). Les candidats participants à cette étape n'ont pas participé à la seconde étape (l'expérimentation à proprement parler). Les membres du projet Spangeo ont été sollicités pour la phase d'expérimentation. Les utilisateurs (d'où proviennent les données) n'avaient pas d'expérience a priori avec les données ou les systèmes de visualisations.

4.4.2 Résultats quantitatifs

Durant l'évaluation, un utilisateur a utilisé les 3 représentations 9 fois chacune, et a répondu à chacune des 3 questions 9 fois. Lors de l'expérimentation, différents paramètres ont été enregistrés tel que le temps mis pour répondre à chaque tâche, le nombre d'interactions effectuées et bien sûr la réponse à la question. Ces informations reflètent la capacité de l'utilisateur à effectuer la tâche.

Cette expérimentation tente de déterminer si une des conditions (visualisations) montre des mesures (les différents paramètres enregistrés) qui sont significativement différentes des autres.

Afin de mettre en évidence de telles différences, deux tests statistiques sont utilisés (dans cette étude) : le test d'analyse de variance ANOVA (ANalysis Of VAriance) et le test de Tukey. L'analyse de la variance utilisée pour la forme particulière des données que nous avons est le *repeated measures design* ANOVA. Le test appliqué sur les mesures collectées calcule une valeur F qui indique ou non si il y a des différences significatives dans les mesures. Ce test repose sur un paramètre p indiquant le niveau de confiance. Différents niveaux sont considérés : $p = 0.1$, 0.05 ou 0.01. Ainsi, pour un $p = 0.1$, le niveau de confiance est de 10%. C'est-à-dire, si le test indique une différence significative, il a une chance sur 10 de se tromper. La valeur de p influe sur la valeur du F requis afin qu'il soit significatif.

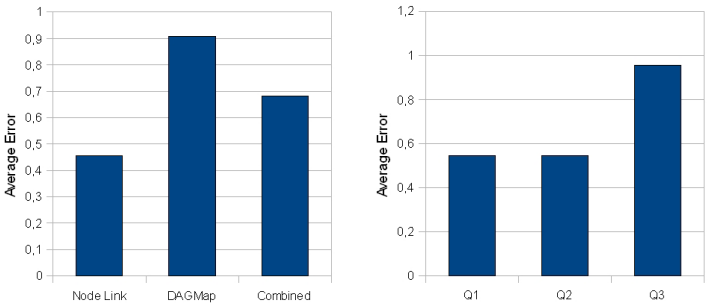
Quand on effectue un test ANOVA sur plus de deux groupes, une valeur significative F n'indique pas où se trouve la différence significative. Cela indique seulement qu'il y a une différence significative entre les conditions (les différentes vues). Cela indique seulement que la condition (la représentation) choisie influe (a un effet sur) les performances de l'utilisateur. Une analyse supplémentaire est nécessaire afin d'identifier la source de la différence significative. Les données collectées doivent subir un nouveau test. Le test de Tukey HSD (Honestly Significant Difference) est un test post hoc (complémentaire - près traitement) du test ANOVA. Il est utilisé pour comparer chacune des conditions entre elles afin d'identifier les paires de conditions dont la différence est significative. Ceci permet d'identifier la source de la valeur significative F.

Justesse des résultats des utilisateurs (Average error)

Dans cette section, sont présentées les différentes analyses effectuées à partir de l'exactitude ou non des réponses des candidats. Notre collaboration avec les membres du projet Spangeo nous laissait penser que la vue combinée aurait la meilleure performance, combinant les avantages de la vue noeud lien et de la vue DAGMap. Les résultats obtenus semblent nous contredire. La figure 4.14 présente un résumé de ces résultats.

Le test ANOVA n'a pas trouvé de différence significative entre les différentes conditions (Noeud lien, DAGMap et Vue combinée). Le test n'a pas pu déterminer si la représentation avait un rôle discriminant sur la performance des utilisateurs en terme de nombre d'erreurs faites. Cela ne signifie pas que la vue n'a pas d'impact sur la performance ou que les représentations ont un rôle équivalent. Un échantillon plus important de candidats serait souhaitable pour répondre à cette question.

Des analyses similaires peuvent être effectuées suivant chaque question et pour les jeux de données. Ainsi on peut déterminer si il y a une différence significative entre les vues sur la performance suivant la question posée ou le jeu de données utilisé. Autrement dit, si l'on considére que les tâches ayant la question Qi, y a t-il une différence significative entre les représentations ? Ces analyses n'ont pas montré de différences significatives concernant les jeux de données. Cependant, pour les questions, le test est positif. Il y a une différence significative sur la performance



(a) (b)

FIG. 4.14 – (a) Moyenne des erreurs faites par les utilisateurs suivant les représentations (Noeud lien, DAGMap et Vue combinée). (b) Suivant la question posée.

des candidats répondant à la question 3 (portant sur le calcul du chemin le plus court) suivant la représentation proposée. En effet la valeur $F = 3.51$ est supérieure au $F(2, 42, p = 0.01) = 3.15$ de référence. Un post traitement (Tukey) montre où est la différence significative avec un niveau de confiance de $p = 0.05$: le DAGMap (Condition 2) a produit de plus mauvais résultats que la vue noeud lien (Condition 1). Le test n'a pas révélé d'autre différence. Cela n'est pas surprenant, la question 3 est une question qui nécessite de chercher un plus court chemin. La vue noeud lien est donc ici avantagée de par l'explicitation de la structure dans cette vue.

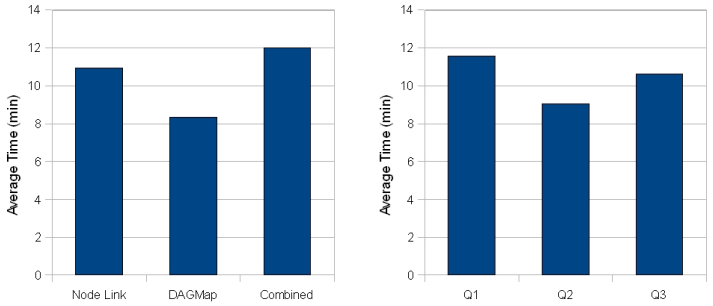
La figure 4.14(b) montre la somme des erreurs moyennes faites par les candidats suivant chacune des différentes questions posées. Afin de déterminer si une question est plus difficile ou plus facile que les autres, un test ANOVA sur la performance en terme d'erreur suivant les questions utilisées est effectué. Ce test n'a pas mis en évidence une différence significative entre les différentes questions (Q1, Q2 et Q3).

Temps de reponse (Time to completion)

Dans cette section, nous nous intéressons au temps mis par les candidats pour répondre aux différentes questions. La figure 4.15 montre les résultats obtenus suivant la vue ou la question : la figure 4.15(a) montre le temps mis par tous les candidats pour compléter les différentes tâches suivant la vue proposée. La figure 4.15(b) rend compte du temps requis suivant la question posée. La représentation DAGMap semble être la plus efficace pour chacune des trois questions.

Afin de déterminer si il y a une différence significative entre les différentes représentations, un test ANOVA est effectué. C'est avec un niveau de confiance $p = 0.01$ que le test confirme qu'il y a une différence significative. Un test en post traitement permet d'identifier la source de cette différence significative. Le test de Tukey indique qu'il y a une différence significative ($p = 0.01$) entre la représentation DAGMap et les deux vues noeud lien et combinée. Le DAGMap a produit de meilleur résultat que les deux autres vues. Par ailleurs, le test n'indique pas de différence significative entre la vue noeud lien et la vue combinée. Ces résultats ne sont pas surprenants concernant la vue combinée. En effet, la vue combinée a souffert du

"double check". Les candidats ont d'abord cherché la réponse sur l'une des deux vues et ont ensuite contrôlé leur réponse sur l'autre.



(a) (b)

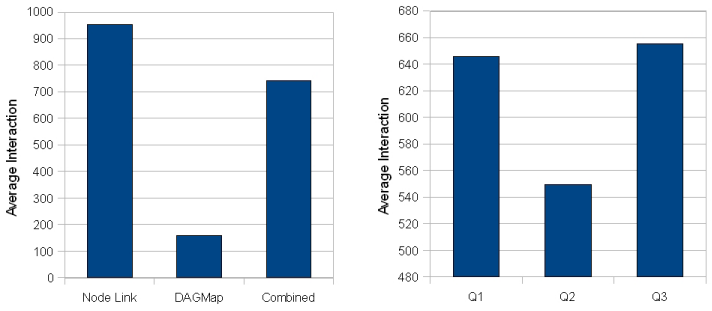
FIG. 4.15 – (a) Temps moyen (en minute) mis par les candidats pour compléter les tâches suivant la représentation (Noeud Lien, DAGMap et Vue Combinée). (b) Temps moyen (en minute) suivant la question (Q1, Q2, Q3).

Un test ANOVA révèle qu'il y a une différence significative sur le temps requis pour effectuer une tâche suivant la question.

Nous pouvons procéder à un test ANOVA/Tukey pour déterminer l'effet des représentations suivant les questions posées. Autrement dit, si l'on considére que les tâches ayant la question Qi, y a t-il une différence significative entre les représentations ? Pour la question 2, le test ANOVA n'a pas révélé de différence significative en terme de temps requis suivant les différentes interfaces. Pour la question 1 et 3, le test fut positif : il y a une différence significative suivant la représentation proposée pour la question 1 ($F = 9.2 > F(2, 42, p = 0.01) = 3.15$). Un post traitement de Tukey révèle que le DAGMap a produit de meilleurs résultats que la vue Noeud lien et que la vue combinée. De même il y a une différence significative pour la question 3 ($F = 6.29 > F(2, 42, p = 0.01) = 3.15$). Le post traitement révèle que le DAGMap a eu de meilleurs résultats que la vue combinée. Il n'y a pas d'autre différence significative.

Nombre d'interactions

Des mesures additionnelles peuvent nous aider à évaluer les différentes vues. Ainsi, la performance des utilisateurs est évaluée par le nombre d'interactions effectuées pour résoudre les tâches. Les interactions prises en compte dans notre étude sont le clic de souris, le drag and drop, le zoom in/out. La figure 4.16 montre un résumé des résultats obtenus suivant la représentation et la question proposées. Quelle que soit la question, le DAGMap est la représentation qui a enregistré le moins d'interactions. Un test ANOVA vient confirmer cette évidence. Un post traitement montre que la différence significative oppose le DAGMap aux autres représentations. On peut donc dire que le DAGMap nécessite moins d'interactions que les deux autres représentations. Le test n'a cependant pas pu différencier la vue noeud lien de la vue combinée.



(a) (b)

FIG. 4.16 – (a) Nombre moyen d'interactions effectuées par les candidats suivant les différentes représentation proposées (noeud lien, DAGMap et vue combinée). (b) Nombre moyen d'interactions effectuées par les candidats suivant la question posée (Q1, Q2, Q3).

L'observation faite lors de l'évaluation permet d'expliquer ce dernier point. Malgré le fait que la vue noeud lien montre de façon explicite la structure même de la hiérarchie, cette représentation ne permet pas toujours d'éviter les croisements d'arêtes et le chevauchement de sommets. Par conséquent, afin de répondre à la question, l'utilisateur doit effectuer un nombre non négligeable de repositionements de sommets (drag and drop) afin d'identifier les différents éléments du DAG. Par ailleurs, afin de mieux visualiser les éléments, l'utilisateur effectue des zoom in/out successifs pour voir le détail (compter par exemple le nombre d'éléments ayant un attribut particulier) et se déplacer dans la vue. De plus, même pour des questions qui normalement avantagent la vue noeud lien, le nombre de croisements d'arêtes oblige l'utilisateur à faire des zoom in et out pour s'assurer de sélectionner le bon élément. La vue combinée a, quant à elle, souffert du "double check". En effet, le candidat a toujours contrôlé sa réponse sur l'autre vue disponible.

La figure 4.16 pourrait laisser supposer qu'il y a une différence de performance suivant la question posée. Un test ANOVA vient confirmer cette supposition. Il y a une différence significative entre les couples de questions (Q1, Q2) et (Q2, Q3). La question Q2 est celle qui a mis le moins de temps. Le test n'a pas révélé d'autre différence significative.

4.5 Discussion

La justesse des réponses pour les trois visualisations a un score élevé (supérieur à 90 %). Le test ANOVA n'a pas indiqué de différence significative entre les représentations au regard des erreurs effectuées (voir section 4.4.2) : 5% pour la vue noeud lien, 7.5% pour la vue combinée et 10% pour le DAGMap. Le DAGMap a cependant enregistré la meilleure performance, en terme de temps pour effectuer les tâches. La visualisation noeud lien a, quant à elle, enregistré le moins d'erreur, mais a nécessité le plus d'interactions et le plus de temps. La vue combinée, qui a souffert du "double check" semble être, après tout, un bon compromis entre les deux autres représentations,

comme nous l'avions pressenti.

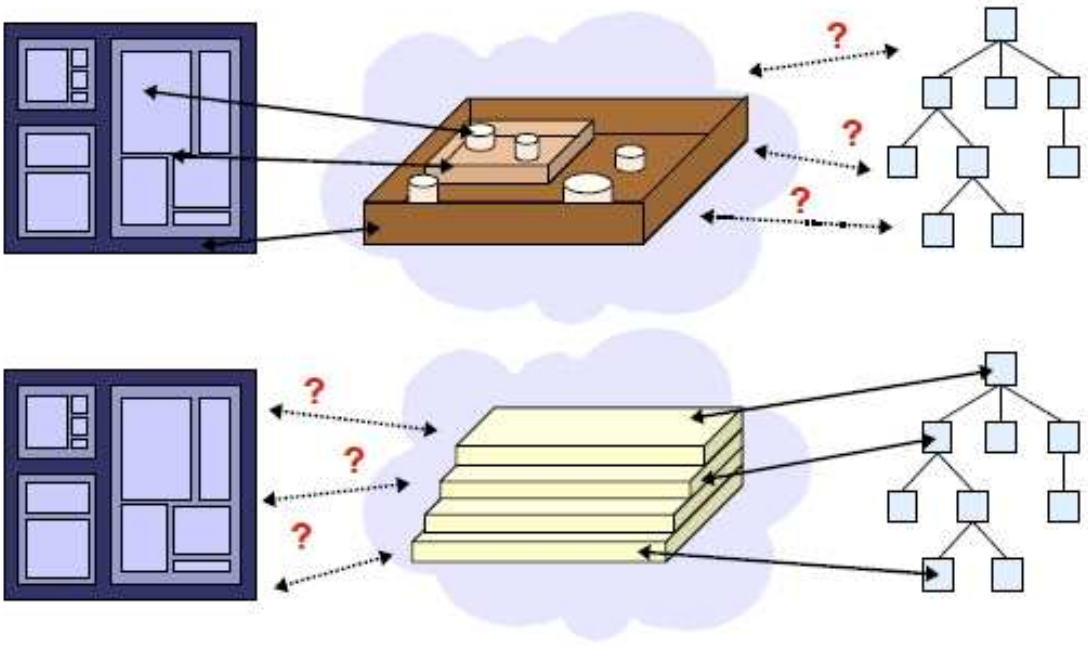


FIG. 4.17 – Compréhension de l'information [97].

Nous nous attendions à ce que les résultats du DAGMap et de la vue combinée ressortent comme étant les visualisations les plus efficaces. Cela n'a pas été confirmé par l'expérimentation. Les travaux de Kosara et al. [97] offrent une piste pouvant expliquer les observations faites à l'issue notre expérience. Kosara et al. ont cherché à mesurer à quel point le vocabulaire employé dans les questions peut favoriser une vue ou une autre. La figure 4.17 illustre ce propos. Si on parle de la notion de contenu, "Qui contient qui ?", alors la réponse est plus évidente sur une TreeMap que sur un diagramme noeud lien. En effet, la TreeMap fait référence explicitement à la notion de contenant. A l'inverse, si on parle de prédécesseur, de niveau, alors c'est la vue noeud lien qui a l'avantage. Dans notre expérience, nous avons utilisé le mot "niveau" dans la question Q1, ou "longueur" dans la question Q3 faisant référence à la structure topologique. Selon Kosara et al., cela favoriserait donc la vue noeud lien et influerait sur l'expérience elle-même et sur les conclusions à en tirer. Les travaux de Kosara et al. n'ont été publié que très récemment, peu après la fin de notre expérimentation. Il aurait fallu pouvoir reprendre l'expérience depuis le début pour tenir compte de leurs résultats – ce que nous n'avons pas pu envisager par manque de temps.

4.5.1 Satisfaction des utilisateurs (User Satisfaction)

Il est coutume de faire un court interview en fin d'expérience pour récolter les impressions des candidats. On rapporte ici les questions posées aux candidats (voir

figure 4.18).

- Q0. Quelle est la visualisation la plus simple, selon vous, pour répondre à la question Q1 ?
- Q1. Quelle est la visualisation la plus simple, selon vous, pour répondre à la question Q2 ?
- Q2. Quelle est la visualisation la plus simple, selon vous, pour répondre à la question Q3 ?
- Q3. Dans quelle visualisation est-il le plus simple de voir la hiérarchie (structure) de données ?
- Q4. Dans quelle visualisation est-il le plus simple de voir les attributs (couleur ...) relatives aux données ?

Les utilisateurs ont clairement exprimé une préférence pour le DAGMap. Ils ont, cependant, opté pour la vue noeud lien pour les tâches visant à explorer la structure. Pour l'exploration des attributs, le DAGMap est la visualisation privilégiée.

Les impressions des utilisateurs ne donnent pas de mesure objective sur les méthodes mais servent à renseigner l'expérimentateur. Il faut donc prendre ces résultats avec des réserves. En effet, souvent les impressions des utilisateurs sont en contradiction avec les mesures observées.

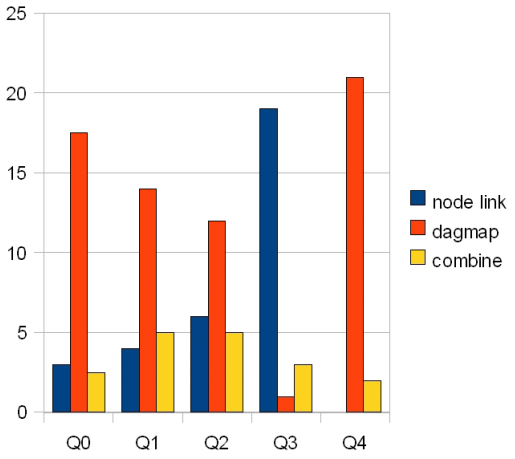


FIG. 4.18 – Présentation des résultats du questionnaire.

4.6 Conclusion

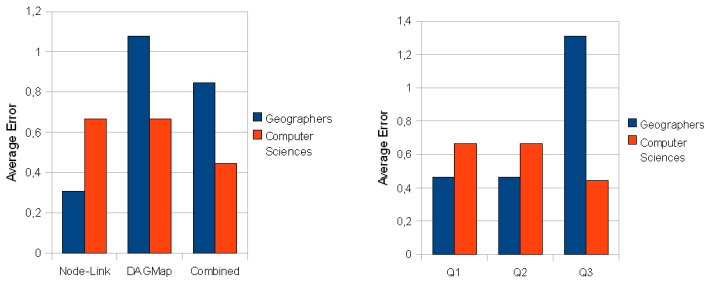
Dans ce chapitre, nous avons présenté le DAGMap qui a été élaboré dans le cadre d'un projet avec les géographes. Sa conception et son développement se sont fait au travers d'échanges constants et fréquents avec eux. Nous avons décrit différentes interactions sur le DAGMap permettant une visualisation par niveau de la hiérarchie par l'introduction d'une bande. Le chapitre 6 fournit des interactions plus élaborées permettant une navigation "focus+contexte".

Nous avons présenté le cadre expérimental permettant d'évaluer le DAGMap. Les questions ont d'abord été élaborées dans le contexte du projet avec les géographes et

ont été ciblées sur des applications en géographie. Nous avons porté attention dans les questions à la couverture des tâches élémentaires [60]. D'autres questions avaient été proposées au départ mais ont été écartées après l'expérience pilote.

Q0a : Soit une compagnie A. Combien de chemin existe t-il connectant cette compagnie aux compagnies mères ? (exerçant un contrôle sur elle).
Q0b : Soient les compagnies A et B. Est ce que l'une contrôle l'autre ? Est ce que l'une est une filiale de l'autre ?
Ces questions n'ont pas été retenues dans l'évaluation puisqu'elles donnaient un avantage évident à la représentation DAGMap. En effet, la tâche à effectuer est trop simple avec cette visualisation.

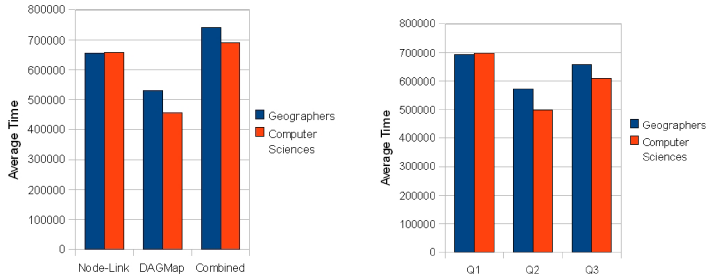
Nous n'avons pas prêté attention dans la section 4.4 au fait que l'ensemble de nos candidats appartiennent à deux communautés différentes : les géographes et les informaticiens. Cependant, notre échantillon est insuffisant pour amener à constater une différence qui est statistiquement valable. Cela dit, par curiosité, on peut rapporter ici les résultats qui semblent indiquer que les informaticiens ont été plus performants (bien que statistiquement ce ne soit pas avéré). Les figures 4.19, 4.20 et 4.21 montre un comparatif entre les deux populations.



(a) (b)

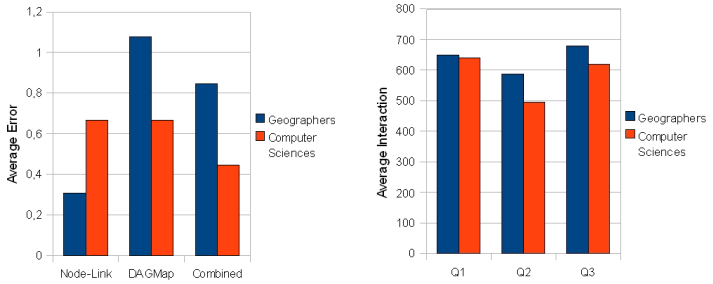
FIG. 4.19 – (a) Nombre d'erreurs moyen fait par les géographes (en bleu) et les informaticiens (en rouge) suivant les différentes vue proposées. (b) Nombre d'erreurs suivant la question posée.

Nos travaux sur le DAGMap ont été publiés au niveau national et international, tant auprès de publics de géographes que d'informaticiens. La publication des résultats de l'expérimentation reste à faire.



(a) (b)

FIG. 4.20 – (a) Somme du temps moyen mis par les géographes (en bleu) et les informaticien (en rouge) suivant la vue proposée. (b) Somme du temps moyen suivant la question posée.



(a) (b)

FIG. 4.21 – (a) Nombre moyen d'interactions effectuées par les géographes (en bleu) et les informaticiens (en rouge) suivant lavue proposée. (b) Nombre moyen d'interactions suivant la question posée.

Chapitre 5

Navigation “focus+contexte”

> *Un graphique ne doit pas seulement montrer les feuilles de l'arbre. Il doit aussi montrer les branches et l'arbre tout entier. L'oeil peut alors aller du détail à l'ensemble et découvrir à la fois la structure générale et ses exceptions.*
>
> J. Bertin [11]

Une représentation d'un graphe peut montrer l'ensemble de ses éléments donnant ainsi une vue globale de la structure des données. Même pour de petits graphes (centaines de sommets et d'arêtes), une telle représentation ne permet pas alors de discerner les détails de la structure. En effet, afin de représenter l'ensemble du graphe, la taille de ses éléments est alors petite. Une solution serait de faire un zoom géométrique sur le dessin afin de grossir la zone d'intérêt. La vue ainsi obtenue est alors une vue partielle de la structure, mais après quelques zooms successifs l'utilisateur peut être perdu : en effet, en effectuant un zoom, on perd le contexte, la vue d'ensemble. Afin d'éviter la perte de repère, il faut donc minimiser le nombre de zooms avant / arrière et de translation de la vue (“pan”) [53].

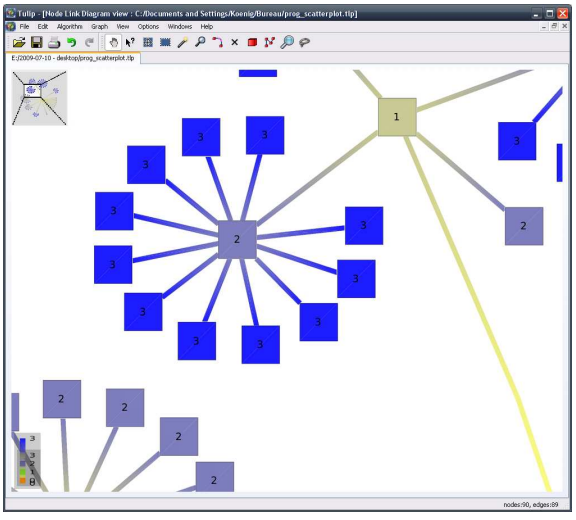


FIG. 5.1 – Vue détaillée d'un graphe après avoir effectué un zoom.

Une alternative est de montrer deux vues simultanément du graphe, une représentant le graphe en entier et l'autre montrant le détail. La figure 5.1 nous montre un exemple de représentation "overview+detail". Une petite zone de l'écran (le carré en haut à gauche) est consacrée à la vue d'ensemble du graphe. Le rectangle blanc dans cette fenêtre de "overview" indique précisément le périmètre de la vue détaillée.

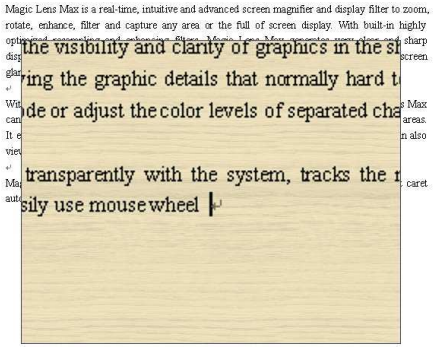


Fig. 5.2 – Magic Lens.

D'autres dispositifs existent impliquant différentes résolutions d'affichage. Dans [6], les auteurs proposent d'afficher l'information sur un mur (tableau) à l'aide d'un vidéo projecteur. Les grandes surfaces permettant de faire des images en plus grand format, évitent de faire des zooms. L'image produite fournit le contexte de la visualisation. L'utilisateur s'intéressant à une zone particulière fait glisser son centre d'intérêt sur un écran ayant une plus forte résolution (voir figure 5.3).

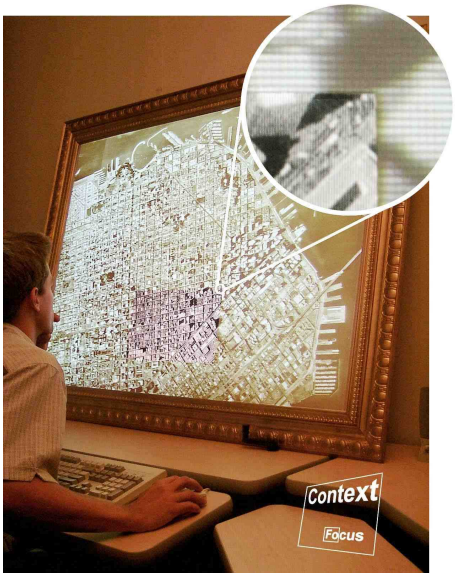


Fig. 5.3 – Dispositif avec plusieurs résolutions [6].

Le fait d'utiliser des grandes surfaces permet d'avoir une visualisation plus détaillée mais l'interaction avec les grandes surfaces amène de nombreuses questions à résoudre

d'un point de vue IHM. Le lecteur intéressé pourra consulter, par exemple, les travaux de Collomb et Hascoët [26, 25].

Les solutions qu'on a mentionnées ici consistent à travailler au niveau de l'écran et de l'affichage graphique soit en disposant d'une résolution très grande soit en proposant des vues combinées (vue détaillée et vue globale). Il y a encore une alternative qui consiste à construire une abstraction et à visualiser cette abstraction en tant que vue globale de la donnée. Le défi consiste alors à construire l'abstraction à partir de la donnée. A travers cette vue abstraite de la structure, le niveau de détail le plus fin doit pouvoir être accédé sur une région d'intérêt.

Cette autre solution, celle qu'on va développer, consiste à travailler non pas au niveau du rendu (écran) mais au niveau de la donnée à visualiser. A partir de la donnée, on fabrique une structure de donnée annexe qui en donne une vue globale comportant moins d'éléments graphiques.

L'image 5.4 illustre bien ce concept. En effet, on a une vue détaillée de la neuvième avenue de New York et une vue globale du reste de l'amérique, de l'océan pacifique et enfin de l'Asie.

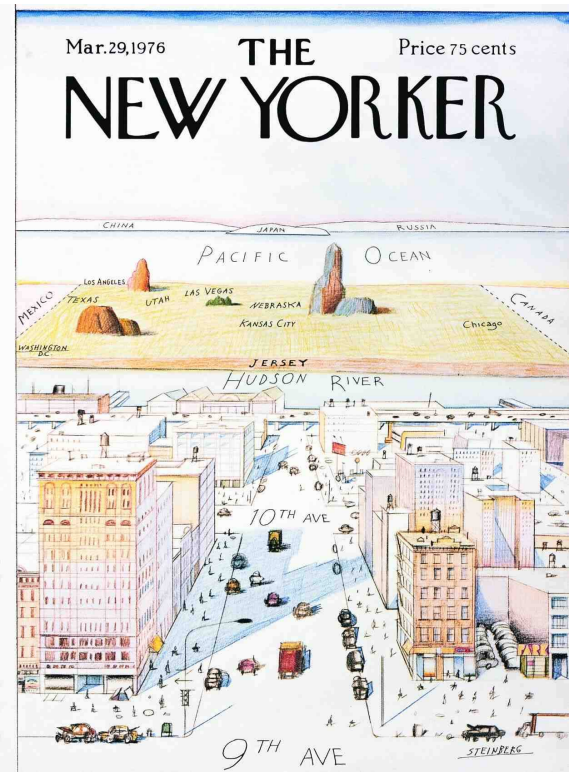


FIG. 5.4 – Saul Steinberg, "View of the World from 9th Avenue" (1976) : Vue "focus+contexte" de l'amérique, le détail étant une rue de New York (en avant plan) et on remarque en arrière plan l'océan pacifique puis le Japon.

Cette idée de présenter le graphe à un niveau de détail faible peut consister à produire une abstraction (un résumé) de celui-ci. Une vue abstraite du graphe permet d'avoir une compréhension globale des données. L'utilisateur peut, dans cette vue abstraite lui donnant un contexte, vouloir localement plus de détails. Ce problème est fondamental dans la visualisation d'informations et a donné lieu à de

nombreuses recherches qui forment à elles seules un domaine qu'on appelle navigation "focus+contexte".

Ce chapitre s'intéresse à ces travaux. Nous allons nous attarder sur les travaux fondateurs de Furnas qui proposent une déformation sémantique de l'espace. Furnas avait nommé lui-même sa technique "fisheye view" faisant écho aux déformations des lentilles des appareils photo. Cette déformation a été comprise comme étant une déformation géométrique, ce qui n'est pas le cas.

5.1 Furnas

Dans "The FISHEYE View : A New Look at Structured Files" [34], Furnas développe un outil de navigation de fichiers code source qui implémente l'idée de Bertin (voir citation 5). En effet, le paradigme focus+contexte par Furnas reprend quasi parfaitement l'idée introduite par Bertin. Le code source est organisé de façon arborescente, sous forme de hiérarchie. Les noeuds internes représentent des blocs qui sont détaillés en sous blocs, ..., jusqu'aux feuilles qui sont des instructions simples (voir figure 5.5). L'édition du code se fait toujours à un niveau de détail le plus fin c'est-à-dire au niveau des feuilles. Dans la figure 5.5, la ligne marquée par des chevrons est celle qui est en cours d'édition. L'idée de Furnas est de permettre à l'utilisateur d'avoir une vue globale sur l'ensemble du code qui tient sur une seule page. Pour ce faire, l'idée est de ne pas montrer (détailler) le contenu des blocs qui sont loin de la ligne en cours d'édition mais d'en donner seulement une indication (voir figure 5.7).

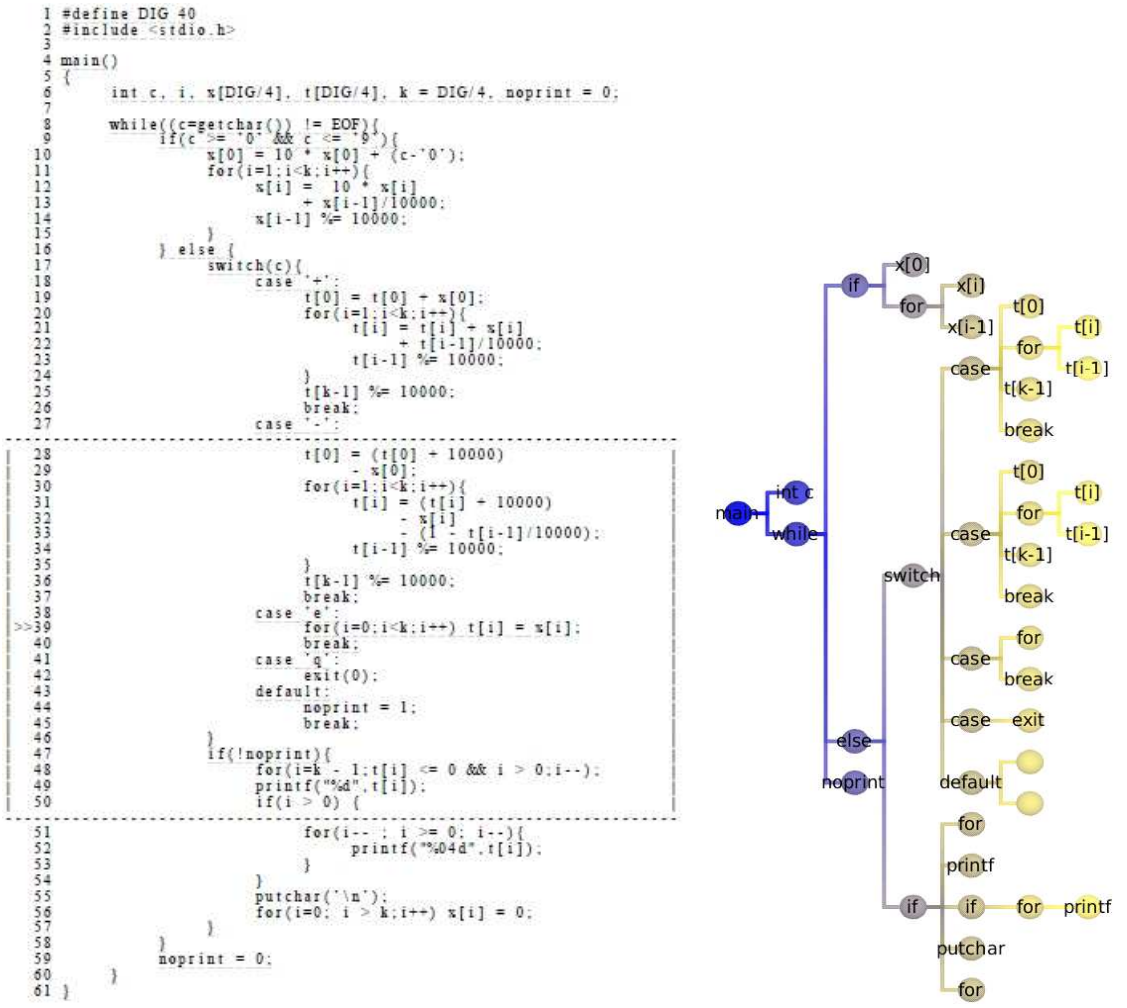
La technique repose sur l'utilisation d'indices qui sont calculés sur la hiérarchie. Ces indices permettent de déterminer le niveau de détail des zones affichées. Chaque élément x de la hiérarchie a un degré à priori d'importance ($API(x)$). Plus on descend dans le détail, plus sa valeur diminue (voir figure 5.6(a)). Par ailleurs, puisque on s'intéresse à un bloc y en particulier (le bloc en cours d'édition par exemple), plus un sommet est loin de y (voir figure 5.6(b)) moins on sera intéressé à l'afficher sauf si par ailleurs il est considéré comme important (à priori). Furnas utilise la distance dans l'arbre $d(x, y)$ pour quantifier cet éloignement mais d'autres approches sont possibles comme nous le verrons plus loin.

Il y a donc un équilibre à trouver entre l'importance à priori (API) et la distance à laquelle on se trouve du centre d'intérêt ($d(x, y)$). L'équilibre est donné par le DOI Degree Of Interest (voir figure 5.6(c)).

$$DOI(x) = API(x) - D(x, y)$$

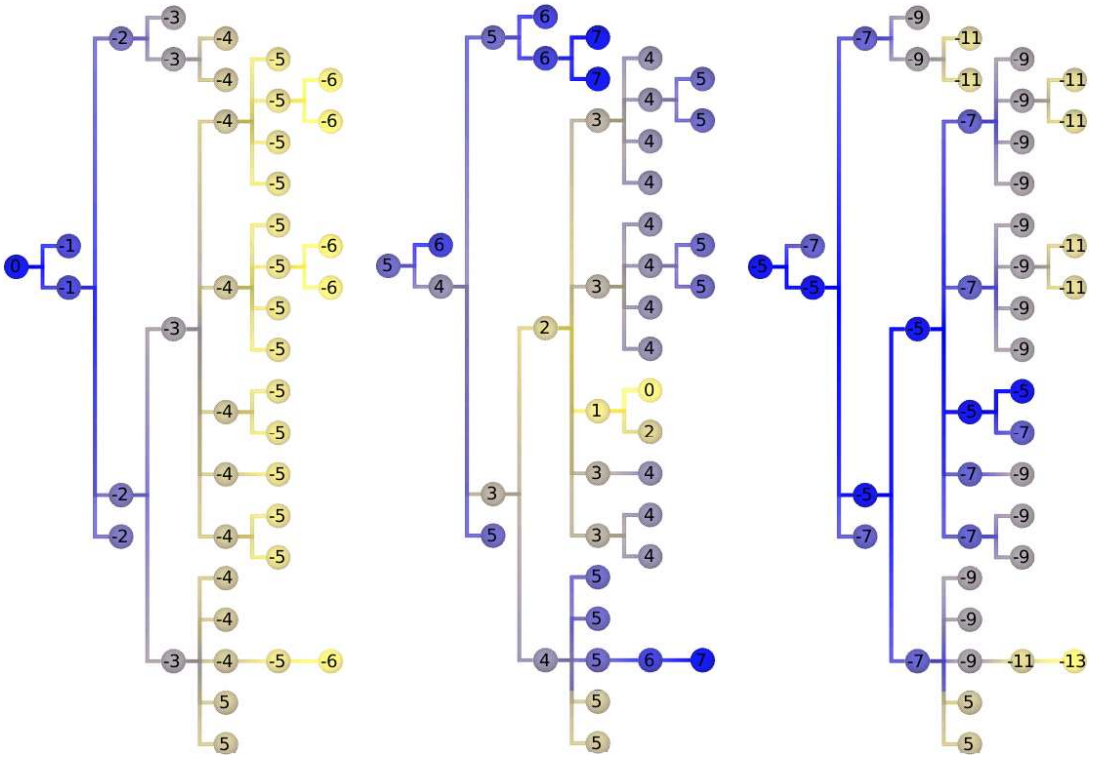
La méthode consiste alors à afficher les sommets dont le DOI est supérieur à un certain seuil. Dans l'exemple de la figure 5.5, si on applique un seuil sur le DOI, on voit toujours la ligne en cours d'édition et le bloc dans lequel elle se situe. Les blocs voisins ne sont pas détaillés. On voit comment ces blocs s'inscrivent dans la boucle "while" qui est directement dans le "main" (voir figure 5.7).

Vue du point de vue du processus de visualisation, la technique proposée par Furnas agit au niveau du "Filtre" (voir figure 5.8). En effet, la technique consiste à filtrer les données suivant leurs positions dans la hiérarchie et une distance au centre d'intérêt.



(a) Code source en format texte (b) Code source sous forme d'arbre

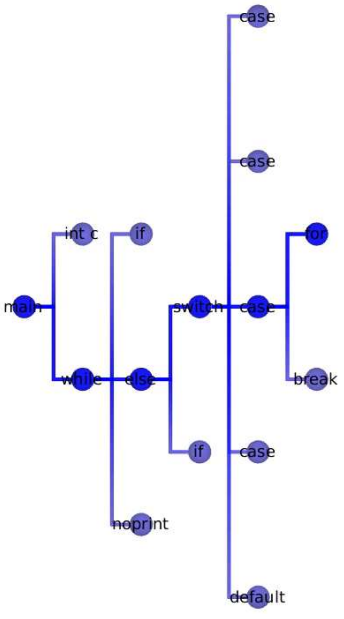
FIG. 5.5 – Code source d'un programme en C.



(a) API (b) Distance de y (c) DOI

FIG. 5.6 – Structures de données permettant la navigation “focus+contexte” de Furnas.

```
    1 #define DIG 40
    2 #include <stdio.h>
...4 main()
    5 {
    6       int c, i, x[DIG/4], t[DIG/4], k = DIG/4, noprint = 0;
...8       while((c=getchar()) != EOF){
    9             if(c >= '0' && c <= '9'){
...16             } else {
   17                   switch(c){
   18                         case '+':
...27                      case '-':
...38                      case 'e':
 >>39                            for(i=0;i<k;i++) t[i] = x[i];
   40                            break;
   41                      case 'q':
...43                      default:
...46                }
   47                if(!noprint){
...57                }
   58             }
   59             noprint = 0;
   60       }
   61 }
```



(a) code filtré (b) hiérarchie filtrée

FIG. 5.7 – Vue “focus+contexte” du code source de la figure 5.5.

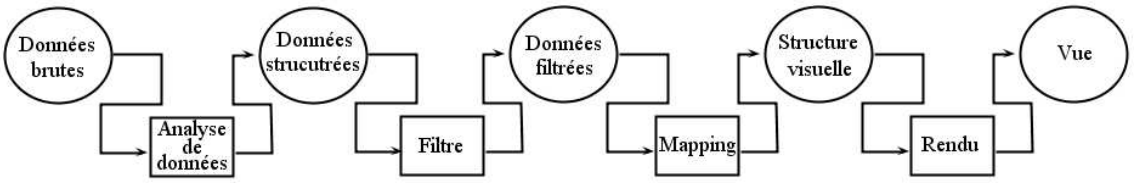


FIG. 5.8 – La navigation “focus+contexte” de Furnas réside dans la partie “filtre” dans le processus de visualisation.

Dans “Degree-of-interest trees : A component of an attention-reactive user interface” [18], puis dans “DOITrees revisited : scalable, space-constrained visualization of hierarchical data” [42], les auteurs présentent une technique s'inspirant des travaux de Furnas afin de visualiser de façon interactive des hiérarchies arborescentes (voir figure 5.9). Ils utilisent la technique de Furnas pour filtrer la hiérarchie et déterminer les éléments à afficher. Cependant, comme ils affichent des graphes (sommets et arêtes) et non pas un fichier texte (comme c'est le cas pour Furnas), ils proposent une solution à la transition de point de vue. Le changement de point de vue est rendu intelligible par une transformation lente et progressive.

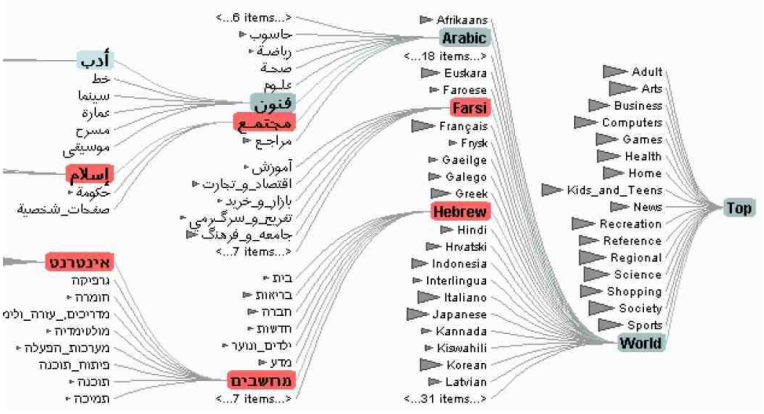


FIG. 5.9 – Visualisation d'une hiérarchie avec DOITree [18].

5.2 Déformation géométrique

D'autres techniques rentrent aussi dans le paradigme “focus+contexte” et procèdent par déformation géométrique. Les méthodes “focus+contexte” sur les graphes en particulier ont été popularisées par des techniques qui pratiquent des déformations géométriques. Le travail de Furnas est resté (même si il a été cité par ces auteurs) un peu à l'écart. Nous allons détailler le travail de déformations géométriques pour montrer le contraste avec la méthode de Furnas.

Les déformations géométriques dites en oeil de poisson (fisheye) sont des techniques qui imitent l'effet produit par la lentille grand angle (comme les lentilles des yeux de poisson). La zone d'intérêt est placée au centre de la lentille et se trouve élargie. Le reste de dessin est alors montré avec de moins en moins de détails plus

on s'éloigne du centre. (voir figure 5.10).

La déformation fisheye de la figure 5.10 montre une déformation qui se passe au point de vue du rendu seulement.

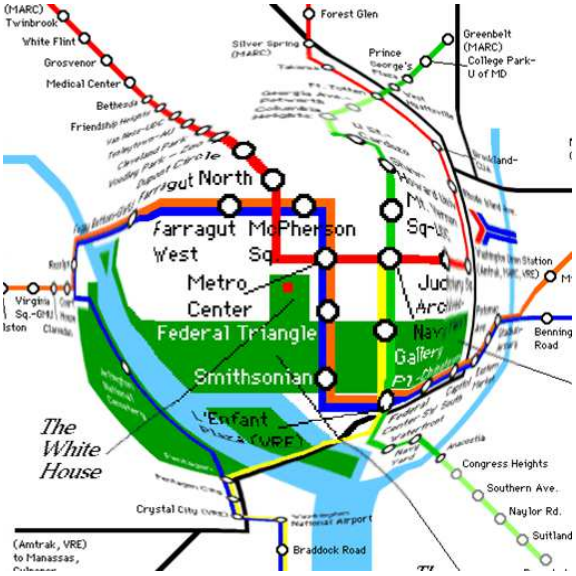
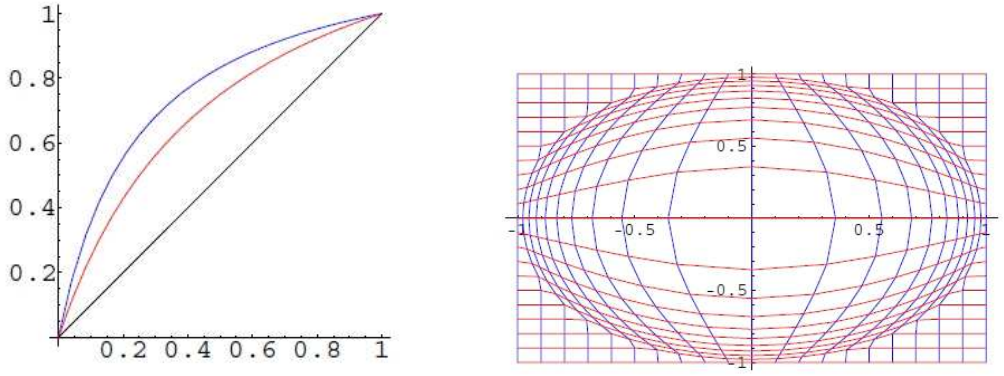


FIG. 5.10 – Vue "fisheye" (en oeil de poisson) d'une carte du métro.

Cette technique consiste à modifier les coordonnées des points dans le plan suivant leur distance à un centre d'intérêt. Partant d'un plongement des données dans le plan, on recalcule la position des sommets au moyen d'une fonction de distorsion $h : [0,1] -> [0,1]$ qu'on étend aux coordonnées des points du plan : $h(x,y) = (h(x), h(y))$.



(a) Fonction de déformation de facteur 2 (rouge) et 4 (bleu)

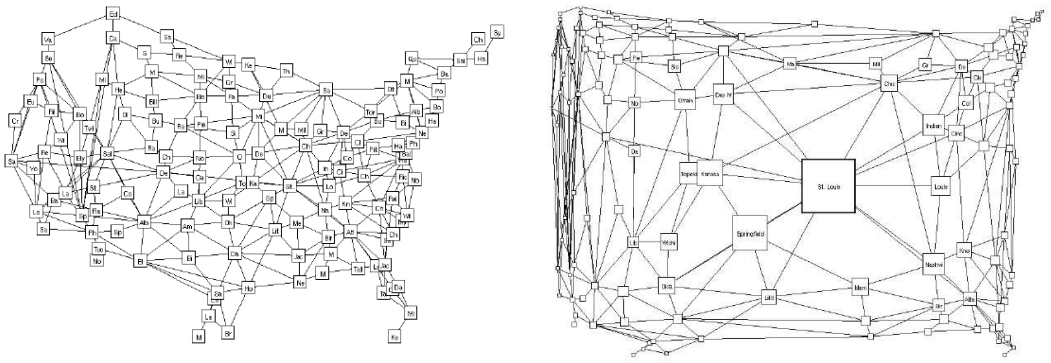
(b) Effet de la déformation sur une grille

FIG. 5.11 – Fonction de déformation appliqué au dessin d'un graphe et effet générique sur le plan.

Pour produire un effet de "fisheye", cette fonction doit être concave et strictement croissante. En effet, proche de 0 (centre d'intérêt), une telle fonction croît rapidement donnant beaucoup d'espace autour du centre d'intérêt. Elle croît beaucoup moins

vite plus on s'approche de 1 donnant ainsi moins d'espace aux points s'éloignant du centre d'intérêt. Dans "Graphical Fisheye Views" [78], Sarkar et Brown proposent une telle fonction avec un paramètre de distorsion d. Ce paramètre doit être positif et plus il est fort, plus la distorsion est importante.

$$h(x) = \frac{(d+1)}{(d+1/x)} \tag{5.1}$$



(a) Image sans déformation (b) Même image avec déformation

FIG. 5.12 – Dessin de graphe des villes des Etats Unis et sa déformation avec le fisheye proposé par Sarkar et Brown. Le centre d'intérêt est ici St. Louis.

Typiquement, la fonction h est appliquée en prenant pour origine le centre d'intérêt qui a été sélectionné ou qui est pointé par la souris. On peut détailler une zone dans le voisinage du centre d'intérêt en modifiant légèrement la fonction et en introduisant un effet de palier (voir figure 5.11(a)). La figure 5.11(b) illustre bien l'effet de la fonction sur une grille. Ce genre de technique peut être étendu.

A ce changement de coordonnées, Sarkar et Brown ajoutent une modification de taille et du niveau de détails (par exemple l'affichage des étiquettes (labels)). La zone proche de l'élément "focus" va avoir une taille plus grande permettant une vue plus claire (cela donne bien l'effet "focus+contexte"). La modification des coordonnées combinée à un affichage adapté des éléments est un processus en deux étapes. La première modifie les coordonnées (déformation géométrique) mettant l'emphase sur le "focus" (étape de Mapping de la figure 5.13). L'étape qui suit implémente le mécanisme de Furnas pour décider de la taille et du niveau de détail (étape de filtrage de la figure 5.13). Cette étape repose sur le DOI qui permet de déterminer si on affiche le label d'un sommet (voir figure 5.12).

De nombreux travaux ont proposé des variantes à divers niveaux sur la forme de la zone, le type de fonction de distorsions appliquées, les filtres utilisés ou encore le nombre de centre d'intérêt appliqué. Nous allons les décrire brièvement ici.

Les travaux de Furnas, Sarkar et Brown ont été suivis par une panoplie de déformations de divers types qui exploitent ces techniques. Dans ces méthodes, le centre d'intérêt est déterminé par un point qui peut être un élément défini par l'utilisateur ou simplement la position du curseur de la souris. On peut étendre ces techniques à une région d'intérêt. Dans "Generalized Fisheye Views of Graphs" [32], les auteurs présentent différentes méthodes afin de sélectionner la zone d'intérêt. En

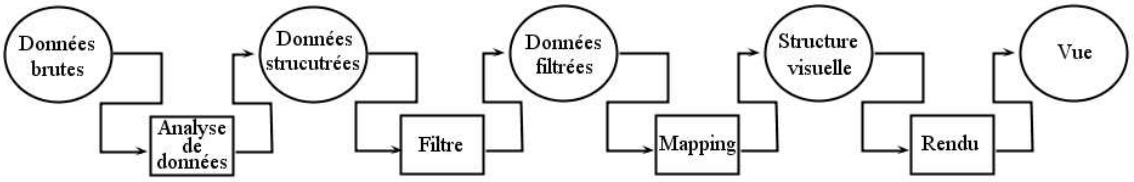
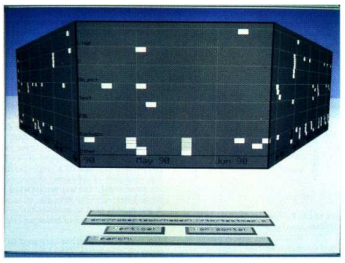


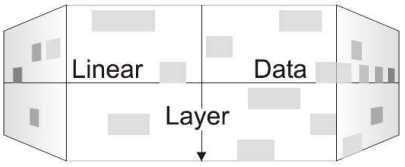
FIG. 5.13 – La navigation “focus+contexte” de Sarkar et Brown réside dans la partie “mapping” pour la déformation géométrique dans le processus de visualisation.

effet, dans [77], la zone d'intérêt est limitée à un polygone convexe. Ils vont étendre la technique aux polygones simples.

Dans “The perspective wall : detail and context smoothly integrated” [63], les auteurs présentent une technique permettant de représenter un grand nombre d'informations à l'aide d'un “mur”. Ce mur est constitué de trois panneaux. Un panneau central montrant le centre d'intérêt de l'utilisateur (le détail) et deux panneaux latéraux montrant le contexte. Les panneaux latéraux sont fuyants pour donner une perceptive 3D (voir figure 5.14). Dans cette perceptive, le niveau de détail du contexte va en décroissant par rapport à la distance du centre d'intérêt (mur central). La méthode Vitesse [69] étend cette idée en projetant les pages web sur différentes surfaces discrétisées (voir figure 5.15).



(a) Perceptive Wall - image d'origine

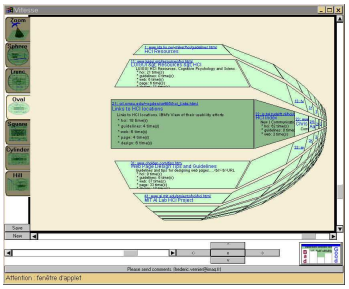


(a) Schematic overview of the Perspective Wall.

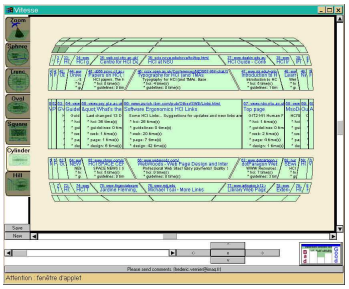
(b) Implementation of the TimeWall by Inxight® Software, 2004.

(b) Perceptive Wall - image plus recente

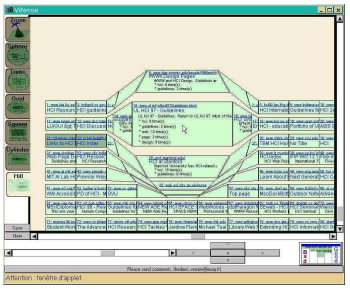
FIG. 5.14 – Différentes vues proposées par la technique Perceptive Wall [63] sur un calendrier.



(a) Vue en ovale



(b) Vue en cylindre



(c) Vue en colline

FIG. 5.15 – Différentes vues sur un ensemble de pages web proposées par l'outil Vitesse [69].

La technique de Sarkar et Brown qui consiste à modifier la taille des éléments peut être appliquée à des tableurs. Dans “The table lens : merging graphical and symbolic representations in an interactive focus+context visualization for tabular information” [74], les auteurs proposent un tableur interactif où l'utilisateur peut déterminer plusieurs centres d'intérêt (voir figure 5.16). Afin de tenir compte des différents “focus”, un indice - DOI (Degree Of Interest) - est calculé pour chaque cellule. Suivant l'idée de Sarkar et Brown [78], cet indice va déterminer la hauteur et la largeur de chaque cellule. La représentation graphique de la cellule est elle aussi déterminée par cet indice. Ainsi, une cellule proche d'un centre d'intérêt aura une hauteur et une largeur grandes et sera pourvue d'une étiquette (label). Une cellule éloignée des centres d'intérêt sera petite et aura comme représentation une couleur.

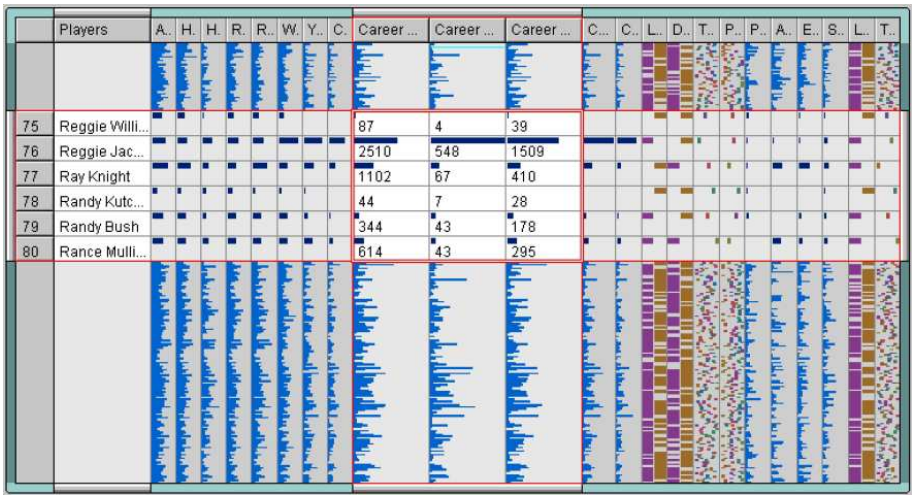


FIG. 5.16 – Vue d'un tableur proposée par TableLens [74].

Cette idée peut tout naturellement être étendue aux calendriers [9]. L'espace gagné et la facilité de navigation prend tout son sens sur des appareils ayant une petite surface d'affichage telle que celle des téléphones portables (voir figure 5.17).

FiCell [1] est un outil permettant une vue avec plusieurs centres d'intérêt sur un tableur. Reprenant l'idée de la vue en colline de Vitesse [69], l'outil offre plusieurs déformations de la ou les zones d'intérêt (voir figure 5.18).

Dans “The Bifocal Tree : a Technique for the Visualization of Hierarchical Information Structures” [21], les auteurs présentent une technique de visualisation “focus + contexte” permettant d'explorer des structures arborescentes. La méthode consiste à agencer deux vues correspondant au focus et au contexte. Chaque vue aura son propre centre d'intérêt d'où le terme “Bifocal”. La vue “focus” aura pour centre d'intérêt un noeud f et montrera l'ensemble de ses fils. La vue “contexte” sera alors centrée sur le père de f et montrera le reste de la hiérarchie. La figure 5.19 présente une telle représentation où le sommet “navbar” est le centre d'intérêt. Une première vue centrée sur “navbar” présente alors l'ensemble du sous-arbre (dont “navbar” est la racine) et constitue la vue “focus”. Une autre vue centrée sur “Heat” (père de “navbar” dans l'arbre) présente le reste de la hiérarchie et constitue la vue “contexte”.

[1] https ://ficell.dev.java.net/

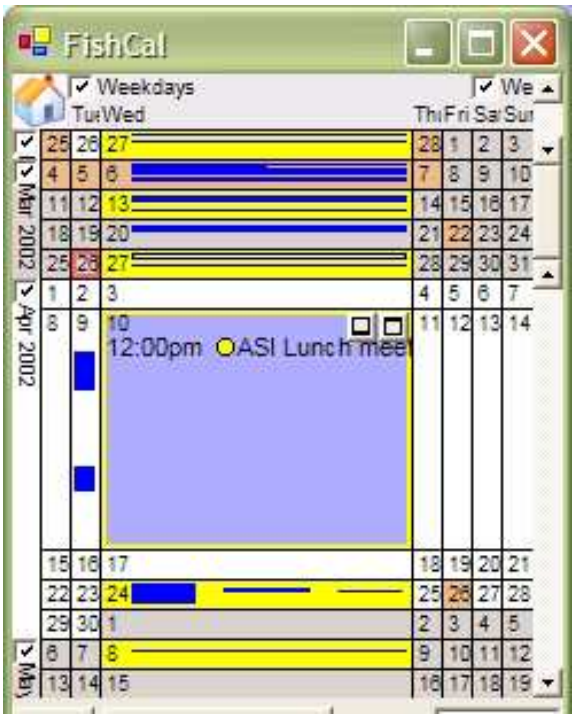
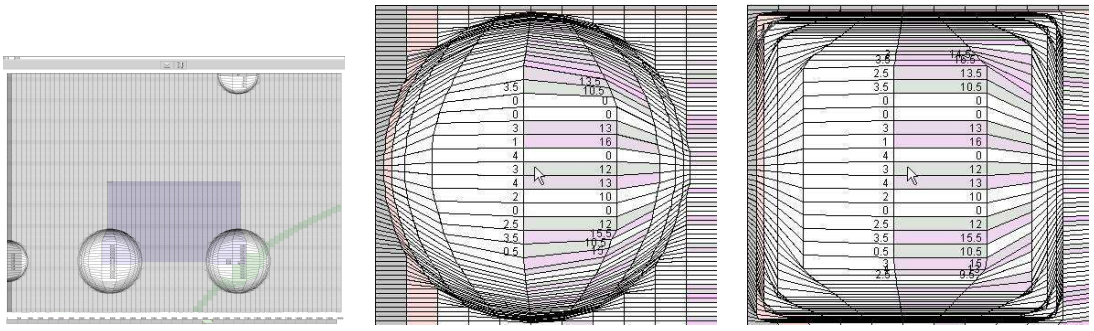


FIG. 5.17 – Vue d'un calendrier sur un PDA proposée par l'outil FishCal [9].



(a) Multi point de vue (b) Focus sphérique (c) Focus pyramidale

FIG. 5.18 – Représentation d'un tableur avec l'outil FiCell mettant en scène plusieurs point de vues (a) et différentes lentilles (b) et (c).

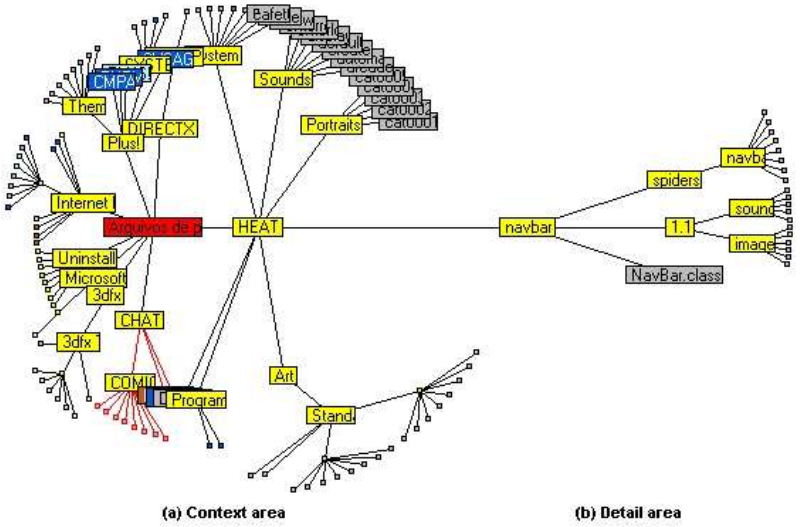


FIG. 5.19 – Représentation d'une hiérarchie (370 noeuds) proposée par l'outil Bifocal Tree [21]. Le dessin est articulé autour de deux noeuds : le centre d'intérêt ("navbar") et son père dans la hiérarchie ("HEAT"). La partie gauche du dessin montre l'ensemble de la hiérarchie en partant du père, cette partie correspond au contexte. La partie droite montre l'ensemble des fils du centre d'intérêt et correspond au détail de la vue.

La méthode de Sarkar et Brown a inspiré directement de nombreux auteurs. Dans "Fisheye menu" [8], la technique présentée n'utilise que la fonction de DOI pour afficher les éléments d'un menu. La méthode définit une zone d'intérêt autour du centre d'intérêt défini par l'utilisateur. La taille de cette zone est paramétrable par l'utilisateur. Dans la figure 5.20, le centre d'intérêt est "Mauritius". Les éléments entourant "Mauritius" ont la même taille. Hors de la zone d'intérêt, la taille des éléments commence à décroître.

Géométrie hyperbolique et illusion de déformation D'autres types de visualisation produisent des vues ressemblant à une déformation fisheye. Dans [57, 58, 68], la déformation "focus+contexte" géométrique découle de l'utilisation de la géométrie hyperbolique. Cette géométrie qui remet en question l'axiome des parallèles dans la géométrie euclidienne, induit une courbure naturelle de l'espace (voir [45, 41, 39, 73] pour plus de détails). La déformation sur l'écran découle alors du passage de la géométrie abstraite à un modèle 2D ou 3D (modèle de Poincaré [57] ou modèle de Klein [68] (voir figure 5.21), et non pas d'un changement des coordonnées des sommets du graphe. L'implémentation de la technique est au niveau du Rendu (voir figure 5.22) par le biais d'instructions données directement à la carte graphique.

5.3 Conclusion

Dans ce chapitre, nous avons présenté les travaux de Furnas et de Sarkar et Brown qui sont les travaux fondateurs de tout le courant de recherche en visualisa-

FIG. 5.20 – Fisheye Menu [8] : le centre d'intérêt est sur "Mauritius". Plus on s'éloigne de l'élément "Mauritius", plus la taille des éléments est petite.

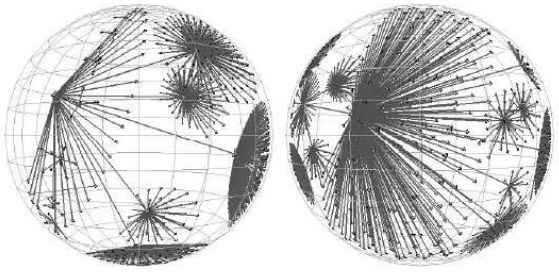


FIG. 5.21 – Déformation Hyperbolique [68].

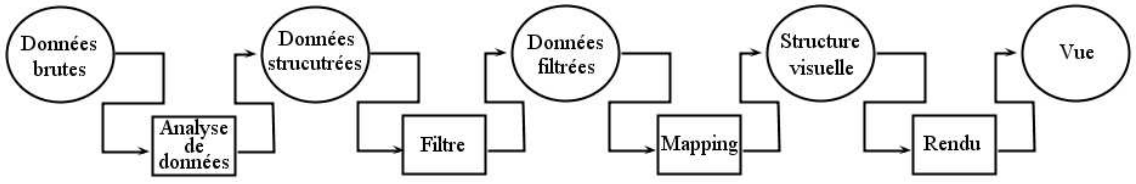


FIG. 5.22 – La navigation "focus+contexte" par la géométrie hyperbolique réside dans la partie "rendu" pour la déformation géométrique dans le processus de visualisation.

tion “focus+contexte”. A première vue, on peut opposer les travaux de Furnas aux travaux de Sarkar et Brown. En effet, Furnas effectue un filtrage sur la donnée alors que Sarkar et Brown calculent une déformation des coordonnées dans l'espace de dessin. Cependant, les différences et les similitudes de leurs travaux sont beaucoup plus fines. Dans ces deux approches, une distance est calculée entre les éléments de la donnée. Dans le cas de Furnas, cette distance correspond à une distance dans l'arbre et dans le cas de Sarker, elle correspond à une distance dans le graphe (plus court chemin entre les éléments).

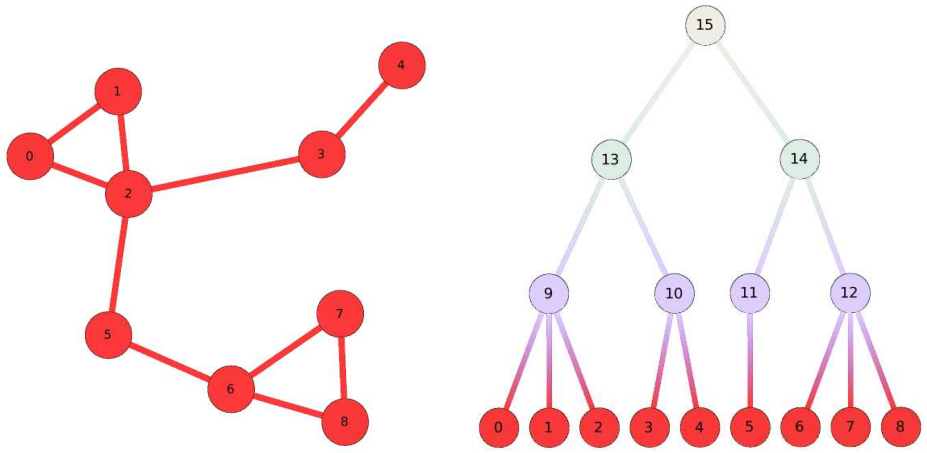
Dans le cas de Furnas, le fait que les données soient hiérarchisées n'intervient que dans le calcul de l'importance à priori des éléments (API). Passé cette étape, le caractère hiérarchique de la donnée n'est plus utilisée. Cela vient du fait que la hiérarchie utilisée coïncide avec les données à visualiser. Dans le cas de Sarkar et Brown, aucune hiérarchie n'est présente, ni dans les données, ni en arrière scène.

Dans le chapitre suivant, une troisième approche sera proposée consistant à calculer et à exploiter la hiérarchisation des données. La hiérarchie ne coïncide plus alors forcément avec les données à visualiser et devient un outil de filtrage qui reste en arrière plan.

Chapitre 6

Visualisation abstraite : contexte

Nous allons nous pencher dans ce chapitre sur la notion d'agrégation. Cette notion apparaît naturellement lorsqu'on veut visualiser des graphes de grande taille. Voyons sur un exemple simple le principe de fonctionnement. On peut, à partir du graphe (par exemple celui de la figure 6.1(a)), agréger les sommets qui sont les plus proches les uns des autres. Le nouveau graphe ainsi construit possède moins de sommets et sa lisibilité est alors meilleure (voir figure 6.2(a)). En réitérant cette construction, on obtient une visualisation avec plusieurs niveaux de détail comme le montre la figure 6.2. Le processus d'agrégation est complètement capturé par une structure annexe 6.1(b). Ce genre de mécanisme est central pour notre propos.

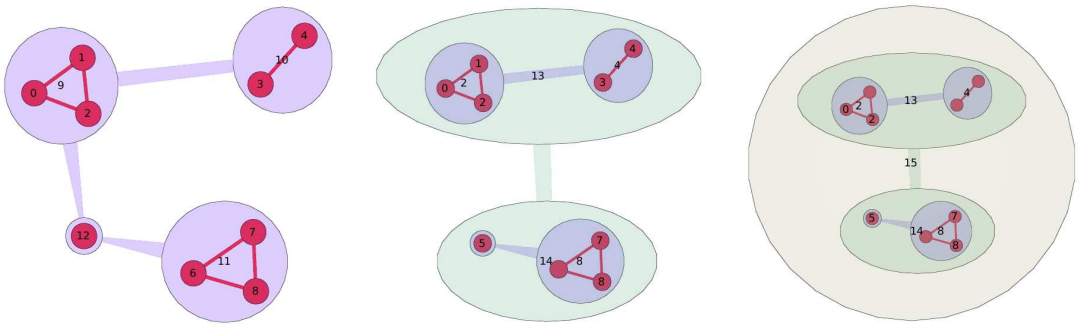


(a) Graphe d'origine (b) Hiérarchie associée

Fig. 6.1 – Graphe et hiérarchie associée.

Dans "Topological Fisheye Views for Visualizing Large Graphs" [36], Gansner et al. présentent une technique pour visualiser de très gros graphes. Les auteurs prennent soin d'agréger des sommets voisins dans le graphe en respectant des critères topologiques : la contraction de sommets en un nouveau sommet préserve globalement la structure (voir figure 6.3).

Des sommets "proches" induisent un méta sommet, et les métas sommets forment ainsi un nouveau graphe donnant une vue abstraite du graphe de départ. Le regroupement suit quatre principes : produire des clusters (méta sommets) de



(a) Abstraction 1 (b) Abstraction 2 (c) Abstraction 3

FIG. 6.2 – Différentes abstractions du graphe de la figure 6.1 suivant la hiérarchie associée. Les abstractions se font ici par niveau dans la hiérarchie.

taille comparable, regrouper des sommets proches dans le dessin initial, respecter la topologie du graphe (ne pas regrouper des sommets qui induirait des cycles qui n'existent pas dans le graphe d'origine) et enfin la méthode doit avoir une complexité en temps raisonnable.

L'introduction d'un méta sommet représentant des sommets du graphe d'origine correspond à la construction d'une hiérarchie. Une coupe dans cette hiérarchie correspond elle à l'une des abstractions à un certain niveau de détail. Les auteurs définissent ainsi différents graphes abstraits du graphe d'origine, correspondant à un certain niveau dans la nouvelle hiérarchie. Le niveau de détail choisi correspond alors à une opération sur la hiérarchie : choisir un niveau de détail revient à déterminer les sommets de la hiérarchie qui vont être affichés.

La figure 6.3 montre différentes abstractions d'un graphe en suivant les principes de Gansner et al. Chaque graphe correspond à une coupe de la hiérarchie décrivant l'agrégation. Dans le cas de la figure 6.2, les graphes proposés sont des abstractions du graphe de la figure 6.1. Ces abstractions correspondent à des coupes exactement horizontales de la hiérarchie.

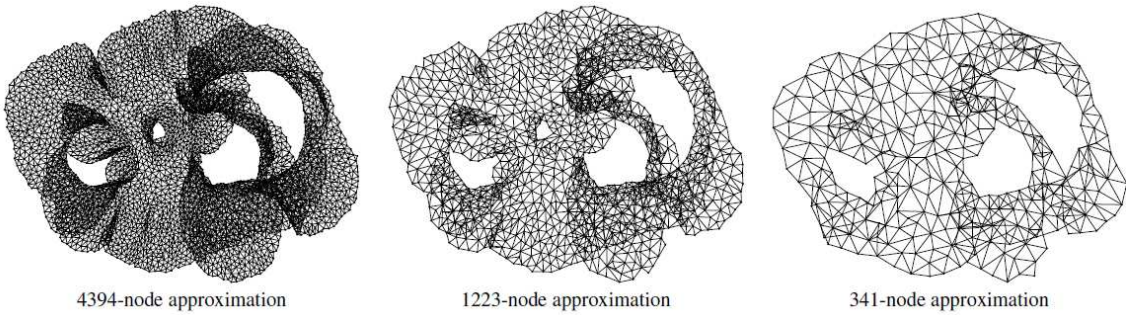


FIG. 6.3 – Trois différentes abstractions d'un même graphe.

Cette approche est très différente de ce qu'on a présenté dans le chapitre précédent. La hiérarchie permet de décider quels sont les éléments à afficher, cependant la hiérarchie elle-même reste en arrière plan : elle n'est pas affichée. Cette situation est différente de Furnas et de Sarkar et Brown qui ne calculent pas de structure annexe

pour piloter la navigation “focus+contexte”. En effet, Furnas utilise la hiérarchie pour le filtrage et affiche cette dernière. Dans le cas de Sarkar et Brown, on dispose d'un graphe et le filtre appliqué consiste à calculer un DOI sur le graphe. La méthode proposée par Gansner et al. calcule un DOI sur la hiérarchie qui est dérivée de la donnée originale.

Cette notion d'agrégation et ses liens avec les techniques “focus+contexte” sont présents dans les travaux de van Wijk et van Ham. Nous allons revenir de façon assez détaillée sur ces travaux dont on s'est inspiré. La présentation qu'on en donne est assez proche du travail d'Abello [1] qui formule cette notion de façon algébrique.

6.1 Abstraction

La hiérarchie est une structure annexe qui est construite et est utilisée pour piloter l'affichage correspondant à une abstraction. On peut calculer une hiérarchie à partir de la donnée de départ. Dans cette hiérarchie, une coupe nous donne un niveau de détail global.

Revenons sur l'exemple de Gansner et al. Le graphe qu'ils visualisent est assez particulier parce qu'il discrétise une surface 2D et présente donc une forte régularité. Le travail d'agrégation consiste à passer d'un maillage d'une certaine granularité à un maillage de granularité plus grosse. Cette régularité dans le graphe et dans le travail d'agrégation fait en sorte que les coupes donnant des abstractions soient quasi horizontales.

On va s'intéresser à des graphes où cette régularité n'est pas présente. Par exemple, dans le graphe de la figure 6.4, l'oeil de façon préemptif fractionne le dessin en zones plus denses. L'oeil capture immédiatement le fractionnement du graphe en plusieurs zones grossières. On peut se donner comme objectif de calculer des abstractions du graphe de départ qui iraient d'une vue détaillée jusqu'à une vue très abstraite. La question est alors de savoir comment on peut déterminer le regroupement des éléments pour faire les différentes transitions.

Nous allons présenter en détail les travaux de van Wijk et van Ham qui ont étendu les idées de Furnas pour définir un indice permettant de piloter ce changement de détail. Les travaux de van Wijk et van Ham ont été à leur tour une inspiration pour nous, ce qui explique la place que nous leur laissons ici.

La méthode proposée par van Wijk et van Ham consiste à définir un indice qui ferait cette transition. Le DOA (Degree Of Abstraction) est une valeur numérique comprise entre 0 et 1 représentant le degré d'abstraction global de la visualisation du graphe. Un DOA de 0 (voir graphe 6.5(a)) correspond au graphe d'origine. Plus on s'approche de 1, plus l'abstraction du graphe est importante, affichant de moins en moins de sommets (voir graphes de la figure 6.7). Quand le $DOA = 1$, la visualisation du graphe se ramène à un seul sommet. Un seul paramètre fait donc varier le niveau de détail de la vue.

6.1.1 Utilisation du DOA

Van Wijk et van Ham se donnent comme point de départ un dessin du graphe. L'agrégation des sommets va se faire en fonction de la distance euclidienne entre

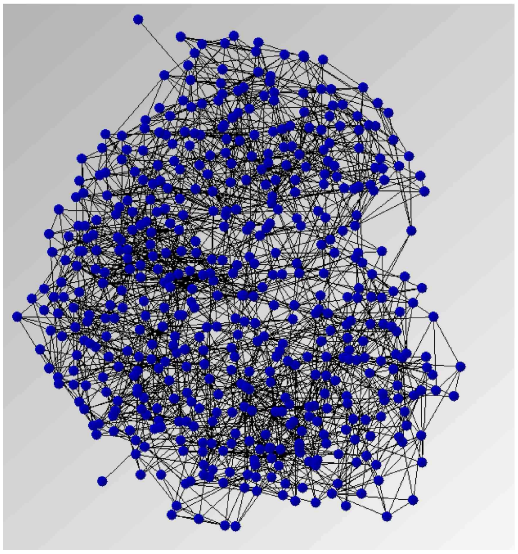
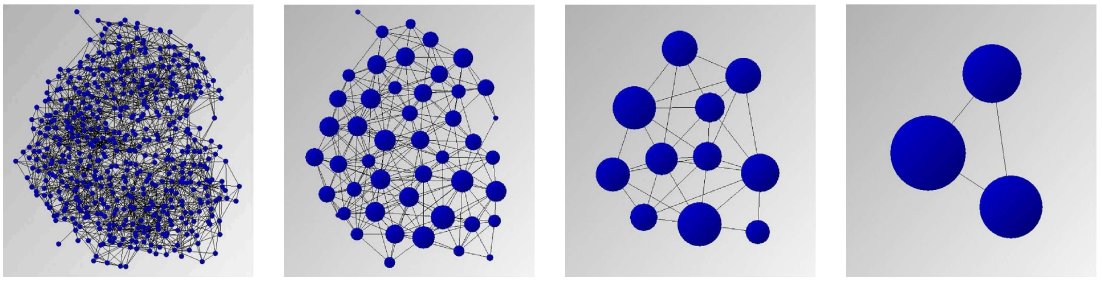


FIG. 6.4 – Graphe représentant un réseau d'artistes peintres. Les liens sont de natures diverses (courants communs, collaborations, etc...).



(a) $DOA = 0$ (b) $DOA = 0.2$ (c) $DOA = 0.4$ (d) $DOA = 0.7$

FIG. 6.5 – Différentes vues abstraites suivant la valeur du DOA.

eux : les sommets proches dans le dessin vont être regroupés en clusters comme l'illustre le passage de la figure 6.5(a) à (b). Pour réitérer la procédure (de (b) à (c)), il faut définir la distance entre les clusters comme étant la distance moyenne entre les éléments les composant. Dans la figure 6.5(b), les sommets représentés ne sont pas des sommets du graphe de départ. Ce sont des sommets qui ont été calculés, des sommets qui représentent un sous-groupe des sommets de départ. On introduit ces sommets (clusters) comme abstraction d'un sous-ensemble de sommets. Les interactions entre ces nouveaux sommets sont déduites des interactions qu'il y avait entre les sommets des sous-groupes dans le graphe de départ. Il y a donc un travail d'agrégation qui est fait en amont de la représentation. La même procédure peut être itérée pour passer de la figure (b) à la (c), et ainsi de suite. Comme pour Gansner et al.[36], le passage du graphe de départ à ses abstractions correspond à calculer un arbre qui devient une structure annexe. Différents algorithmes peuvent être utilisés pour effectuer l'agrégation selon les propriétés du graphe. On ne fera pas ici un état de l'art sur ces techniques. Le lecteur intéressé pourra consulter les aricles [16, 79].

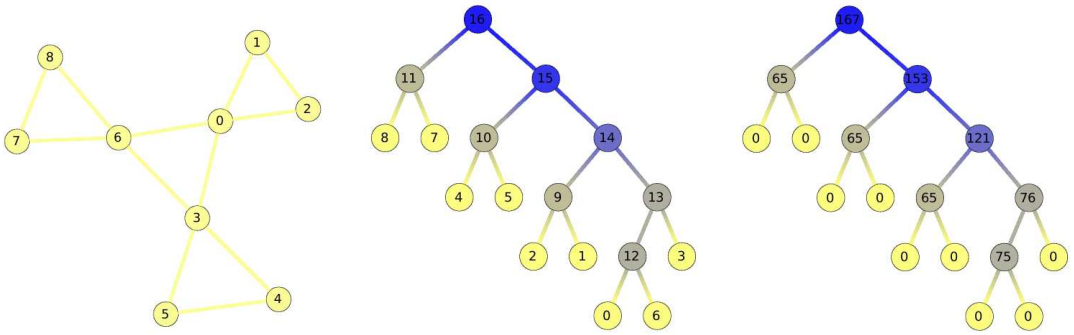
On peut alors utiliser cette hiérarchie pour n'afficher qu'un certain nombre de sommets représentant le graphe. On exploite l'arbre à des fins d'affichage de manière assez simple. Les affichages de la figure 6.5 se déduisent de l'arbre de clustering en sélectionnant les sommets à un certain niveau dans l'arbre. A l'évidence, la racine de cette arbre comprend trois fils puisque le graphe de la figure 6.5(d) comporte 3 sommets (3 clusters). Cette vue correspond au niveau de détail le plus haut.

Nous allons maintenant introduire un mécanisme permettant de sélectionner ces coupes dans la hiérarchie. Ce mécanisme ne se sert pas du paramètre de profondeur dans l'arbre mais plutôt d'un paramètre numérique entre 0 et 1 : le DOA.

Pour chacun des clusters k de notre arbre est calculé un attribut *dispersion*, noté d. Cet attribut d correspond à la distance entre ses clusters fils. Les clusters feuilles de la hiérarchie correspondent aux sommets d'origine du graphe. N'ayant pas de cluster fils, la *dispersion* des clusters feuilles est fixée à 0. L'attribut d est croissant. En effet, la distance moyenne entre sommets est de plus en plus grande quand on monte dans la hiérarchie : le nombre de sommets qu'un cluster représente augmente. Ainsi, un cluster k doit avoir une *dispersion* d_k supérieure ou égale à la *dispersion* de tous ses fils, la *dispersion* la plus grande étant celle de la racine.

La figure 6.6(a) montre un graphe G. La hiérarchie associée est illustrée en 6.6(c). Les sommets du graphe G sont les feuilles de cette hiérarchie et ses sommets internes représentent les clusters. La figure 6.6(b) présente les étiquettes ou identifiants des clusters. La figure 6.6(c), quant à elle, présente l'attribut *dispersion* des clusters. La hiérarchie est ici calculée en regroupant les sommets deux à deux suivant la distance euclidienne. Ce type de regroupement produit une hiérarchie binaire. L'attribut *dispersion* représente ainsi la distance (en terme de pixels) entre les clusters. Les sommets du graphe G ont bien une *dispersion* égale à 0 étant des feuilles.

Soit *root* le cluster racine de notre hiérarchie, d_{root} sa *dispersion*. Soit k un cluster, d_k sa *dispersion*, fk le cluster père de k de *dispersion* d_{fk}. La dispersion étant croissante, on a donc $d_k \leq d_{fk} \leq d_{root}$. van Wijk et van Ham [87] propose une façon simple d'obtenir une abstraction d'un graphe clusterisé pour chaque valeur de DOA. Ne seront affichés à l'écran que les clusters k pour lesquels :

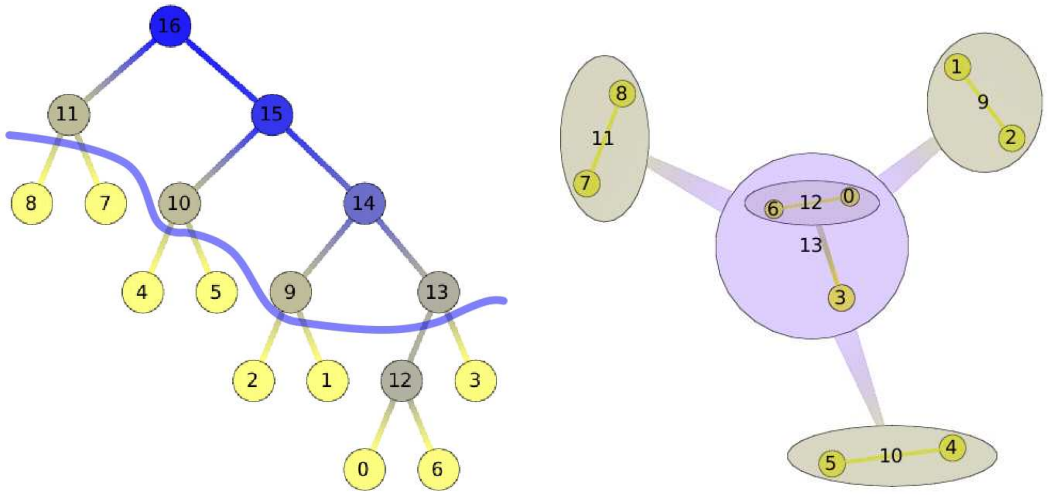


(a) Graphe d'origine (b) Identifiant (c) Dispersion

FIG. 6.6 – Graphe et hiérarchie associée. La dispersion est ici la distance en pixels entre les sommets.

$$d_k \leq DOA \times d_{root} < d_{fk}. \tag{6.1}$$

La figure 6.7 montre la sélection sur la hiérarchie des clusters respectant ces inégalités pour un $DOA = 0.5$ et l'abstraction du graphe en résultant.



(a) Coupe dans la hiérarchie (b) Abstraction du graphe

FIG. 6.7 – Coupe dans la hiérarchie et abstraction pour un DOA égal à 0.5.

La valeur d_{root} étant strictement positive et aussi la plus grande *dispersion* des clusters, on a donc :

$$\frac{d_k}{d_{root}} \leq DOA < \frac{d_{fk}}{d_{root}} \text{ avec } \frac{d_k}{d_{root}}, \frac{d_{fk}}{d_{root}} \in [0, 1]$$

Le ratio $\frac{d_k}{d_{root}}$ correspond au degré d'abstraction global du cluster k. On peut alors fixer comme seuil le DOA et ne prendre que les clusters dont ce ratio est inférieur ou égal à ce seuil ($\frac{d_k}{d_{root}} \leq DOA$) et dont le ratio des pères est lui supérieur strictement à ce seuil ($DOA < \frac{d_{fk}}{d_{root}}$).
Il est à noter qu'un cluster k représente un ensemble e_k de sommets du graphe

d'origine. Le père fk de ce cluster représente lui aussi cet ensemble e_k plus les ensembles e_i de tous ses autres fils. Donc, si on affiche un cluster, on ne peut pas afficher en même temps un de ses descendants ou un de ses ancêtres, auquel cas on aurait redondance d'informations. La condition posée par van Wijk et van Ham [87] pour afficher un cluster respecte cette observation :

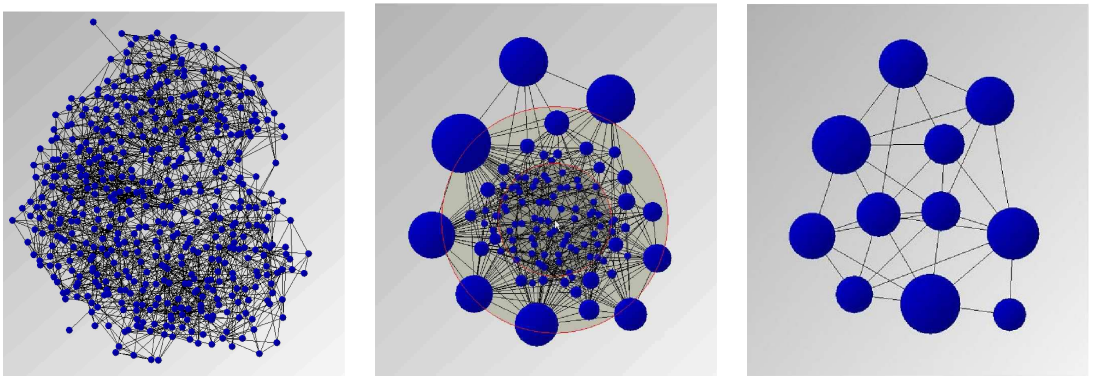
- si on a $\frac{d_k}{droot} \leq DOA < \frac{d_{fk}}{droot}$, on ne peut avoir d_{ck} *dispersion* du fils ck du cluster k telle que $\frac{d_{ck}}{d_{root}} \leq DOA < \frac{d_k}{d_{root}}$ car $\frac{d_k}{d_{root}} \leq DOA$. Ainsi on ne peut afficher en même temps un cluster et son descendant.
- De même, les ancêtres de ce cluster ne seront pas affichés : on ne peut avoir d_{ffk} *dispersion* du père ffk de fk telle que $\frac{d_{fk}}{d_{root}} \leq DOA < \frac{d_{ffk}}{d_{root}}$ car $DOA < \frac{d_{fk}}{d_{root}}$.

En d'autres mots, les clusters sélectionnés déterminent une coupe dans l'arbre. C'est-à-dire que chaque feuille est dominée par un élément de la coupe et par exactement un seul. Deux éléments de la coupe ne peuvent pas se dominer entre eux.

Remarque On peut voir la hiérarchie comme le diagramme de Hasse d'un ensemble ordonné. Les sommets sont ordonnés de la racine jusqu'aux feuilles et l'arbre ne représente que les relations de couverture minimale : on supprime toutes relations qui se déduisent par transitivité. Une coupe correspond alors à une antichaine maximale. Cette notion algébrique a été exploitée dans les travaux d'Abello [1] pour optimiser l'affichage et la manipulation de très gosses données.

6.1.2 *DOA* et navigation "focus+contexte"

Grâce à l'indice *DOA* associé à une structure annexe, on peut calculer une abstraction d'un graphe. Nous allons maintenant voir comment faire évoluer cet indice vers une fonction pour tenir compte d'un centre d'intérêt. A l'image de ce qu'ont fait Sarkar et Brown, on veut obtenir un niveau d'abstraction (DOA) faible quand on est proche du centre d'intérêt afin d'avoir le maximum de détails et fort lorsqu'on s'en éloigne. Mais cette fois, on va exploiter la hiérarchie à cette fin. Le *DOA* considéré pour filtrer un sommet doit donc varier suivant la distance de ce sommet au centre d'intérêt.



(a) Graphe d'origine (b) Vue "focus+contexte" (c) Graphe abstrait

FIG. 6.8 – Vue "focus+contexte" d'un graphe.

Dans un graphe $G = (V, E)$, supposons donné un centre d'intérêt $f \in V$ dit point focal. Basée sur la distance s d'un sommet au point focal f, van Wijk et van Ham [87] définit une fonction $DOA(s)$ permettant de faire varier le DOA suivant la distance au point focal f. Ainsi, proche du centre d'intérêt, le degré d'abstraction servant à filtrer la hiérarchie est faible et plus on s'éloigne, plus il est fort (voir figure 6.11).

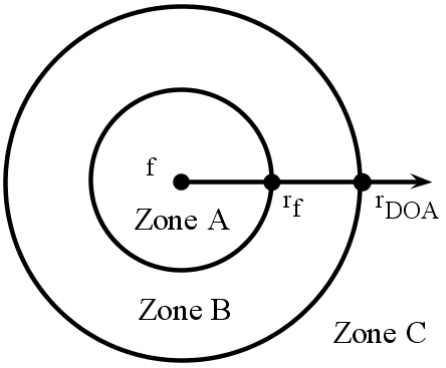


FIG. 6.9 – Définition des zones suivant un centre d'intérêt f.

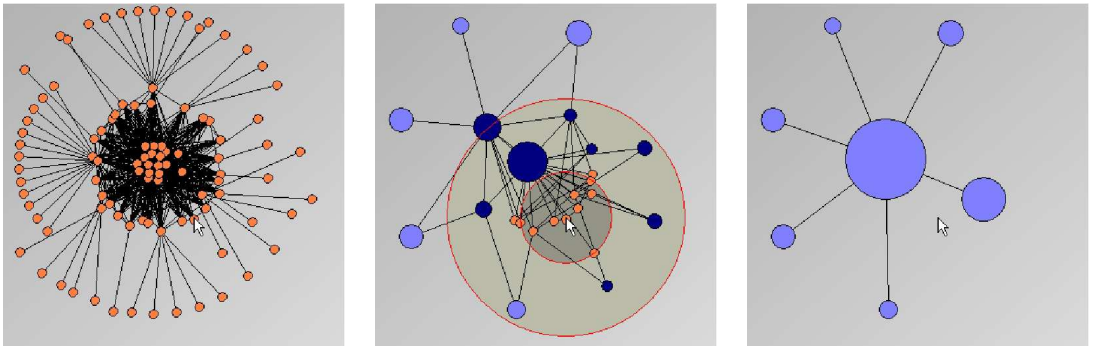
Cette fonction est définie par morceaux divisant l'écran en trois zones concentriques de centre f. La figure 6.9 explicite ces zones. Ces zones ont des niveaux d'abstraction de plus en plus forts à mesure qu'on s'éloigne du centre d'intérêt. Une première zone A de rayon r_f montre un niveau de détails constant maximum équivalant à une abstraction de $DOA = 0$. Cette zone correspondant au centre d'intérêt de l'utilisateur. Une deuxième zone B en anneau autour de la zone A de largeur $r_{DOA} - r_f$ montre un niveau de détails variable moins élevé. Enfin, une dernière zone C correspondant au reste de l'écran, montre un niveau de détails constant correspondant au DOA global. On calcule $DOA(s)$ de la façon suivante :

$$DOA(s) = \begin{cases} 0 & \text{si} \quad s \leq r_f \\ a(s) & \text{si} \quad r_f < s < r_{DOA} \\ DOA & \text{si} \quad s \geq r_{DOA} \end{cases}$$

où $a(s)$ est une fonction croissante qui fait la transition entre les zones A et C. Un candidat typique pour la fonction $a(s)$ est une fonction linéaire consistant à faire varier le détail de façon linéaire comme le montre la figure 6.11.

Zone A La zone A est la zone d'intérêt de l'utilisateur, il faut donc lui donner le maximum de détails. Pour ce faire, dans cette zone, on fixe le DOA à 0 affichant ainsi les sommets d'origine. Dans la figure 6.10, les clusters en orange représentent les clusters feuilles (sommets d'origine). On retrouve bien dans la zone A ces sommets oranges.

$$DOA(s) = 0 \qquad \text{si} \quad s \leq r_f$$



(a) Graphe d'origine (b) Vue "focus+contexte" (c) Graphe abstrait

FIG. 6.10 – Graphe de $DOA = 0.6$, $r_f = 50$ et $r_{DOA} = 130$.

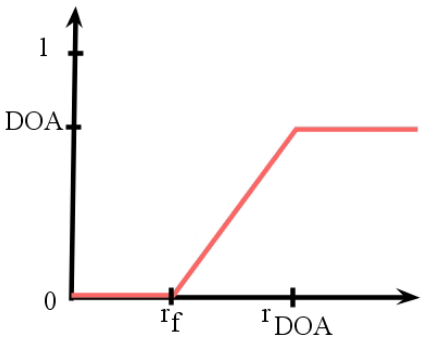


FIG. 6.11 – Fonction de distortion $DOA(s)$ utilisée par van Wijk et van Ham.

Zone C La zone C est la zone donnant le contexte général. Cette zone a le moins de détails, son degré d'abstraction est fixé à DOA global. Dans la figure 6.10, les sommets en bleu pâle représentent les clusters de DOA global. On retrouve bien dans la zone C ces sommets bleu pâle.

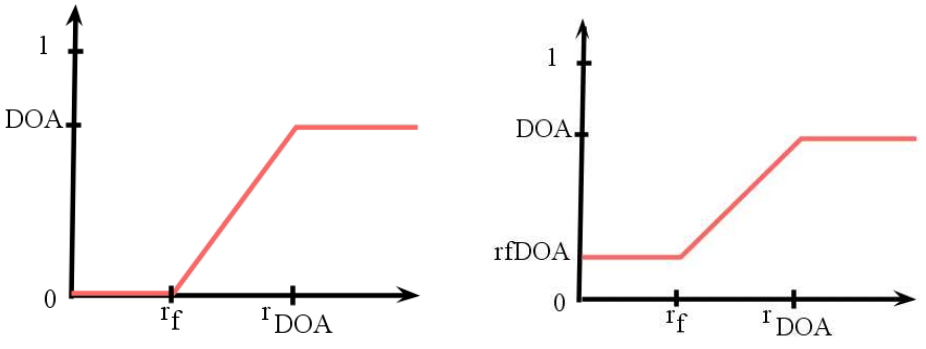
$$DOA(s) = DOA \qquad \text{si} \quad s \geq r_{DOA}$$

Zone B La zone B est une zone qui doit faire la transition entre la zone A où l'on a un maximum de détails et la zone C où l'on a le moins de détails. Le niveau d'abstraction d'un cluster k varie donc de 0 (degré d'abstraction de la zone A) à DOA global (degré d'abstraction de la zone C) suivant sa distance au point focal. Cette transition est faite grâce à une fonction $a(s)$ croissante. Ainsi, plus on s'approche de la zone A, plus les détails augmentent et plus on s'approche de la zone C, plus les détails diminuent. On remarque dans la figure 6.10(b) qu'en bordure de la zone A on a des clusters feuilles et en bordure de la zone C on a des clusters de DOA global. La fonction $a(s)$ est dans cette figure une fonction linéaire.

$$DOA(s) = a(s) \qquad \text{si} \quad r_f < s < r_{DOA}$$

Introduction d'un seuillage Pour des graphes très grands, un niveau de détails maximal ($DOA = 0$) dans la zone A peut nuire à la lisibilité. Ainsi, on doit pouvoir faire varier le niveau de détails dans cette zone. On introduit un seuil $rfDOA$ correspondant au niveau d'abstraction minimal souhaité pour cette zone (voir figure 6.13). On modifie donc la fonction $DOA(s)$ (voir figure 6.12) :

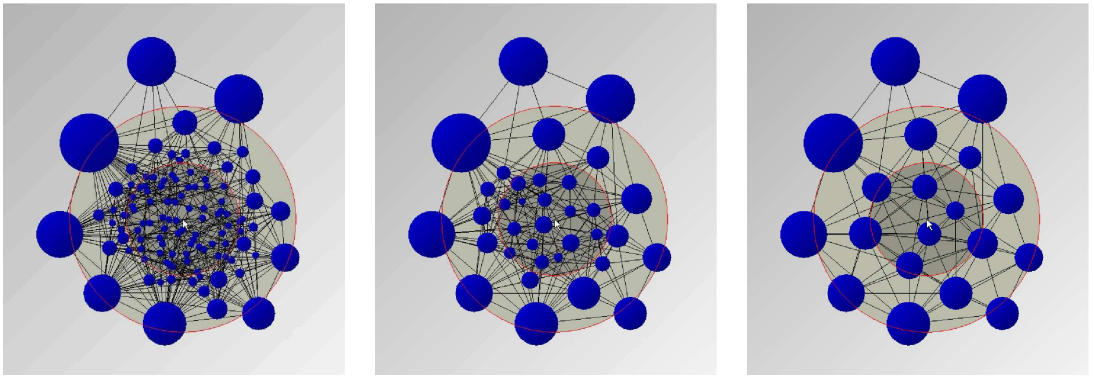
$$DOA(s) = \begin{cases} rfDOA & \text{si} \quad s \leq r_f \\ a(s) & \text{si} \quad Rf < s < r_{DOA} \\ DOA & \text{si} \quad s \geq r_{DOA} \end{cases}$$



(a) Fonction de distorsion (b) Fonction de distorsion avec seuil

FIG. 6.12 – Fonction de distortion $DOA(s)$ avec l'introduction d'un seuil.

La zone B est, elle aussi, modifiée. La transition entre la zone A et la zone C, fait varier le DOA de $rfDOA$ à DOA global affichant ainsi moins de détails dans la zone B.



(a) $rfDOA = 0$ (b) $rfDOA = 0.1$ (c) $rfDOA = 0.2$

FIG. 6.13 – Visualisation d'un graphe pour $rfDOA$ égal à 0, 0.1 et 0.2 pour un $DOA = 0.4$ et $a(s)$ linéaire.

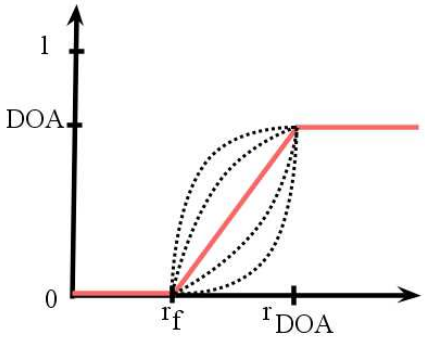
Cette fonction $DOA(s)$ est analogue à la fonction h de Sarkar. Sarkar et Brown utilisent la fonction h pour recalculer les coordonnées des points dans le graphe. Ici, la fonction sert à filtrer la hiérarchie. En effet, elle fait varier le niveau de détails suivant la distance au centre d'intérêt. Sarkar et Brown n'utilisent toutefois pas de pallier comme ici ; la variation par pallier est aussi présente dans [8]. La figure 6.14 montre l'effet de différentes fonctions $a(s)$.

Van Wijk et van Ham montrent comment les inégalités (6.1) peuvent être adaptées pour tenir compte d'un centre d'intérêt dans un graphe. Afin de décider si un sommet C' doit être sélectionné, on prend en compte sa distance $|C' - f|$ au focus. On utilise alors la fonction $DOA(s)$ définie précédemment pour modifier les inégalités :

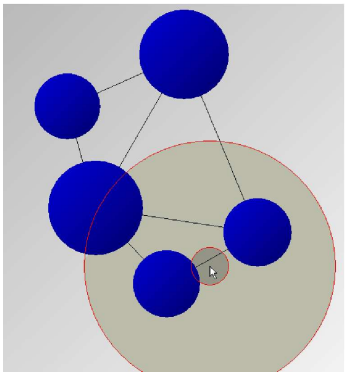
$$d_{C'} \leq d_{root} \cdot DOA(|C' - f|) \ ; \ d_{root} \cdot DOA(|C - f|) < d_C \tag{6.2}$$

La figure 6.15 montre la sélection des éléments à afficher et le graphe correspondant à cette vue abstraite tenant compte d'un centre d'intérêt. Le point focal est ici l'élément 3 qui est placé au centre du graphe et qui se retrouve dans l'abstraction de la figure 6.15(a) dans le cluster du centre. Le fait d'en faire le point focal va provoquer l'éclatement de ce cluster pour en donner le détail. Son voisinage est alors lui aussi détaillé. Ainsi, le cluster 10 est aussi éclaté.

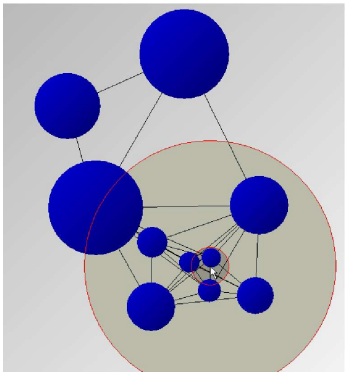
Parallèle avec Furnas L'approche de van Wijk et van Ham ne diffère pas fondamentalement de celle de Furnas. En effet, nous pouvons transformer l'exemple de Furnas en attribuant à chaque sommet v dans l'arbre (i.e. une instruction) une valeur égale à $d_v = 1/(1 - \ell(v))$, où $\ell(v)$ est le niveau du sommet v dans l'arbre. L'API de Furnas est alors analogue à la *dispersion*. 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 $1/a < DOA \leq 1/(a+1)$ on sélectionne les sommets au niveau $\ell(v) = -a$. La vue proposée par Furnas nécessite alors de sélectionner les sommets par les inégalités (6.2). Cependant, comme le montre la figure 6.16, les sommets à afficher sont ceux sélectionnés par les inégalités et l'ensemble de leurs ancêtres.



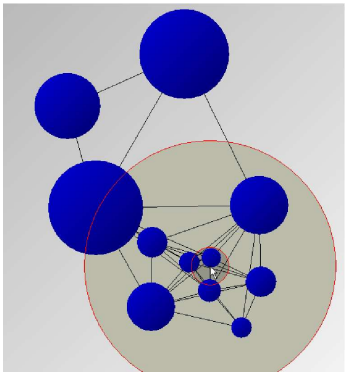
(a) Fonctions $a(s)$



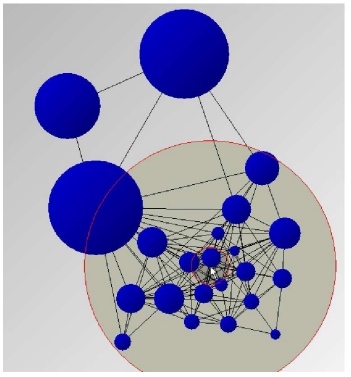
(b) Fonction logarithmique



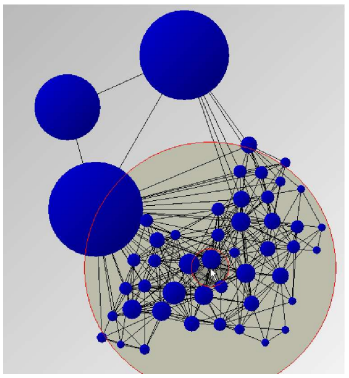
(c) Fonction racine carrée



(d) Fonction linéaire

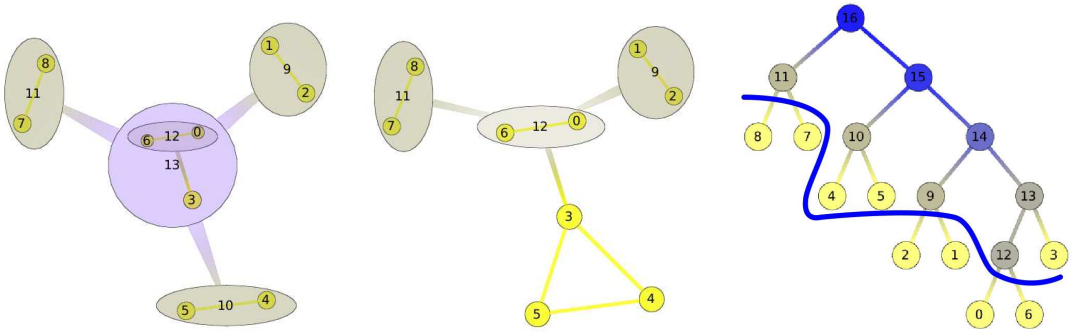


(e) Fonction carrée



(f) Fonction exponentielle

FIG. 6.14 – Différentes fonctions a(s) pour la fonction de distortion de van Wijk et van Ham (dans l'ordre du plus haut au plus bas : $ln(x)$, $\sqrt{x}$, x, x^2 et e^x). Visualisation d'un graphe suivant différentes fonctions $a(s)$.



(a) Abstraction du graphe (b) Vue "focus+contexte" du graphe (c) Hiérarchie associée

FIG. 6.15 – Vue "focus+contexte" du graphe et coupe associée avec comme centre d'intérêt le sommet 3 au centre du graphe.

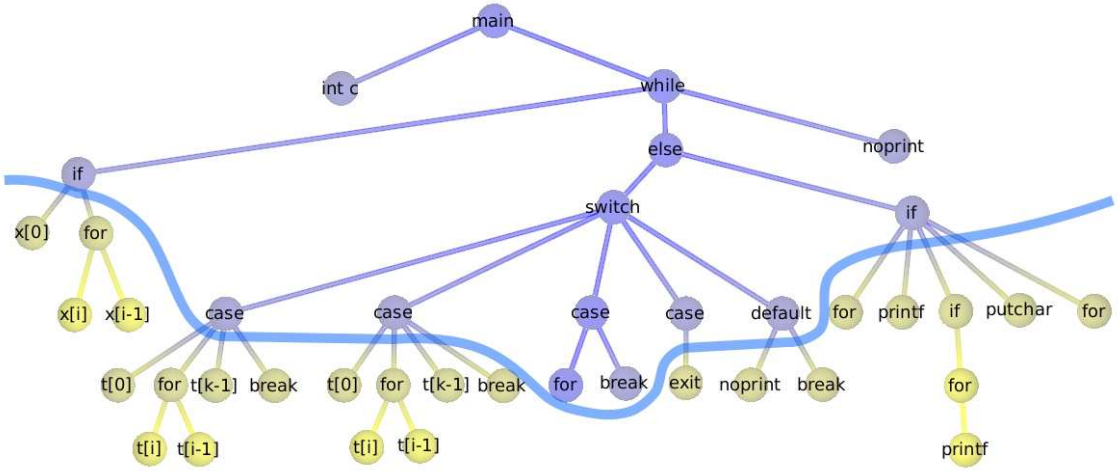


FIG. 6.16 – Sélection d'une coupe sur la hiérarchie de code.

6.2 DAGMap : Visualisation par niveau de détail

Dans cette section, nous allons nous intéresser aux techniques "focus+contexte" pour la visualisation des DAGs. La méthode que nous allons décrire est à mi chemin entre les techniques de Furnas et van Wijk et van Ham qu'on étend aux DAGs.

Notre méthode est proche de celle de Furnas dans le sens où la hiérarchie servant à filtrer la donnée est confondue avec la donnée à visualiser. Cependant, on se distingue de sa méthode de par la nature plus complexe de la hiérarchie : le DAG.

Lorsqu'on applique un seuil sur le DOI de Furnas dans un arbre, cela revient à faire une coupe dans la hiérarchie, un choix des sommets de l'arbre qui couvre toutes les feuilles. Nous généralisons cette notion de coupe aux DAGs en se servant d'un indice qui s'inspire directement de DOA de van Wijk et van Ham. La technique que nous allons décrire utilise le calcul du DOA, tel que l'a défini van Wijk et van Ham, qu'on généralise aux DAGs et donc au DAGMap. Ce mécanisme étendu au DAG a été exploité avec le DAGMap comme nous allons le voir dans la section suivante.

6.2.1 Généralisation du DOA aux DAGs

Pour pouvoir appliquer le mécanisme de van Wijk et van Ham à un DAG, il faut deux ingrédients. Il faut pouvoir calculer un indice d (une dispersion) pour tous les sommets de la hiérarchie et ensuite appliquer les inégalités (6.1) (voir page 95).

Supposons l'indice d calculé (voir section 6.2.2). Dans le cas d'un arbre, l'inégalité (6.1) fait intervenir un sommet et son père. Dans le cas d'un DAG, les inégalités sont plus complexes. En effet, un sommet pouvant avoir plusieurs pères, nous avons un ensemble d'inégalités (une inégalité pour chacun de ses pères). Par exemple, dans la figure 6.17, le sommet étiqueté 10 a deux pères, les sommets 13 et 14. Il y a donc deux inégalités qui doivent être simultanément respectées :

$$d_{10} \leq DOA \times d_{root} < d_{13}$$
$$d_{10} \leq DOA \times d_{root} < d_{14}$$

Ainsi modifiées, l'ensemble d'inégalités (6.1) peut être appliqué aux DAGs pour déterminer une coupe à un certain degré d'abstraction (DOA). Étant donné un indice sur les sommets dans le DAG (croissant des feuilles aux racines) et une valeur de DOA préalablement choisie, les éléments satisfaisant les inégalités forment alors une coupe. Comme dans le cas de Furnas, le DOA nous permet de filtrer une partie de la structure pour en avoir une version simplifiée.

La figure 6.18 montre un exemple de coupe où l'indice de dispersion est le nombre de Strahler (voir section suivante) et le niveau d'abstraction est $DOA = 0.5$. Les inégalités (6.1) sélectionnent ici les sommets qui ont un indice inférieur ou égal à 2 (indice de la racine 4 multiplié par le $DOA = 0.5$) et dont les pères sont supérieurs strictement à cette valeur. Ainsi, dans notre exemple, le sommet 10 d'indice 2 est sélectionné car ses pères sont strictement supérieurs.

De façon analogue, l'introduction d'un centre d'intérêt peut être pris en compte. Les inégalités (6.2) se transforment alors en un ensemble d'inégalités à satisfaire simultanément. La figure 6.19 présente une fonction de distorsion et une vue tenant

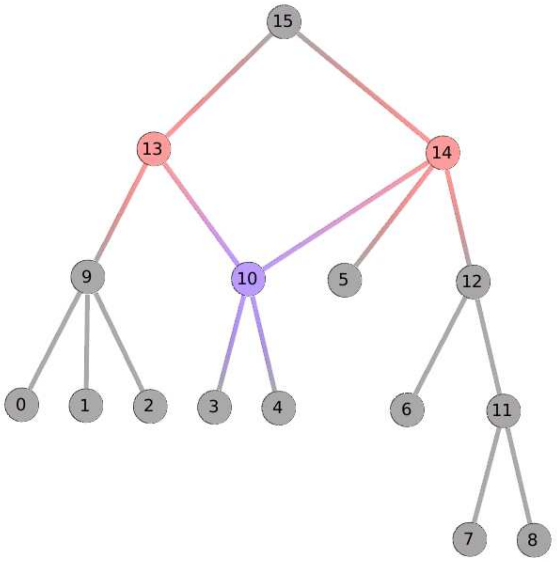


FIG. 6.17 – Exemple de hiérarchie sous forme de DAG.

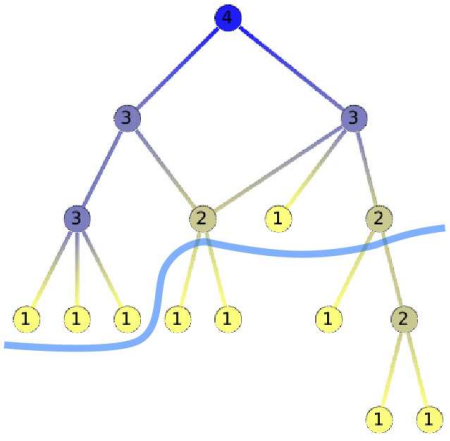
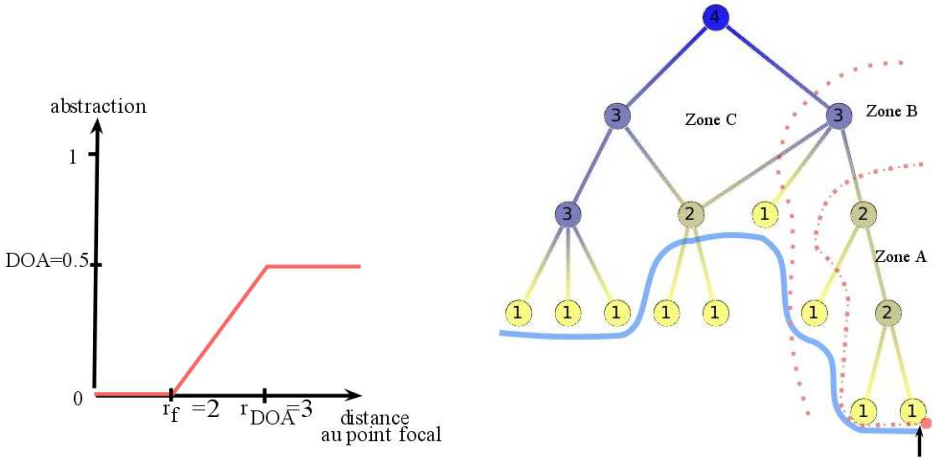


FIG. 6.18 – Sélection d'une coupe dans la hiérarchie. La dispersion est ici le nombre de Strahler et le $DOA = 0.5$.

compte d'un centre d'intérêt (indiqué par une flèche). La distance utilisée ici est la distance dans le graphe.



(a) Fonction de distorsion (b) Coupe dans la hiérarchie

FIG. 6.19 – Sélection d'une coupe dans la hiérarchie en tenant compte d'un centre d'intérêt suivant les paramètres $DOA = 0.5$, $r_f = 2$, $r_{DOA} = 3$ et $a(s)$ linéaire.

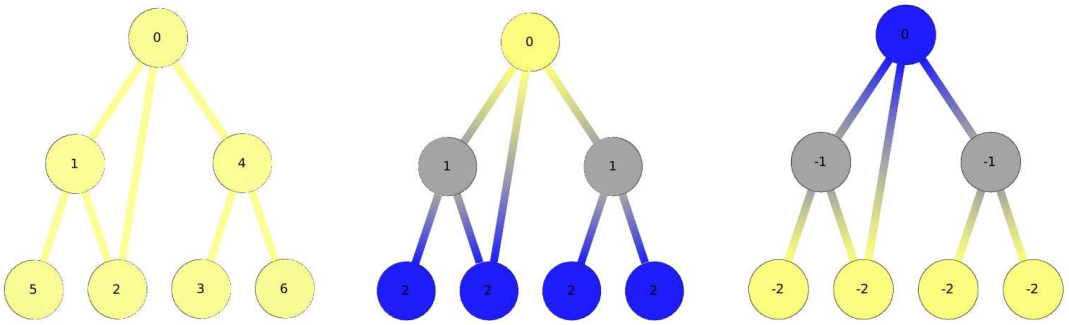
Il y a une grande différence entre les deux situations : la coupe calculée à partir d'un arbre forme une *antichaine* : les sommets sélectionnés ne se comparent pas deux à deux et les autres sommets ne sont dominés que par un seul élément de la coupe. Ce qui n'est pas le cas lorsqu'on applique le DOA au DAG comme on l'introduit. La coupe dans un DAG couvre simplement l'ensemble des sommets feuilles.

6.2.2 Dispersion et statistique sur les sommets du DAG

L'un des ingrédients pour appliquer le mécanisme de van Wijk et van Ham est l'indice de *dispersion*. Cet indice, comme cela a été décrit, doit être croissant des feuilles à la racine dans le cas d'un arbre. Le passage aux DAGs implique que cet indice doit être croissant des feuilles aux sources. Nous présentons dans cette section différents indices pouvant être calculés sur les DAGs et répondant au critère de croissance.

La notion de profondeur ou de rang est définie comme la longueur du plus long chemin entre le sommet et une source (voir figure 6.20). L'opposé de cet indice est croissant des feuilles aux sources et peut donc être utilisé comme indice de *dispersion*. Cette notion est la généralisation de la profondeur d'un sommet dans un arbre. Comme cela a été décrit, la technique de Furnas fait intervenir différents indices dont le degré à priori d'importance (l'API). L'API correspond à l'opposé de la profondeur d'un sommet dans l'arbre. La racine ayant un indice égale à 0, ses fils une valeur -1, leur fils -2 ... .

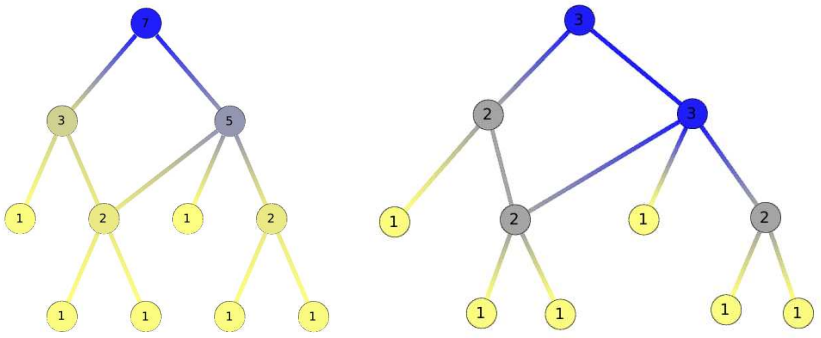
Le nombre de feuilles d'un arbre suit une distribution gaussienne. Cela permet de déterminer si un arbre est de forme très plate ou très allongée (à la verticale) [44]. Ce nombre correspond au nombre de feuilles couvertes par un sommet. Une feuille a un indice de 1 : elle se couvre elle-même. Plus on monte dans la hiérarchie



(a) DAG (b) Rang (c) Opposé du rang

FIG. 6.20 – Rang des sommets dans un DAG.

plus cette valeur augmente car les clusters couvrent de plus en plus de feuilles. Cette statistique peut être étendue aux DAGs est donc un bon candidat pour notre indice de *dispersion* (voir figure 6.21(a)).



(a) Couverture des feuilles (b) Nombre de Strahler

FIG. 6.21 – Exemple de hiérarchie sous forme de DAG.

Le nombre de Strahler est introduit pour la première fois en hydrographie par Robert Horton[48] et Arthur Newell Strahler [83]. Dans un réseau hydrographique modélisé par un arbre, on peut attribuer une valeur à chaque arête caractérisant son importance. Les feuilles de notre arbre représentent des drains n'ayant pas d'affluent, on leur attribue la valeur 1. Un drain interne de la hiérarchie correspond à la confluence des drains fils. Dans le cas d'un arbre binaire, si ses drains fils n'ont pas la même valeur, on lui attribue le maximum des valeurs. Si les drains fils ont la même valeur, alors on ajoute 1 à cette valeur. Le nombre de Strahler peut être étendu aux arbres d'arité quelconque [2].

En informatique, ce nombre est utilisé pour compter le nombre de registres nécessaires afin d'effectuer une expression parathésées. En effet, l'ordre d'exécution est modélisé par un arbre. Ce nombre est croissant des feuilles à la racine, il peut donc être utilisé comme indice de *dispersion*. Ce nombre a été étendu aux DAGs [43] et constitue un autre bon candidat (voir figure 6.21(b)).

Les indices présentés ci-dessus sont des indices topologiques. En effet, ils se basent que sur la structure de la hiérarchie et non pas sur des attributs des données. Cependant, suivant le domaine d'application, ces indices peuvent être pondérés par des

attributs des données ou encore être un attribut des données. En effet, la seule condition à respecter est la croissance de l'indice. Un exemple sera fourni dans notre cas d'étude.

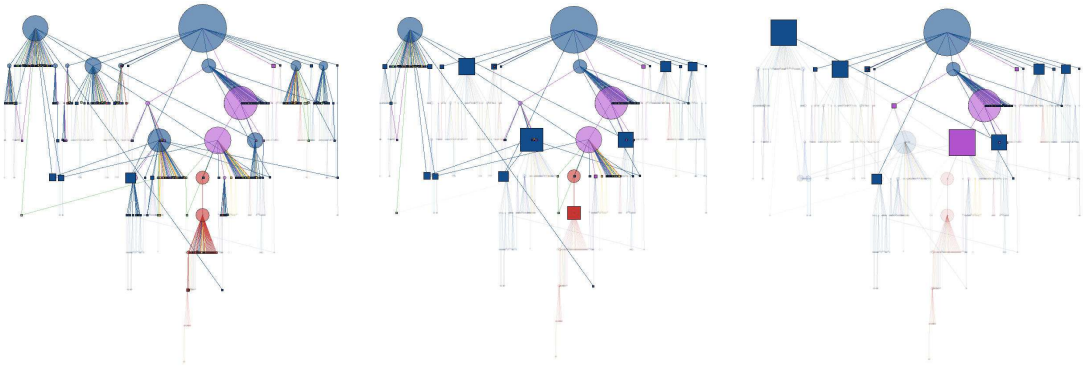
6.2.3 Cas d'étude

L'analyse et l'exploration des relations d'héritage, relations de gouvernance des sociétés, constituent notre cas d'étude que nous avons déjà abordé dans le chapitre 4. Les relations d'héritage peuvent être modélisées par des DAGs. La visualisation de ces DAGs est une première étape du processus d'analyse et de compréhension.

Un graphe de gouvernance est un graphe qui modélise les relations entre les différentes filiales d'une société. Ce graphe est une hiérarchie. On peut donc appliquer sur cette structure des techniques de visualisation "focus+contexte" à l'image de Furnas sur le code source d'un programme. Nous allons maintenant décrire pourquoi les concepts de visualisation "focus+contexte" peuvent être appliqués à ce type de graphe.

Dans l'exploration du graphe de gouvernance d'une société, le géographe peut vouloir, dans un premier temps, avoir une vue plus ou moins générale de la hiérarchie. Il construit alors une abstraction plus ou moins forte de son graphe en appliquant un filtre sur la hiérarchie. Ces abstractions peuvent lui indiquer un comportement global de la société et permet une analyse de ses stratégies de gouvernance.

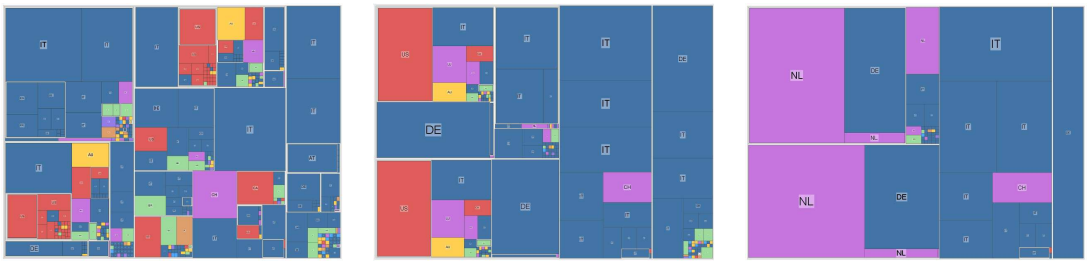
L'exploration des données peut se faire suivant des scénarios naturels. Sur notre exemple de sociétés et filiales, supposons que l'utilisateur s'intéresse aux filiales, mais seulement à celles se situant à une certaine distance des sociétés de plus haut niveau. Ceci peut être obtenu facilement en fixant le degré d'abstraction du graphe à un niveau plus ou moins fort en choisissant l'indice de dispersion adéquat. La figure 6.22 montre les différentes vues noeud lien produites suivant l'abstraction.



(a) $DOA = 0,25$ (b) $DOA = 0,5$ (c) $DOA = 0,75$

FIG. 6.22 – Représentation noeud lien du graphe de gouvernance de Fiat à différents niveaux d'abstraction.

Le DAGMap est alors 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. La figure 6.23 montre différentes abstractions de la figure 6.22 suivant le niveau d'abstraction choisi.



(a) $DOA = 0,25$ (b) $DOA = 0,5$ (c) $DOA = 0,75$

FIG. 6.23 – Représentation DAGMap du graphe de gouvernance de Fiat à différents niveaux d'abstraction.

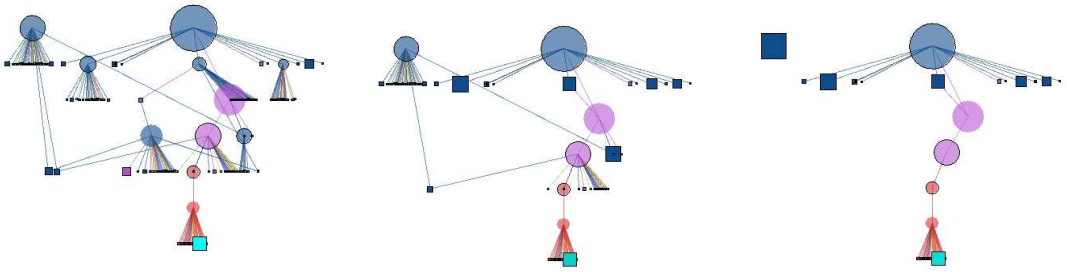
La donnée qu'on manipule ici est riche d'informations. Ainsi, on peut vouloir filtrer le graphe suivant la profondeur mais en tenant compte du capital des sociétés ou encore de la participation au capital d'une filiale. Le géographe filtre la donnée grâce à une coupe qui n'est pas forcément horizontale. 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.

Si on s'intéresse à la société Fiat et à ses filiales, on souhaite avoir une vue sur les sociétés que contrôle Fiat, ou avoir une vue de l'empire Fiat. Cela dit, il est fort probable que Fiat et Peugeot aient un équipementier en commun. Donc Peugeot ou des filiales de Peugeot sont liés à des données qui nous intéressent. Cela dit, on ne souhaite peut-être pas avoir une vue détaillée des filiales impliquées avec Peugeot mais seulement avoir une indication de la présence de Peugeot dans l'environnement de Fiat. Ce type de filtrage de la donnée peut typiquement être obtenue à l'aide d'une coupe.

De plus, si nous nous intéressons à une filiale de Fiat en particulier, nous souhaitons avoir une vue claire des sociétés qu'elle contrôle et qui la contrôle ainsi que de voir sa filiation avec la société mère. Cependant, une autre filiale contenue dans l'empire de la société mère ne nous intéresse pas forcément. Les figures 6.24 et 6.25 montrent le DAG de gouvernance de Fiat à différents niveaux d'abstraction où l'on a défini un centre d'intérêt. Ce point d'intérêt est la société de couleur turquoise située en bas au centre de la vue noeud lien.

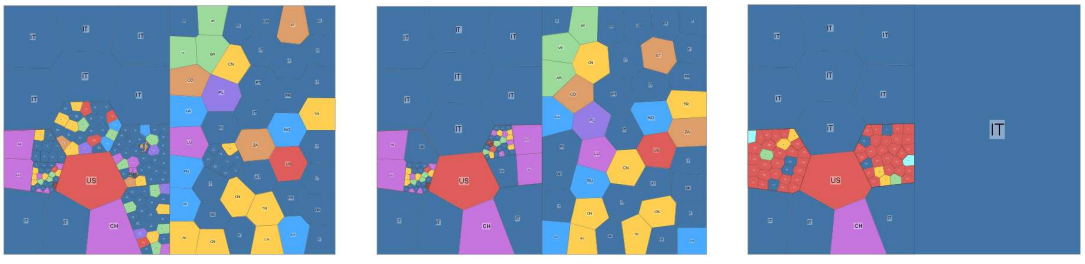
6.2.4 Interacteur

Les différents concepts abordés dans ce chapitre sont assez complexes. Pour une exploration facile et intuitive des données, nous ne souhaitons pas imposer à l'utilisateur de manipuler ces concepts en l'état, et d'avoir à manipuler explicitement les indices DOA, nombre de Strahler, et la fonction $a(s)$. Auquel cas, un temps non négligeable d'apprentissage serait nécessaire. Nous avons donc conçu un interacteur spécifique qui capture les différents paramètres de la visualisation et permet de les



(a) (b) (c)

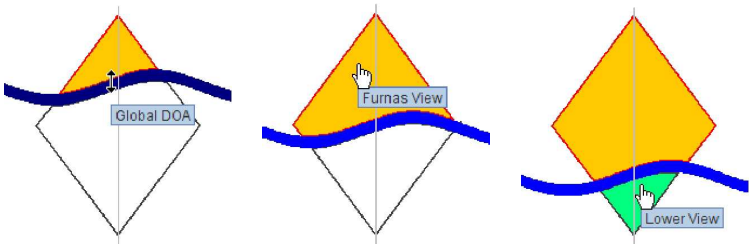
FIG. 6.24 – Représentation noeud lien du graphe de gouvernance de Fiat à différents niveaux d'abstraction en tenant compte d'un centre d'intérêt.



(a) (b) (c)

FIG. 6.25 – Représentation DAGMap du graphe de gouvernance de Fiat à différents niveaux d'abstraction en tenant compte d'un centre d'intérêt.

modifier de façon très intuitive.



(a) (b) (c)

FIG. 6.26 – Un interacteur en forme de losange est utilisé pour induire une coupe du DAG, en le déplaçant de haut en bas (images (a) à (c)).

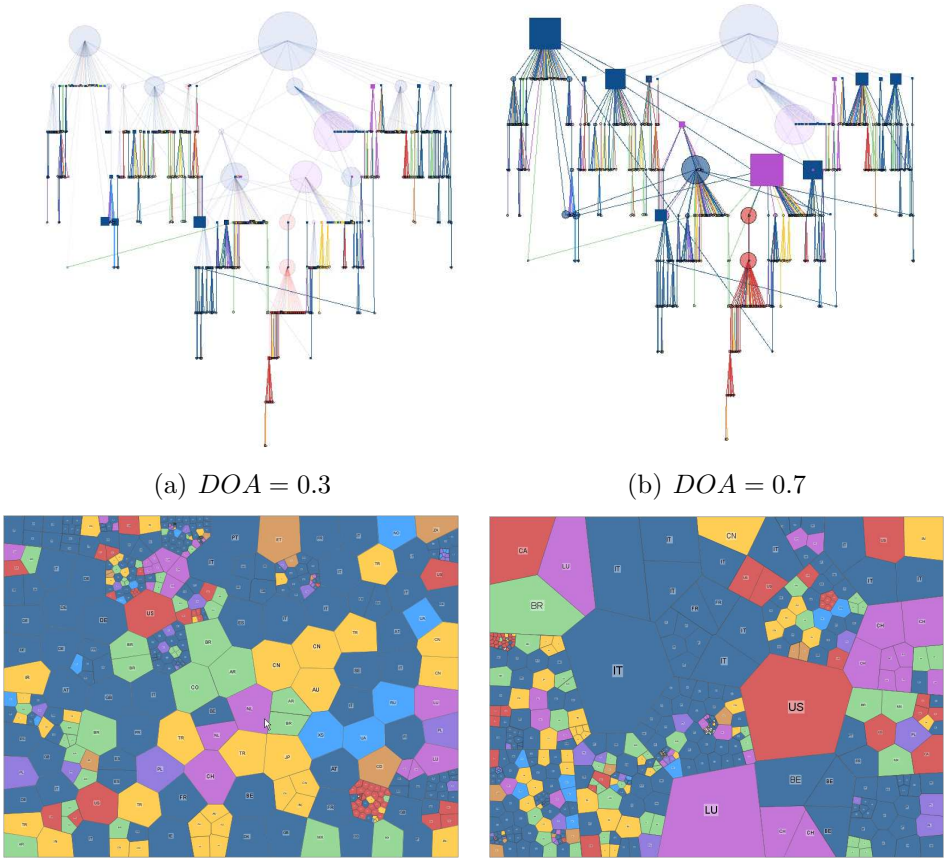
Le premier ingrédient que nous utilisons pour notre visualisation repose sur la notion de "coupe" dans le DAG. Le DAGMap est construit à partir d'une sélection de certains éléments du DAG se trouvant à des niveaux contigus (mais pas forcément situés sur un *même* niveau) et recouvrant toute sa largeur.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

Le calcul de cette "coupe" repose sur différents paramètres qui peuvent être modifiés par le biais de notre interacteur en forme de losange. 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 détails) auquel le DAGMap sera construit (voir figure 6.26).

Notre interacteur permet de modéliser différentes situations. Un simple clic sur la partie supérieure de l'interacteur provoque la construction de la vue en ne sélectionnant que les éléments de la coupe et l'ensemble de leurs ancêtres. C'est la situation qui reprend le cas développé par Furnas. Un clic sur la ligne incurvée elle même nous place alors dans le cas de van Wijk et van Ham et seuls les éléments de la coupe seront affichés. L'interacteur permet aussi de cliquer sur la partie en dessous de la ligne incurvée, afin de construire la vue à partir des sommets de la coupe et l'ensemble de ses successeurs (voir figure 6.27).

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.

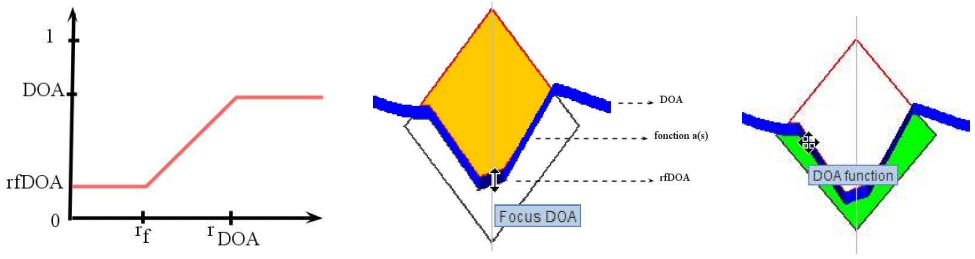
Cette situation est modélisée dans l'interacteur en permettant à l'utilisateur de jouer avec la forme de la ligne incurvée, permettant à la coupe de plonger autour du sommet sélectionné tout en gardant un niveau de détails élevé pour les éléments éloignés de celui-ci (voir figure 6.28). L'application se met en mode "focus" en cochant un bouton de contrôle dans l'interface (voir figure 6.29), l'utilisateur peut jouer avec la ligne incurvée et définir combien de détails sont requis autour de l'élément focus.



(a) $DOA = 0.3$ (b) $DOA = 0.7$

(c) $DOA = 0.3$ (d) $DOA = 0.7$

FIG. 6.27 – Vues noeud lien et DAGMaps associés construits en prenant les éléments de la coupe et l'ensemble des successeurs.



(a) (b) (c)

FIG. 6.28 – Fonction de distortion pour le DAG et manipulation de l'interacteur.

FIG. 6.29 – Application mettant en scène une vue combinée d'une représentation noeud lien et du DAGMap. Les différents outis de paramétrage sont placés en haut à gauche.

6.3 Conclusion

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 figure 6.30). 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.

Dans le cas d'un DAG volumineux comme le cas d'Ubuntu, le choix de la coupe pour extraire un DAG de taille raisonnable devient alors capital. Ce qui est moins le cas des DAG étudiés avec les géographes qui sont relativement modestes.

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. Lors de l'installation d'un package particulier, différents packages dont il dépend seront eux aussi installés. Ainsi, la

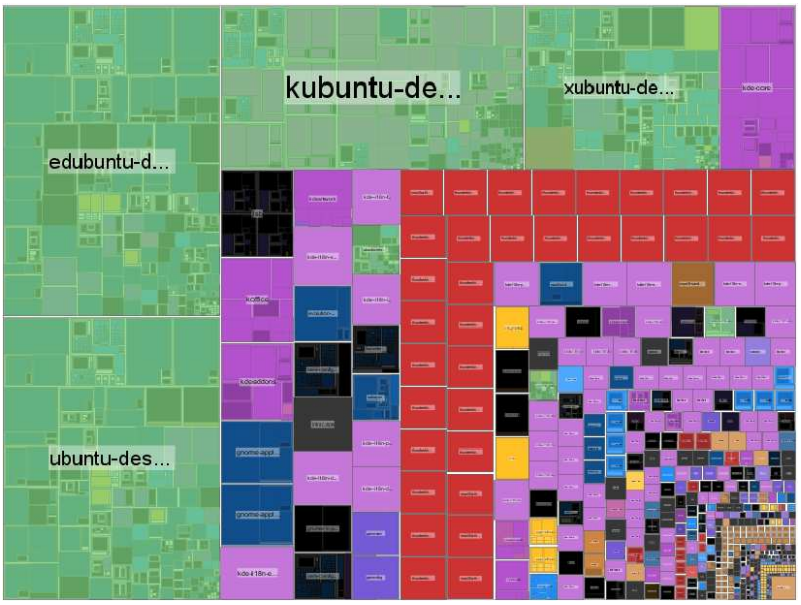


FIG. 6.30 – DAGMap du graphe de dépendance de la distribution Ubuntu (Linux).

visualisation préalable du sous-dag de dépendance permet de décider de son installation. Cela correspond alors à centrer la vue sur ce package et à filtrer les packages éloignés de lui. Ces opérations sont facilités par le DAGMap.

Notre travail est dans une certaine mesure proche des Flexible TreeMaps proposés par [23], 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 [31]).

Le cas étudié ici est particulier. Le DAG sur lequel s'effectue la coupe coïncide avec le DAG qui est visualisé. On peut imaginer d'autres situations où le DAG émerge d'une procédure d'agrégation. Dans ce cas, l'agrégation produit des clusters qui se chevauchent. Il faut alors imaginer des visualisations qui tiennent compte du chevauchement. Une prochaine étape sera l'étude de ces situations où des données sont groupées avec des recouvrements qui s'imbriquent et se chevauchent. 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 appartienne 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.

Chapitre 7

Conclusion et perspectives

La visualisation d'information est une approche des plus prometteuses pour l'exploration, l'analyse et la compréhension de données, parfois massives [52, 84]. La conception d'une bonne visualisation nécessite le choix de la bonne représentation des données et d'un ensemble d'interactions adaptées. Dans le cadre de cette thèse, les différentes collaborations avec les utilisateurs finaux de nos méthodes (biologistes et géographes) ont permis d'élaborer de nouvelles visualisations et interactions adaptées aux besoins spécifiques de leurs domaines.

C'est dans une perspective de navigation "focus+contexte" que s'est articulée cette thèse. Une vue abstraite permet une compréhension globale des données qu'elle représente. Les différents niveaux de granularité de cette vue donnent un contexte dans lequel on peut vouloir localement plus de détails (focus). Ou inversement, s'intéressant à un élément des données, on peut vouloir une vue détaillée de son voisinage et une idée ou abstraction des éléments éloignés de lui.

Nous avons présenté une méthode permettant de visualiser et d'interagir sur des données multidimensionnelles. L'objectif premier de cette méthode est de proposer une visualisation permettant d'identifier visuellement des corrélations non linéaires dans des données multidimensionnelles. Tirant profit de la perception multi-échelle de l'image, nous avons rendu plus facile la sélection de régions d'intérêts en structurant l'image. Notre méthode a été implémentée et testée en collaboration avec des biologistes qui l'ont jugée utile et complémentaire des autres techniques de statistiques plus classiques comme l'analyse en composante principale (ACP). Cette méthode entre dans le paradigme "focus+contexte". En effet, la structuration de l'image en images de granularités différentes nous permet d'avoir différentes abstractions des données. Dans cette vue structurée, la sélection d'une zone donne plus de détails sur celle-ci et son rôle dans la matrice de graphes de corrélations.

Dans le cadre de notre collaboration avec des géographes, nous nous sommes intéressés à la représentation de données hiérarchiques. Notre but étant de visualiser des hiérarchies sous forme de DAG, nous avons présenté un état de l'art sur les techniques de représentation des hiérarchies. Ces techniques se concentrent sur le cas des arbres. L'approche par pavage de l'espace (les TreeMaps) a retenu notre attention et nous l'avons étendu au DAG. Nous avons opté pour cette technique du fait de notre cas d'étude. En effet, les géographes souhaitaient un outil centré sur les attributs des données. Cependant, s'intéressant à la structure même de la

hiérarchie, il nous fallait une représentation permettant son exploration. On s'est donc dirigé vers une vue combinant les deux aspects. D'un côté, une vue noeud lien et de l'autre, une TreeMap étendue au DAG. Différentes interactions sur le DAGMap ont été proposées permettant notamment une visualisation par niveau de la hiérarchie par l'introduction d'une "bande". L'utilisabilité de la méthode et des différentes interactions ont été évaluées de façon formelle opposant une vue noeud lien, le DAGMap et une vue combinée. Les résultats de cette expérimentation semblent indiquer que la vue combinée serait un bon compromis entre temps requis et interactions effectuées.

Afin d'améliorer l'exploration des données, nous nous sommes intéressés aux navigations "focus+contexte" sur les arbres et les graphes. Un état de l'art est proposé présentant les travaux de Furnas et de Sarkar et Brown qui sont les travaux fondateurs de tout le courant de recherche en visualisation "focus+contexte". Travaillant sur le filtrage de la donnée, Furnas introduit un indice, le DOI, pour afficher ou ne pas afficher les sommets s'éloignant d'un centre d'intérêt. Cet indice repose sur la distance de chaque élément à la racine et la distance dans l'arbre des éléments au centre d'intérêt. Sarkar, pour sa part, utilise la distance dans le graphe pour recalculer la position des éléments dans le dessin. Cependant, dans ces deux approches, la structure hiérarchique des données n'est pas exploitée en tant que telle. Une troisième approche est étudiée consistant à exploiter davantage la hiérarchisation de l'information. Dans cette approche, la hiérarchie ne coïncide pas forcément avec les données à visualiser. La hiérarchie est alors un outil de filtrage qui reste en arrière plan. Nous avons étendu ce mécanisme aux DAGs visualisés à travers une adaptation de la TreeMap : le DAGMap.

Changement de point de vue

L'exploration des données implique de faire varier le niveau de détail afin d'avoir une vue globale. Dans cette vue donnant le contexte, on explore localement des éléments dont le voisinage proche est alors accessible. Ces différentes interactions ont été, dans le cas du DAGMap, modélisées par un interacteur permettant de piloter une coupe dans la hiérarchie de données.

Cependant, l'une des difficultés lors de la manipulation du DAGMap et de la manipulation interactive de la coupe réside dans le changement de point de vue. En effet, si l'utilisateur porte son attention sur un nouvel élément et donc, implicitement détermine le calcul d'une nouvelle coupe, la transition se fait brutalement (instantanément). Une nouvelle présentation du graphe est alors amenée à l'utilisateur sans repère par rapport à la vue précédente. Un temps d'adaptation est alors nécessaire pour reconstruire la carte mentale du graphe. Il y a donc un vrai défi à permettre à l'utilisateur de changer de point de vue de façon intelligible.

L'algorithme calcule les cellules de Voronoi associées aux éléments de la TreeMap et ajuste itérativement leurs positions, modifiant ainsi dynamiquement la taille des cellules [3]. On peut donc chercher à paramétrer dans le temps la taille des cellules du nouveau point de vue à atteindre et miser sur l'animation pour effectuer la transition. Cette piste fait l'objet de travaux en cours. Cette stratégie paraît plus difficile à implémenter sur les TreeMaps plus classiques. Les travaux de Blanch et

Lecolinet [14] pourraient être un point de départ intéressant.

Domaines d'application

Le domaine d'application du DAGMap ne se limite pas au contexte de la géographie. En effet, les DAGs apparaissent dans différentes disciplines telle que la biologie avec la taxonomie des protéines, les composés chimiques ou encore le clustering de gènes [71]. On peut aussi penser à l'ontologie de la génomique décrite par un graphe acyclique orienté (GO)[1]. Le DAGMap tel qu'il est construit ici en vue combinée avec la vue noeud lien pourrait convenir puisque les vues noeuds liens sont assez populaires en biologie. On pourrait penser mettre à profit le DAGMap dans une application comme Proviz[2] [49] qui visualise des graphes d'interactions entre protéines simultanément avec une vue sur l'ontologie des protéines. En génie logiciel, on retrouve les DAGs avec le diagramme de classes d'un programme ou encore dans l'analyse de documents avec la classification des concepts. Son utilisation dans d'autres cadres est déjà amorcée. Par exemple, dans le génie logiciel avec le graphe de dépendance des packages dans la distribution Ubuntu (voir section 6.3).

Couplage avec d'autres représentations

Dans le cadre du projet avec les géographes, le DAGMap a été associé à la vue noeud lien. Cependant, on peut imaginer la vue combinée avec d'autres types de représentations. Par exemple, pour intégrer d'avantage de dimensions, on peut coupler le DAGMap avec une représentation axée sur les attributs telle que les coordonnées parallèles. Les différents paramètres de la construction du DAGMap seraient alors pilotés par une sélection d'attributs sur les coordonnées parallèles. L'intégration du DAGMap au sein d'un environnement complet tel que Tulip[3] nous permettrait d'expérimenter d'avantage de combinaisons.

[1] `http://www.geneontology.org/`
[2] `http://cbi.labri.fr/eng/proviz.htm`
[3] `http://tulip.labri.fr/`

Publications

Publications en informatique

Guy Melançon, Fabien Jourdan, Paris Alain, and Pierre-Yves Koenig. Multiscale scatterplot matrix for visual and interactive exploration of metabonomics data. In François Poulet, editor, *Pixelization Paradigm*, volume 4370/2007 of *Lecture Notes In Computer Sciences*, pages 202-215. Springer Berlin / Heidelberg, 2006.

Pierre-Yves Koenig, Guy Melançon, Charles Bohan, and Bérengère Gautier. Combining dagmaps and sugiyama layout for the navigation of hierarchical data. In *IV '07 : Proceedings of the 11th International Conference Information Visualization*, pages 447-452, Washington, DC, USA, IEEE Computer Society, 2007.

Pierre-Yves Koenig. Dagmap : un outil d'exploration adaptée aux relations d'héritages multiples. In *IHM '07 : Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine*, pages 237-240, New York, NY, USA, 2007. ACM.

Pierre-Yves Koenig and Guy Melançon. Dagmap : exploration interactive de relations d'héritage. In *Hermes-Lavoisier, R. (ed.) Revue d'Intelligence Artificielle*, 2008, 22 , 355-370.

Paolo Simonetto, Pierre-Yves Koenig, Faraz Zaidi, Daniel Archambault, Frédéric Gilbert, Trung Tien Phan Quang, Morgan Mathiaut, Antoine Lambert, Jonathan Dubois, Ronan Sicre, Mathieu Brulin, Remi Vieux, and Guy Melançon. Solving the Traffic and Flitter Challenges with Tulip. In *Proceedings of IEEE Symposium on Vast 2009 IEEE Symposium on Visual Analytics Science and Technology 2009*, pages ?, Atlantic City, New Jersey, USA United States, 10 2009.

Publications en géographie

Melançon, G., C. Rozenblat, P.-Y. Koenig and C. Discazeaux. Multiscale Analysis of Small World Networks : Visual and Interactive Exploration of Graph Hierarchies. *Sunbelt XXVII, 8th European Social Network Conference*, Corfu Island, Greece, International Network for Social Network Analysis 2007.

Charles Bohan, Bérengère Gautier, Céline Rozenblat and Pierre-Yves Koenig. Networks of Urban Centres and of Multinational Corporations : A Multi-Level Graph Approach. In *15TH European Colloquium on Theoretical and Quantitative Geography*, September 7-11, 2007, Montreux, Switzerland.

Céline Rozenblat, Pierre-Yves Koenig and Guy Melançon. Territorial and Topological Levels in Worldwide Air Transport Networks. In *15TH European Colloquium on Theoretical and Quantitative Geography*, September 7-11, 2007, Montreux, Switzerland.

Bozzani-Franc S., Conesa A., L'Hostis A., Auber D., Rozenblat C., Mary P., Melançon G., Koenig P.-Y. Relating the organisation of the cities network and the organisation of the air transport network. An approach through graph visualisation. In *15TH European Colloquium on Theoretical and Quantitative Geography*, September 7-11, 2007, Montreux, Switzerland.

Ducruet César, Rozenblat Céline et Koenig Pierre-Yves Analysing the world maritime system : An approach through graph visualization *Annual Conference of the Association of American Geographers (AAG)*, Boston, April 15-17 2008.

Posters

Projet SPANGEO. Le Projet Spangeo pour la visualisation des graphes en géographie. Poster de présentation du projet Spangeo. *STIC 2007*, Paris France

Pierre-Yves Koenig. Exploration de Hiérarchies. Poster de vulgarisation. *Doctiss'08*, Montpellier France.

Equipe gravité. GRAVITE Graph Visualization and Interactive Exploration. Poster de présentation de l'équipe de recherche GRAVITE. *Inria 2008* Bordeaux France

Bibliographie

[1] J. Abello and J. Korn. Mgv : a system for visualizing massive multidigraphs. *IEEE Transactions on Visualization and Computer Graphics*, 8(1) :21–38, 2002.

[2] D. Auber, M. Delest., J.M. Fedou, J.P. Domenger, and P. Duchon. New strahler numbers for rooted plane trees. In Michael Drmota, Philippe Flajolet, Danièle Gardy, and Bernhard Gittenberger, editors, *Third Colloquium on Mathematics and Computer Science, Trends in Mathematics*, pages 203–215, Wien, Austria, 2004.

[3] Michael Balzer and Oliver Deussen. Voronoi treemaps. In *IEEE Symposium on Information Visualization InfoVis '05*, Washington, DC, USA, 2005. IEEE Computer Society.

[4] Atanu Basu and Shivani Malhotra. Error detection of bathymetry data by visualization using GIS. *ICES J. Mar. Sci.*, 59(1) :226–234, 2002.

[5] G. di Battista, Peter Eades, Roberto Tamassia, and Ian G. Tollis. *Graph Drawing : Algorithms for the Visualisation of Graphs*. Prentice Hall, 1998.

[6] Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus plus context screens : combining display technology with visualization techniques. In *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 31–40, New York, NY, USA, 2001. ACM.

[7] Richard A. Becker and William S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2) :127–142, 1987.

[8] Benjamin B. Bederson. Fisheye menus. In *UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 217–225, New York, NY, USA, 2000. ACM.

[9] Benjamin B. Bederson, Aaron Clamage, Mary P. Czerwinski, and George G. Robertson. Datelens : A fisheye calendar interface for pdas. *ACM Trans. Comput.-Hum. Interact.*, 11(1) :90–119, 2004.

[10] Benjamin B. Bederson, Ben Shneiderman, and Martin Wattenberg. Ordered and quantum treemaps : Making effective use of 2d space to display hierarchies. *ACM Trans. Graph.*, 21(4) :833–854, 2002.

[11] J. Bertin. Sémiologie graphique les diagrammes- les réseaux les cartes. *Edition de l'EHESS ISSN 1294-1107 ISBN 2-7132-1277-4*, 1973.

[12] Thomas Bladh, David A. Carr, and Matjaz Kljun. The effect of animated transitions on user navigation in 3d tree-maps. In *IV '05 : Proceedings of the Ninth International Conference on Information Visualisation*, pages 297–305, Washington, DC, USA, 2005. IEEE Computer Society.

[13] Thomas Bladh, David A. Carr, and Jeremiah Scholl. Extending tree-maps to three dimensions : A comparative study. In *Computer Human Interaction*, volume 3101/2004 of *Lecture Notes in Computer Science*, pages 50–59. Springer Berlin / Heidelberg, 2004.

[14] Renaud Blanch and Éric Lecolinet. Treemaps zoomables : techniques d'interaction multi-échelles pour les treemaps. In *IHM '07 : Proceedings of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine*, pages 131–138, New York, NY, USA, 2007. ACM.

[15] Charles Bohan, Bérengère Gautier, Céline Rozenblat, David Auber, and Pierre-Yves Koenig. Cities networks through multinational firms networks : A multi-level graph approach. In *15th European Colloquium on Theoretical and Quantitative Geography*, Zurich, CH, 2007.

[16] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Engineering graph clustering : Models and experimental evaluation. *Journal of Experimental Algorithmics*, 12 :1.1, 2007. http ://doi.acm.org.gate6.inist.fr/10.1145/1227161.1227162 ACM Press.

[17] Mark Bruls, Kees Huizing, and Jarke J. van Wijk. Squarified treemaps. In W. de Leeuw and R. van Liere, editors, *Joint Eurographics and IEEE TCVG Symposium on Visualization (Data Visualization '00)*, pages 33–43, Amsterdam, 2000. Springer-Verlag.

[18] S. Card and D. Nation. Degree-of-interest trees : A component of an attention-reactive user interface, 2002.

[19] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization.* Morgan Kaufmann Publishers, San Francisco, 1999.

[20] Jeromy Carriere and Rick Kazman. Interacting with huge hierarchies : Beyond cone trees. In *Proc. IEEE Information Visualization '95, IEEE Computer Press, Los Alamitos, CA*, pages 74–81. IEEE, 1995.

[21] Ricardo A. Cava, Paulo R. G. Luzzardi, Carla M. D. S. Freitas, and Pelotas Rs Brasil. The bifocal tree : a technique for the visualization of hierarchical information structures, 2002. Fortaleza, Brazil.

[22] Chaomei Chen. Top 10 unsolved information visualization problems. *IEEE Comput. Graph. Appl.*, 25(4) :12–16, 2005.

[23] Gouthami Chintalapani, Catherine Plaisant, and Ben Shneiderman. Extending the utility of treemaps with flexible hierarchy. In *Eighth International Conference on Information Visualisation (IV'04)*, pages 335–344. IEEE Computer Society, 2004.

[24] Yves Chiricota, Fabien Jourdan, and Guy Melancon. Metric-based network exploration and multiscale scatterplot. In *INFOVIS '04 : Proceedings of the IEEE Symposium on Information Visualization*, pages 135–142, Washington, DC, USA, 2004. IEEE Computer Society.

[25] M. Collomb and M. Hascoët. Implementing Drag-and-Drop Like Manipulations in a Collaborative and Large Screen Surface Context. Technical report, 2006.

[26] Maxime Collomb, Mountaz Hascoët, Patrick Baudisch, and Brian Lee. Improving drag-and-drop on wall-size displays. In *GI '05 : Proceedings of Graphics Interface 2005*, pages 25–32, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[27] Selan dos Santos and Ken Brodlie. Gaining understanding of multivariate and multidimensional data through visualization. *Computers & Graphics*, 28(3) :311–325, 2004.

[28] P. Eades. Drawing free trees. *Bulletin of the Institute for Combinatorics and its Applications*, pages 10–36, 1992.

[29] P. Eades and K. Sugiyama. How to draw a directed graph. *Journal of Information Processing*, 13(4) :424–437, 1990.

[30] Stephen G. Eick. Information visualization at 10. *IEEE Computer Graphics and Applications*, 25(1) :12–14, 2005.

[31] J.D. Fekete, D. Wang, N. Dang, A. Aris, and C. Plaisant. Overlaying graph links on treemaps. In *IEEE Information Visualization 2003 Symposium Poster Compendium*, pages 82–83. IEEE Press, 2003.

[32] A. Formella and J. Keller. Generalized fisheye views of graphs. In *Symposium on Graph Drawing GD '95*, pages 242–253, Berlin, 1995. Springer-Verlag.

[33] Michael Friendly. A brief history of the mosaic display. *Journal of Computational and Graphical Statistics*, 11(1) :89–107, 2002.

[34] G. W. Furnas. The fisheye view : a new look at structured files. *Bell Laboratories Technical Memorandum*, October 12, 1981. 81-11221-9.

[35] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Trans. Softw. Eng.*, 19(3) :214–230, 1993.

[36] Emden R. Gansner, Yehuda Member-Koren, and Stephen C. Senior Member-North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4) :457–468, 2005.

[37] Bérengère Gautier. Integration of the moroccan cities by the multinational firms of agro-alimentary sector. In *IGU Commission of Monitoring cities of tomorrow"*, pages 193–209, Guangzhou, China, 2007. Sun Yat Sen University Press.

[38] S. Grivet, D. Auber, J.P. Domenger, and G. Melançon. Bubble tree drawing algorithm. In *ICCVG'04 : International Conference on Computer Vision Graphics*, pages 633–641, 2004. 11362.

[39] C. Gunn. Visualizing hyperbolic space. In *Eurographics Workshop on Computer Graphics and Mathematics*, pages 299–313. Springer-Verlag, 1992.

[40] C. Gutwenger and P. Mutzel. An experimental study of crossing minimization heuristics. In B. Liotta, editor, *Graph Drawing 2003*, volume 2912 of *Lecture Notes in Computer Science*, pages 13–24. Springer-Verlag, 2004.

[41] B. Hausmann, B. Slopianka, and H.-P. Seidel. Exploring plane hyperbolic geometry. In *Workshop on Visualization and Mathematics*, pages 21–36. Springer-Verlag, 1998.

[42] Jeffrey Heer and Stuart K. Card. Doitrees revisited : scalable, space-constrained visualization of hierarchical data. In *AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, pages 421–424, New York, NY, USA, 2004. ACM.

[43] Ivan Herman, M. Scott Marshall, Guy Melançon, David J. Duke, Maylis Delest, and Jean-Philippe Domenger. Skeletal images as visual cues in graph visualization. In E. Gröller, H. Löffelmann, and W. Ribarsky, editors, *Data Visualization '99, Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization*, pages 13–22. Springer Verlag, 1999.

[44] Ivan Herman, Guy Melançon, and Maylis Delest. Tree visualisation and navigation clues for information visualisation. *Computer Graphics Forum*, 17(2) :153–165, 1998.

[45] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization : A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1) :24–43, 2000.

[46] Bill Hibbard. Top ten visualization problems. *SIGGRAPH Comput. Graph.*, 33(2) :21–22, 1999.

[47] Danny Holten. Hierarchical edge bundles : Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :741–748, 2006.

[48] R. E. Horton. Erosioned development of streams and their drainage basins, hydrophysocal approach to quantitative morphology. *Bulletin of the Geologic Society of America*, 56 :275–370, 1945.

[49] Florian Iragne, Macha Nikolski, Bertrand Mathieu, David Auber, and David Sherman. Proviz : protein interaction visualization and exploration. *Bioinformatics*, 21(2) :272–274, January 2005.

[50] II J. Q. Walker. A node-positioning algorithm for general trees. *Softw. Pract. Exper.*, 20(7) :685–705, 1990.

[51] Liqun Jin and David C. Banks. Tennisviewer : A browser for competition trees. *IEEE Computer Graphics and Applications*, 17(4) :63–65, 1997.

[52] C. Johnson, R. Moorhead, T. Munzner, H. Pfister, P. Rheingans, and T. S. Yoo, editors. *NIH-NSF Visualization Research Challenges Report.* IEEE Computer Society, 2006.

[53] Susanne Jul and George W. Furnas. Critical zones in desert fog : aids to multiscale navigation. In *UIST '98 : Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 97–106, New York, NY, USA, 1998. ACM.

[54] W. Jungmeister and D. Turo. Adapting treemaps to stock portfolio visualization. Technical report , cartr648, cs-tr-2996, src-tr-92-120, Dept. of Computer Science, University of Maryland, 1992.

[55] Michael Kaufmann and Dorothea Wagner, editors. *Drawing Graphs, Methods and Models*, volume 2025 of *Lecture Notes in Computer Science.* Springer-Verlag, London, UK, 2001.

[56] Robert Kosara, Silvia Miksch, and Helwig Hauser. Focus+context taken literally. *IEEE Computer Graphics and Applications*, 22(1) :22–29, 2002.

[57] J. Lamping and R. Rao. The hyperbolic browser : A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, 7(1) :33–55, 1996.

[58] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Human Factors in Computing Systems, CHI'95*, pages 401–408. ACM Press, 1995.

[59] Bongshin Lee, Cynthia S. Parr, Catherine Plaisant, Benjamin B. Bederson, Vladislav D. Veksler, Wayne D. Gray, and Christopher Kotfila. Treeplus : Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6) :1414–1426, 2006.

[60] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *BELIV '06 : Proceedings of the 2006 AVI workshop on BEyond time and errors*, pages 1–5, New York, NY, USA, 2006. ACM.

[61] Bill Lorensen. On the death of visualization. *NIH/NSF Fall 2004 Workshop*, 2004.

[62] Hao Lü and James Fogarty. Cascaded treemaps : examining the visibility and stability of structure in treemaps. In *Graphics Interface 2008*, volume 322 of *ACM International Conference Proceeding Series*, pages 259–266, Windsor, Ontario, Canada, 2008. Canadian Information Processing Society.

[63] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall : detail and context smoothly integrated. In *CHI '91 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 173–176, New York, NY, USA, 1991. ACM.

[64] Allen R. Martin and Matthew O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *VIS '95 : Proceedings of the 6th conference on Visualization '95*, page 271, Washington, DC, USA, 1995. IEEE Computer Society.

[65] Dumas ME, Canlet C, Debrauwer L, Martin P, and Paris A. Selection of biomarkers by a multivariate statistical processing of composite metabonomic data sets using multiple factor analysis. *Journal of Proteome Research*, 4(5) :1485–92, 2005.

[66] Guy Melançon and Ivan Herman. Dag drawing from an information visualization perspective. In W. de Leeuw and R. van Liere, editors, *Joint Eurographics and IEEE TCVG Symposium on Visualization (Data Visualization '00)*, pages 3–13, Amsterdam, 2000. Springer-Verlag.

[67] Guy Melançon, Fabien Jourdan, Paris Alain, and Pierre-Yves Koenig. Multiscale scatterplot matrix for visual and interactive exploration of metabonomics data. In François Poulet, editor, *IEEE/ACM VIEW 2006 Visual Information Expert Workshop*. Springer, 2006.

[68] T. Munzner. H3 : Laying out large directed graphs in 3d hyperbolic space. In *IEEE Symposium on Information Visualization (InfoVis '97)*, pages 2–10. IEEE CS Press, 1997.

[69] Laurence Nigay and Frédéric Vernier. Design method of interaction techniques for large information spaces. In *AVI '98 : Proceedings of the working conference on Advanced visual interfaces*, pages 37–46, New York, NY, USA, 1998. ACM.

[70] Stephen C. North and Gordon Woodhull. Online hierarchical graph drawing. *Graph Drawing*, pages 77–81, 2002.

[71] Frank Olken. Graph data management for molecular biology. *OMICS : A Journal of Integrative Biology*, 7 (1) :75–78, 2003.

[72] Stefan Schlechtweg Petra Neumann and M.Sheelagh T. Carpendale. Arctrees : Visualizing relations in hierarchical data. In *Eurographics, IEEE VGTC Symposium on Visualization (EuroVis2005). Eurographics*, pages 53–60, 2005.

[73] Mark Phillips and Charlie Gunn. Visualizing hyperbolic space : unusual uses of 4x4 matrices. In *Symposium on Interactive 3D Graphics*, SIGGRAPH : ACM Special Interest Group on Computer Graphics and Interactive Techniques, pages 209 – 214, Cambridge, Massachusetts, United States, 1992. ACM Press New York, NY, USA.

[74] Ramana Rao and Stuart K. Card. The table lens : merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *CHI '94 : Conference companion on Human factors in computing systems*, page 222, New York, NY, USA, 1994. ACM.

[75] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Trans. Softw. Eng.*, 7(2) :223–228, 1981.

[76] Jun Rekimoto and Mark Green. The information cube : Using transparency in 3d information visualization. In *In Proceedings of the Third Annual Workshop on Information Technologies and Systems (WITS'93*, pages 125–132, 1993.

[77] M. Sarkar and M.H. Brown. Graphical fish-eye views of graphs. In *Human Factors in Computing Systems, CHI '92 Conference Proceedings*, pages 83–91. ACM Press, 1992.

[78] M. Sarkar and M.H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12) :73–84, 1994.

[79] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1 :27–64, 2007.

[80] David W. Scott. *Multivariate Density Estimation : Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, September 1992.

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

[82] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[83] A. N. Strahler. Hypsometric (area-altitude) analysis of erosional topology. *Bulletin of the Geologic Society of America*, 63 :1117–1142, 1952.

[84] James J. Thomas and Kristin A. Cook, editors. *Illuminating the Path : The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2006.

[85] W. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(13) :743–768, 1963.

[86] Jarke J. van Wijk and Huub van de Wetering. Cushion treemaps : visualisation of hierarchical information. In Graham Wills and Daniel Keim, editors, *IEEE Symposium on Information Visualization (InfoVis '99)*, pages 73–78. IEEE CS Press, 1999.

[87] Jarke J. van Wijk and Frank van Ham. Interactive visualization of small world graphs. In Tamara Munzner and Matt Ward, editors, *IEEE Symposium on Information Visualisation*, Austin, TX, USA, 2004. IEEE Computer Science press.

[88] J.J. van Wijk. Views on visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 12(4) :1000–433, July-Aug. 2006.

[89] Roel Vliegen, Jarke J. van Wijk, and Erik-Jan van der Linden. Visualizing business data with generalized treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :789–796, 2006.

[90] S. Wan. Blog treemap visualizer. http ://www.samuelwan.com/information/archives/000159.l

[91] Colin Ware. *Information visualization : perception for design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[92] Martin Wattenberg. Arc diagrams : Visualizing structure in strings. *Information Visualization, IEEE Symposium on*, 0 :110, 2002.

[93] Martin Wattenberg and Danyel Fisher. A model of multi-scale perceptual organization in information graphics. In Stephen C. North and Tamara Munzner, editors, *IEEE Symposium on Information Visualization*, Seattle, USA, 2003. IEEE Computer Press.

[94] M. Weskamp. Newsmap. http ://marumushi.com/apps/newsmap/newsmap.cfm.

[95] K. Wetzel. Using circular treemaps to visualize disk usage. `http://lip.sourceforge.net/ctreemap.html`.

[96] Shengdong Zhao, Michael J. McGuffin, and Mark H. Chignell. Elastic hierarchies : Combining treemaps and node-link diagrams. *Information Visualization, IEEE Symposium on*, 0 :8, 2005.

[97] Caroline Ziemkiewicz and Robert Kosara. The shaping of information by visual metaphors. *Transactions on Visualization and Computer Graphics (Proceedings InfoVis)*, 14(6) :1269–1276, 2008.