Méthodes et modèles pour la visualisation de grandes masses de données multidimensionnelles nominatives dynamiques

Frédéric Gilbert

N^o d'ordre : 4498

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par Frédéric GILBERT

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : Informatique

Méthodes et modèles pour la visualisation de grandes masses de données multidimensionnelles nominatives dynamiques

---

Soutenue le : 21 Mars 2012

Après avis de :

Mme.	Anne Doucet	Professeure	**Rapporteur**
M.	Yves Chiricota	Professeur	**Rapporteur**

Devant la Commission d'Examen composée de :

M.	David Auber	Maître de conférence.	**Co-directeur de thèse**
M.	Yves Chiricota	Professeur	**Rapporteur**
Mme.	Maylis Delest	Professeure	**Directrice de thèse**
M.	Jean-Philippe Domenger	Professeur	**Président**
Mme.	Anne Doucet	Professeure	**Rapporteur**
M.	Noël Novelli	Maître de conférence.	**Examinateur**

- 2012 -

N^o d'ordre : 4498

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par Frédéric GILBERT

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : Informatique

Méthodes et modèles pour la visualisation de grandes masses de données multidimensionnelles nominatives dynamiques

---

Soutenue le : 21 Mars 2012

Après avis de :

Mme.	Anne Doucet	Professeure	**Rapporteur**
M.	Yves Chiricota	Professeur	**Rapporteur**

Devant la Commission d'Examen composée de :

M.	David Auber	Maître de conférence.	**Co-directeur de thèse**
M.	Yves Chiricota	Professeur	**Rapporteur**
Mme.	Maylis Delest	Professeure	**Directrice de thèse**
M.	Jean-Philippe Domenger	Professeur	**Président**
Mme.	Anne Doucet	Professeure	**Rapporteur**
M.	Noël Novelli	Maître de conférence.	**Examinateur**

- 2012 -

Remerciements

Mes remerciements vont tout d'abord à l'Université de Bordeaux et au LaBRI pour m'avoir accueilli et permis de réaliser mes travaux de thèse dans de bonnes conditions. Je tiens également à remercier Maylis Delest et David Auber d'avoir accepté d'encadrer mes travaux. Je remercie plus particulièrement David de m'avoir donné l'opportunité de partager sa vision de l'évolution du monde de la visualisation d'informations. Je remercie Maylis d'avoir su me guider lors de la rédaction de ce manuscrit.

Je voudrais remercier Anne Doucet et Yves Chiricota d'avoir accepté d'être les rapporteurs de cette thèse. Je tiens à les remercier pour leurs remarques avisées tant aussi bien sur le fond que sur le forme. Je remercie également Jean-Philippe Domenger et Noël Novelli d'avoir accepté de faire partie de mon jury et d'avoir témoigné un vif intérêt pour les problématiques lors de la soutenance.

Je tiens aussi à remercier l'ensemble des membres du thème "Visualisation des grandes masses de données" pour les moments partagés au cours de ce doctorat, et plus particulièrement Guy Melançon qui m'a permis de découvrir la visualisation d'informations et d'intégrer cette équipe. Je tiens aussi à remercier les ingénieurs, tout particulièrement Jonathan Dubois, pour les discussions que nous avons eu au sujet de mes (nombreux) questionnements sur les choix de développement.

Je remercie les membres de ma famille et de ma belle-famille pour les encouragements et le soutien qu'ils m'ont témoigné au cours de ce doctorat. Je voudrais remercier mon Grand-Père qui tout au long de mon enfance m'a transmis cette envie de connaître et de comprendre les choses. Je pense que sans cette envie je n'aurais jamais imaginé un jour présenter un Doctorat. Et enfin, je voudrais témoigner toute ma reconnaissance et mon amour à ma femme Pauline sans qui cette thèse n'aurait certainement jamais été réalisée. Merci à elle d'avoir participé activement à la relecture de ce manuscrit, d'avoir cru en moi et de m'avoir redonné confiance dans les moments difficiles tout au long de ces trois années de thèse.

Méthodes et modèles pour la visualisation de grandes masses de données multidimensionnelles nominatives dynamiques

Résumé : La visualisation d'informations est un domaine qui connait un réel intérêt depuis une dizaine d'année. Dernièrement, avec l'explosion des moyens de communication, l'analyse de réseaux sociaux fait l'objet de nombreux travaux de recherches. Nous présentons dans cette thèse des travaux sur l'analyse de réseaux sociaux dynamiques, c'est à dire que nous prenons en compte l'aspect temporel des données. Nous nous sommes particulièrement intéressé à la mise en évidence des communautés dans les réseaux et à leurs évolutions dans le temps. Nous présentons également un algorithme permettant de construire une hiérarchie d'influence qui identifie le rôle occupé par les individus au sein du réseau.

Le second axe de recherche abordé dans cette thèse traite de l'obstacle que représente la diversité des formats de stockage de données. Cette diversité complique grandement l'import de données dans les systèmes de visualisation par des utilisateurs novices. Nous proposons dans cette thèse deux méthodes permettant de manipuler des données dans le but de produire des visualisations de type nœud-lien : une basée sur la construction d'une taxonomie des dimensions des données, l'autre sur la mise en évidence de domaines d'entités. Ces méthodes mettent l'accent sur les interactions avec l'utilisateur, afin de tirer profit de ses connaissances sur les données.

Mots-clef : Analyse de données, Génération de graphes, Analyse de réseaux sociaux dynamiques.

Discipline : Informatique

Methods and model for huge amount of nominative multidimendionnal dynamic data visualization

Abstract : Since ten years, informations visualization domain knows a real interest. Recently, with the growing of communications, the research on social networks analysis becomes strongly active. In this thesis, we present results on dynamic social networks analysis. That means that we take into account the temporal aspect of data. We were particularly interested in communities extraction within networks and their evolutions through time. We present an algorithm building an influence hierarchy which identifies the roles of the actors within the network.

The second area of research approached in this thesis deals with the variety of data storage formats. This variety spoils the import data for non expert users. In this thesis, we propose two methods allowing to manipulate data in order to produce node-link diagram visualizations. The first one is based on the construction of a dimensions taxonomy of the data. The second one tries to discover entities domains. These methods focus on user's interactions, to take avantages from his data knowledge.

Keywords : Data analysis, Graph generation, Dynamic social networks analysis.

Field : Computer Science

Table des matières

Table des figures

Liste des Algorithmes

Liste des tableaux

Chapitre 1

Introduction

1.1 Motivation

On pourrait penser que la visualisation d'informations est apparue en même temps que les premiers moyens de communications. Or, les hommes préhistoriques n'ont pas encore l'usage d'un langage évolué qu'ils sont déjà capables d'identifier le rôle de chacun à l'aide de leur tenue vestimentaire. L'association de type de peau ou fourrure avec un rôle social pourrait être considéré comme un des premiers moyens de visualisation d'informations ou d'identifications. De nos jours, on trouve encore de tels moyens d'identification. Par exemple, dans le milieu hospitalier chaque corps de métier, médecin, infirmière, aide soignante, porte une tenue d'une couleur différente, afin d'être plus facilement identifiable. Ce n'est que bien plus tard que les premières représentations visuelles abstraites ont vu le jour, avec l'apparition de la cartographie. C'est durant l'Antiquité que Ptolémée prépare, vers 150 ap. J.C, une carte générale du monde connu avec des cartes secondaires qui fournissent des noms de fleuves, de peuples et de villes. Bien qu'approximatives, ces cartes sont le point de départ de la cartographie et donc dans un certain sens, celui de la visualisation d'informations.

Au cours des siècles, les techniques de relevés et de tracés ont permis de fortement améliorer la cartographie, permettant d'y adjoindre de plus amples informations. Dans la Figure 1.1, Charles Minard propose en 1869 une représentation visuelle qui illustre l'évolution des effectifs de l'armée Napoléonienne au fur et à mesure de son avancée, lors de la campagne de Russie de 1812. On peut clairement identifier le passage de la rivière Bérézina lors du retour de la campagne qui donnera l'expression de la langue française associée à cette rivière de Biélorussie. On voit qu'avec la forte chute des températures et le franchissement de l'eau glacée de la Bérézina sur des ponts de fortune, les effectifs diminuent brutalement de moitié. La Figure 1.2 présente une carte de Londres répertoriant les cas d'une épidémie de choléra. Elle a été réalisée en 1854 par John Snow, un célèbre médecin britannique, considéré comme l'un des fondateurs de l'épidémiologie moderne. Le fait d'indiquer sur la carte les positions des cas recensés, ainsi que leur nombre, lui permettra d'émettre une hypothèse quant à la source de cette épidémie. De par la répartition géographique, il identifiera une zone "centrale" comme pouvant être le point de départ de l'épidémie. Après vérification de cette hypothèse, il s'est avéré que dans la zone concernée se trouvait un puits dont l'eau était contaminée. Il s'agit là d'un des premiers cas dans lequel la visualisation d'informations a permis d'analyser des données, d'émettre une hypothèse non soupçonnée jusqu'alors et d'en vérifier l'exactitude.

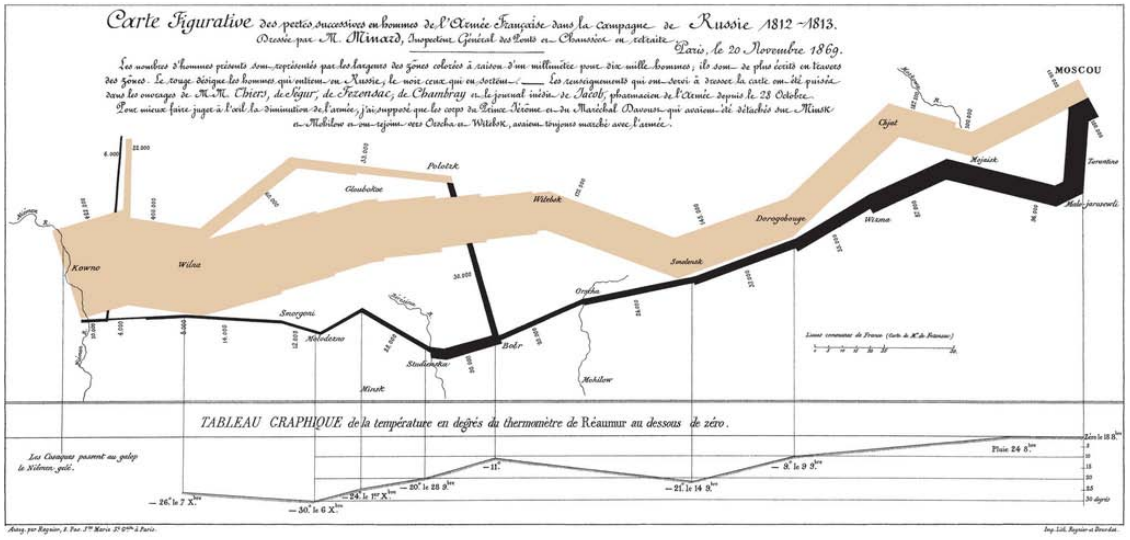


FIG. 1.1: Carte de Charles Minard de 1869 montrant les pertes en hommes de la Grande Armée de Napoléon 1^{er}, leurs mouvements et la température durant la campagne de Russie de 1812.Les effectifs sont représentés par la épaisseur du trait et un code couleur permet d'identifier l'aller et le retour. [65]

Avec le rôle majeur que joue la communication dans notre société et l'omniprésence des statistiques, la visualisation d'informations a connu un essor important sur ces douze dernières années. En effet, elle se trouve être à la fois une bonne manière d'exploiter des données et un très bon outil de communication. Depuis bien plus longtemps encore sont utilisées des méthodes mathématiques pour analyser des données (PCA [50], LSA [56]...) mais celles-ci ne proposaient que deux manières de visualiser les résultats : des tableaux ou des graphiques (nuages de points, courbes). Or, il est aisé de voir les limitations que peuvent présenter ces modes de visualisation. C'est en essayant de mettre à la portée de tous des outils de communication que ces limitations se sont dissipées. Ainsi, on a pu voir l'utilisation d'histogrammes ou des digrammes circulaires se banaliser et faire leur entrée dans les programmes scolaires. Puis avec le temps, le fait de s'être familiarisé avec ces techniques a permis d'en introduire de nouvelles telles que les diagrammes radar ou les diagrammes de Gant. D'autres ont évolué afin de devenir plus compacts et/ou intégrant des interactions. Par exemple, les tracés en mosaïque définis en 1981 par Hartigan *et al.* [43] ont été étendus à plusieurs reprises et dans des voies différentes. On peut noter les améliorations apportées entre 1994 et 1999 par Friendly [33] qui ont permis d'aborder le traitement de données catégorielles, parallèlement à celles réalisées à partir de 1991 par Shneiderman [49] qui ont donné naissance aux Tree-Maps. Les Tree-Maps commencent à être utilisées pour présenter des informations au grand public comme en témoigne le diagramme de la Figure 1.4 représentant le budget Américain de 2011 publié dans le New-York Times [85].

Mais, si la plupart des jeux de données possèdent des attributs quantitatifs sur les objets à étudier, une dimension relationnelle est souvent présente entre ces objets. Les moyens de communications tels qu'internet, les e-mails, les sms, les téléphones portables ou les réseaux sociaux, ont un réel aspect relationnel qui joue un rôle important dans leur analyse. Facebook ou Twitter génèrent à eux seul des quantités impressionnantes

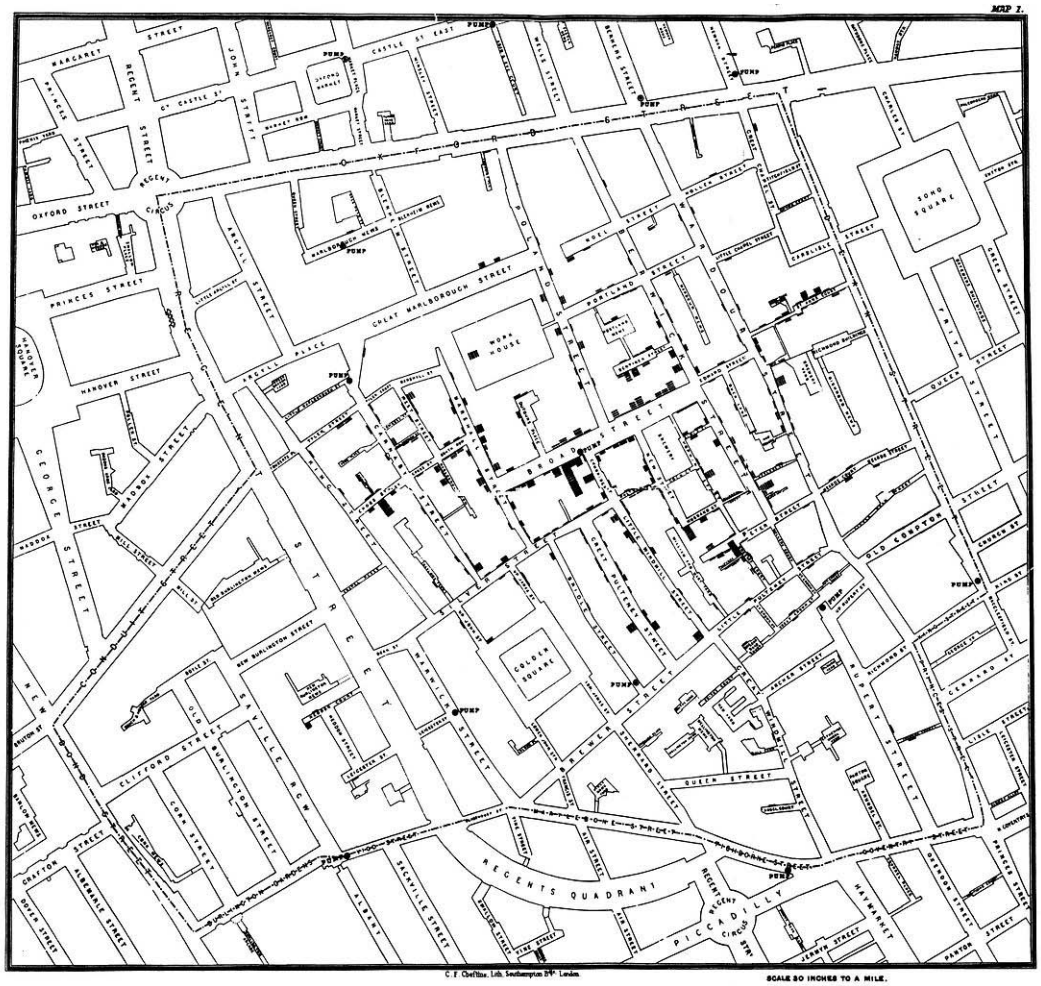


FIG. 1.2: Carte de John Snow montrant les régions de cas de choléra lors de l'épidémie de Londres en 1854. Les barres noires indiquent les localisations des cas et la longueur de chaque barre le nombre de cas recensés à cette adresse. [81]

d'échanges d'informations. L'enjeu qu'il y a derrière la compréhension de ces réseaux sociaux est énorme d'un point de vue financier. En effet, pouvoir comprendre comment sont tissés ces réseaux, quels en sont les points centraux et les points d'entrées, sont des informations primordiales pour toute entreprise souhaitant commercialiser un produit. Pour les services commerciaux ou de communications de ces entreprises, il est très important de pouvoir cibler sa clientèle, savoir comment entrer en contact avec elle et quelle est la meilleure période pour lancer leur produit. C'est aussi un bon moyen pour suivre les tendances, la création de communauté et leurs évolutions.

Depuis 2003, la communauté de visualisation d'informations a organisé des concours ([46]) afin de confronter à la fois les algorithmes et les systèmes de visualisation. C'est dans ce cadre que la conférence VAST (Visual Analytics Science and Technology) a proposé en 2008 un concours [73] visant à répondre à des tâches précises. Celles-ci portaient sur l'analyse de données multi-dimensionnelles dynamiques pouvant permettre de construire des réseaux sociaux. Un tel concours s'inscrit directement dans la problématique de l'évolution de communautés au sein des réseaux sociaux. Ce concours était divisé en quatre

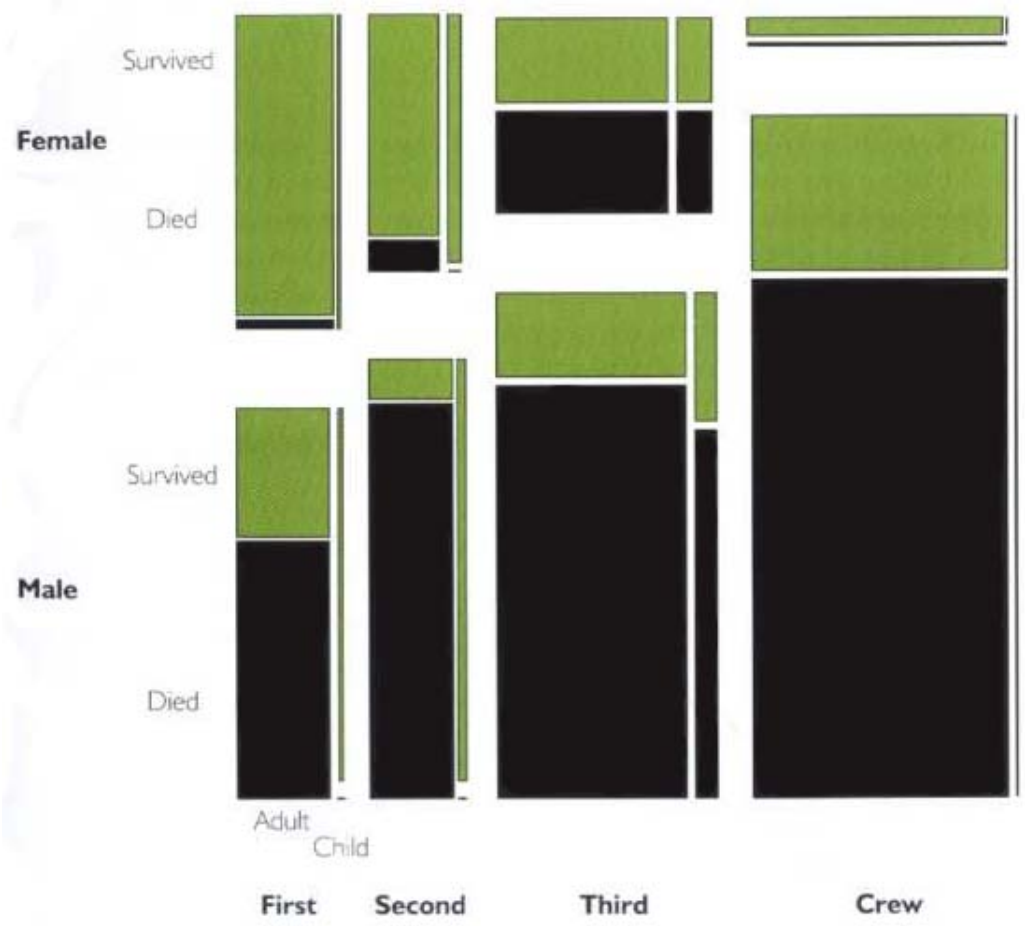


FIG. 1.3: Tracé mosaïque des données concernant les passager du Titanic. La partie supérieure représente la totalité des femmes présentes à bord, la partie inférieure les hommes. Horizontalement, les données sont découpées en fonction des classes des passagers et au sein de chaque classe il y a une distinction des tranches d'âge entre les adultes et les enfants. Dans chaque barre obtenue suivant ce découpage, la partie verte représente les survivants et le partie noire les passagers décédés. [33]

"Mini Challenges" qui une fois combinés permettaient de répondre à un "Grand Challenge". Nous avons pris part à ce concours et nous nous sommes concentrés sur le "Mini Challenge" n^o 3. Celui-ci consistait à traiter des informations relatives à des appels téléphoniques pour tenter des faire correspondre les numéros de téléphones portables avec des personnes. Il était aussi question de mettre en évidence les changements structuraux au sein des réseaux au cours de la période d'enregistrement des appels. De par la nature des appels téléphoniques qui mettent en relations deux personnes, la résolution de cette tâche fait appel aux techniques d'analyse de données relationnelles et des réseaux sociaux. C'est pour cela que nous avons dans un premier temps cherché des logiciels permettant un tel traitement. Devant l'absence de solutions, nous avons utilisé des mesures et des algorithmes existants permettant d'analyser des réseaux sociaux [18, 42] au sein de la plateforme Tulip (cf. Section 3.2.5). Nous avons aussi mis en place une méthode permettant d'étudier l'évolution de la structure d'un réseau social dans le temps. Lors de ce travail, nous nous sommes rendus compte que certains systèmes permettent d'effectuer des mesures sur les réseaux, d'autres de les dessiner ou de les animer, comme nous le présentons

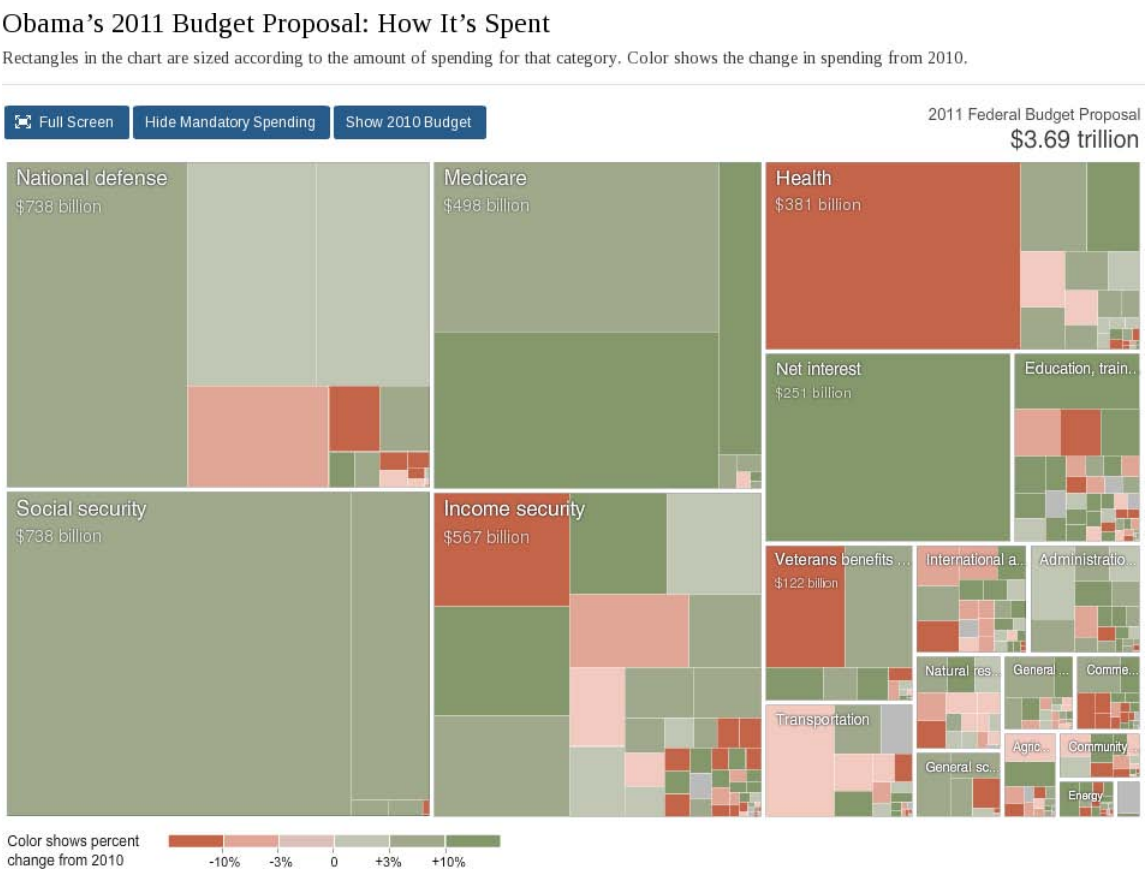


FIG. 1.4: Proposition de budget faite par le Président Obama pour l'année 2011. Les rectangles montrent les fonds autorisés à être dépensés chaque année pour chaque poste. La couleur des rectangles montre les écarts de dépense avec l'année 2010. Il s'agit d'une visualisation proposant des interactions. En cliquant sur un rectangle, il est possible d'obtenir des information plus détaillées sur le budget alloué au poste concerné. http://www.nytimes.com/interactive/2010/02/01/us/budget.html [85]

dans la Section 3.1. Mais nous avons constaté qu'il n'existait pas de logiciel ou système permettant de mener de bout en bout des telles analyses. Nous avons donc réalisé DySNAV (cf. Chapitre 4) un système proposant d'analyser des réseaux sociaux dans le temps au travers d'une exploration visuelle.

Au cours des travaux réalisés dans le cadre du concours VAST et du développement de DySNAV, nous nous sommes heurtés au problème que représente la diversité des formats de stockage des données. De nos jours, il existe un grand nombre de raisons qui nous pousse à collecter de l'information. Chaque raison qui a engendré une collecte d'information amène le plus souvent à stocker cette information sous une forme qui lui est propre. Par exemple, dans le cadre du projet ANR SPANGEO[1], les chercheurs en géographie collectent des données sur des routes maritimes qu'empruntent les navires marchands [28]. Il semble naturel que les données soient stockées de manière à ce que les entités soient les navires et que les points de passages (position GPS et ports maritimes), les dates de passages et les cargaisons soient des arguments propres aux navires. Ainsi comparer deux

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

navires ne représente aucune difficulté, mais dès que les chercheurs souhaiteront mettre en évidence le rôle qu'ont les capacités d'accueil de certains ports sur l'utilisation des routes marchandes, cela peut devenir impossible pour eux. En effet le nombre de routes, la précision de leurs relevés et le nombre de navires compliquent grandement la tâche. L'utilisation d'un système de visualisation d'informations semble alors tout indiqué, mais il ne sera pas évident pour l'utilisateur de parvenir à structurer ses données pour mettre en évidence le rôle attractif de ports ou les contraintes liées aux saisons. Il sera peut-être encore moins évident pour lui de manipuler les données afin qu'elles puissent produire une visualisation capable de l'aider dans son analyse.

Anisi, il n'existe pas de norme de stockage de l'information et chaque jeu de données soulève un grand nombre de questions. Il semble donc être impossible, dans l'état actuel des connaissances, de construire un système de visualisation capable d'importer automatiquement tout type de jeu de données et suggérer de voies de réponse à la question que se pose l'utilisateur. Le thème de l'import de données et de la construction automatique de visualisation représente donc un enjeu capital dans l'ouverture des techniques de visualisations au plus grand nombre. Ce domaine a fait l'objet de très peu de travaux. Jusqu'à présent, la communauté a principalement concentré ses efforts sur le développement de techniques de visualisation, d'interactions et de représentations ainsi que sur le dessin de graphes et la mise au point de mesure. Or, lorsqu'on se trouve confronté au problème que posent l'import des données et la construction de visualisation pour des utilisateurs novices, il est capital d'y répondre si on souhaite élargir l'utilisation de ces techniques.

Nous essayons d'apporter quelques éléments de réponse sur le sujet dans les Chapitre 5 et 6.

1.2 Processus de visualisation

L'intérêt que connait le domaine de la visualisation d'informations depuis une dizaine d'années et la multiplication des systèmes de visualisation ont amené dos Santos et Brodlie à présenter dans [27], une modélisation du processus de visualisation (*"visualization pipeline"*). Jusqu'alors, la communauté utilisait, tel un mantra, les travaux de Shneiderman [76] guidés par la phrase :

« Overview first, zoom and filter, details on demand ».

Le processus proposé par dos Santos et Brodlie (cf. Figure 1.5) a très vite fait l'unanimité dans la communauté pour réaliser des algorithmes de visualisation. Il met en évidence les différents composants intervenant dans le processus de visualisation d'informations. Celui-ci part des données brutes pour arriver à une image représentant les données. Pour cela quatre étapes sont nécessaires :

1. l'Analyse des données (*"Data Analysis"*)
2. le Filtrage (*"Filtering"*)
3. le Plongement Visuel (*"Mapping"*)
4. le Rendu (*"Rendering"*)

L'étape d'analyse transforme les données de manière à ce qu'elles deviennent exploitables par le système et les autres étapes du processus. On obtient alors les "données préparées". Ces dernières passent ensuite par une étape de filtrage qui fournira les "données

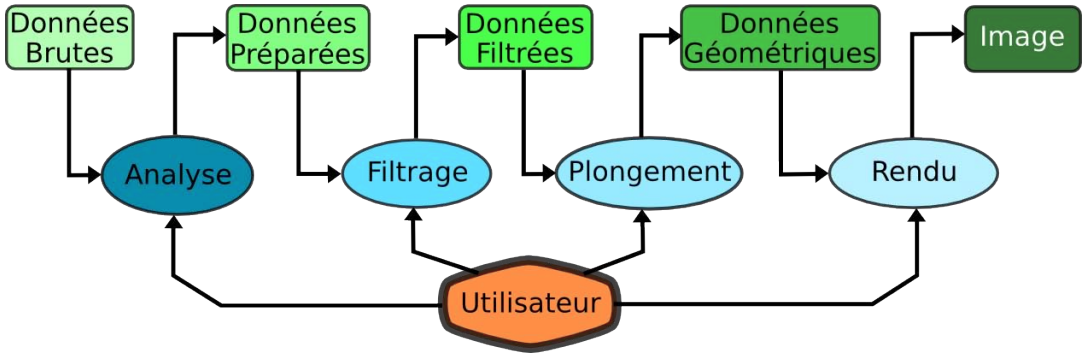


FIG. 1.5: Processus de visualisation tiré des travaux de dos Santos et Brodlie. Il utilise en entrée les données brutes qui passent au travers de quatre étapes de traitement avant de proposer une image représentant ces données. Ces étapes sont : l'analyse des données, le filtrage, le plongement visuel et le rendu. On peut voir qu'après chaque traitement les données sont dans une nouvelle forme et deviennent à leur tour le point d'entrée du traitement suivant. On peut voir que l'utilisateur tient un rôle important car il a la possibilité d'agir sur chacune des quatre étapes.

filtrées" à l'étape de plongement visuel. Cette étape produit des "données géométriques" auxquelles sera appliqué l'étape de rendu pour finalement obtenir l'image finale des données. Même si ces quatre étapes ont chacune un rôle précis, on peut voir qu'elles forment deux blocs au sein du processus. En effet, d'un coté l'analyse et le filtrage sont très liés et de l'autre le plongement visuel et le rendu sont complémentaires. C'est donc en s'appuyant sur cette notion de bloc au sein du processus que nous présenterons ces étapes. La dernière chose que l'on peut noter dans ce processus, est que l'utilisateur joue un rôle important tout au long du processus. En effet, il a la possibilité d'agir sur chacune de ces étapes. Étant donné qu'il est le seul à avoir connaissance de la tâche qu'il souhaite accomplir, il doit pouvoir agir sur les paramètres du processus ou bien le processus doit le solliciter lors de certains choix.

1.3 Analyse des données et Filtrage

L'analyse des données a pour but de produire une abstraction des données. Dans cette thèse, nous nous intéressons aux données relationnelles. Il existe de nombreuses manière de représenter de telles données. Par exemple dans des sports d'opposition, on a pour habitude de représenter l'ensemble des confrontations entre les participants sous la forme d'une matrice. Dans le cas des relations de filiation on a plus l'habitude d'utiliser un arbre. Ce qui a donné le terme arbre généalogique qui depuis est passé dans le langage courant.

Dans cette thèse, nous nous concentrons sur des représentations des graphes de type nœud-lien. Nous considérons également que les données sont stockées sous forme de table. Lors de l'analyse des données, un rôle doit être affecté à chaque élément de la table. Par exemple, il faut identifier dans la table quels sont les éléments qui joueront le rôle de sommets, mais aussi définir les relations qui existent entre les sommets afin de les modéliser à l'aide d'arêtes. De plus, il est important de construire le graphe le plus complet possible afin de ne pas manquer d'informations (attributs) par la suite. Pour cela, l'analyse de données détermine également un certains nombre d'attributs qui vont apporter de l'information supplémentaire sur les sommets ou sur les arêtes.

Cette étape d'analyse des données est à notre sens le point le plus important du processus. En effet, une mauvaise analyse engendre une mauvaise abstraction de ces données, et peut amener l'utilisateur à faire des mauvaises conclusions lors de l'étape d'analyse visuelle. Le plus souvent, l'utilisateur est amené à définir les règles qui vont permettre d'abstraire ses données. On se rend compte aussi qu'il doit, souvent, réaliser lui-même cette abstraction. Ce qui donne un rôle encore plus crucial à l'étape d'analyse.

Le filtrage, lui, intervient sur les données abstraites. Il a pour but de permettre à l'utilisateur de sélectionner certains éléments. Pour cela, l'utilisateur va pouvoir agir sur les attributs identifiés lors de l'étape d'analyse. Il pourra ainsi décider de conserver uniquement les sommets ou arêtes ayant une valeur d'attributs inférieur/supérieur à une valeur seuil, ou alors ne conserver qu'une partie des sommets car la tâche qu'il doit résoudre porte uniquement sur un groupe identifié.

C'est bien l'action combinée de ces deux étapes qui va permettre à l'utilisateur de construire une représentation de ses données et d'y sélectionner certaines informations afin de répondre aux tâches ou questions qu'il se pose. C'est pour cela que nous voyons ces deux étapes du processus comme un bloc indissociable. C'est ce bloc qui se trouve être le moteur de l'analyse que l'utilisateur pourra effectuer.

1.4 Plongement visuel et Rendu

Le plongement visuel et le rendu forment quant à eux le second bloc du processus de visualisation. Ce bloc a un rôle complémentaire au précédant. En effet, celui-ci va avoir pour but de produire une image des résultats obtenus par le bloc précédant.

Dans un premier temps, un plongement visuel est appliqué aux données sélectionnées. Lors de cette étape, on va chercher à affecter à chaque sommet et arête du graphe une position géométrique dans le plan ou l'espace à trois dimensions, en utilisant des algorithmes de dessins de graphe. Ces algorithmes utilisent des travaux de deux domaines : la théorie des graphes [25] et l'algorithmique géométrique [9]. Certaines propriétés des graphes permettent d'argumenter sur l'algorithme à utiliser. Mais le plus souvent c'est la tâche que doit accomplir l'utilisateur qui détermine de fait l'algorithme de dessin. Par exemple, dans le cas des réseaux sociaux, on sait que les algorithmes à modèle de force sont particulièrement performants lorsqu'il s'agit de mettre en évidence des communautés [41, 69, 84].

Une fois les positions des sommets et arêtes déterminées, on passe à l'étape de rendu. Celle-ci produit l'image finale qui sera affichée sur l'écran. Les principes de cette étape relèvent plus du domaine de la synthèse d'images dans le sens où l'ensemble des informations collectée lors des étapes précédentes vont servir à produire des objets graphiques. Par exemple, le fait que l'utilisateur ait choisi d'utiliser des disques bleus pour représenter les sommets et des courbes de Bézier rouges pour les arêtes, constitue les paramètres de l'affichage. Le moteur graphique utilise ces informations pour effectivement dessiner un disque bleu pour chacune des positions de sommets déterminées lors de l'étape de plongement visuel et les relier le cas échéant par des courbes de Bézier rouges. On peut agir sur la qualité du rendu en jouant sur des paramètres tels que les textures, l'ombrage ou la transparence.

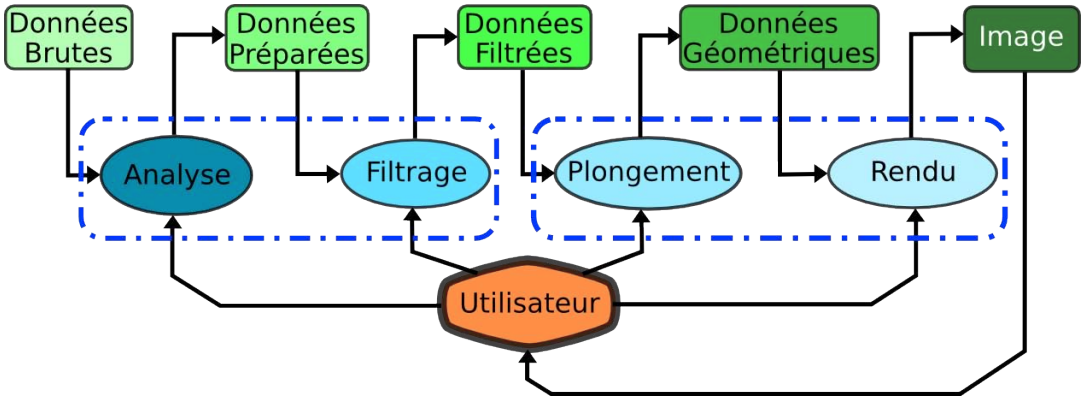


FIG. 1.6: Processus de visualisation intégrant la notion de boucle de rétroaction. Celui-ci reste très proche du processus de dos Santos et Brodlie. On retrouve en entrée les données brutes, puis les quatre étapes de traitement, et enfin une image représentant les données. Nous avons fait apparaitre sur ce diagramme en pointillé les blocs décrits dans les Sections 1.4 et 1.5. On peut voir que l'utilisateur tient un rôle encore plus important que dans le processus de la Figure 1.5. En plus d'agir sur chaque étape, il peut tirer des conclusions de l'image produite pour agir de nouveau sur les étapes du processus.

1.5 Interaction avec l'utilisateur

Le terme d'interaction avec l'utilisateur désigne l'ensemble de actions proposées à l'utilisateurs par le système. Cela comprend aussi bien, le choix de certaines valeurs seuils lors de l'étape de filtrage que le choix d'un algorithme de dessin particulier. On voit nettement la notion d'action s'inscrivant dans le processus, mais il existe une autre catégorie d'interactions : celles liées à l'image finale. En effet, une fois que l'image finale est produite, l'utilisateur peut vouloir agrandir cette image afin d'en observer une zone précise ou encore obtenir les informations propres à un sommet particulier du graphe. Pour cela l'utilisateur a généralement accès à un mécanisme de zoom et de défilement de l'image. Mais il est possible d'imaginer un grand nombre d'interactions, comme le fait de pouvoir déplacer manuellement quelques sommets, en changer la couleur ou la forme.

On retrouve aussi dans les interactions les requêtes faites par le système. Par exemple, la plupart des algorithmes de dessin de graphe requièrent un réglage de paramètres avant de s'exécuter. Pour cela, le système va solliciter l'utilisateur afin que celui-ci le renseigne.

On voit alors apparaitre une boucle entre l'utilisateur et le système (cf. Figure 1.6). En effet, le choix de certains paramètres pour un algorithme va produire une image qui conviendra ou non à l'utilisateur. Si jamais celle-ci ne convient pas, l'utilisateur pourra modifier les paramètres afin de générer une nouvelles image. Il est possible de remonter aussi loin qu'on le désire dans le processus, d'où le terme de boucle de rétroaction. On peut imaginer que le choix d'un algorithme de dessin produise une image qui amène l'utilisateur à formuler une nouvelle hypothèse. Pour vérifier cette hypothèse, l'utilisateur pourra être amené à modifier une valeur de filtrage qui se trouve être en amont dans le processus de visualisation.

1.6 Organisation du Manuscrit

Nous décrivons dans cette section l'organisation de ce manuscrit et nous présentons brièvement les sujets abordés dans chacun des six chapitres qui vont suivre.

Dans le Chapitre 2, nous présentons l'ensemble des définitions et notations que nous allons utiliser tout au long de ce manuscrit. Les définitions de la Section 2.3 interviennent indifféremment dans les Chapitres 4, 5 et 6. Alors que celles de la Section 2.2 introduisent des notions auxquelles nous feront plus particulièrement appel dans les Chapitres 5 et 6.

Puis dans le Chapitre 3, nous présenterons les travaux de références qui forment l'état de l'art des domaines abordés dans cette thèse. Ceux-ci sont regroupés en deux parties, une portant sur l'analyse de réseaux sociaux dynamiques et l'autre sur les méthodes et systèmes permettant de générer automatiquement des visualisations. En ce qui concerne l'analyse de réseaux sociaux dynamiques, nous dressons plus particulièrement un état des lieux des techniques permettant de prendre en compte l'aspect temporel des données dans les réseaux sociaux. La plupart des travaux existant se concentrent sur la visualisation de l'évolution des réseaux dans le temps plutôt que sur l'analyse dans le temps des réseaux. Pour ce qui est des méthodes et systèmes de générations de graphes à partir de données, nous avons recensé trois types de méthodes que nous présentons accompagnées de systèmes qui incarnent ces méthodes.

Nous présentons dans le Chapitre 4, les travaux que nous avons réalisés sur le domaine de l'analyse de réseaux sociaux [16, 17, 36]. Les quatre étapes décrites composent notre processus d'analyse. On trouve tout d'abord la discrétisation du graphe, qui permettra dans un second temps de le décomposer. Par la suite nous détectons les changements qui s'y sont produits avant de terminer par la construction d'une hiérarchie d'influence. Nous décrivons également le système que nous avons développé afin d'expliciter les interactions mises en place pour guider l'utilisateur dans la processus d'analyse. Puis nous proposons deux cas d'études, un réalisé sur un jeu de données de publications et l'autre réalisé sur un jeu de données regroupant des appels téléphoniques.

Dans les Chapitres 5 et 6, nous présentons les travaux que nous avons menés sur l'import de données dans les systèmes de visualisation et la génération automatique de visualisation.

Nous abordons dans le Chapitre 5 la génération automatique de graphes à partir de tables [37]. Pour cela, nous cherchons à ordonner les dimensions du jeu de données afin d'en simplifier le traitement. L'ordre obtenu permet de déterminer si une dimension fournit des informations plus ou moins précises sur les entités (lignes) de la table. Ainsi l'utilisateur peut choisir le degré de précision suivant lequel il souhaite construire le graphe qui modélisera les relations entre les entités de la table. Nous présentons tout d'abord le processus que nous avons mis en place avant de détailler chacune des étapes de ce processus. Enfin nous présentons les interactions mises en place pour que l'utilisateur puisse agir sur le processus de création. Nous proposons également un exemple d'utilisation réalisé à partir d'un jeu de données que nous avons synthétisé.

Nous présentons dans le Chapitre 6 [35] des travaux portant sur la même problématique que celle du Chapitre 5. Dans le Chapitre 5, nous considérons que chaque ligne de la table est une entité et que les dimensions de la table définissent les attributs de ces dimensions. Ici, nous considérons que chaque ligne de la table est une relation et que les dimensions définissent les entités en relation. Nous cherchons à mettre en évidence

l'existence d'intersections entre les dimensions qui définiront ainsi des domaines d'entités. Nous présentons tout d'abord le processus que nous avons mis en place avant de détailler chacune des étapes de ce processus. Nous décrivons les interactions mis en place avec l'utilisateur afin que celui-ci puisse utiliser ses connaissances sur le jeu de données dans le processus de création des graphes. Nous présenterons aussi un cas d'étude mené sur un jeu de données regroupant des publications.

Enfin, nous présentons dans le Chapitre 7 les conclusions que l'on peut tirer sur les travaux menés dans cette thèse, ainsi que les perspectives de recherche des domaines abordés.

Chapitre 2
Notations et définitions

Nous présentons dans ce chapitre les principales notions, notations et définitions nécessaires à la compréhension de ce manuscrit. Dans un premier temps, nous donnons des notions ensemblistes dans la Section 2.1, puis nous donnons des définitions associées à l'utilisation de tables dans la Section 2.2. Pour finir, nous définissons des notions de la théorie des graphes dans la Section 2.3

2.1 Ensemble

Dans cette partie, nous définissons des notions qui nous serviront à la fois pour les définitions se rapportant aux tables (Section 2.2) ainsi qu'aux définitions se rapportant aux graphes (Section 2.3).

Définition 2.1 (Ensemble) *Un ensemble est une collection ou un groupement d'objets distincts. Ces objets s'appellent les éléments de cet ensemble.*

Définition 2.2 (Multi-ensemble) *On appelle multi-ensemble, un ensemble pouvant contenir plusieurs fois le même élément.*

Définition 2.3 (Alphabet d'un multi-ensemble) *Soit E un multi-ensemble. L'alphabet associé à E, noté Σ_E, est le plus grand ensemble contenu dans E. La $n^{ième}$ valeur de l'alphabet associée à E sera notée $\Sigma_{E,n}$.*

Définition 2.4 (Alphabet d'un ensemble de multi-ensembles) *Soit $\mathcal{E} = \{E_1, E_2, \dots E_n\}$ un ensemble de n multi-ensembles. L'alphabet associé à $\mathcal{E}$, noté $\Sigma_{\mathcal{E}}$, est défini par :*

$$\Sigma_{\mathcal{E}} = \bigcup_{i=1}^{n} \Sigma_{E_i} \ .$$

Définition 2.5 (Cardinalité) *Soit E un ensemble ou un multi-ensemble. La cardinalité de E, notée $|E|$, est le nombre d'éléments qui compose E. On note $|E|_e$ le nombre de fois que l'élément e apparait dans E.*

Définition 2.6 (Décomposition d'ensemble) *Soient E un ensemble et E_i des sous-ensembles de E tels que $\forall i \in [1 \dots n], E_i \subseteq E$. Soit $D = \{E_i\}_{1 \leq i \leq n}$ un ensemble d'ensembles E_i. D est une décomposition de E si et seulement si :*

$$\bigcup_{i=1}^{n} E_i = E \ .$$

Définition 2.7 (Partition d'ensemble) *Soient E un ensemble et $P = \{E_1, E_2, \ldots, E_n\}$ une décomposition de E. P est une partition de E si et seulement si $\forall i \in [1 \ldots p], \forall j \in [1 \ldots p], i \neq j$, on a $E_i \cap E_j = \emptyset$.*

Définition 2.8 (Paire d'éléments) *Soit E un ensemble. Une paire d'éléments de E est un ensemble non-ordonné de deux éléments $\{u, v\}$ tel que $\{u, v\} \in E$. Une paire d'éléments de E peut aussi être vue comme un sous-ensemble de E de cardinalité 2.*

Définition 2.9 (Couple d'éléments) *Soient E un ensemble et u, v deux éléments de E. Le couple formé de u et de v est la donnée de ces deux éléments dans un ordre déterminé. Un tel couple est noté (u, v).*

Définition 2.10 (Application) *Une application est une relation entre deux ensembles E et F, pour laquelle chaque élément de ensemble de départ E est relié à un unique élément de l'ensemble d'arrivée F.*

Définition 2.11 (Injection) *Une application f de E dans F est dite injective ou est une injection si pour tout élément y de l'ensemble d'arrivée F, il existe au plus un élément x dans l'ensemble de définition E tel que $f(x) = y$. On dit encore dans ce cas que tout élément y de F admet au plus un antécédent x (par f), c'est à dire :*

$$\forall (x, y) \in X^2, \, (f(x) = f(y) \implies x = y) \ .$$

On notera $i : E \to F$ une injection de E vers F.

Définition 2.12 (Surjection) *Une application f de E dans F est dite surjective ou est un surjection si pour tout élément y de l'ensemble d'arrivée F, il existe au moins un antécédent x par f, c'est-à-dire si :*

$$\forall y \in F, \, \exists \ x \in E, \, f(x) = y \ .$$

On notera $s : E \to F$ une surjection de E vers F.

Définition 2.13 (Bijection) *Une application f de E dans F est dite bijective ou est une bijection si et seulement si tout élément y de l'ensemble d'arrivée F a exactement un antécédent x par f dans l'ensemble de départ E, c'est-à-dire si :*

$$\forall \ y \in F, \ \exists! \ x \in E / \ f(x) = y \ .$$

Donc une application est dite bijective si elle est à la fois injective et surjective.

On notera $b : E \to F$ une bijection de E vers F.

2.2 Table/Données

2.2.1 Type de données

On désigne par type de données le rôle que va jouer une donnée lors de l'analyse et du traitement d'informations. On considère qu'il existe trois types de données décrites par Ware dans [93] : les entités, les relations et les attributs.

2.2.1.1 Entité

Les entités sont les objets que l'on étudie et sur lesquels on souhaite conserver des informations. Une entité peut aussi bien représenter un objet concret qu'un objet abstrait tant que l'objet peut être reconnu distinctement. Par exemple, pour un géographe qui souhaite réaliser un recensement dans un région géographique, les entités seront les habitants de cette région.

2.2.1.2 Relation

Les relations donnent des informations entre les entités, elles les structurent. On peut considérer qu'il n'existe qu'un seul type de relation, les relations binaires. Ce sont les attributs qui nous permettent de les différencier. Par exemple lors du recensement, le géographe pourra créer une relation ayant comme attribut le terme "habite dans le même foyer" afin d'identifier les familles. Il peut exister aussi une relation "est le voisin de" ou encore "travaille avec". Il est aussi possible de définir des relations plus spécifiques à un domaine d'application. Par exemple dans la modélisation de phénomènes biologiques, les relations peuvent modéliser la synthèse de protéines.

Nous considérons uniquement le cas des relations binaires. En effet, dans le cadre de cette thèse nous nous limitons aux représentations de type nœud-lien, et celles-ci conviennent parfaitement puisque les liens entre les nœuds matérialisent les relations. De plus, il est intéressant de noter que même si l'on souhaite modéliser des hypergraphes à l'aide de telles représentations, les relations binaires restent encore valables. Il suffit d'effectuer une transformation de l'hypergraphe en graphe biparti. Les hypersommets et les hyperarêtes deviennent les sommets du graphe biparti. On ajoute une arête entre deux sommets du graphe biparti si l'hypersommet correspondant était une extrémité de l'hyperarête.

2.2.1.3 Attribut

Les entités et les relations peuvent comporter elles-mêmes un grand nombre d'informations appelées attributs. Ces attributs sont des caractéristiques intrinsèques de l'objet. Par exemple, dans le cas d'un recensement, la taille, le sexe et l'âge d'un individu sont des attributs des habitants. La date depuis laquelle deux habitants sont voisins est un attribut de la relation identifiée par "est le voisin de". Si on considère ne serait-ce que la taille, le sexe et l'âge d'un individu, on voit alors qu'il existe plusieurs type d'attributs. On peut classer les attributs en trois groupes.

- Les attributs nominaux : Ce sont des mots qui peuvent avoir deux rôles distincts. Le premier est un rôle d'étiquetage. Dans la plupart des cas c'est un attribut de ce type qui permet de différencier les entités. En effet, il est plus simple pour identifier une entité de lui fournir une étiquette unique plutôt que d'utiliser un sous-ensemble de ses attributs. Les attributs nominaux peuvent jouer un second rôle de classification. Par exemple, la marque ou la couleur d'un véhicule sont des attributs nominaux propres à chaque véhicule, toutefois il est possible de les classer suivant leur marque ou suivant leur couleur.

- Les attributs ordinaux : Un ordre doit pouvoir être appliqué sur ces attributs. Il est aussi possible de comparer deux à deux des attributs ordinaux mais il est toutefois impossible d'en calculer la distance. Par exemple, on peut définir des températures grâce aux valeurs : chaud, tiède et froid. Et on peut dire avec certitude que $chaud > tiède > froid$, mais on ne peut pas estimer de combien de degrés varient les températures entre un objet "chaud" et un objet "tiède".

- Les attributs quantitatifs : Ce sont des attributs qui prennent des valeurs continues. Ainsi contrairement aux autres types, il est possible d'effectuer des opérations entre de tels attributs (addition, multiplication...). Il existe donc une distance entre deux attributs. Dans le cas d'un recensement effectué par un géographe, la taille et le poids d'un individu seront des attributs quantitatifs, à condition de les standardiser.

2.2.2 Table

C'est dans une table que sont souvent stockées des informations, que ce soient des résultats d'expériences scientifiques, des sondages ou des relevés d'appels téléphoniques. Il y a d'autres cas où les informations sont stockées dans une structure plus élaborée : une base de données. Mais une base de données peut très bien être vue ou transformée en table.
Ainsi, une table est une représentation graphique et/ou conceptuelle de données, sous forme de colonnes et de lignes. Usuellement, la première ligne et/ou la première colonne servent à identifier ou décrire les informations qu'elles contiennent. L'intersection d'une colonne et d'une ligne s'appelle une cellule et chaque cellule contient une et une seule information. Toutefois cette information peut avoir une structure complexe. Par exemple, une position géographique peut être stockée en utilisant deux cellules, une pour la latitude et une pour la longitude, mais il est possible de stocker dans la même cellule le couple $(latitude, longitude)$. Il est alors évident qu'il est important de fournir avec chaque table un moyen de la lire et de l'exploiter, si la première ligne et/ou la première colonne ne sont pas suffisamment explicites.

Nous considérerons dans ce manuscrit que les tables ont n lignes et m colonnes, et sont notées $T_{n,m}$. Ainsi d'un point de vue plus formel, une table est un ensemble de n multi-ensembles de cardinalité m ou une matrice $M_{n,m}$.

Définition 2.14 (Entrée de la table (Ligne)) *On désigne par "entrée de la table", une ligne de la table. Le plus souvent il s'agit d'un ensemble d'attributs décrivant une entité. Ce qui signifie qu'une ligne de la table correspond à une entité. Elle sera identifiée par la suite de la manière suivante : L_n, la $n^{ième}$ ligne.*

Définition 2.15 (Dimension (Colonne)) *On désigne par "dimension", une colonne de la table. Le plus souvent il s'agit d'un multi-ensemble de valeurs décrivant un domaine. Elle sera identifiée par la suite de la manière suivante : C_n, la $n^{ième}$ colonne. L'alphabet associé à la $n^{ième}$ colonne sera quant à lui noté Σ_n*

Définition 2.16 (Valeur (Cellule)) *On désigne par valeur, le contenu d'une cellule de la table. Elle sera identifiée par la suite de la manière suivante : $T_{i,j}$, la $j^{ème}$ valeur de la $i^{ème}$ dimension de la table.*

2.3 Graphe

La majeure partie des définitions utilisées dans cette section sont tirées de Graph Theory [25].

Définition 2.17 (Graphe non-orienté) *Soient V et E deux ensembles, tels que $E \subseteq \{\{u, v\} \mid u \in V, v \in V\}$. On appelle graphe, noté $G = (V, E)$, la structure $< V, E >$. Les éléments de V (resp. de E) sont appelés des sommets (resp. des arêtes). Soit $e = \{u, v\}$ une arête, les sommets u et v sont appelés les extrémités de l'arête e.*

Définition 2.18 (Graphe orienté) *Soient V et A deux ensembles, tels que $A \subseteq \{(u, v) \mid u \in V, v \in V\}$. On appelle graphe orienté, noté $G = (V, A)$, la structure $< V, A >$. Les éléments de V (resp. de A) sont appelés des sommets (resp. des arcs). Soit $a = (u, v)$ un arc, on appelle le sommet u (resp. v) la source (resp. la destination) de l'arc a.*

Définition 2.19 (Pseudo-graphe ou graphe à arête multiple) *Soient V un ensemble et E un multi-ensemble, tels que $E \subseteq \{\{u, v\} \mid u \in V, v \in V\}$. On appelle pseudo-graphe, noté $G = (V, E)$, la structure $< V, E >$.*

Définition 2.20 (Arêtes parallèles ou arêtes multiples) *Soient $G = (V, E)$ un pseudo-graphe, et $e_0 = \{u_0, v_0\} \in E$ et $e_1 = \{u_1, v_1\} \in E$ deux arêtes de G. On dit que e_0 et e_1 sont parallèles si et seulement si $u_0 = u_1$ et $v_0 = v_1$.*

Par la suite nous utiliserons le terme de graphe pour pseudo-graphe, conformément à l'usage fait dans la communauté. Toutefois, lorsque nous souhaiterons distinguer les graphes définis en 2.17 et 2.18, des pseudo-graphes, nous utiliserons respectivement les termes de graphes simples et multi-graphes. Il est également à noter que les définitions suivantes sont aussi bien applicables aux graphes simples qu'aux multi-graphes.

Définition 2.21 (Graphe dynamique) *Soient $G = (V, E)$ un graphe et $T = [0, T]$ un intervalle. On dit que G est un graphe dynamique si et seulement si à chaque arête $e \in E$ est associé un intervalle $[t_i, t_j] \subseteq [0, T]$. On notera $G_{[t_h, t_l]}$ le graphe représentant tous les sommets de V ainsi que toutes les arêtes telles que l'intervalle leur étant associé vérifie une des deux conditions suivantes : $t_i \in [t_h, t_l]$ ou $t_j \in [t_h, t_l]$.*

Définition 2.22 (Graphe de concept) *Un graphe de concept est un graphe pour lequel l'ensemble V des sommets représente des classes. C'est à dire que chaque sommet représente une classe qui elle-même contient plusieurs objets.*

Définition 2.23 (Hypergraphe) *Un hypergaphe est une généralisation des graphes pour lesquels les arêtes peuvent connecter un nombre arbitraire de sommets.*
Soient V et E deux ensembles, tels que $E \subseteq \mathcal{P}(V)$ où $\mathcal{P}(V)$ est l'ensemble des parties de V. On appelle hypergraphe, noté $HG = (V, E)$, la structure $< V, E >$. Les éléments de V (resp. de E) sont appelés des hypersommets (resp. des hyperarêtes). Soit $e = \{u, v, w_1, \ldots, w_n\}$ une hyperarête, les sommets u, v, w_1, $\ldots$, w_n sont appelés les extrémités de l'hyperarête e.

Définition 2.24 (Sous-graphe) *Soit $G = (V, E)$ un graphe. On appelle $G' = (V', E')$ sous-graphe de G si et seulement si les trois conditions suivantes sont vérifiées :*
- *$V' \subseteq V$.*

- $E' \subseteq E$.
- $\forall (u,v) \mid (u,v) \in E', u \in V' et\ v \in V'$.

Définition 2.25 (Sous-graphe induit) *Soient $G = (V,E)$ et $G' = (V',E')$ deux graphes, tels que G' est un sous-graphe de G. G' est un sous-graphe induit de G si et seulement si $E' = \{(u,v) \mid (u,v) \in E, u \in V' et\ v \in V'\}$.*

Définition 2.26 (Voisinage d'un sommet) *Soient $G = (V,E)$ un graphe et $v \in V$ un sommet de G. On appelle voisinage entrant (resp. sortant), noté $N^-(v)$ (resp. $N^+(v)$) le multi-ensemble $\{u \mid (u,v) \in E\}$ (resp. $\{u \mid (v,u) \in E\}$). On appelle alors voisinage de v le multi-ensemble $N^-(v) \cup N^+(v)$, noté $N(v)$.*

Définition 2.27 (Adjacence d'un sommet) *Soient $G = (V,E)$ un graphe et $v \in V$ un sommet de G. On appelle adjacence entrante (resp. sortante), notée $adj^-(v)$ (resp. $adj^+(v)$) le multi-ensemble $\{(u,v) \mid (u,v) \in E\}$ (resp. $\{(v,u) \mid (v,u) \in E\}$). On appelle alors adjacence de v le multi-ensemble $adj^-(v) \cup adj^+(v)$, noté $adj(v)$.*

Définition 2.28 (Degré d'un sommet) *Soient $G = (V,E)$ un graphe et $v \in V$ un sommet de G. La quantité $|adj^-(v)|$ (resp. $|adj^+(v)|, |adj(v)|$) est appelée degré entrant (resp. degré sortant, degré) de v, notée $deg^-(v)$ (resp. $deg^+(v), deg(v)$). On appelle alors adjacence de v le multi-ensemble $adj^-(v) \cup adj^+(v)$, noté $adj(v)$.*

Définition 2.29 (Valuation) *Soient $G = (V,E)$ un graphe et K un ensemble. On appelle valuation des sommets (resp. des arêtes) du graphe toute application $f_V : V \mapsto K$ (resp. $f_E : E \mapsto K$). On dit alors que G est sommet-valué (resp. arête-valué) et on le note $G = (V,E,f_V)$ (resp. $G = (V,E,f_E)$).*

Définition 2.30 (Chemin non-orienté) *Soit $G = (V,E)$ un graphe non-orienté. On appelle chemin non-orienté (ou chemin) dans G, une séquence $(v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k)$ avec :*

- $\forall i \in [1 \ldots k], v_i \in V$
- $\forall i \in [1 \ldots k-1], e_i = \{v_i, v_{i+1}\} \in E$
- $\forall i,j \in [1 \ldots k], i \neq j, v_i \neq v_j$
- $\forall i,j \in [1 \ldots k-1], i \neq j, e_i \neq e_j$.

Définition 2.31 (Chemin orienté) *Soit $G = (V,A)$ un graphe orienté. On appelle chemin orienté dans G, une séquence $(v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k)$ avec :*

- $\forall i \in [1 \ldots k], v_i \in V$
- $\forall i \in [1 \ldots k-1], e_i = (v_i, v_{i+1}) \in A$
- $\forall i,j \in [1 \ldots k], i \neq j, v_i \neq v_j$
- $\forall i,j \in [1 \ldots k-1], i \neq j, e_i \neq e_j$.

Définition 2.32 (Connexité) *Soit G un graphe non-orienté. Le graphe G est* connexe *si pour tous sommets u et v de G, il existe un chemin (non-orienté) allant de u à v.*

Définition 2.33 (Connexité forte) *Soit G un graphe orienté. Le graphe G est* fortement connexe *si pour tous sommets u et v de G, il existe un chemin (orienté) allant de u vers v et un autre allant de v vers u.*

Définition 2.34 (Composante connexe) *Une composante connexe d'un graphe non orienté $G = (V,E)$ est un sous-graphe $G' = (V',E')$ connexe maximal de ce graphe. Le terme maximal signifie qu'il n'est pas possible d'ajouter de sommets à V' sans briser la condition de connexité du sous-graphe.*

Définition 2.35 (Composante fortement connexe) *Une composante fortement connexe d'un graphe orienté $G = (V, A)$ est un sous-graphe $G' = (V', A')$ fortement connexe maximal de ce graphe. Le terme maximal signifie qu'il n'est pas possible d'ajouter de sommets à V' sans briser la condition de connexité forte du sous-graphe.*

Définition 2.36 (Cycle et DAG) *Soient $G = (V, A)$ un graphe et $v \in V$ un sommet. On appelle cycle tout chemin de v à v. S'il n'existe pas de tel chemin dans G, alors le graphe G est un graphe acyclique aussi appelé DAG pour "Directed Acyclic Graph".*

Définition 2.37 (Arbre enraciné) *Soit $G = (V, A)$ un graphe orienté. On dit que G est un arbre enraciné si et seulement si :*
- *G est un DAG.*
- *$|A| = |V| - 1$.*
- *$\exists! r \in V, deg^-(r) = 0$.*
- *$\forall v \in V \backslash \{r\}, deg^-(v) = 1$.*

Le sommet r est appelé racine de l'arbre et l'ensemble des sommets v tels que $deg^+(v) = 0$ constitue les feuilles de l'arbre.

Définition 2.38 (Hiérarchie) *Soit $G = (V, A)$ un arbre enraciné. On dit que le graphe G est une hiérarchie si $\forall (u, v) \in A$ on peut définir une relation de subordination de u sur v.*

Définition 2.39 (Forêt) *Soit $G = (V, A)$ un graphe. On dit que le graphe G est une forêt si toutes ses composantes connexes sont des arbres. Un tel graphe peut aussi être vu comme une union disjointe d'arbres.*

Définition 2.40 (Graphe complet) *Soit $G = (V, E)$ un graphe. On dit que le graphe G est complet si et seulement si $\forall u \in V$ et $\forall v \in V$, $\{u, v\} \in E$. Un graphe complet à n sommets sera noté K_n et possède $\dfrac{n(n-1)}{2}$ arêtes.*

Définition 2.41 (Clique) *Soient $G = (V, E)$ un graphe et $G' = (V', E')$ un sous-graphe de G. On dit que le sous-graphe G' est une clique si et seulement si G' est un graphe complet.*

Chapitre 3

État de l'art

Nous présentons dans ce chapitre les principales méthodes et les travaux se rapportant au sujet que nous traitons dans cette thèse. Nous avons découpé ce travail de recherche bibliographique en deux sections. La première regroupe les travaux relatifs à l'élaboration d'un système d'analyse de réseaux sociaux dynamiques, que ce soit du point de vue de la détection de communautés, de la visualisation de réseaux sociaux ou l'analyse et la visualisation de graphes dynamiques. Dans la seconde section, nous présenterons les principaux travaux sur la génération de graphes et la visualisation de graphes complexes pour des utilisateurs novices. Nous présenterons cinq logiciels/plateformes. Pour chacun d'eux, nous ferons d'abord une description globale, puis nous en rappellerons les points forts, avant de poursuivre par une discussion ayant pour but de les comparer afin de les positionner les uns par rapport aux autres.

3.1 Analyse de réseaux sociaux dynamiques

Afin de pratiquer une analyse complète d'un réseau social dynamique, il faut respecter certaines étapes. La première consiste à identifier des communautés dans le réseau, la seconde à détecter les changements qui s'opèrent au sein ou entre les communautés au cours du temps et enfin à proposer une visualisation qui prend en compte ces communautés ainsi que les changements. Nous présentons dans cette section les travaux déjà réalisés par la communauté sur chacun de ces trois points. Puis nous présenterons quatre systèmes essayant de répondre à ces trois phases de l'analyse.

La détection de communautés dans des réseaux sociaux est l'objet de nombreux travaux notamment dans le domaine de la sociologie. Les sociologues utilisent le terme de communauté [23] qui a la même signification et portée que le terme cluster qui est plus souvent utilisé par les statisticiens et les experts de l'extraction de données [86]. Il existe plusieurs travaux de synthèse sur les problèmes rencontrés lors de la détection de clusters ou communautés [12, 47, 72]. Deux types d'approches cohabitent pour ce genre de questions. Les approches dites agglomératives et les approches dites divisives.

Les approches agglomératives se décomposent en deux étapes. La première consiste à calculer une similarité entre toutes les paires de sommets, et la seconde à ajouter des arêtes entre les sommets d'un graphe initialement vide suivant un ordre décroissant de similarité. Un des avantages de ces méthodes est qu'elles peuvent être stoppées à tout moment lors de l'étape d'ajout des arêtes. De plus, quel que soit le moment où la procédure

d'ajout des arêtes a été stoppée, on obtient des communautés. Par contre, ces méthodes sont très dépendantes du choix de la mesure de similarité. Chaque communauté sera révélatrice d'une caractéristique souhaitée par l'utilisateur. Par exemple, si on choisit pour similarité le degré des sommets, on obtiendra une fragmentation des sommets qui ne sera pas forcement corrélée à la topologie du réseau initial. En effet, deux sommets ayant un même degré se retrouveront liés alors qu'ils ne l'étaient pas nécessairement dans le réseau initial. Les méthodes divisives, puisque partant du réseau initial pour en retirer certaines arêtes, respecteront d'avantage l'information liée à la topologie du réseau. En effet, dans ces méthodes la première étape détermine également une similarité entre les sommets, mais pour cela elles utilisent obligatoirement les arêtes présentes dans le réseau. Dès lors que l'on souhaite travailler avec des réseaux sociaux, la topologie de celui-ci doit être prise en compte. Les méthodes divisives correspondront donc mieux à l'analyse qui devra être menée. Parmi les approches divisives celles décrites dans [4, 68, 38] sont les plus utilisées par les chercheurs du domaine en ce qui concerne la détection de communautés dans des réseaux sociaux.

La communauté a fait preuve d'un intérêt moins marqué en ce qui concerne la détection de changements dans les clusters présents dans les données dynamiques [51] ainsi que pour les fragmentations de données évoluant par flux [2]. Les techniques existantes bien que performantes dans le domaine de la fragmentation de données dynamiques, sont soit insuffisantes, soit inefficaces pour caractériser des changements dans des structures communautaires. En effet, comme les interactions qui ont lieu entre deux individus sont modélisées à l'aide d'une relation (par exemple une arête valuée), les interactions entre des communautés sont caractérisées à l'aide d'un grand nombre de ces interactions sur une période de temps. Il faut alors être capable d'identifier les communautés tout au long d'un intervalle de temps mais aussi être capable de comparer les interactions inter-communautaires et non pas seulement les interactions entre les individus..

La visualisation de réseaux sociaux a suscité beaucoup d'intérêt car l'obtention d'images représentant des réseaux sociaux offre aux investigateurs de nouvelles perspectives [32]. Il existe de nombreux logiciels de visualisation permettant l'analyse de réseaux sociaux ([7, 14, 44, 75]). On peut trouver dans [32] un travail de veille scientifique de la littérature existante sur la visualisation des réseaux sociaux. Mais aucun de ces outils ([7, 14, 44, 75]) ne prend en compte l'aspect temporel qui peut exister dans les données. Or, il s'agit d'un point important à coté duquel il ne faut pas passer pour appréhender et analyser correctement ce genre de réseaux. Si ces systèmes ne prennent pas en compte cet aspect temporel, ce n'est pas uniquement à cause d'un manque d'intérêt pour ce problème. C'est aussi parce que, comme nous l'avons souligné, il y a un retard au niveau des techniques de détection de changement dans les réseaux sociaux dynamiques. Il est alors logique de retrouver ce même retard en ce qui concerne la visualisation de tels réseaux.

La recherche sur le domaine de l'analyse et la visualisation de graphes dynamiques a suscité un réel intérêt, comme le montre le grand nombre de systèmes s'attelant à cette tâche. Par exemple Kang *et al.* [52] ont proposé un outil appelé C-Group permettant de réaliser une analyse temporelle de réseaux sociaux. Cet outil utilise un modèle biparti pour construire les réseaux, d'un côté il considère les acteurs du réseaux et de l'autre les événements. La construction du graphe à partir du modèle aboutit toujours à un graphe de type acteur-acteur mais repose sur une des trois règles suivantes pour relier les acteurs entre eux :

– deux acteurs sont reliés s'ils ont au moins un attribut en commun,

– deux acteurs sont reliés s'ils ont pris part à des événement ayant un attribut en commun,
– deux acteurs sont reliés s'ils apparaissent dans des participations à des événements ayant elles-mêmes un attribut en commun.

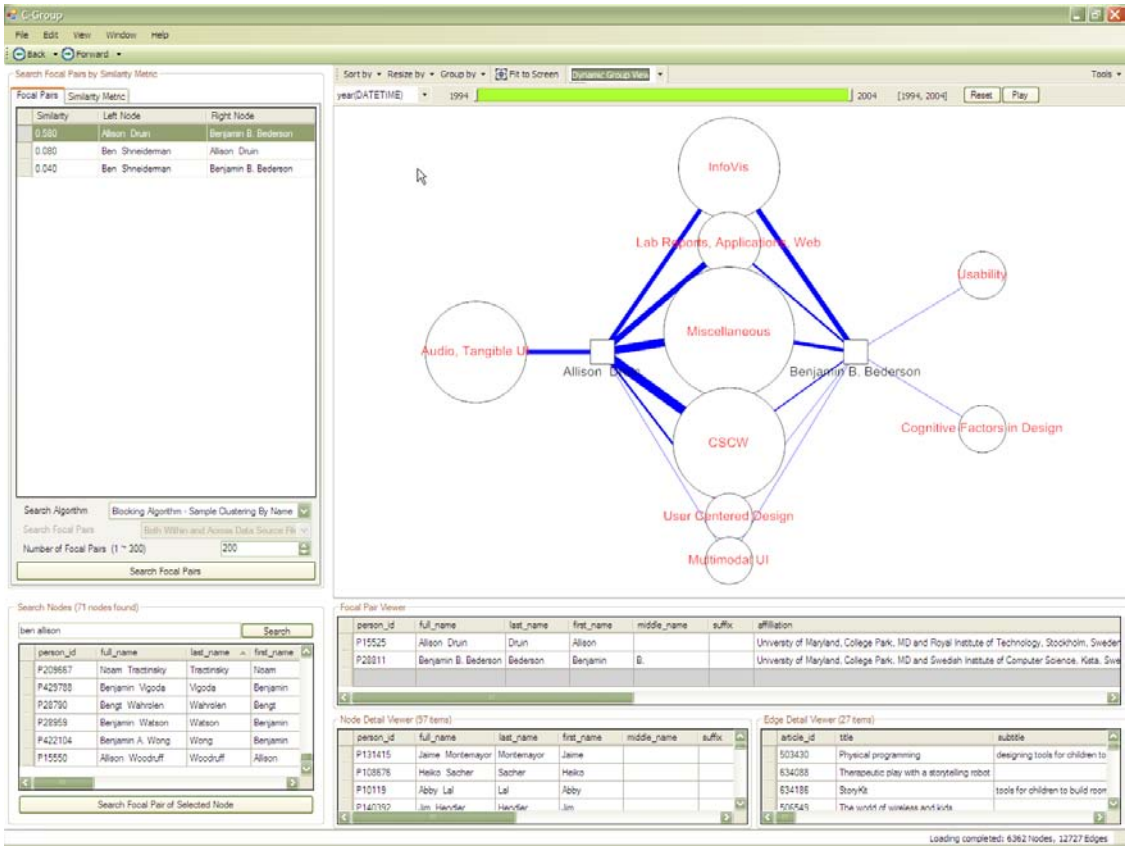


FIG. 3.1: Logiciel C-Group. Il permet d'analyser la portion du réseau concernée par la paire de sommets représentés par des carrés. Les sommets ayant des valeurs d'attributs égales pour une dimension choisie sont regroupés dans un cercle. Plus une arête reliant un sommet de la parie étudiée à un des cercles est large, plus le nombre de connexions avec ce groupe est grand.

En ce qui concerne l'analyse du réseau, C-Group se concentre sur une paire d'individus. Étant donné qu'il n'est pas possible d'obtenir une vue globale du réseau, les individus de la paire que l'on souhaite étudier sont sélectionnés directement dans la table des acteurs du réseau. Une fois la paire d'individus choisie, le système propose de visualiser le voisinage de ces individus c'est à dire l'ensemble des événement auxquels ils ont pris part. Ces voisinages sont divisés en trois parties dans le dessin :

– le voisinage commun aux deux individus, qui sera dessiné entre les deux individus,
– le voisinage propre au membre gauche de la paire qui sera dessiné à sa gauche,
– le voisinage propre au membre droit de la paire qui sera dessiné à sa droite.

Dans chacune de ces parties, des groupes d'acteurs sont constitués à l'aide de filtrage sur les valeurs de la règle de construction choisie. Par exemple, s'il s'agit d'un réseau de co-auteurs, il sera possible de constituer des groupes en fonction des affiliations (laboratoire, entreprise, ...) des acteurs. Par la suite, il est possible d'utiliser le fait que les événement sont datés pour animer les voisinages. Ainsi, on peut voir apparaitre ou disparaitre certaines arêtes. Une arête peut disparaitre (respectivement apparaitre) si l'événement associé s'est terminé

(respectivement commence) ou si l'acteur ne prend plus part (respectivement prend part) à l'événement.

Cet outil propose donc une analyse dynamique d'un réseau. Mais en se positionnant à l'échelle d'une paire d'individus, l'analyse faite sera une analyse locale et ne pourra en aucun cas proposer une visualisation globale de l'évolution du réseau ou des groupes du réseau.

Gloor *et al.* [39] proposent quant à eux un algorithme à fenêtre de temps glissante pour afficher les connections entre des acteurs sur un intervalle de temps. Pour un graphe donné, un dessin du graphe dans sa globalité est calculé. Les sommets se voient alors attribué une position définitive. Puis une échelle de temps (jours, heure, ...) est définie, l'intervalle de temps est alors découpée en fonction de cette échelle. Pour chaque tranche du découpage, un graphe contenant tous les sommets est construit. Les sommets sont positionnés aux coordonnées déjà déterminées et seules les arêtes concernées par ce laps de temps sont ajoutées au graphe. Ainsi, plusieurs "photos" (snapshots) sont créées et à l'aide d'une barre de défilement, l'utilisateur peut choisir de visualiser le graphe correspondant à une période. Cette approche fonctionne très bien lorsqu'il s'agit de suivre l'évolution de relations entre des individus mais ne permet pas de capturer l'évolution des structures communautaires en considérant le réseau dans sa globalité.

Sarkar et Moore [71] présentent une méthode pour modéliser les relations qui changent au cours du temps. L'idée qu'ils proposent consiste à développer une compréhension des données historiques et d'en déduire les futures interactions. Pour cela, plusieurs graphes sont construits à partir du jeu de donnée. Chacun de ces graphes correspond à un intervalle de temps donné et ces intervalles ne possèdent pas d'intersection. On retrouve ici la notion de "photos" (snapshots) du graphe global évoqué dans le système précédant. Puis à l'aide de méthodes de Multidimentionnal scaling (MDS) [15] et de chaine de Markov [40], en comparant des photos successives du graphe, un modèle d'apprentissage est construit. Sa robustesse tient dans le fait qu'il est possible d'affiner ce modèle à chaque comparaison de photos. Ainsi, le modèle produit sera capable de prédire des futures évolutions du réseau, aussi bien les disparitions d'arêtes que leurs apparitions. Toutefois, il est important de noter que le graphe initial ne comporte pas d'arêtes. Celles-ci sont déterminées à l'aide de la proximité des entités définie par les méthodes MDS. On peut alors s'interroger sur la fiabilité de la méthode à construire le réseau tel que l'utilisateur souhaitait le visualiser. De plus, ce modèle peut être utilisé pour étudier le comportement d'une relation particulière entre deux entités, mais demande à être radicalement modifié afin de pouvoir appréhender le comportement d'une communauté.

SoNIA (Social Network Image Animator) [8] est une plateforme qui permet d'animer les représentations de réseaux dynamiques au cours du temps. Elle permet de pratiquer une exploration de données relationnelles dynamiques et a pour vocation de permettre la comparaison de différentes techniques de représentations (dessins) fournissant des animations fiables du réseau. En aucun cas elle ne capture l'évolution dynamique d'un groupe de personnes (cluster) mais agrège et transforme les données dynamiques pour donner une dimension sociale stable nécessaire à la création d'une visualisation significative. Le travail réalisé dans le cadre du développement de cette plateforme traite plus de la représentation, de l'animation et des performances des algorithmes de dessins que de l'analyse de réseau sociaux dynamiques. En effet, elle ne comporte pas les outils (mesures) nécessaires à l'analyse de tels réseaux. Si la plateforme SoNIA ne propose pas de processus d'analyse, elle présente en revanche de bons outils (dessins et animations) pour explorer ces réseaux.

Moody *et al.* [66], introduisent deux types de visualisations : Les "flip books" et les "dynamic movies". Les "flip books" consistent à attribuer une fois pour toute une position aux sommets puis ce sont les arêtes qui vont évoluer (apparaitre ou disparaitre) entre les sommets à l'instar des travaux de Gloor *et al.* que nous avons présentés. Les "dynamic movies" proposent d'animer les déplacement des sommets en fonction des changements qui s'opèrent dans le réseau. Mais encore une fois, l'accent est mis sur le traitement des données dynamiques, mais aucun apport n'est réalisé en ce qui concerne l'analyse des réseaux sociaux construits et animés.

Il existe donc un grand nombre de méthodes permettant de construire à partir de données dynamiques des réseaux sociaux, mais très peu permettent d'en faire une analyse dans le temps. Les méthodes ou plateformes traitant de l'analyse se limitent quant à elles, le plus souvent, à l'analyse d'une paire de sommets. Cette analyse peut porter sur la paire exclusivement ou offrir une vue un peu plus élargie en y ajoutant son voisinage. Mais aucune méthode n'est capable de capturer les structures communautaires des réseaux et d'en mettre en évidence les évolutions au cours du temps.

3.2 Des données à la visualisation

Comme on peut le voir dans la Figure 1.2 les données constituent le point d'entrée de tout système de visualisation. Il est donc primordial que leurs interprétations par le système soient aisées et sans erreur. Il se trouve que les systèmes de visualisation ont, pour la majeure partie d'entre eux, développé leurs propres formats de données. Ce qui a pour effet d'obliger les utilisateurs à manipuler leurs données afin qu'elles répondent aux exigences du système. Ces manipulations peuvent être à l'origine d'erreurs pouvant fausser l'analyse et donc l'exploitation des résultats sans que cela ne soit perçu par l'utilisateur. De plus, les données doivent être formatées de manière à décrire le plus possible d'informations dès leur mise en forme pour ne plus être manipulées par la suite. Il serait pourtant plus simple d'utiliser un format générique qui permettrait aux utilisateurs de ne pas avoir à formater leurs données avant de pouvoir entamer un processus d'analyse et de visualisation ou bien même une analyse visuelle. La charge serait alors attribuée à chaque système. Il faudrait mettre en place un mécanisme d'import de données qui guiderait et simplifierait les manipulations. Si chaque système de visualisation mettait en place de tels mécanismes, il serait alors plus facile d'appréhender un système de visualisation. Cette difficulté d'appréhension et d'apprentissage représente souvent un réel obstacle à l'utilisation des systèmes de visualisation qui sont pourtant maintenant complet et très performant pour mener des analyses poussées de grandes quantités de données.

Une telle approche visant à simplifier la mise en forme des données par le système lui-même n'a pas fait l'objet d'une grande quantité de travaux. Beaucoup de logiciels préfèrent à l'heure actuelle s'affronter sur le terrain de la fluidité, du rendu et de la performance. De ce fait, il existe peu de travaux portant sur le problème de l'import ou de l'adaptation des données afin qu'elles puissent être utilisées par les systèmes de visualisations. À l'heure actuelle, on peut trouver trois types d'approches, chacune portée par un système de référence. Nous présentons maintenant ces trois approches, ainsi que la parade utilisée par les systèmes ne s'intéressant pas encore à ce problème.

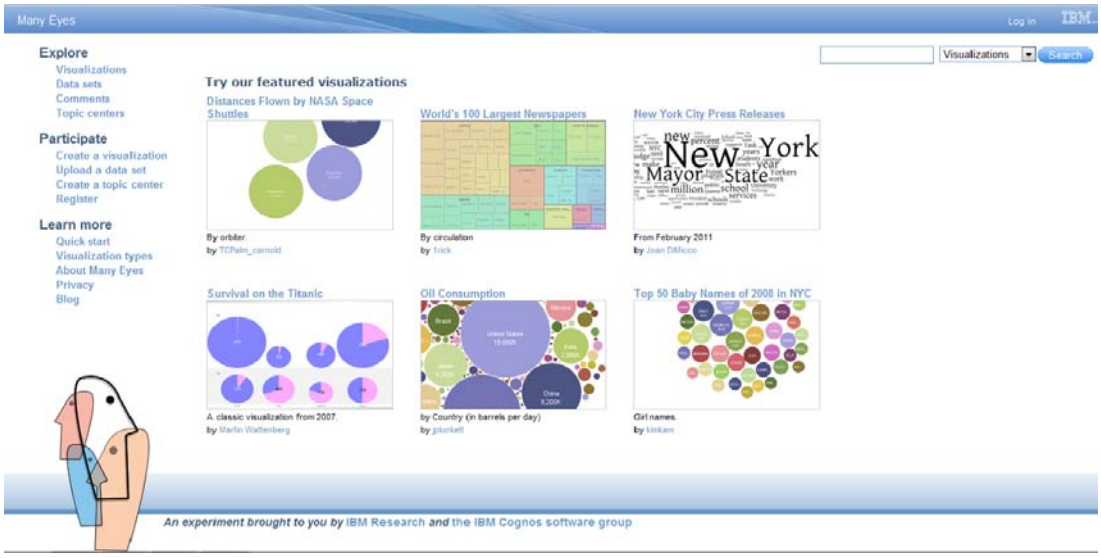


FIG. 3.2: Plateforme Many Eyes. Il s'agit là de la page d'accueil de la plateforme. On peut voir qu'il est possible d'y créer des visualisations de types différents. Par exemple des Tree-Map ou des Nuage de mots clés (Tag Cloud).

3.2.1 Many Eyes [90].

Description : Many Eyes [90] est un service Web lancé en 2007 qui permet aux utilisateurs experts ou non-experts de déposer des jeux de données, de construire une visualisation de ces mêmes données et enfin de discuter de la pertinence de la visualisation choisie avec d'autres utilisateurs.

D'après l'équipe qui développe cette plateforme, il est important que plusieurs personnes puissent travailler sur les mêmes données et débattre sur les différentes visualisations générées à partir de celles-ci. L'argument principal qu'avancent les auteurs repose sur le fait que d'après eux, il est difficile de donner une bonne manière de visualiser un jeu de données. C'est pour cela qu'il a été ajouté à la chaine de production le fait de laisser en ligne et visible par tous les visualisations produites. Pour chaque production, il existe un fil de discussions dans lequel les autres utilisateurs experts ou non-experts peuvent donner leur avis et conseils sur les jeu de données et la visualisation. Ainsi, au final, en ce servant de ces discussions, un utilisateur pourra construire une visualisation plus pertinente. Au lieu d'essayer de construire "la" bonne visualisation pour un jeu de données, Many Eyes préfère offrir l'opportunité d'avoir une visualisation élaborée par plusieurs experts. Cette idée de production collaborative vient parfaitement soutenir le but premier de Many Eyes qui est d'ouvrir le monde de la visualisation d'informations au plus grand nombre.

Il est clair que pour cela, la plateforme doit être conçue pour que des utilisateurs novices puissent arriver à produire des visualisations. Les développeurs à l'origine de cette plateforme ont donc cherché un moyen de guider l'utilisateur novice tout au long de l'élaboration de visualisations. Ils sont partis du constat que si deux jeux de données diffèrent ne serait-ce que d'une dimension, la visualisation des informations qu'ils contiennent ne se fera probablement pas de la même manière. Ils ont donc mis en place un prototypage de différents types de visualisations. Pour chaque type de visualisation, un ensemble de paramètres indispensables a été défini. Si l'utilisateur souhaite visualiser son jeu de données à l'aide d'une visualisation, il doit au préalable s'assurer que son jeu de données répond

bien au prototype défini. Par exemple si l'utilisateur souhaite visualiser ses données sous la forme d'une Treemap, il devra s'assurer que dans son jeu de données il y aura des dimensions nominales définissant la hiérarchie contenue dans les données ainsi que deux dimensions quantitatives qui serviront à déterminer la taille et la couleur des éléments. Si jamais l'utilisateur veut changer de mode de visualisation et que dans l'état actuel son jeu de données ne répond pas au prototype du nouveau mode, la plateforme propose un module de manipulation des données qui sous la forme d'un tableur permet à l'utilisateur de remédier à cela. Ainsi, l'accent est placé sur le fait de vouloir prendre en charge toutes les étapes de production au sein la plateforme. Comme tous les services sont accessibles en ligne, l'utilisateur n'aura pas à jongler entre plusieurs logiciels.

Les points forts de la plateforme Many Eyes sont :

- l'intégration de la manipulation des données au processus
- la génération de visualisation simplifiée grâce à un prototypage des visualisations
- l'élaboration collaborative de visualisation
- un système intégralement sur le Web

Comparaison/Positionnement : La plateforme Many Eyes propose une approche novatrice dans le sens où elle essaie de donner un prototype de chaque mode de visualisation existant. C'est en utilisant ces prototypes que la plateforme va pouvoir guider l'utilisateur dans ses choix. Mais on voit alors qu'il va être amené à manipuler ses données afin que celles-ci répondent au prototype. Or, il n'est pas évident pour un utilisateur novice de transformer un jeu de données. Il est parfois difficile pour lui d'identifier les dimensions qu'il va devoir choisir afin de répondre aux exigences du prototype. Par exemple, dans le cas des Treemaps, il est difficile pour l'utilisateur de déceler la présence des dimensions qui formeront la hiérarchie des éléments. De plus, s'il essaie de visualiser son jeu de données à l'aide d'un autre mode de visualisation, il devra à nouveau manipuler ses données afin que celles-ci satisfassent le nouveau prototype. On peut alors se demander si ces prototypes qui amènent une certaine automatisation de la production de visualisations ne risquent pas de nuire à l'appréhension des méthodes de visualisation par les utilisateurs novices.

Nous pensons qu'il serait plus simple pour l'utilisateur que le système essaie de détecter les dimensions qui peuvent répondre à un prototype. Ainsi, l'utilisateur se verrait proposer des paires composées d'une part d'un mode de visualisation (nœud-lien, histogramme, coordonnées parallèles ...) et de l'autre l'ensemble des dimensions présentes dans les données qui permettent de générer cette visualisation. Ainsi, l'utilisateur n'aura plus besoin de manipuler le jeu de données. Il n'aura plus besoin d'essayer de comprendre comment on construit une visualisation pour s'en servir. Un tel mécanisme de recherche de compatibilité entre les données et des modes de visualisation pourrait amener l'utilisateur à découvrir des associations auxquelles il n'aurait jamais pensé.

Sur la plateforme Many Eyes, ces associations peuvent lui être proposées. En effet, grâce aux fils de discussion associés à chaque production, les autres utilisateurs peuvent suggérer l'utilisation d'autres prototypes. Ils peuvent même indiquer à l'utilisateur comment modifier ses données pour qu'elles correspondent au prototype. Ils peuvent même effectuer toute cette tâche puisque les jeux de données sont publics. Mais alors on voit que l'utilisateur se trouve être dépendant de la communauté d'utilisateurs et de leurs compétences. Il s'agit d'un problème qui ne se pose pas avec un système qui essaie de faire coïncider de manière automatique les dimensions des données avec les prototypes.

Un autre point qui limite la plateforme dans son usage est le fait qu'il n'est pas possible de choisir certains paramètres de la visualisation. Par exemple, une fois qu'il a formaté ses données pour obtenir une visualisation en représentation nœud-lien, l'utilisateur n'a pas la possibilité de choisir un algorithme de dessin. Il doit se contenter de celui fourni par la plateforme. Alors qu'on sait très bien que certains algorithmes sont plus performants (en terme de rendu) dans certains cas que d'autres. Il serait donc plus intéressant de produire une première visualisation que l'utilisateur pourra manipuler par la suite. Il pourra alors choisir un algorithme particulier de dessins, paramétrer des filtres ou encore utiliser des outils d'analyse tels que des fragmentations.

3.2.2 Tableau Software [60].

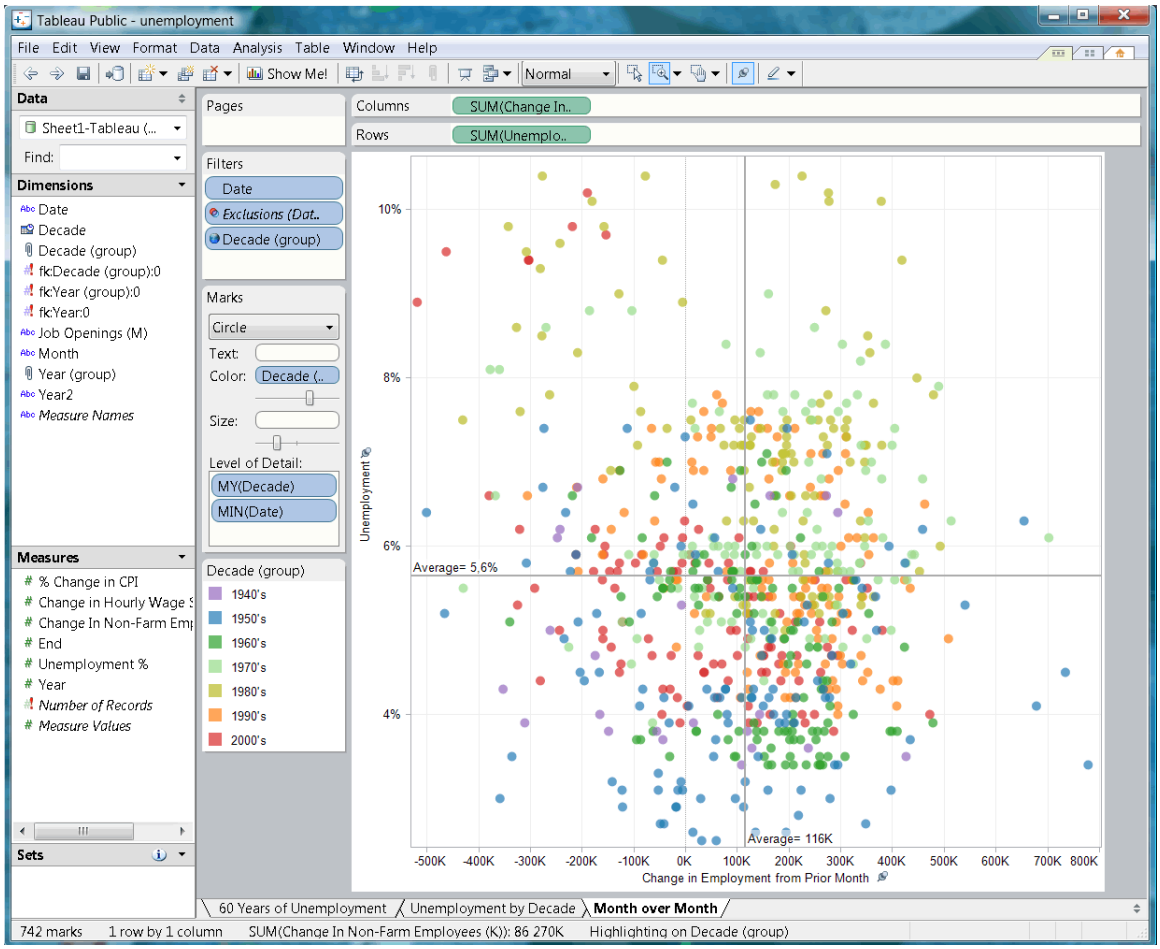


FIG. 3.3: Logiciel Tableau. On peut voir dans le panneau latéral gauche l'ensemble des éléments qui vont intervenir dans l'élaboration d'une visualisation. Il s'agit du choix de la source des données, du noms des dimensions ainsi que les mesures réalisées sur les données. Dans la partie droite de l'interface, on voit les étiquettes (en bleu et vert) de dimensions ou mesures. Suivant leur positionnement dans les champs de l'interface, la visualisation produite ne sera pas la même. Ici le fait d'avoir placé en colonne et en ligne des données quantitative (étiquette verte) a pour effet de produire une visualisation en nuage de points.

Description : Tableau Software [60] est un logiciel développé par Mackinlay. Il est le résultat de plusieurs travaux menés successivement ou parallèlement avec pour objectif commun de simplifier l'utilisation des systèmes de visualisations. Le système permet de travailler avec de grandes bases de données. Pour cela, la vue des données est résumée à une liste des noms de dimensions. Afin qu'il soit plus simple pour l'utilisateur de s'y retrouver, chaque dimension se voit attribuer un type. Ce type, lors de combinaison de dimensions, sera pris en compte par une algèbre (présentée dans Polaris [83]) codant les règles de la sémiologie graphique de Bertin [13].

Dans cette approche, le type des dimensions intervient aussi dans un mécanisme de règles visant à encoder des méthodes de visualisation, de manière à guider l'utilisateur novice dans son choix de mode de visualisation. Par exemple, pour certaines combinaisons de type de dimensions l'utilisateur se verra proposer une visualisation en nuage de points, alors que pour d'autres un histogramme sera plus approprié. Pour générer les visualisations, l'utilisateur manipule exclusivement les noms des dimensions à partir des listes établies initialement. Puis, grâce à la mise en place d'une interface et d'interactions basées sur le "Drag and Drop", l'utilisateur déplace les dimensions qu'il souhaite utiliser dans trois nouvelles listes : une pour les dimensions qui seront utilisées en tant que colonnes dans la visualisation, une pour celles utilisées comme lignes et une pour les attributs. Une fois qu'une première visualisation est créée, il est possible de la modifier en ajoutant ou supprimant des dimensions des différentes listes.

Lors de l'ajout d'une dimension, une règle vient s'assurer que l'utilisateur n'essaie pas de combiner des dimensions qui ne le peuvent pas (pour des raisons de types). Ainsi, le système garantit le maintien des règles de l'algèbre ainsi que la cohérence de la visualisation. L'approche mise en place dans Tableau propose donc un système capable de guider un utilisateur novice de manière à ce qu'il puisse construire une visualisation correcte d'un point de vue sémiologique. Il propose également une abstraction des données brutes permettant de manipuler des grandes quantités de données. Toutefois, ce système ne propose pas tous les types de visualisations qui existent, même s'il en propose un très grand nombre. En effet, il n'est pas possible de générer des visualisations ou de présenter des données relationnelles sous forme de graphe nœud-lien.

Les points forts du logiciel Tableau sont :

- l'abstraction des données pour pouvoir manipuler de grands jeux de données
- la génération de visualisation guidée par le typage automatique des dimensions
- la construction des visualisations évolutive grâce à un mécanisme d'interactions.

Comparaison/Positionnement : L'apport réalisé dans le logiciel Tableau propose une approche totalement différente de celle réalisée dans Many Eyes. On pourrait même dire qu'elle comble les lacunes laissées par Many Eyes. En effet, Tableau met en place des étiquetages automatiques des dimensions du jeu de données. Et suivant les dimensions choisies par l'utilisateur, en se référant à un ensemble de règles, le système propose un panel de mode de représentation à l'utilisateur. Ainsi, aucun effort de manipulation des données n'est demandé à l'utilisateur. Celui-ci manipule uniquement des étiquettes représentant les dimensions. Mais il n'est pas toujours évident pour l'utilisateur de savoir comment organiser les dimensions choisies. Par exemple, il n'est pas évident de cerner comment l'ordre de sélection des dimensions influe sur la visualisation produite. Cela demande un peu d'apprentissage, mais l'aisance que donnent les interactions du système permet à

l'utilisateur d'en comprendre le fonctionnement en procédant par des essais successifs. En effet, le fait de pouvoir manipuler les données par "Drag and Drop" simplifie les interactions et donc la génération de nouvelles représentations.

Toutefois, il est important de noter que le système ne propose pas de vue complète sur les données, l'utilisateur ne manipule que les étiquettes de dimensions. Ce qui signifie que l'utilisateur doit connaître suffisamment bien ses données pour interagir avec le système. De plus, Tableau revendique le fait de pouvoir travailler avec de très grandes bases de données. Mais bien souvent les grandes bases de données comptent un grand nombre de dimensions. Or si ce nombre devient trop grand l'utilisateur ne sera peut-être plus capable d'identifier efficacement les différentes dimensions. Nous pensons donc qu'il est important qu'un système de construction automatique de visualisation propose une vue directe et globale des données. Les étapes suivantes du processus de production (cf. Figure 1.2) sont bien présentes dans Tableau. Contrairement à Many Eyes, une fois qu'une représentation a été choisie, il est alors possible de choisir des paramètres d'affichage tels que les dessins, les tailles des objets etc... On peut donc dire qu'il n'y a pas de discontinuité entre la construction de la représentation et l'analyse qui va pouvoir être menée.

D'un point de vue du panel de représentation que propose Tableau, on peut dire que celui est moins complet que celui de Many Eyes. En effet, lorsque Many Eyes propose des représentation en courbes, nuages de point, Treemaps [49] ou nœud-lien, Tableau se limite aux représentations que définissent les règles de la sémiologie graphique de Bertin. Donc tout le pan que représentent les visualisations de type nœud-lien n'est pas présent dans Tableau.

3.2.3 nodeXl Software [80]

Description : NodeXl [80] est un module développé par Smith, qui s'intègre dans le logiciel Microsoft Excel. Celui-ci reprend l'utilisation de feuilles de calcul du tableur pour y afficher dans un premier temps des données qui permettront de produire un graphe sous la forme nœud-lien. L'utilisation des feuilles de calcul permet un import souvent direct ou plus aisé des données, car comme le font remarquer les auteurs, la plupart des jeux de données sont stockés sous forme de table ou de base de données facilement exportable dans un format tabulaire.

Toutefois, comme il s'agit de produire un graphe relationnel sous la forme nœud-lien, il est demandé à l'utilisateur de mettre en évidence les relations (arêtes) entre les entités. Pour cela, les données doivent être présentées suivant un format strict. Chaque ligne de la table décrit une arête qui sera identifiée à l'aide des deux entités (sommets) qu'elle met en relation. Les sommets sont stockés dans les deux premières colonnes de la ligne puis suit une liste d'attributs qui seront affectés à l'arête. Une fois une telle table chargée dans le module, l'utilisateur peut entamer une analyse visuelle du réseau contenu dans les données. Cette analyse est rendue plus accessible par le calcul automatique de mesures sur le graphe. Ces calculs peuvent être fait directement à partir de la table puisque le fait d'avoir défini un format de présentation des données permet d'utiliser la force d'Excel, à savoir la définition de formule dans le tableur. Ainsi, cet outil ne nécessite pas l'utilisation d'un langage de programmation.

Pour ce qui est de la représentation du réseau sous forme nœud-lien, l'utilisateur à le choix entre plusieurs algorithmes de dessin qu'il peut relancer à chaque fois qu'il apporte une modification au graphe. Ces modifications peuvent être faite soit directement dans

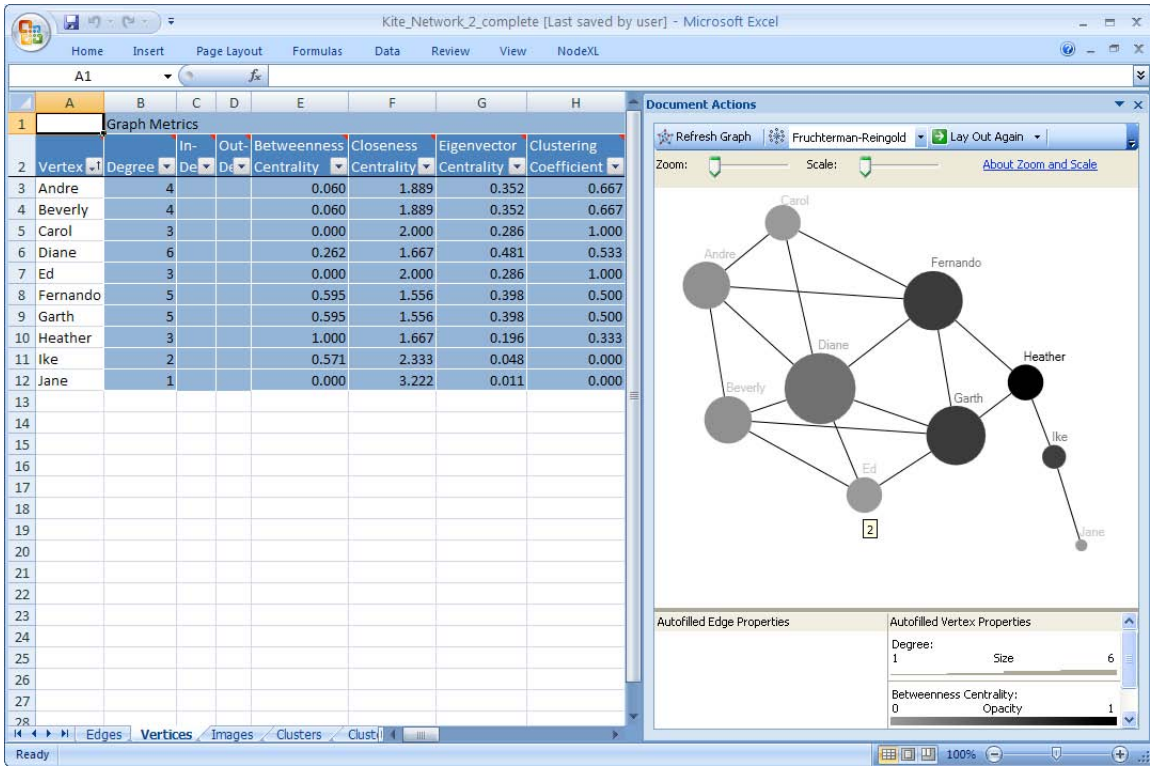


FIG. 3.4: Logiciel NodeXL. L'interface de ce logiciel est composée de deux panneaux. Celui de droite contient la représentation du graphe. Celui de gauche contient les informations sur les différents objets du graphe. Ces informations sont présentées sous le forme de feuille de calcul de type Excel. On peut voir en bas de ce panneau qu'il existe une feuille pour les arêtes (Edges), une pour les sommets (Verticies), ainsi que des pages pour les éventuels clusters calculés.

la représentation nœud-lien soit dans la table. Il est possible d'appliquer des filtres sur le graphe. Chacune des mesures possède son propre filtre afin de permettre à l'utilisateur d'exploiter au mieux les résultats obtenus grâce aux calculs fait par la partie tableau/-formule. Il est également possible d'appliquer des fragmentations sur le graphe. Chaque cluster construit sera observable soit sous la forme nœud-lien soit dans sa propre table respectant le format définit. Ce module permet donc d'importer un réseau (mail, téléphone ...), d'en construire une représentation et enfin de pratiquer une analyse visuelle.

Les points forts du module nodeXL sont :

- la construction d'un graphe nœud-lien à partir d'une table
- le calcul automatique de métriques grâce à la puissance du tableur
- l'analyse visuelle du réseau
- un module intégré à Microsoft Excel.

Comparaison/Positionnement : Contrairement à Tableau qui ne s'intéresse pas du tout aux représentations sous forme nœud-lien, nodeXL, lui, se concentre exclusivement sur ce type de représentation. Il peut se permettre cela car le logiciel Excel propose déjà toute une série de visualisations telles que les histogrammes ou les nuages de points ... Mais au lieu de proposer une manipulation simple des données comme Tableau ou un

prototype codant ce mode de représentation comme le fait Many Eyes, nodeXL définit un format de présentation des données. Ce format est très strict, ce qui a pour effet de mettre l'utilisateur face à une mise en forme des données qui est bien souvent différente des formats obtenus lors de la collecte de ces données. L'utilisateur est bien souvent amené à essayer d'extraire les informations nécessaires afin de créer lui-même la liste d'arêtes. Il est alors impossible d'assurer que les utilisateurs non experts seront capables de construire un graphe valide. On voit que ce choix de format entraine une faille dans le processus de visualisation. En effet, quand Many Eyes et Tableau s'efforcent de guider l'utilisateur pour garantir des constructions cohérentes et correctes, nodeXl lui laisse le champ libre. Nous nous demandons alors pourquoi ne pas reprendre les méthodes de Tableau afin d'ajouter une règle à l'algèbre pour faire en sorte que les listes d'arêtes soient détectées dans les données.

3.2.4 Gephi [6]

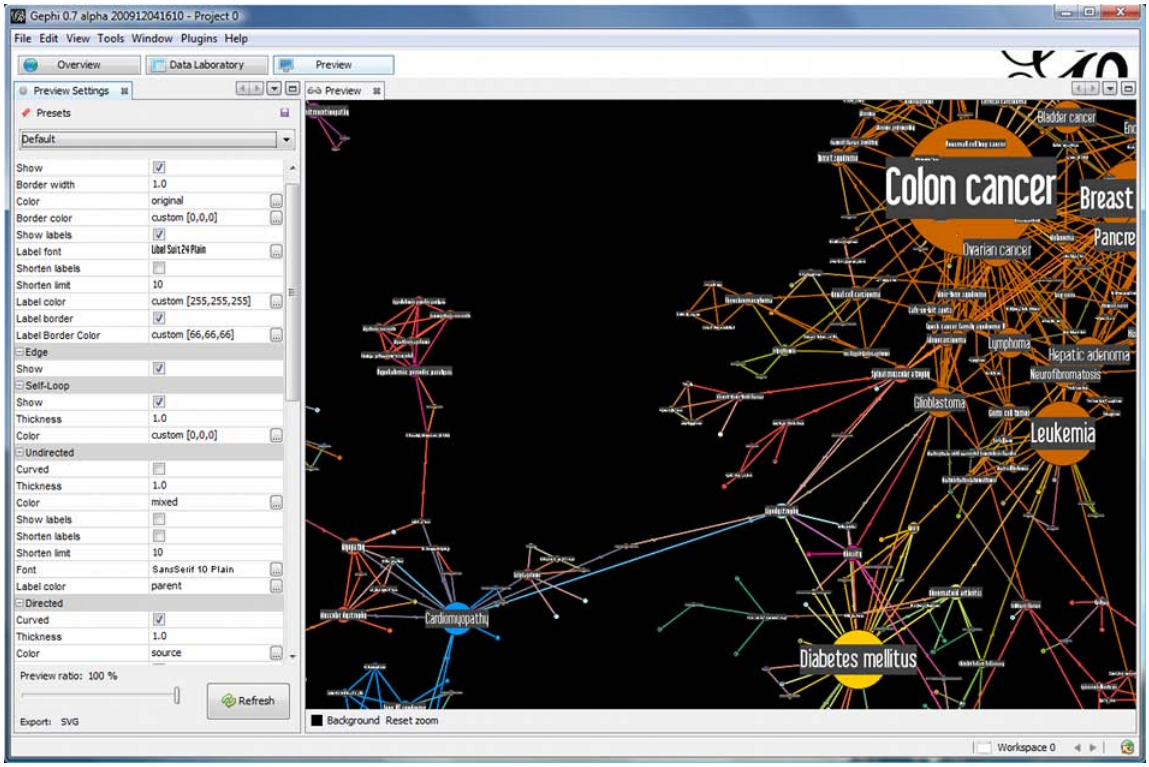


FIG. 3.5: Plateforme Gephi. L'interface de cette plateforme est composée de trois onglets que l'on peut distinguer sur le haut de l'image. Il y a un onglet "Data Laboratory" qui permet à l'utilisateur de manipuler les données. L'onglet "Preview" permet à l'utilisateur de paramétrer la représentation, il peut y modifier la couleur des sommets et des arêtes, leurs tailles ... Le dernier onglet est l'onglet "Overview". C'est dans ce dernier que l'utilisateur pourra pratiquer une exploration du graphe ou du réseau.

Description : Gephi [6] est une plateforme de visualisation et d'exploration de réseaux et graphes complexes et/ou hiérarchiques ainsi que des graphes dynamiques. Elle reprend une grande partie des algorithmes d'analyse, de fragmentation et de dessins présents dans le domaine. Son principal objectif est de permettre à l'utilisateur de générer des images

d'une grande qualité. Pour cela, l'accent est mis sur le rendu final et l'aspect global de l'image. Le point novateur de la plateforme est son approche entièrement tournée vers la carte graphique de la machine ne surchargeant pas le processeur. Cela permet une certaine fluidité dans le traitement des interactions et des dessins. En ce qui concerne les données, Gephi peut importer la plupart des graphes générés par d'autres systèmes de visualisation. Il propose aussi un module de manipulation de données, prenant en entrée un fichier au format CVS (Comma-Separated Values, table). Ces tables doivent représenter les graphes sous certaines formes : une liste d'arêtes, des listes d'adjacence ou une matrice d'adjacence. Au sein du module de manipulation de données, il est possible de modifier les données d'un format vers un autre, de donner un poids aux arêtes.

Les points forts de la plateforme Gephi sont :

- une approche tournée vers le calcul à l'aide du processeur graphique
- l'import des formats les plus utilisés dans le domaine ainsi que le format CVS
- l'analyse de graphes dynamiques

Comparaison/Positionnement : C'est parce qu'il fait partie des logiciels les plus récents que nous avons choisi le logiciel Gephi pour illustrer la parade qu'utilise les systèmes concernant la construction automatique de visualisations. Gephi, présente une technologie basée sur le calcul à l'aide de carte graphique pour améliorer les qualités de rendu. C'est en se basant sur cet argument qu'il propose de se substituer aux autres systèmes pour la partie dessin et visualisation du processus de visualisation. Comme beaucoup de systèmes, il délaisse le problème de l'import de données et de la construction de visualisation et laisse à chaque système la responsabilité de produire des visualisations et de les enregistrer dans leur propre format. Ainsi, ces systèmes se renvoient mutuellement le problème en espérant qu'un jour peut-être, un d'entre eux se dotera d'un mécanisme de production automatique et débloquera la situation pour tous les autres. En attendant, l'ensemble de ces logiciels s'affronte dans une course à la performance.

3.2.5 Tulip [3]

Description : Tulip [3] est une plateforme de visualisation et d'analyse d'informations. Elle permet aux utilisateurs de suivre le processus exploratoire définit par une hypothèse, puis une expérimentation et finalement une découverte. Pour cela, elle intègre un grand nombre de visualisation différentes : nuage de points, coordonnées parallèles, carte auto-organisatrice (Self Organized Map), graphe nœud-lien, histogramme, carte de chaleur (Heat Map) ... Pour chacune de ces visualisations l'utilisateur a accès à de nombreux algorithmes d'analyse permettant d'obtenir des mesures sur les données, ainsi que, le cas échéant de nombreux algorithmes de dessins. Cette plateforme repose principalement sur une architecture de type plug-in qui lui permet d'avoir une grande malléabilité. Il est possible de créer une interface dédiée à un type de tâches particulier. Par exemple, lorsqu'un utilisateur travaille avec des données arborescentes, il est alors possible, grâce à la notion de plateforme déclinable, de ne proposer à l'utilisateur uniquement des algorithmes de dessins d'arbre. Il semble aussi pertinent de lui mettre à disposition uniquement les mesures applicables à ce type de graphe. Tulip permet également d'avoir des vues simultanées sur les données, et ces vues peuvent être synchronisées. Par exemple, il est possible d'avoir une vue des données sous la forme d'un graphe de type nœud-lien, ainsi qu'une vue en

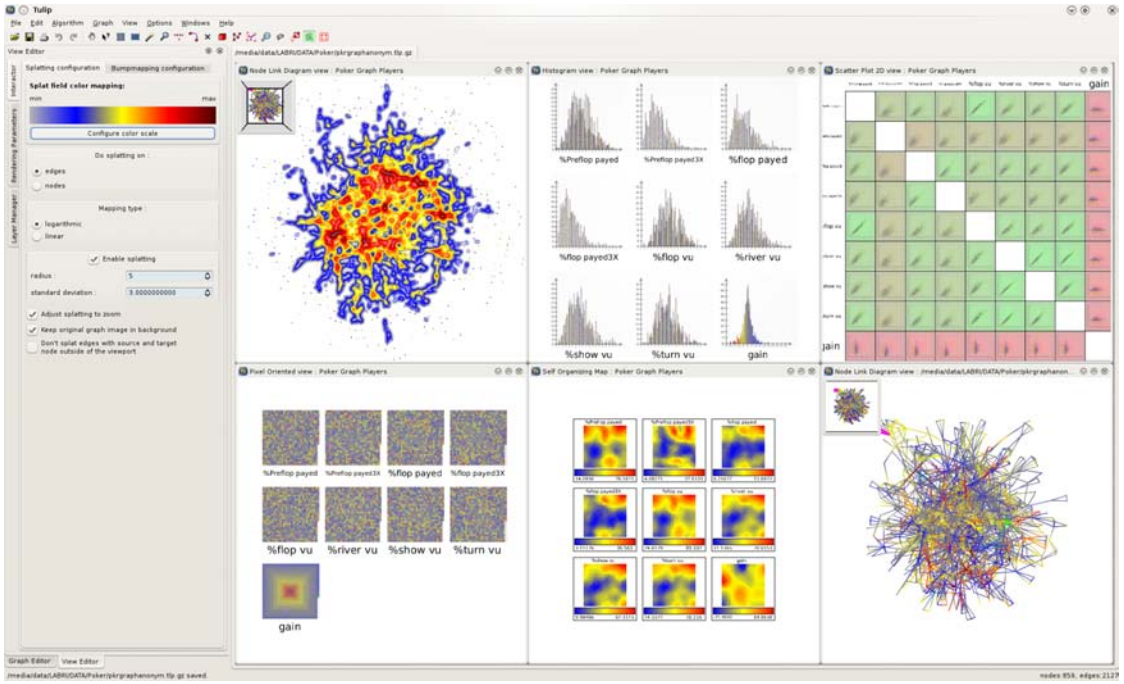


FIG. 3.6: Plateforme Tulip. L'interface est composée d'un panneau de contrôle/configuration sur la gauche et d'un panneau de visualisation sur la droite. On peut voir qu'il est possible d'obtenir des vues simultanées d'une même jeu de données. Il y a ici six représentation différentes. On peut également voir que les représentations communiquent entre elles. En effet, dans la vue inférieure gauche, une coloration a été appliquée au sommet d'après la composante "gain". Cette coloration des sommets est répercutée dans chacune des autres composantes de cette vue mais également dans chacune des autres représentations.

carte auto-organisatrice. Lorsque l'utilisateur sélectionnera un nœud du graphe, l'élément correspondant sera mis en surbrillance dans l'autre vue.

Tulip possède un système performant de gestion des objets qui lui permet de travailler avec plusieurs millions de nœuds et arêtes sans pour autant avoir de ralentissement lors de la phase exploratoire de l'expérimentation menée par un utilisateur. Tulip permet d'autre part d'importer la plupart des graphes générés par d'autres systèmes de visualisation, il est également possible de manipuler les données au format CSV. Pour cela, l'utilisateur doit créer son propre plug-in qui lui permettra de d'importer des données de manière à construire une des représentations.

Les points fort de la plateforme Tulip sont :

- un regroupement d'une grande variété de type de visualisation : nuage de points, coordonnées parallèles, Self Organized Map ...
- un système de plug-in qui permet de l'étendre et d'y développer ses propres méthodes
- une modularité de la plateforme qui permet de la rendre spécifique pour un certain type de tâche[1][2]
- la possibilité d'avoir des vues différentes simultanées sur les données

[1] http://nossi.gforge.inria.fr/
[2] http://tulip.labri.fr/TulipDrupal/?q=systrip

Comparaison/Positionnement : Comparé à Many Eyes, Tableau Software ou nodeXL, Tulip ne propose pas d'outils permettant de générer des graphes à partir d'un fichier sous forme de table, sans avoir à écrire un algorithme. À l'heure actuelle, on retrouve dans la plateforme Tulip la notion de prototype associé au type de visualisation présente dans Many Eyes. Pour construire une représentation, l'algorithme de construction doit répondre à un certain nombre de contraintes sur des paramètres propres à une représentation. On retrouve sur la plate forme Tulip sensiblement le même nombre de type de visualisations que dans la plateforme Many Eyes. Mais l'aspect modulaire du Tulip lui permet d'intégrer facilement les nouveautés et un utilisateur peut même y développer de nouvelles visualisations. C'est un aspect qu'on ne retrouve dans aucun des autres logiciels/plateformes cités précédemment. Tulip est une plateforme qui contrairement à Gephi propose les outils nécessaires pour mener une analyse de bout en bout. Elle ne se concentre pas uniquement sur le rendu, l'exploration et les performances. De plus, grâce à son système de gestion des objets, le niveau de rendu et de performances atteint par Tulip est supérieur à celui proposé par Gephi.

Le fait de pouvoir réutiliser des modules de l'interface et des modules de calculs nous rend la tâche plus aisée pour le développement du logiciel DySNAV présenté dans le Chapitre 4. Quant à la possibilité d'y créer nos propres algorithmes d'import de données, de dessins ou d'interactions avec les visualisations, celle-ci est particulièrement intéressante pour la mise au point de méthodes de génération de graphes présentées dans les Chapitres 5 et 6.

Chapitre 4

Analyse de réseaux sociaux dynamique

4.1 Motivation

Un réseau social est un ensemble de personnes reliées entre elles par un ensemble de relations sociales [74, 94] telles que l'amitié [70] ou des collaborations professionnelles [67, 95]. D'un point de vue formel, ces réseaux peuvent être modélisés à l'aide de graphes dans lesquels les nœuds représentent les personnes et les arêtes les relations. Les travaux de recherches antérieurement menés sur l'analyse de réseaux sociaux [94] ont montrés que le fait de connaitre la structure des communautés ainsi que la force des relations au sein de ces réseaux ont d'importantes applications dans le domaine de l'analyse d'Internet [21], lors d'analyse de marchés [26], dans la sécurité nationale [92, 96] ou la modélisation de maladie ou d'épidémie [29, 53].

L'analyse visuelle de réseaux sociaux fait partie intégrante du domaine de l'analyse visuelle [31]. Visualiser les structures communautaires présentes dans des réseaux sociaux et identifier les personnes qui jouent un rôle important à l'intérieur de ces réseaux, permet de mettre en évidence des informations importantes. Il est possible de donner encore plus de poids à ces informations si on utilise l'évolution temporelle des relations. En effet, il est possible de prendre en compte des modifications au cours du temps de plusieurs manières. On peut observer la fréquence d'une relation, sa durée totale ou encore sa durée moyenne. On peut aussi s'intéresser aux apparitions simultanées de relations ainsi qu'à des disparitions simultanées. Par exemple, une personne peut changer régulièrement d'entreprise parce qu'elle travaille en tant qu'intérimaire. De ce fait, cette personne va être amenée à lier régulièrement de nouvelles relations avec des personnes puis finira par en rompre certaines lors de la fin de son contrat. De plus, les relations peuvent représenter des événements qui ont une importance toute particulière à un instant précis. Par exemple, on peut mettre en relation toutes les personnes qui se sont rendues à une projection en avant-première au cinéma. Puis en observant successivement les différentes fréquentations de telles projections, une communauté de cinéphiles pourra peut-être être mise en évidence. Il s'agit bien du fait d'observer le réseau à des instants précis qui nous permet de détecter une telle communauté.

Plus récemment, l'analyse de réseaux sociaux a fait l'objet d'études et d'applications dans le cadre du contre-terrorisme [1, 61, 64, 96]. Étudier les réseaux sociaux de terroristes potentiels peut nous permettre de révéler la structure organisationnelle de ces réseaux,

prévoir des actes terroristes en identifiant une activité anormale et peut-être divulguer les identités des têtes pensantes se cachant derrière des activités criminelles.

Le problème initial qui a motivé ces travaux de recherche est l'analyse de données correspondant à des appels téléphoniques de téléphone portable à téléphone portable (pour de plus ample détails sur ces données, veuillez vous référer à la Section 4.2). Le but était d'analyser les changements qui ont lieu au sein de ce réseau social au cours du temps et d'arriver à en extraire une hiérarchie sous-jacente.

Parmi les autres types de réseaux sociaux que l'on peut étudier, on peut citer les réseaux basés sur les emails [24] utilisant les horaires d'envoi, les réseaux de co-auteurs dans les publications scientifiques [67] utilisant les années de publication ainsi que les réseaux basés sur les collaborations entre acteurs dans des films [5] utilisant les années de sorties de films. On constate alors que dans tous ces exemples de réseaux sociaux, il y a un aspect temporel fort et que celui-ci doit être utilisé pour analyser et comprendre ces réseaux.

Dans ce chapitre, nous présentons le système DySNAV qui est l'acronyme de **Dy**namic **S**ocial **N**etwork **A**nalysis and **V**isualization. Ce système a été conçu pour aider les utilisateurs dans leurs tâches d'analyse des dynamiques liées aux structures communautaires dans les réseaux sociaux. Les gens forment des structures communautaires en communiquant ou collaborant plus fréquemment avec certaines personnes que d'autres dans un réseau. Ces communautés changent au cours du temps en fonctions des personnes présentes, de leur relation ainsi que de leur rôle au sein du réseau. Nous essayons d'identifier ces changements en nous concentrant sur deux aspects :

- les communautés et leurs changements les unes par rapport aux autres au travers de visualisations
- la détection d'événements importants en observant des changements radicaux dans la structure du réseau.

Nous extrayons aussi une hiérarchie en identifiant les personnes les plus influentes du réseau.

Ce chapitre est organisé de la manière suivante : dans la Section 4.2, nous présentons différents jeux de données que nous avons utilisé pour l'élaboration du système. Dans la Section 4.3 nous présentons le système au travers des quatre étapes qui le compose :

1. la discrétisation des données
2. la décomposition dans laquelle les structures communautaires sont identifiées
3. la détection des changements dans ces structures, au moyen de visualisations
4. la détermination d'une hiérarchie d'influence présente dans le réseau.

Enfin, nous présentons dans la Section 4.4 l'analyse de deux réseaux sociaux faite à l'aide de notre système.

4.2 Jeux de données

Nous utilisons deux jeux de données pour l'évaluation empirique de notre système.

Le jeu de données Catalano/Vidro est un jeu de données fictif rendu public à l'occasion du IEEE VAST 2008 CHALLENGE [1] [73] qui portait sur la visualisation et extraction

[1] http://www.cs.umd.edu/hcil/VASTchallenge08/

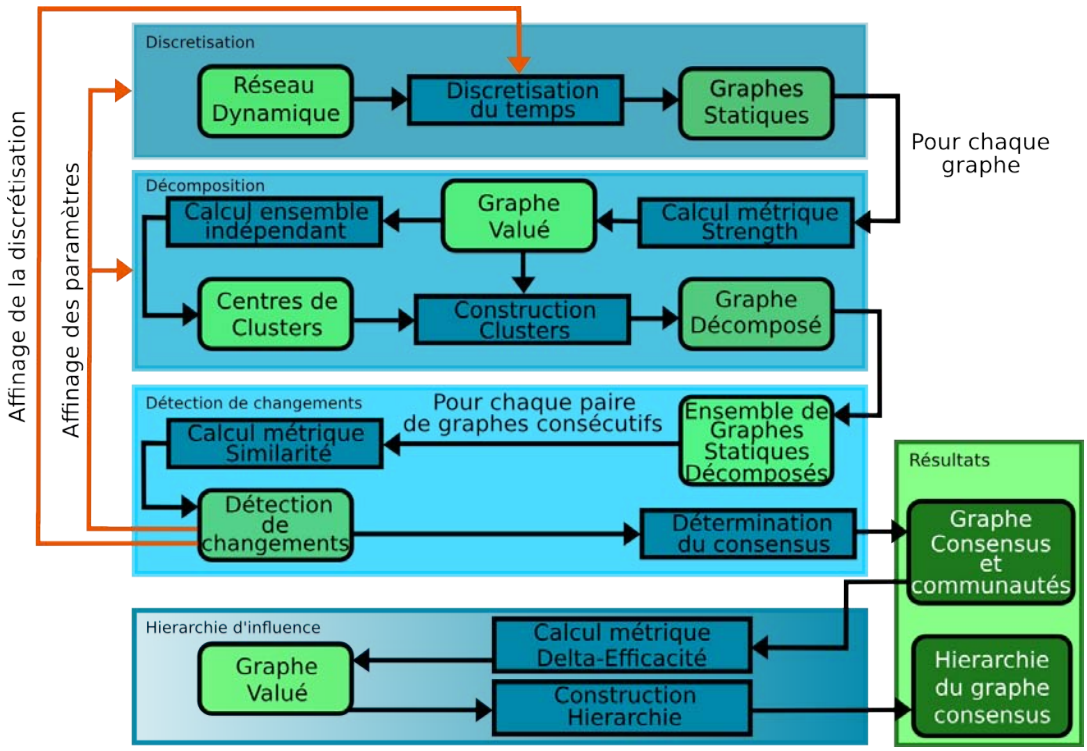


FIG. 4.1: Structure du système proposé. On peut voir les quatre étapes majeures qui le composent ainsi que sur la partie droite les informations extraites à l'aide du système : le graphe consensus et ses communautés ainsi que la hiérarchie d'influence sous-jacente à ce graphe.

d'informations sur un groupe terroriste au sein d'un réseau social. Ce jeu de données est constitué de 9834 appels téléphoniques entre 400 numéros de téléphones portables sur un pédiode de 10 jours en Juin 2006 sur l'ile Isla Del Sueno. Les données sont stockées sous la forme d'un 5-tuple :

- from_user_id, numéro de téléphone émettant l'appel
- to_user_id, numéro de téléphone recevant l'appel
- timestamp, horodatage du début de l'appel
- call_duration, durée de l'appel
- cell_tower_location, position géographique de la tour relais émettrice qui a permis d'assurer l'appel.

Ce jeu de données est un bon exemple illustrant précisément comment peuvent être stockés des informations sur des appels téléphoniques à travers n'importe quel réseau de téléphones portables. De plus, l'utilisation de la dimension temporelle associée aux appels téléphoniques peut nous aider à détecter ou prévoir un événement grâce à une inattendue augmentation de la fréquence des appels, la propagation d'une information importante, l'identité des personnes à l'origine de la propagation etc ...

L'autre exemple est le réseau de co-auteurs. C'est un réseau composé de chercheurs où deux personnes sont reliées l'une à l'autre si elles ont co-signés une production scientifique. L'année de publication de cette production constitue l'information temporelle dans ce jeu de données. Nous avons collecté ces données bibliographiques à partir du site web "DBLP

Computer Science Bibliography" [2] [58]. Elles contiennent les publications jusqu'à l'année 2008. À partir du jeu de données complet, nous avons extrait un sous-ensemble de publications en ne sélectionnant que celles associées au chercheur *Ulrik Brandes*, celles associées aux chercheurs ayant co-signés un travail avec lui, ou celles associées aux chercheurs ayant co-signés un travail avec un chercheur ayant co-signé un travail avec lui. Ce sous-ensemble de publications couvre les années 1997 à 2008 et contient environ 900 chercheurs différents et 6500 paires de co-signatures. Les données sont stockées sous la forme d'un 5-tuple :

- Author1, nom du chercheur ayant co-signés la publication
- Author2, nom du chercheur ayant co-signés la publication
- Year, année de publication
- Strength, notion de force ou de cohésion entre les deux auteurs
- Title_of_Artifact, titre de la publication

Par défaut, le terme "Strength" a une valeur de 1 pour toutes les publications. Mais on peut affiner cette valeur de telle sorte que si une publication est co-signée par exactement deux personnes, alors cette publication aura une forte signification en terme de cohésion, alors qu'une publication ayant un grand nombre de co-auteurs donnera une connotation plus faible à la cohésion entre chacune des paires d'auteurs possibles. Quant au titre de publication, un même titre peut apparaître plusieurs fois car par exemple pour une publication comprenant 3 auteurs, il y aura 3 paires de co-signatures.

4.3 Description du système

Un réseau social dynamique peut être modélisé à l'aide d'un graphe dynamique (cf. Définition 2.21). Pour cela, il suffit de considérer un graphe $G = (V, E)$ pour lequel l'ensemble V décrit les personnes de ce réseau social et l'ensemble E décrit des relations entre ces personnes.

Ce sont de tels graphes que nous utilisons dans ce système. Ils sont le point d'entrée du système et vont être manipulés successivement par les quatre étapes qui composent le système. La Figure 4.1 présente le diagramme illustrant la composition et l'enchainement de ces quatre étapes.

La première étape consiste à transformer le graphe dynamique en un ensemble de graphes statiques où chaque graphe statique correspond à un intervalle de temps présent dans le graphe dynamique. Le facteur de discrétisation qui va définir l'intervalle de temps est ajusté de manière interactive par l'utilisateur.

La seconde étape fragmente chaque graphe statique séparément à l'aide d'un algorithme de fragmentation chevauchante (overlapping clustering), pour produire une fragmentation floue (Fuzzy Clusters). Cette étape nous permet d'identifier des communautés au sein du réseau ainsi que les pivots de ce réseau (les sommets partagés par plusieurs communautés).

La troisième étape a pour but de détecter les changement structuraux majeurs dans le réseau. Pour cela, nous comparons les fragmentations obtenues pour chaque paire de graphes statiques successifs à l'aide d'une mesure de similarité décrite dans la Section 4.3.3. Une similarité faible indique qu'il y a eu de forts changements durant la période qui sépare les deux graphes statiques, alors qu'une forte valeur de similarité correspond à une période

[2] http://www.informatik.uni-trier.de/~ley/db/

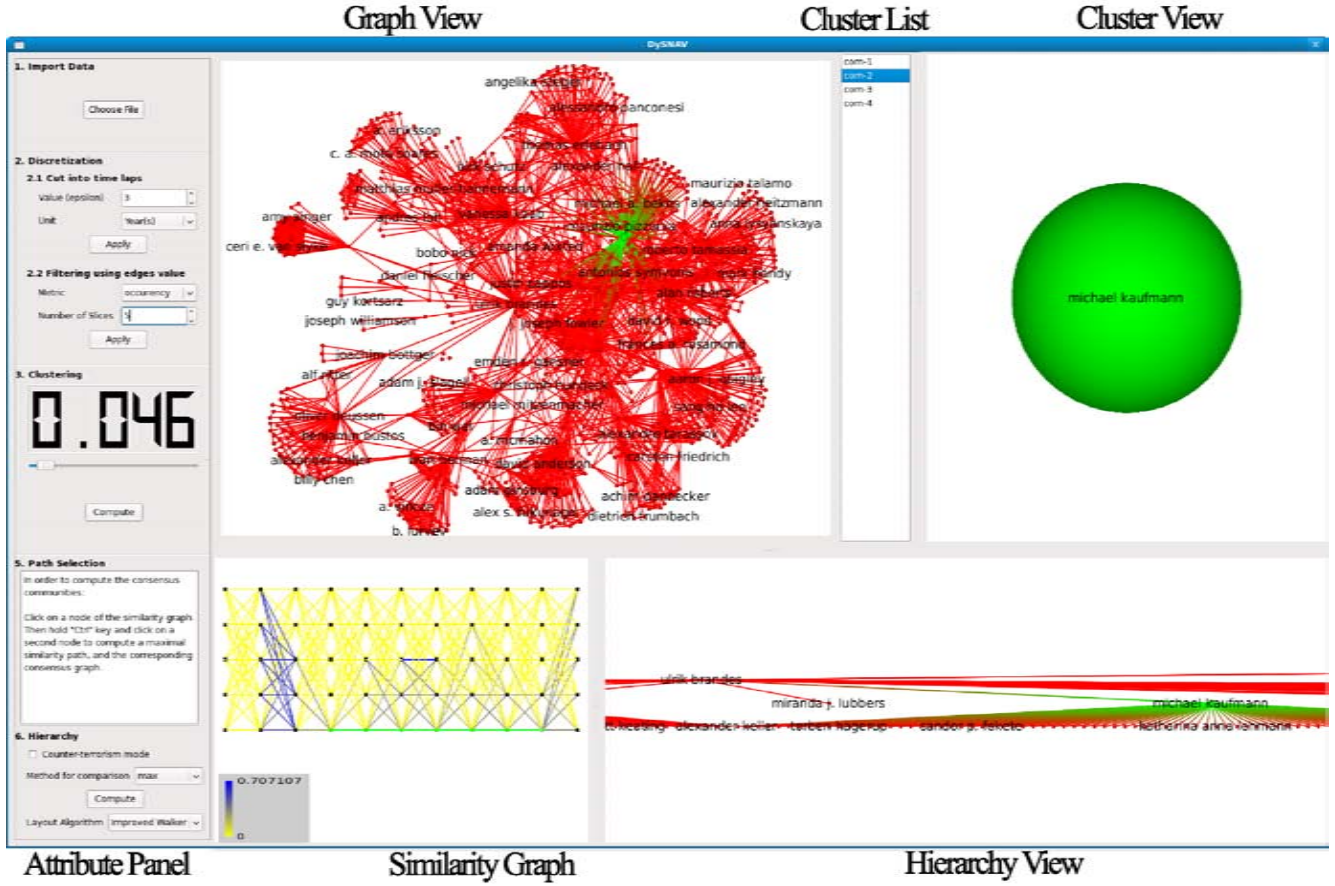


FIG. 4.2: Capture d'écran du système proposé. Il est composé de plusieurs fenêtres qui proposent toutes une visualisation du résultat d'une étape précise du processus. Celle intitulée "Graph View" propose la vue d'un graphe pour un intervalle de temps donné. Seules les arêtes correspondant à cet intervalle sont affichées. Celle intitulée "Similarity Graph" présente le graphe modélisant la découpe du graphe en fonction des intervalles de temps (selon un axe horizontal) et en fonction des valeurs de filtrage (selon un axe vertical). La fenêtre "Cluster List" propose la liste de tous les clusters calculés à partir du graphe affiché dans la fenêtre "Graph View". La fenêtre "Cluster View" propose une vue sur le contenu d'un cluster particulier. Et la fenêtre "Hierarchy View" affiche la hiérarchie d'influence calculée par le système. La partie gauche de l'interface intitulée "Attribute Panel" est un panneau de configuration qui guide l'utilisateur tout au long du processus.

de stabilité durant laquelle il n'y pas eu de changements significatifs de la structure topologique du réseau. De plus, une fois que nous avons la matrice des valeurs de similarité des fragmentations calculés à l'étape précédente, nous pouvons décomposer les changements temporels dans le réseau initial en périodes de forte activité et en communautés de consensus durant les périodes stables.

La dernière étape consiste à trouver une hiérarchie d'influence dans les communautés de consensus que l'on a identifiées lors de l'étape précédente. On définit la hiérarchie d'influence comme un arbre dans lequel la hauteur d'un sommet dans l'arbre représente l'influence qu'a ce sommet dans le réseau (cf. Définition 2.37 et Définition 2.38). Notre technique est basée sur la métrique de Delta Efficiency [62, 63] qui calcule l'importance d'un sommet en fonction du flux global d'informations dans le réseau. En utilisant la métrique de Delta Efficiency et l'algorithme d'arbre couvrant de poids minimum de Kruskal [54], nous sommes capable de produire une hiérarchie d'influence pour ce réseau.

Les données qui sont utilisées en entrée de notre système sont présentées sous la forme d'un fichier texte au format informatique ouvert CSV (Comma-Separated Values). Chaque ligne du fichier est un 5-tuple :

- user_id_1,
- user_id_2,
- timestamp,
- relationship_strength,
- relationship_class.

Les termes "user_id_1" et "user_id_2" sont des chaînes de caractères ou des nombres qui servent à identifier deux personnes dans le réseau social. Le terme "timestamp" décrit avec précision le moment où les deux personnes sont entrées en relation. Ce terme suit le format "yyyy/mm/dd-hr :mn :sc" avec dans l'ordre l'année codée sur 4 chiffres, le mois codé sur 2 chiffres, le jour codé sur 2 chiffres puis l'heure, les minutes et les secondes toutes codées sur 2 chiffres. Toutefois, l'utilisateur n'est pas dans l'obligation de remplir toutes ces informations temporelles. Par exemple, si seuls l'année et le mois sont connus pour un jeu de données particulier, l'utilisateur peut saisir les données dans un format "yyyy/mm" pour chaque tuple.

Le terme "relationship_strength" est un entier qui sert à donner un poids à la relation existant entre les deux personnes. Cette valeur peut être utilisée en tant que métrique pour distinguer les relations fortes des plus faibles. Par exemple, dans le cas d'un réseau construit sur des échanges d'emails, la taille du message peut être utilisée comme un attribut de la relation. Si jamais il n'existe pas de telle valeur à assigner aux relations, il suffit de donner la même valeur à chaque relation (tuple). Le terme "relationship_class" est une valeur nominale qui sert à classer les relations. Toujours dans le cas des échanges d'emails, l'adresse IP de l'expéditeur peut être utilisée pour former des classes de relations. Comme précédemment, il est possible de donner une même valeur pour chaque tuple si une telle information n'existe pas pour un jeu de données. Afin de charger un jeu de données dans le système, l'utilisateur doit sélectionner un fichier dans le format décrit à l'aide du bouton "Choose File" se trouvant dans le panneau de configuration du système décrit dans la Figure 4.2.

Le système est composé de 5 fenêtres servant à afficher les informations et d'un panneau de configuration qui permet de régler les valeurs des différents attributs comme on peut le voir dans la Figure 4.2. Les détails sur les rôles joués par les différents attributs sont décrits dans les sections suivantes. La fenêtre intitulée *Graph View* sert à afficher le réseau

social sous la forme d'un graphe nœud-lien dessiné à l'aide d'un algorithme à modèle de force mis au point par Hachul and Junger [41]. Les algorithmes à modèle de force sont très efficaces lorsqu'il s'agit de visualiser des structures de communautés, car ils ont tendance à regrouper les nœuds qui sont densément connectés entre eux et les nœuds peu connectés sont écartés. La fenêtre intitulée *Similarity Graph* est utilisée pour afficher le graphe de graphes. Ce graphe de graphes est construit de la manière suivante : chaque nœud de ce graphe représente un graphe obtenu après le découpage temporel de l'intervalle de temps (selon un axe horizontal) ainsi qu'un filtrage des arêtes à l'aide de l'attribut *Number of Slices* du panneau de configuration (selon un axe vertical). Cette visualisation est utilisée pour visualiser l'évolution du réseau au cours du temps (la manière dont les changements sont détectés est expliquée dans les sections suivantes).

En cliquant sur un des sommets du graphe de la fenêtre *Similarity Graph*, le graphe correspondant est affiché dans la fenêtre *Graph View*. On peut également voir dans le coin inférieur gauche de cette fenêtre une légende donnant les valeurs minimum et maximum de similarité entre les graphes ainsi qu'un gradient de couleur associé à ces valeurs. La fenêtre intitulée *Cluster List* contient la liste de tous les clusters qui ont été trouvés dans le réseau. En cliquant sur le nom d'un des clusters présent dans cette liste a pour effet d'afficher le contenu de ce cluster dans la fenêtre *Cluster View*. La dernière fenêtre est intitulée *Hierarchy View* et permet d'afficher la hiérarchie d'influence extraite du réseau social. Celle-ci donne une idée de l'influence que peut avoir une personne dans le réseau. Cette hiérarchie est dessinée à l'aide de l'algorithme proposé par Buchheim *et al.* [20]. Il est également possible de dessiner cette hiérarchie avec l'algorithme "Radial Tree" [48]. Le choix de l'algorithme de dessin se fait dans le panneau de configuration. Chaque algorithme a ses avantages et ses inconvénients. Le premier, de part le fait qu'il dessine la hiérarchie de haut en bas, transmet bien la notion de subordination mais lorsqu'un niveau de la hiérarchie compte beaucoup de nœuds, il n'est pas toujours évident de les différencier. C'est un problème qui ne se pose pas avec le second algorithme car celui-ci dispose de plus d'espace pour représenter un même niveau.

Le système propose un grand nombre d'interactions avec les différentes fenêtres. Il est possible de changer la taille de toutes les fenêtres, d'agrandir (zoom-in) et réduire (zoom-out) les dessins des graphes à l'aide de la molette de la souris. Les valeurs des attributs dans le panneau de configuration peuvent être modifiées à tout moment, mais la modification n'est effective (modification du graphe ou de son dessin) qu'une fois que les boutons *compute* ou *apply* sont cliqués. Par exemple, si la valeur de l'attribut *Clustering* est modifiée, le calcul de la fragmentation sera effectué une fois que le bouton de défilement des valeurs sera relâché. Il est à noter aussi que si le $n^{ième}$ attribut est modifié alors les étapes suivantes du processus ne sont plus valables et doivent être ré-exécutées puisque comme on peut le voir sur la Figure 4.1 chacune des trois dernières étapes a besoin des résultats de l'étape précédente. Nous avons également mis en place une manipulation spécifique au calcul des communautés de consensus entre deux graphes d'intervalle de temps différents. Car pour ce calcul, une valeur d'attribut n'est pas suffisante, l'utilisateur doit spécifier les graphes qu'il souhaite prendre comme référence. Pour cela, il doit sélectionner (cliquer) un premier sommet dans le graphe de similarité, puis en sélectionner un second tout en maintenant la touche "Ctrl" du clavier enfoncée. Le chemin passant par les plus fortes valeurs de similarité entre les deux sommets sélectionnés est calculé ainsi que le graphe consensus correspondant.

Nous allons maintenant présenter en détail les quatres étapes du système.

4.3.1 Discretisation du graphe

Une fois que les données sont chargées dans le système, la première étape du système consiste à convertir le graphe dynamique G en un ensemble de graphes. Chacun des graphes de l'ensemble peut être vu comme une photographie du graphe originel sur un certain intervalle de temps. Ainsi, à partir de G on obtient une séquence de graphes $G_{[0,\epsilon]}, \ldots, G_{[T-\epsilon,T]} = G_1, \ldots, G_\alpha$, où α est le nombre de graphes obtenus et ϵ est le facteur de discrétisation. Le graphe $G_{[t,t+\epsilon]}$ correspond alors à une image statique du graphe sur l'intervalle de temps $[t, t+\epsilon]$ (i. e. le graphe contenant tous les sommets et toutes les arêtes apparaissant durant la période $[t, t+\epsilon]$). Le nombre total de graphes α qui seront générés de cette manière est égal à la durée totale de temps $[0, T]$ divisé par le facteur de discrétisation ϵ. Le système permet à l'utilisateur de fixer une valeur pour ϵ qui dépend de la granularité de l'échelle temps présente dans le jeu de données.

Durant cette étape, nous utilisons les informations "relationship_strength" pour calculer une métrique pour chacune des arêtes de chaque graphe. Le système propose trois façons de calculer cette métrique : *Total time*, *Average time* et *Occurrency*. La méthode *Total time* consiste à additionner toutes les valeurs de "relationship_strength" concernant le couple de sommets (u, v). La méthode *Average time* consiste à calculer la moyenne des valeurs de "relationship_strength" concernant le couple de sommets (u, v). Et la méthode *Occurency* consiste à calculer la fréquence avec laquelle le couple de sommets (u, v) apparait. L'utilisation de ces métriques dépend du jeu de données ainsi que de l'interprétation que peut avoir l'utilisateur des valeurs associées à une relation. Afin de choisir le type de métrique que veut obtenir l'utilisateur, celui-ci doit indiquer son choix à l'aide du menu déroulant intitulé *Metric* dans le panneau de configuration.

Le système propose aussi une méthode pour filtrer les arêtes ayant une relation dite faible. C'est à dire une arête qui ne présente que peu d'intérêt pour l'utilisateur ou bien qui ne joue pas un rôle de premier plan dans le réseau. Mais il n'est pas possible de fournir une valeur prédéfinie de seuil, donc plusieurs valeurs sont utilisées pour filtrer les arêtes. Pour cela, il est demandé à l'utilisateur de saisir une valeur pour le champs *Number of slices* (ω) qui doit être un entier positif. Le système utilise alors les valeurs minimum et maximum de métriques et divise l'écart entre ces deux valeurs pour obtenir les ω différentes valeurs qui seront utilisées pour filtrer les arêtes. Ainsi pour chaque graphe de l'ensemble $G_{[0,\epsilon]}, \ldots, G_{[T-\epsilon,T]} = G_1, \ldots, G_\alpha$, on obtient ω graphes. Au final, on obtient en tout $\omega \times \alpha$ graphes qui sont dessinés dans la fenêtre *Similarity Graph* (cf. Figure 4.2). Dans cette fenêtre chaque graphe est représenté par un nœud dont le placement sur l'axe horizontal dépend des différents intervalles de temps ($G_1, \ldots, G_\alpha$ de gauche à droite) et le placement sur l'axe vertical dépend du filtrage des arêtes (de bas en haut). Nous appelons ce graphe le graphe de similarité dans lequel chaque nœud représente un graphe et où les nœud sont placés dans l'espace de manière à former une grille. Nous expliquerons en détails dans une prochaine section pourquoi ce dessin en forme de grille simplifie l'évaluation de la similarité. Il est toutefois possible à ce stade du processus d'interagir avec ce graphe. En effet, cliquer sur un nœud de ce graphe de similarité, aura pour effet d'afficher le contenu du graphe correspondant à ce nœud dans la fenêtre *Graph View* (cf. Figure 4.2).

4.3.2 Décomposition des graphes

Il ne faut pas perdre de vue que notre idée de base est que s'il existe une communauté, alors il est fort probable qu'il existe une structure topologique dans le réseau qui n'évolue

pas ou peu dans le temps. Donc, si on observe deux graphes (statiques) correspondant à des intervalles de temps consécutifs, ils devraient avoir des structures topologiques proches s'il existe une communauté. Nous devons donc extraire les topologies de ces graphes afin de pouvoir plus tard les comparer entre elles. Afin de capturer la topologie d'un graphe, nous utilisons un algorithme de décomposition qui fragmente le graphe en plusieurs composantes (classe, clusters). Nous allons maintenant décrire en détails le processus de décomposition qui est appliqué à chacun des $\omega \times \alpha$ graphes obtenus à l'étape précédente.

4.3.2.1 La métrique Strength

Notre algorithme de décomposition est basé sur la métrique Strength, introduite par Chiricota *et al.* [4, 22]. Cette métrique quantifie la cohésion du voisinage d'une arête donnée et permet aussi de déterminer si cette arête est une arête intra-communautaire ou une arête inter-communautaire. $w_s(e)$ la valeur Strength d'une arête e est définie par :

$$w_s(e) = \frac{\gamma_{3,4}(e)}{\gamma_{max}(e)} \tag{1}$$

où $\gamma_{3,4}(e)$ est le nombre de cycles (cf. Définition 2.36) de taille 3 ou 4 auxquels l'arête e appartient et $\gamma_{max}(e)$ est le nombre maximum de tels cycles qu'il est possible d'obtenir. Nous pouvons définir une valeur Strength pour un sommet u de la manière suivante :

$$w_s(u) = \frac{\sum_{e \in adj(u)} w_s(e)}{deg(u)} \tag{2}$$

où $adj(u)$ est l'ensemble des arêtes adjacentes à u (cf. Définition 2.27) et $deg(u)$ est le degré du sommet u (cf. Définition 2.28). Le complexité en temps de calcul d'une valeur de Strength pour chaque sommet et chaque arête est en $\mathcal{O}(|E| \cdot (deg_{max})^2)$ où deg_{max} est le degré maximum du graphe.

4.3.2.2 Extraction d'un ensemble maximal indépendant

Afin d'identifier les centres des communautés au sein du réseau, nous utilisons une méthode inspirée de *MISF* [34] (Maximal Independent Set Filtering). Pour cela nous extrayons un ensemble maximal ν de sommets tel que $\forall u, v \in \nu, dist_G(u, v) \geq 2$. L'utilisation de cet algorithme donne deux avantages. Premièrement, il donne un nombre de clusters qui respecte la topologie du réseau et deuxièmement, cette technique garantit l'unicité de chaque cluster trouvé puisque un centre ne peut appartenir qu'à un seul cluster à la fois.

Il est intéressant de noter que puisque les sommets dans ν sont les centres des communautés, ces sommets ne peuvent pas être en même temps des pivots du réseau. Dès lors, les sommets pivots du réseau peuvent être identifiés comme ceux ayant une valeur de Strength basse puisqu'ils sont partagés par plusieurs communauté à la fois. De plus, les sommets avec une forte valeur de Strength doivent être ajoutés à l'ensemble ν. Pour extraire un tel ensemble, nous utilisons l'Algorithme 4.1.

Proposition 4.1 *La complexité de l'Algorithme 4.1 est de l'ordre de* $\mathcal{O}(|V| \cdot log(|V|) + |E|)$.

Preuve : La complexité en temps de l'algorithme de tri trieSommetStrength(Graph, vector⟨node⟩) est en $\mathcal{O}(|V| \cdot log(|V|))$, quant à la complexité de la boucle "Pour", elle est en

```
Entrées : Un graphe G = (V, E)
Sorties : Un ensemble maximal ν de sommets à distance au moins 2
vector⟨node⟩ sommets_triés ;
trieSommetStrength(G, sommets_triés) ;
pour i allant de 0 à (nombre de sommets dans G) faire
    sommet u = sommets_triés[i] ;
    si u est dans G alors
        ajouter u à ν;
        pour tous les sommets v dans le voisinage de u faire
            retirer v de G;
        fin
        retirer u de G;
    fin
fin
```

Algorithme 4.1 : Calcul de l'ensemble ν. La méthode trieSommetStrength(G, *sommets_triés*) trie les sommets par valeur de Strength décroissante et stocke le résultat dans *sommets_triés*.

$\mathcal{O}(|V| + |E|)$. Mais l'ensemble V utilisé dans la boucle "Pour" est l'ensemble trié donc au final on a une complexité en temps pour l'Algorithme 4.1 qui est en $\mathcal{O}(|V| \cdot log(|V|) + |E|)$. □

4.3.2.3 Extraction de communautés

Nous utilisons l'ensemble ν de sommets de forte valeur Strength pour extraire les communautés à partir du réseau initial. L'idée que nous avons utilisée ici est de construire des "boules" de rayon 1 autour de chaque sommet de ν. Pour chaque sommet $u \in \nu$, si une arête (u, v) dans le graphe a une valeur Strength supérieure à un seuil donné τ, alors cette arête est considérée comme une arête intra-communautaire et le sommet v est ajouté à la communauté centrée sur u. Plusieurs valeurs de seuils sont utilisées, $\tau_1, \ldots, \tau_m$, de manière à obtenir m fragmentations différentes pour chaque intervalle de temps. Les valeurs de seuils $\tau_1, \ldots, \tau_m$ sont calculées en fonction du nombre de sommets et d'arêtes dans le réseau. La complexité en temps d'extraction des communautés est clairement en $\mathcal{O}(|E|)$, ce qui donne qu'au final la complexité de notre algorithme de décomposition est en $\mathcal{O}(|E| \cdot deg^2_{max} + |V| \cdot log(|V|))$.

A la fin d'étape de décomposition des graphes, nous obtenons m fragmentations pour chaque graphe du graphe de similarité. L'utilisateur peut alors choisir une valeur de τ dans le panneau de configuration à l'aide d'un bouton de défilement.

4.3.3 Détection des changement

Nous allons désigner par C l'ensemble des fragmentations effectuées précédemment, et par $C_{i,j}$ la décomposition du graphe G_i selon la valeur τ_j. Étant donné que les graphes G_i sont ordonnés en fonction du temps, l'évolution la plus logique d'un cluster peut être mise en évidence en comparant chaque graphe $C_{i,j}$ avec chaque graphe $C_{i+1,k}$ $\forall i, j, k$ tel que $1 \leq i < n$, $1 \leq j, k \leq m$. Nous décrivons dans la section suivante une métrique de similarité permettant d'évaluer les similarité entre chaque paire de fragmentations de C.

4.3.3.1 Métrique de similarité

La métrique de similarité doit évaluer la similarité entre deux fragmentations effectuées sur des graphes assez proches. Cette métrique se rapproche de celle utilisée dans les fragmentations de réseau d'interaction de type protéine-protéine [19]. Elle est basée sur le concept de représentativité. On dit qu'un cluster $c_a \in C_{i,j}$ est un bon représentant de $c_b \in C_{i+1,k}$ si et seulement si c_a contient un grand nombre des éléments de c_b et très peu d'éléments n'appartenant pas à c_b.

Définition 4.1 (Représentativité de clusters orientée)
La représentativité de clusters orientée est donnée par :

$$\rho_{c_a \to c_b} = c_a \cap c_b \ / \ |c_b| \qquad \rho_{c_b \to c_a} = c_a \cap c_b \ / \ |c_a|. \tag{3}$$

Elle correspond à la proportion normalisée d'éléments commun à deux clusters.

Nous définissons également la représentativité de clusters non-orientée, ou plus simplement la représentativité de clusters.

Définition 4.2 (Représentativité de clusters non-orientée)
La représentativité de clusters est donnée par :

$$\rho_{c_a, c_b} = \sqrt{\rho_{c_a \to c_b} \cdot \rho_{c_b \to c_a}}. \tag{4}$$

Elle correspond à la moyenne géométrique de la représentativité orientée de chacun des clusters par rapport à l'autre.

Maintenant, nous allons étendre la définition de représentativité de cluster à des groupes de clusters ou même à des fragmentations. On dit que la fragmentation $C_{i,j}$ est une bonne représentante de la fragmentation $C_{i+1,k}$ si la première contient un cluster suffisamment représentatif pour chaque cluster dans la seconde. De plus, comme les clusters de petite taille ont tendance à fausser les valeurs de représentativité, nous donnons plus d'importance aux clusters représentatifs de clusters de grande taille. On définit alors la représentativité orientée d'une fragmentation comme le barycentre (moyenne pondérée sur la cardinalité de la fragmentation) des valeurs de chaque cluster le plus représentatif se trouvant dans $C_{i,j}$ pour chaque cluster dans $C_{i+1,k}$:

Définition 4.3 (Représentativité orientée d'une fragmentation)
La représentativité orientée d'une fragmentation est définie comme le barycentre (moyenne pondérée sur la cardinalité de la fragmentation) des valeurs de chaque cluster le plus représentatif se trouvant dans $C_{i,j}$ pour chaque cluster dans $C_{i+1,k}$. Elle est donnée par :

$$\sigma_{C_{i,j} \to C_{i+1,k}} = \frac{\sum_{c_b \in C_{i+1,k}} \max_{c_a \in C_{i,j}} \rho_{c_a, c_b} \cdot |c_b|}{\sum_{c_b \in C_{i+1,k}} |c_b|}. \tag{5}$$

Comme précédemment, on définit la représentativité non-orientée d'une fragmentation.

Définition 4.4 (Représentativité non-orientée d'une fragmentation)
La représentativité non-orientée d'une fragmentation ou métrique de similarité par :

$$\sigma_{C_{i,j}, C_{i+1,k}} = \sqrt{\sigma_{C_{i,j} \to C_{i+1,k}} \cdot \sigma_{C_{i+1,k} \to C_{i,j}}}. \tag{6}$$

Par la suite, nous utiliserons le terme de métrique de similarité pour désigner la représentativité non-orienté d'une fragmentation.

4.3.3.2 Visualisation des fragmentations

Sous l'hypothèse que l'évolution des clusters a besoin d'une certaine inertie avant d'opérer des changements drastiques (c'est à dire qu'un cluster ne change pas sévèrement à chaque intervalle de temps), la similarité entre les différentes fragmentations permet d'identifier une meilleure valeur pour le paramètre τ. À l'heure actuelle, nous sommes dans l'incapacité de choisir la valeur optimale τ_j qui nous fournira la meilleure fragmentation pour un graphe G_i. Néanmoins, en guise d'heuristique pour estimer une bonne valeur de τ, nous détectons une séquence de fragmentations $C_{i,j}, C_{i+1,k}, C_{i+2,l} \dots$ dont chacun d'entre eux a une valeur de similarité supérieure à la moyenne des valeurs de similarité.

Pour étudier le comportement entre deux intervalles de temps consécutifs, nous calculons la valeur maximale et la valeur moyenne de similarité de ces deux fragmentations consécutives à partir de la formule de la Définition 4.4 en faisant varier j et k indépendamment entre 1 et m. Si la différence entre la valeur maximale et la valeur moyenne est grande, cela signifie qu'il y a des changements radicaux dans le réseau entre ces deux intervalles de temps. Alors que si la différence est petite, nous pouvons en déduire qu'aucun changement significatif a eu lieu dans le réseau entre ces deux intervalles.

Pour faciliter cette analyse, nous utilisons une représentation visuelle des valeurs de la métrique de similarité entre les fragmentations. Comme on peut le voir dans la Figure 4.2, pour chaque paire de fragmentations $(C_{i,j}, C_{i+1,k})$, nous ajoutons une arête entre les deux sommets correspondants aux fragmentations comparées à l'aide de la métrique de similarité. Ces arêtes sont valuées avec la valeur σ de la métrique de similarité et graphiquement les arêtes sont épaissies et coloriées avec une couleur variant en fonction de la valeur de la métrique.

Pour visualiser chaque cluster, l'utilisateur peut sélectionner une sommet dans la fenêtre *Similarity Graph*. Tous les clusters présents dans ce graphe sont alors listés dans la fenêtre *Cluster List*, et l'utilisateur peut alors explorer individuellement chacun de ces clusters et en visualiser le contenu en en sélectionnant un de la liste. Le cluster correspondant est alors affiché dans la fenêtre *Cluster View* comme on peut le voir dans la Figure 4.2.

4.3.3.3 Extraction d'une communauté de consensus

Avec le temps, les communautés peuvent s'étendre et incorporer des nouveaux sommets, fusionner avec d'autres communautés, décroitre en perdant des sommets ou encore se couper en plusieurs sous-groupes. Cela représente un problème lorsqu'on veut identifier une communauté. Pour résoudre ce problème, et obtenir une idée globale de la composition d'une communauté, nous calculons les communautés de consensus dans le réseau initial. À chaque intervalle de temps, chaque communauté est représentée par un cluster, donc nous pouvons suivre l'évolution des communautés en faisant coïncider les bons clusters entre des intervalles de temps consécutifs.

Soient $\overline{C}_x, \overline{C}_{x+1}, \overline{C}_{x+2} \dots$ les fragmentations $C_{x,j}, C_{x+1,k}, C_{x+2,l} \dots$ le long d'un chemin de similarité dans le graphe de similarité. Nous connaissons la valeur de similarité σ

pour chaque paire de fragmentations consécutives $\overline{C}_i, \overline{C}_{i+1}$ et de ce fait nous connaissons aussi la représentativité de la fragmentation entre chaque cluster $c_a \in \overline{C}_i$ et $c_b \in \overline{C}_{i+1}$. Nous allons donc utiliser ces valeurs pour faire coïncider les clusters et identifier les clusters c_b qui sont représentatifs du cluster c_a.

Le calcul de σ entre deux fragmentations $C_{i,j}, C_{i+1,k}$ est borné par le calcul de l'intersection entre chaque paire de clusters $c_a \in C_{i,j}$, $c_b \in C_{i+1,k}$. Cette étape requiert au moins $Q^2|V|$ comparaisons où Q est la cardinalité maximale de $C_{i,j}$ et V le nombre de sommets dans le réseau. Étant donné que chaque fragmentation $C_{i,j}$ est comparée avec toutes les fragmentations de l'intervalle de temps suivant, nous calculons $(\alpha-1)\omega^2$ valeurs de similarité. En général, α et ω ne sont pas très grands, donc la complexité totale est acceptable pour en faire une utilisation interactive. Quant au calcul des communautés de consensus, celui-ci dépend de l'algorithme de filtrage choisi par l'utilisateur. Mais le plus souvent ce calcul est borné par le calcul du graphe de similarité.

4.3.4 Hiérarchie d'influence

On définit la hiérarchie d'influence (cf. Définition 2.38) comme un arbre $G_T = (V_T, E_T)$ où $V_T \subset V$ est un sous-ensemble des sommets du réseau social. La hauteur d'un sommet $v \in V_T$ dans l'arbre représente l'influence que le sommet a sur le reste du réseau. Notre technique est basée sur la métrique de Delta Efficiency [63, 62] qui calcule l'importance qu'a un sommet au vue du flux d'information qui circule globalement dans le réseau. Pour quantifier l'efficacité avec laquelle les sommets dans le réseau échangent de l'information, nous utilisons l'idée de V. Latora *et al.* [57]. Nous savons que tous les sommets échangent des informations dans un réseau représenté par un graphe $G = (V, E)$, et cette information peut-être collectée par d'autres sommets si besoin. Dans nos jeux de données expérimentaux, chaque appel téléphonique ou chaque production co-signée représente de tels échanges d'information. L'efficacité de communication ε_{ij} du réseau entre les sommets i et j est inversement proportionnelle à la longueur du plus court chemin dans le graphe entre i et j :

$$\forall i, j \in V \ \varepsilon_{ij} = 1/d_{ij} \tag{7}$$

où d_{ij} est la longueur du plus court chemin entre i et j. Si jamais il n'y a pas de chemin entre i et j, alors $d_{ij} = +\infty$ et $\varepsilon_{ij} = 0$. Nous pouvons alors quantifier l'efficacité de communication globale du réseau en calculant ε_{ij} pour chaque paire de sommets. Ces efficacité sont normalisées par le nombre maximum d'arêtes présentent dans un graphe comptant le même nombre de sommets (cf. Définition 2.40), car dans un tel graphe les efficacités de communication seront les meilleures possibles compte tenu du fait que tous les sommets sont à distance 1 les uns des autres. L'efficacité moyenne du graphe G est alors définie par :

$$Eff(G) = \sum_{i \neq j \in V} \varepsilon_{ij} \ / \ |V| \cdot (|V| - 1) \tag{8}$$

Cette métrique nous donne une idée de l'efficacité de la communication dans le réseau. Afin de trouver une hiérarchie dans le réseau, nous avons besoin d'évaluer l'efficacité ou la perturbation qu'engendre chaque sommet comme cela est proposé dans [57]. L'idée est que si un membre important du réseau est retiré, l'efficacité de communication du graphe devrait décroitre. Nous définissons la Delta Efficiency(DE) d'un sommet comme :

$$I(node_i) = \Delta Eff_i = Eff(G) - Eff(G \backslash \{i\}). \tag{9}$$

Une fois que nous avons la Delta Efficiency de chaque sommet, nous pouvons l'utiliser pour assigner des poids aux arêtes du graphe. Il existe plusieurs façons d'assigner un poids à une arête si les sommets du graphes sont valués. Une manière consiste à utiliser la moyenne des valeurs des deux sommets connectés par une arête. Mais ici nous nous contentons d'utiliser la maximum des valeurs des deux sommets connectés à une arête. On obtient ainsi :

$$Weight(e_{ij}) = Max\{\Delta Eff_i, \Delta Eff_j\}. \tag{10}$$

Le terme e_{ij} représente une arête entre les sommets i et j et les termes ΔEff_i et ΔEff_j sont respectivement leur valeur de Delta Efficiency. Maintenant, à partir de ce graphe valué, nous utilisons l'algorithme d'arbre couvrant de poids minimum de Kruskal [54] pour générer un arbre. Cet arbre dévoile alors la hiérarchie d'influence du réseau en ne sélectionnant que les arêtes ayant la plus grande efficacité de communication dans le réseau.

Dans le cadre du problème posé sur le jeu de données Catalano/Vidro [17], nous devions être capable d'identifier dans le réseau des personnes ayant des rôles spécifiques : le chef, ses bras droits ainsi que son frère. Comparé à un réseau social classique, dans un réseau terroriste les meneurs essaient de se cacher dans le réseau et n'ont donc pas de grandes valeurs de Delta Efficiency. Habituellement, le meneur est seulement en contact avec quelques personnes qui sont responsables de la diffusion des informations dans le réseau. Dès lors, calculer une hiérarchie pour trouver le meneur dans un tel réseau demande quelques modifications du processus qui peuvent être activées en cochant dans le panneau de configuration le mode anti-terrorisme. Les détails sur la manière de calculer une hiérarchie dans un tel cas sont donnés dans l'étude de cas de la Section 4.4.

4.4 Cas d'étude

4.4.1 Réseau de co-auteurs

Nous rappelons (cf. Section 4.2) que le réseau de co-auteurs a été construit en prenant *Ulrik Brandes* comme point de départ puis en collectant tous les chercheurs qui lui sont connectés à distance 2. Le graphe de similarité présenté dans la Figure 4.3 montre les zones ayant de l'importance afin de déduire d'importantes informations sur le réseau. Dans ce graphe, les arêtes de couleur bleue représentent une similarité parfaite entre les deux sommets qu'elles relient. Les jaunes représentent une similarité faible. La zone étiquetée (a) montre (grâce à la couleur bleue des arêtes) une forte similarité entre les deux périodes définies par les années de publications 1979-81 et 1982-84. Le phénomène qu'il est important de noter est celui représenté par la zone (b). En effet, on peut constater que la similarité sur l'ensemble des périodes entre 1988 et 1999 possède une valeur forte relativement constante (la notion de constance de la valeur de similarité, vient des teintes bleues très proches). Il se trouve que c'est durant cette période qu'*Ulrik Brandes* effectue sa première publication (1997) et qu'il termine sa Thèse de doctorat (1999). Étant donné que le graphe a été construit de manière à ne contenir que des personnes étroitement liées à lui, nous pouvons faire l'hypothèse que ce sont des personnes travaillant dans le même domaine que lui à savoir : le dessin de graphe et la visualisation d'informations. Il est même probable que beaucoup d'entre eux appartiennent à la même université que celle où il a réalisé son doctorat. Ce sont des hypothèse qui pourraient justifier la similarité obtenue entre les graphes de cette période.

```
Entrées : Un graphe G = (V, E), Delta Efficiency n.Δ de chaque sommet n, une matrice
          de listes de sommets M et un arbre couvrant T
Sorties : Le sommet racine boss de la hiérarchie et une orientation de l'arbre couvrant T
int nombre_de_sommets = 3 / 100 x(nombre de sommets dans G) ;
vector⟨node⟩ sommets_triés ;
trieSommetDelta(G, sommets_triés) ;
pour i allant de 1 à number_of_nodes faire
    vector⟨node⟩ voisinage = donneVoisinage(sommets_triés[i]) ;
    pour j allant de 1 à voisinage.taille faire
        M[i][j] = voisinage[j];
    fin
fin
list résultat;
pour i allant de 1 à nombre_de_sommets − 1 faire
    pour j allant de (i + 1) à nombre_de_sommets faire
        pour k allant de 1 à M[i].taille faire
            pour l allant de 1 à M[j].taille faire
                si ( M[i][k] == M[j][l]) alors
                    result.push(M[i][k])
                fin
                ;
            fin
        fin
    fin
fin
vector⟨node⟩ sommets_résultat_triés ;
trieRésultatNœud(résultat, sommets_résultat_triés) ;
node boss = sommetApparaitMaxFois(sommets_résultat_triés) ;
constructionArbreOrienté(T, boss) ;
```

Algorithme 4.2 : Construction d'une hiérarchie d'influence à partir d'un réseau. La méthode trieSommetDelta(*G*, *sommets_triés*) trie les sommets de *G* par valeur de Delta Efficiency décroissante et stocke le résultat dans *sommets_triés*. De même, la méthode trieRésultatNoeud(*résultat*, *sommets_résultat_triés*) trie les sommets de *résultat* en fonction de nom et stocke le résultat dans *sommets_résultat_triés*.

Entre les périodes 1997-99 et 2000-02, il y a une diminution des valeurs de similarité. Cette diminution se poursuit entre les périodes 2000-02 et 2003-2005 comme l'indique la zone étiquetée (c) dans la Figure 4.3. Il se trouve que durant cette période, *Ulrik Brandes* s'est déplacé tout d'abord à l'Université de Sydney puis à celle de Brown pour y réaliser des travaux de recherche post-doctoraux. Suivant les mêmes hypothèses que précédemment, le fait qu'il ait changé de milieu de travail (collègue, équipe) a marqué une rupture dans son travail ce qui se traduire logiquement par une baisse des valeurs de similarité. Cette hypothèse se trouve alors renforcée quand on sait que depuis 2003, il travaille à l'université de Constance. Cette stabilité de travail se traduit immédiatement dans le graphe de similarité par une hausse des valeurs de similarité, comme le montre la zone étiquetée (D).

La hiérarchie que nous avons pu construire à partir de ce réseau est visible dans la fenêtre *Hierarchy View* de la Figure 4.2. On y retrouve *Ulrik Brandes* comme racine de la hiérarchie ce qui reste cohérent au regard de la manière dont les données ont été collectées.

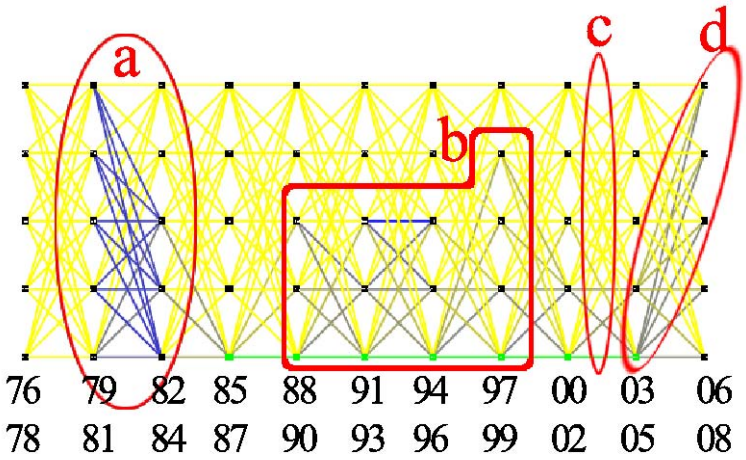


FIG. 4.3: Graphe de similarité obtenu à partir du réseau de co-auteur centré *Ulrik Brandes*. Les arêtes de couleur bleue représentent une similarité parfaite entre les deux sommets qu'elles relient. Les jaunes représentent une similarité faible. La zone étiquetée (a) suggère une très forte similarité entre les intervalles de temps 1979-81 et 1982-84. La zone (b) suggère une longue période de forte similarité entre 1988 et 1999. La zone (c) marque un changement structurel majeur puisqu'entre les deux périodes la similarité chute. La zone (d) montre qu'il y a de nouveau une forte similarité entre les périodes 2003-05 et 2006-08.

En effet, étant donné qu'il joue un rôle central dans la manière dont l'information (publication) circule dans le réseau, il n'est pas surprenant qu'il soit la personne ayant la plus grande valeur de Delta Efficiency. La hiérarchie produite possède une profondeur de 2, ce qui reste encore cohérent avec le fait que nous ayons collecté uniquement les chercheurs étant à distance au plus 2 d'*Ulrik Brandes* d'un point de vue des publications. D'un autre côté, on peut constater que la hiérarchie est très large, ce qui n'est pas très surprenant quand on sait qu'*Ulrik Brandes* a dans ce jeu de données co-signé des publications avec environ 200 personnes différentes.

4.4.2 Réseau Catalano/Vidro

Afin de visualiser la hiérarchie présente dans le réseau Catalano/Vidro, qui est un réseau terroriste, l'utilisateur doit activer le mode de traitement *counter terrorism mode*. L'idée qui est utilisée dans ce processus est basée sur l'étude de réseaux sociaux. On sait que dans ce genre de réseaux on peut distinguer trois types de rôles :

- les meneurs, qui sont les têtes pensantes,
- les portiers, qui contrôlent la diffusion de l'information dans le réseau,
- les suiveurs, qui ne font qu'exécuter les ordres.

Ceux qui ont la plus grande activité dans le réseau sont les portiers, il est donc évident que ce sont eux qui auront les plus grandes valeurs de Delta Efficiency. D'un autre coté, les meneurs et les suiveurs ont des activités de communications très restreintes. Les meneurs ne font qu'émettre l'information et les suiveurs ne font que recevoir. C'est pour cela qu'ils ont des valeurs de Delta Efficiency très basses. Les travaux [63, 62] déjà réalisés sur le domaine montrent que les meneurs essaient de se cacher parmi les suiveurs. Cela se traduit d'un point de vue de la Delta Efficiency par des valeurs d'un même ordre de grandeur. Ainsi, il est clair que pour construire une hiérarchie dans ce type de réseaux, l'objectif

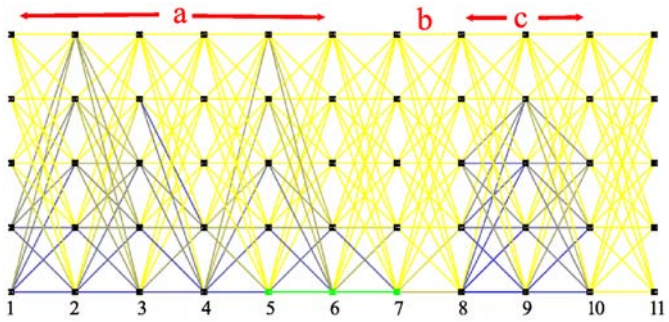


FIG. 4.4: Graphe de similarité du réseau Catalano/Vidro sur une période de 10 jours. Les arêtes bleues représentent une similarité parfaite et les jaunes, une similarité très faible. La plage (a) montre une forte similarité du jour 1 au jour 6. L'intervalle (b) montre une forte différence entre les graphes des jours 7 et 8 qui est synonyme d'un changement structurel majeur dans le réseau. La plage (c) montre un retour à la stabilité avec des valeurs de similarité fortes entre les jours 8, 9 et 10.

est d'identifier à quel rôle appartient un individu et le plus compliqué sera d'identifier les meneurs.

L'idée que nous avons eu pour répondre à ce cas de figure repose sur le fait que les arêtes de la hiérarchie doivent être des arêtes existantes dans le réseau initial. On sait que les portiers sont très souvent les bras droits des meneurs, ils sont donc par définition très proches des meneurs dans le réseau. Nous allons produire une hiérarchie à partir d'un arbre couvrant du réseau où chaque arête est pondérée par l'importance de la relation qui lie les deux sommets. C'est la Delta Efficency qui nous permettra de déterminer la notion d'importance de chaque arête.

Nous associons un poids à chaque arête qui est la différence entre les valeurs de Delta Efficiency des deux sommets qu'elle connecte. Une grande différence entre les valeurs indique que les deux sommets connectés ne peuvent pas être placés sur un même niveau de la hiérarchie puisqu'ils n'ont pas la même influence sur le réseau. L'arête qui lie ces sommets doit donc figurer dans la hiérarchie. Nous prenons la valeur absolue inversement proportionnelle à la valeur associée à chaque arête, ainsi nous pourrons construire un arbre couvrant de poids minimum pour construire la hiérarchie. L'algorithme 4.2 décrit toute les étapes qui permettent d'obtenir une hiérarchie d'influence dans le cas d'un réseau terroriste.

En observant la Figure 4.4, on peut facilement identifier qu'un changement structurel majeur s'est produit durant les jours 7 et 8. Il se trouve que les membres de ce réseau ont changé de téléphone à partir du jour 8, détruisant ainsi toutes les communautés précédentes. Mais on voit que très vite des nouvelles communautés se sont formées en l'espace de 3 jours (jours 8, 9 et 10). Ce graphe est un très bon exemple de l'utilité que représente la visualisation des similarités entre les graphes. La rupture qui se produit entre les jours 7 et 8 est une information importante sur le réseau que nous n'aurions pas pu déceler sans cela.

La Figure 4.5 présente un cluster trouvé dans le graphe consensus formé à partir de des jours 1 à 6. Les sommets qui ont les plus fortes valeurs de Delta Efficency sont les sommets 1, 2 et 5. On sait que ce sont les portiers, qui se trouvent être aussi les bras droits des meneurs, qui possèdent les plus fortes valeurs de Delta Efficency étant donné que ce sont eux qui diffusent l'information dans le réseau. Nous en déduisons donc que les sommets

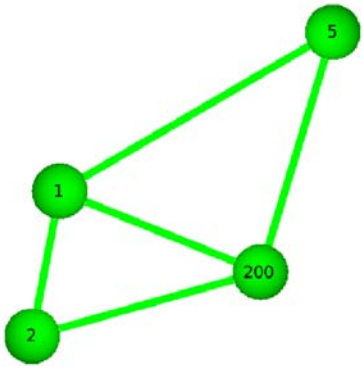


FIG. 4.5: Un cluster trouvé lors de l'analyse du jeu de données Catalano/Vidro. Ce cluster a été trouvé dans le graphe consensus formé à partir des jours 1 à 6. Il est composé de quatre sommets dont trois sont les sommets ayant obtenu les plus grandes valeurs de Delta Efficiency : les sommets 1, 2 et 5. Les sommets de fortes Delta Efficiency sont les bras droits du meneur dans ce genre de réseau. Donc ce cluster confirme le fait avéré que le sommet 200 est le meneur du réseau terroriste et les sommets 1, 2, 5 sont les bras droits du meneur.

1, 2 et 5 jouent le rôle de portier dans ce réseau. Cependant, il est clair que les portiers reçoivent directement les informations du ou des meneurs et ce sur des périodes plutôt longues. En effet, puisque les rôles de portiers et de meneurs sont des rôles clefs du réseau, pour conserver une efficacité de communication ces rôles doivent changer le moins souvent possible. Il est donc logique de penser que les meneurs communiquent ponctuellement mais directement avec les portiers et que les meneurs doivent être dans le voisinage des portiers dans le graphe consensus. Ainsi le sommet 200 répond à tous les critères d'un meneur. Il possède une valeur de Delta Efficency très basse pour essayer de se cacher au milieu des suiveurs et il appartient au voisinage de tous les portiers du réseau comme on peut le voir dans le cluster présenté dans la Figure 4.5. Nous avons pu vérifier la véracité les résultats que nous venons d'avancer en ce qui concerne les portiers et le meneur du réseau Catalano/Vidro puisqu'un référentiel des rôles a été publié pour ce réseau.

Maintenant que nous sommes capable de déterminer un arbre couvrant du réseau, pour obtenir une hiérarchie d'influence et les rôles joués par chacun, nous devons identifier le ou les meneurs. Nous savons que les meneurs ont une valeur de Delta Efficency faible étant donné qu'ils essaient de se dissimuler dans le réseau en communiquant le moins possible. De plus, il est clair que les portiers reçoivent directement les informations du ou des meneurs et donc que les meneurs se trouvent dans le voisinage des portiers. Nous allons alors identifier en premier lieu les portiers car ceux-ci sont plus visibles de par leurs fortes valeurs de Delta Efficency comme le suggère V. Latora *et al.* dans [57]. Mais tous les sommets ayant une fortes valeurs de Delta Efficency ne sont pas forcement des portiers, nous avons donc fixé à 3% du nombre de sommets dans le réseau le ratio de portiers. Ce ratio donne de bon résultat dans cet exemple. Une fois que nous avons sélectionné les 3% de sommets ayant les plus fortes valeurs de Delta Efficency, nous étudions leurs voisinages. Nous allons comparer les voisinages des sommets deux à deux et à chaque fois qu'un sommet apparait dans les deux voisinages nous incrémentons de 1 la valeur de présence qui lui est associée. Après avoir comparé toutes les paires de voisinage, nous élisons comme meneur le sommet ayant la plus grande valeur de présence. Si jamais plusieurs sommets se trouvent à égalité après cela, nous élisons le sommet ayant la plus faible valeur de Delta

Efficency comme seul meneur.

Une fois que nous avons un arbre couvrant et que nous avons réussi à identifier un meneur, nous donnons à l'arbre couvrant une orientation. Pour cela, comme le meneur doit se trouver au sommet de la hiérarchie, nous orientons les arêtes de manière à ce qu'il ait une degré entrant nul (cf. Définition 2.28). Puis on oriente les arêtes non-orientées connectées à ses voisins de manière à ce qu'elle soient des arêtes sortantes. On pratique de même pour tous les sommets de l'arbre de proche en proche.

La hiérarchie ainsi obtenue est affichée dans la fenêtre *Hierarchy View* comme on peut le voir dans la Figure 4.2.

4.5 Conclusion

Dans ce chapitre, nous avons présenté un système permettant d'analyser les réseaux sociaux dynamiques. Il est possible d'y détecter les changements s'effectuant au cours d'un intervalle de temps à l'aide d'une discrétisation du graphe ainsi que d'une fragmentation de ce dernier. Nous avons aussi présenté un algorithme permettant de mettre en évidence une hiérarchie d'influence présente dans un réseau social, basée sur l'efficacité des communications et les arbres couvrant de poids minimum. Notre système a été utilisé pour analyser deux jeux de données différents et les résultats obtenus sont satisfaisants. En effet, nous sommes capables d'identifier correctement les moments où des changements structuraux majeurs du réseau se sont produits. De plus, nous mettons en évidence une hiérarchie d'influence révélant les rôles des acteurs d"un réseau social. Toutefois, le système a quelques limitations. Étant donné que le nombre de graphes générés lors de l'étape de discrétisation est élevé, les performances du système dépendent fortement de l'algorithme de fragmentation utilisé. Or, générer un nombre plus restreint de graphes conduirait à une perte d'informations.

Chapitre 5

Génération de graphes basée sur une taxonomie de dimensions

5.1 Motivation

De nos jours, il est extrêmement facile de collecter des données. Il existe d'ailleurs un grand nombre de méthodes et moyens pour collecter et stocker des données. En ce qui concerne le stockage, les bases de données sont les moyens les plus utilisés et contiennent des données basiques telles que des mots, des nombres, des dates, des horaires ... Ces données sont stockées dans le but de conserver certaines informations comme : qui, quoi, quand, combien de fois etc ... Mais ces facilités de collecte et stockage ont pour conséquence de voir augmenter la quantité de grandes bases de données. Quant à leur taille, celle-ci rend leur exploitation quasi impossible pour un utilisateur non expert.

Il existe des systèmes de visualisation comme DEVise [59], Polaris [83] ou Tableau [60] qui permettent de visualiser ces grandes quantités de données. Ces systèmes suivent le processus d'analyse exploratoire défini par une hypothèse, puis une expérimentation et finalement une découverte. Mais avant cela, il est nécessaire de transformer les données d'un format brut vers une structure de données utilisable par les différents composants de visualisation de ces systèmes. Or cette étape représente une barrière empêchant l'accès des utilisateurs non experts à une riche variété de techniques de visualisation. Par exemple, il peut être techniquement impossible pour un utilisateur non expert de transformer des données arborescentes en un modèle de graphe pouvant utiliser une représentation à base de TreeMap [49]. En plus de cela, chaque système de visualisation utilise un modèle de graphe qui lui est propre donc l'utilisation des spécificités de chaque système demanderait de grandes capacités techniques aux utilisateurs non experts qui viennent de domaines différents tels que la biologie, la géographie ou encore la sociologie.

Considérons le cas d'une entreprise qui cherche à gérer son stock de marchandise du mieux possible. Pour chaque article, elle a collecté les information suivantes : le lieu de stockage, des informations sur l'article (taille, poids), le nom du fournisseur ainsi que les personnes responsables de la manutention de cet article. Les responsables du stock sont capable de déterminer où se trouve les articles d'un certain fournisseur. Mais avec un grand nombre d'article et la multiplication des lieux de stockage, même des tâches aussi simples que celles-ci peuvent devenir difficiles. De plus, repérer des motifs dans un tel jeu de données peut devenir très coûteux en temps voire même impossible.

En utilisant comme données initiales une table extraite d'une base de données, nous proposons dans ce chapitre de générer de manière automatique des graphes valués. Nous proposons aussi un système qui permet à l'utilisateur d'explorer visuellement l'ensemble des graphes construits. La principale contribution décrite dans ce chapitre est une méthode qui simplifie une table et met en évidence les relations liant les entités présentes dans la table. Nous pensons que la méthode proposée rend le processus exploratoire plus efficace et facilite les tâches exécutées par l'utilisateur. Notre méthode de simplification est basée sur la construction d'une taxonomie des dimensions de la table. Cette taxonomie permet de souligner les propriétés d'imbrication des dimensions du jeu de données. Puis à l'aide de cette taxonomie, nous sommes capables d'identifier des relations entre les entités et donc de générer des visualisations simplifiées du jeu de données.

Ce chapitre est structuré de la manière suivante : nous présentons tout d'abord le processus que suit notre méthode, les différentes étapes menant à la construction d'une taxonomie des dimensions et enfin nous présentons les interactions mises à disposition de l'utilisateur au travers d'un cas d'étude.

5.2 Processus de la méthode

Si l'on se rapporte au processus complet de visualisation présenté dans la Section 1.2, la méthode présentée ici se concentre sur la première étape de ce processus. Nous proposons de diviser cette étape en trois sous-étapes comme présenté dans la Figure 5.1 : une première dite de nettoyage des données, une seconde où l'utilisateur va sélectionner les données ou les parties des données avec lesquelles il souhaite travailler et enfin une étape d'analyse des données pour produire une taxonomie des dimensions et des graphes.

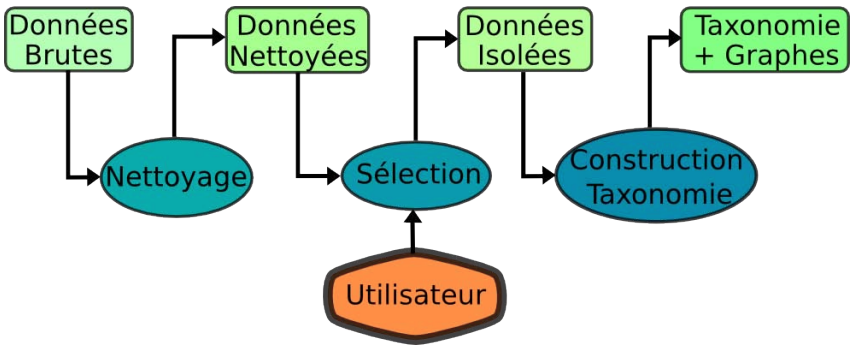


FIG. 5.1: Processus utilisé pour la construction d'une taxonomie de dimensions d'une table. A partir des données brutes de la table, on obtient une taxonomie des dimensions ainsi que des graphes. Ce processus se décompose en trois étapes : le nettoyage, la sélection de données lors de laquelle l'utilisateur peut interagir et la construction de la taxonomie des dimensions et de graphes.

Nous allons au cours de ce chapitre détailler l'ensemble de ces sous-étapes. Tout d'abord, nous définirons ce que la taxonomie des dimensions modélise et comment elle est construite. Ensuite, nous verrons comment sont construits des graphes en s'appuyant sur la taxonomie. Puis, nous présenterons un moyen d'accéder facilement aux graphes générés avant d'expliquer en quoi il est intéressant de nettoyer les données avant de construire

cette taxonomie et de discuter de l'utilité de la sélection de certaines dimensions par l'utilisateur. Enfin nous présenterons les interactions que l'on offre à l'utilisateur et nous finirons par l'étude d'un exemple d'utilisation.

5.3 Taxonomie des dimensions

5.3.1 Hiérarchisation des dimensions

Nous décrivons ici la façon suivant laquelle nous simplifions les données sans qu'il n'y ait pas de pertes d'informations. Notre idée de départ est de détecter les dimensions qui ont plus d'importance que les autres dans le but de trouver par la suite des relations entre les entités. Afin d'obtenir une sorte de classement par ordre d'importance des dimensions, nous allons rechercher s'il existe des surjections (cf. Définition 2.12) entre les dimensions de la table.

Pour vérifier qu'il existe une surjection entre les valeurs de deux dimensions de la table, on peut utiliser la Définition 5.1. Cette définition introduit également la notion de hiérarchisation des dimensions ainsi que la notation que l'on utilisera pour les identifier.

Définition 5.1 *Considérons C_i et C_j deux dimensions de la table et leurs alphabets Σ_i, Σ_j. S'il existe une surjection $s : \Sigma_j \rightarrow \Sigma_i$ de Σ_j vers Σ_i. On dit que C_i hiérarchise C_j et on notera $C_i \geq C_j$.*

En comparant deux à deux les dimensions, nous sommes capable de hiérarchiser les dimensions les unes par rapport aux autres ce qui nous permet d'élaborer des hiérarchies de dimensions. Les différentes hiérarchies obtenues peuvent être fusionnées en une taxonomie. Nous avons choisi de représenter cette taxonomie à l'aide d'un graphe orienté (cf. Définition 2.18). Dans ce graphe, chaque dimension de la table sera matérialisée par un un nœud (sommet) étiqueté du nom de la dimension qu'il représente. Lorsque l'on aura déterminé que la dimension A hiérarchise la dimension B, alors on ajoutera au graphe un arc allant du nœud étiqueté A vers le nœud étiqueté B. Dans cette taxonomie, les dimensions ayant un degré entrant (cf. Définition 2.28) nul sont les dimensions qui fournissent les informations les plus globales et les dimensions ayant un degré sortant nul fournissent les informations les plus précises.

5.3.2 Exemple de taxonomie de dimensions

Considérons la Table 5.1, où chaque ligne contient des informations sur des adresses de personnes. Les différentes dimensions de la table sont : Continent, Pays, Ville, Rue, Nom. En appliquant notre méthode, on obtient comme on peut le voir dans la Figure 5.2(a) "Continent" $\geq$ "Pays" $\geq$ "Ville" $\geq$ "Nom" d'une part et "Continent" $\geq$ "Rue" $\geq$ "Nom" d'autre part. On ne peut pas avoir une hiérarchie composée des cinq dimensions car la rue "Louis Pasteur" et la rue "Grand" empêchent de valider les hiérarchisations entre les dimensions "Ville" et "Rue" et entre les dimensions "Pays" et "Rue".

Continent	Pays	Ville	Rue	Nom
Europe	France	Paris	Louis Pasteur	Dupont
Europe	France	Paris	Champs-Elysée	Durand
Europe	France	Bordeaux	Louis Pasteur	Martin
Europe	Allemagne	Berlin	Max Planck	Muller
Europe	Allemagne	Munich	Max Planck	Fischer
Europe	Espagne	Madrid	Louis Pasteur	Fernandez
Amérique du Nord	États-Unis	New York	Grand	Smith
Amérique du Nord	États-Unis	Boston	Beacon	Do
Amérique du Nord	Canada	Calgary	Grand	Wilson

TAB. 5.1: Un exemple de table regroupant des adresses de personnes

5.3.3 Propriétés et simplifications la taxonomie des dimensions

Propriété 5.1 *Considérons deux dimensions C_i et C_j. Si $C_i \geq C_j$ et $C_j \geq C_i$, alors C_i et C_j sont égales.*

Preuve : Comme on a que $C_i \geq C_j$, il existe une surjection $s_1 : C_j \rightarrow C_i$ et comme on a aussi que $C_j \geq C_i$, il existe aussi une surjection $s_2 : C_i \rightarrow C_j$. Donc il existe une surjection $S : C_i \rightarrow C_i$ telle que $S = s_1 \circ s2$. Montrons par l'absurde que S est nécessairement une bijection (cf. Définition 2.13).

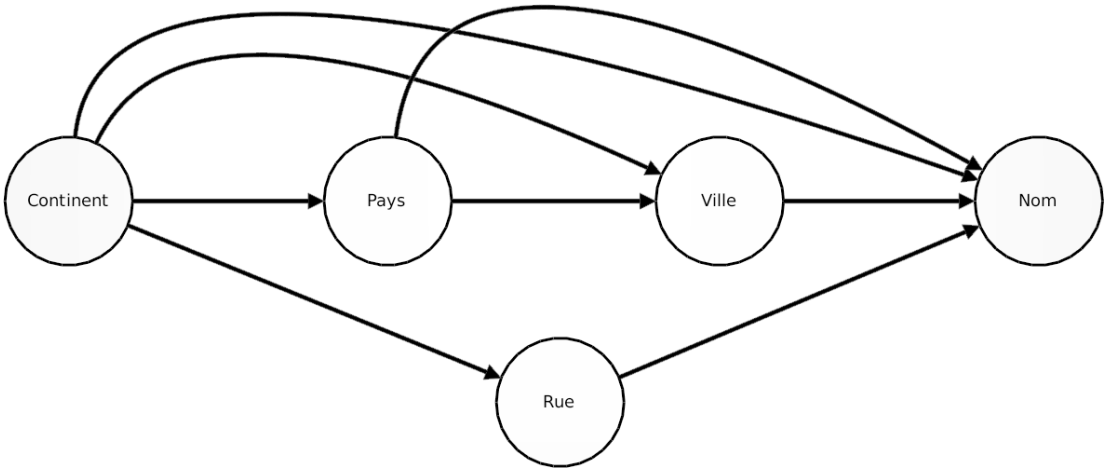
Si S n'est pas une bijection, alors il existe au moins un élément x de C_i ayant au moins deux antécédents x_1 et x_2. Soit $|C_i| = n$ la cardinalité de C_i, alors il reste $n-1$ éléments de C_i pour lesquels il faut déterminer les antécédents. Mais ces $n-1$ éléments ne peuvent en aucun cas avoir x_1 ou x_2 comme antécédent car comme S est une application (cf. Définition 2.10) x_1 et x_2 ne peuvent pas être les antécédents de deux éléments. Donc il resterait $|C_i - \{x_1, x_2\}| = n - 2$ antécédents pour $|C_i - \{x\}| = n - 1$ éléments, ce qui est absurde. Donc S est nécessairement une bijection.

Or on sait que si une application $S = s_1 \circ s2$ est une bijection alors s_2 est une injection (cf. Définition 2.11). Donc s_2 est à la fois une injection est une surjection donc c'est une bijection. Donc, puisqu'il existe entre C_i et C_j une bijection, on peut dire que ces deux dimensions sont égales. □

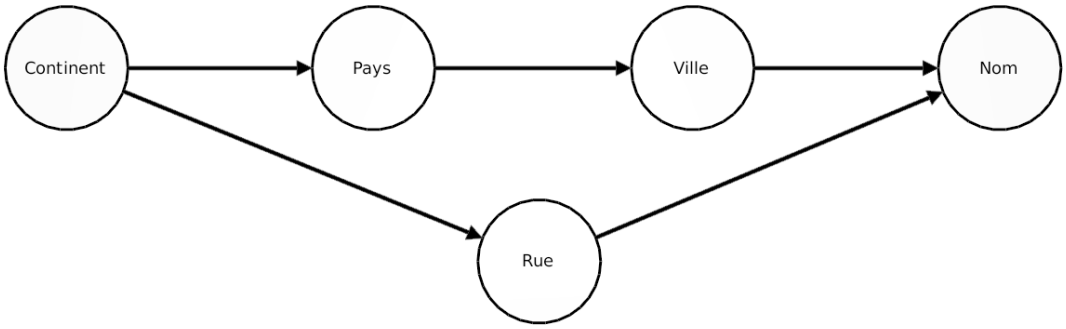
Propriété 5.2 *Considérons trois dimensions C_1, C_2 and C_3. Si on a $C_3 \geq C_2 \geq C_1$, alors $C_3 \geq C_1$.*

Preuve : Comme on a que $C_2 \geq C_1$, il existe une surjection $s_1 : 1 \rightarrow 2$ et comme on a aussi que $C_3 \geq C_2$, il existe une autre surjection $s_2 : 2 \rightarrow 3$. Alors il existe par composition de surjections une surjection $S : 1 \rightarrow 3$ telle que $s_{1,3} = s_{1,2} \circ s_{2,3}$. On obtient alors que $C_3 \geq C_1$. □

Propriété 5.3 *Considérons une taxonomie de dimensions construite suivant la Définition 5.1. Toutes les dimension d'une même composante fortement connexe sont égales.*



(a) La taxonomie obtenue à partir de la Table 5.1 en comparant les dimensions deux à deux. S'il y a une arête allant du nœud A vers le nœud B, cela signifie que la dimension A hiérarchise la dimension B. On peut donc dire que la dimension B fournit des informations plus précises que celles de la dimension A.



(b) La taxonomie obtenue à partir de la Table 5.1 une fois débarrassée des arcs superflus grâce notamment à la Propriété 5.2. On peut distinguer deux branches dans cette taxonomie. Une contenant les dimensions "Continent", "Pays", "Ville", "Nom", et une seconde contenant les dimensions "Continent", "Rue"et "Nom".

FIG. 5.2: Figures représentant la construction de la taxonomie des dimensions associées à la Table 5.1. La Sous-Figure (a) montre la taxonomie calculée sans tenir compte des propriétés sur les taxonomies ou des simplifications possibles. La Sous-Figure (b) montre la même taxonomie, mais en tenant compte des propriétés et simplifications.

Preuve : En effet, si l'on a les inéquations $C_1 \geq C_2 \geq C_3 \geq C_1$ cela signifie qu'il y a un cycle dans la taxonomie. Considérons les inégalités $C_1 \geq C_2 \geq C_3$. On peut donc en conclure que $C_1 \geq C_3$ et comme on a déjà dans l'inégalité de départ que $C_3 \geq C_1$, on a grâce à la Propriété 5.1 que C_1 est équivalente à C_3. Par extension on a que $C_1 \geq C_2 \geq C_1$, donc C_1 est équivalente à C_2. Donc finalement on a que C_1 est équivalente à C_2 qui est elle même équivalente à C_3. Donc les dimensions d'un cycle sont toutes équivalentes. Or, par définition (cf. Définition 2.33) une composante fortement connexe contient des cycles. Ce qui signifie qu'à chaque fois que l'on rencontre une composante fortement connexe dans une taxonomie, toutes les dimensions de cette composante fortement connexe sont équivalentes. □

En utilisant les différentes propriétés énoncées précédemment, il est possible de simplifier les taxonomies construites à l'aide de la Définition 5.1. Tout d'abord, nous avons

montré dans la Propriété 5.1 qu'il était possible que deux dimensions soient équivalentes. Si deux dimensions sont équivalentes, il n'est pas nécessaire de conserver ces deux dimensions dans la suite du processus puisque les informations qu'elles apportent feront doublon. Ainsi, la suppression d'une de ces dimensions ne génèrera pas de perte d'informations et en conserver une seule réduira le nombre total de dimensions (sommets) dans la taxonomie. Étant donné que cette taxonomie sera parcourue pour générer des graphes, nous améliorerons la complexité de la génération de graphes.

Au niveau de la taxonomie, la suppression d'une dimension revient à supprimer un des deux sommets. Comme les deux dimensions sont équivalentes, elles hiérarchisent et sont hiérarchisées par les mêmes dimensions donc les arcs correspondants dans la taxonomie font doublon. Donc tous ceux qui vont disparaitre avec la suppression du sommet ne provoqueront pas de perte d'informations. Et moins il y aura d'arcs dans la taxonomie, moins le nombre de parcours possibles de la taxonomie sera élevé. Ce qui, encore une fois, réduira la complexité de la suite du processus.

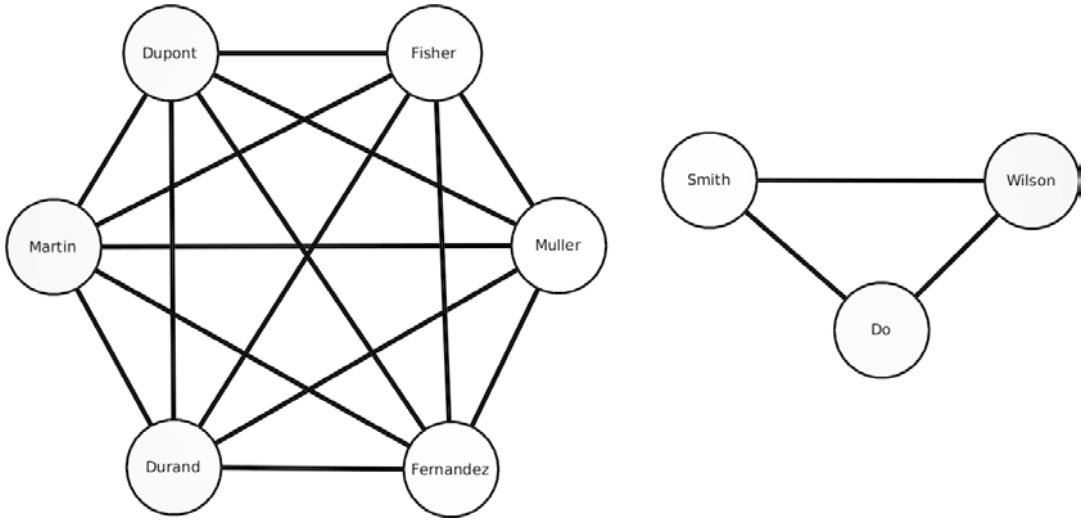
Maintenant que nous avons mis en évidence l'intérêt de ne conserver qu'une des deux dimensions équivalentes, nous allons pouvoir étendre cette simplification aux composantes fortement connexes. Nous avons montré dans la Propriété 5.3 que toutes les dimensions d'une même composante fortement connexe sont équivalentes. Donc d'après la simplification que nous avons proposé pour deux dimensions équivalentes, il est intéressant de ne considérer qu'une seule de des dimensions d'une même composante fortement connexe dans la suite du processus. Ce qui simplifiera encore plus grandement la taille de la taxonomie, sa compréhension et les futurs parcours que l'on fera sur elle. La simplification du graphe s'effectue sur le même principe que celui utilisé dans la cas de deux dimensions équivalentes. De plus, comme nous pouvons retirer tous les cycles de la taxonomie, nous pouvons affirmer que toutes les taxonomies construites sont des DAG (cf. Définition 2.36).

Une dernière simplification qu'il est possible de faire sur la taxonomie pour en réduire la taille repose sur la Propriété 5.2. En effet, comme nous comparons chaque dimension avec chacune des autres dimensions de la table, la taxonomie possède des arêtes superflues. Dans notre exemple (cf. Figure 5.2(a)) on a : "Continent" $\geq$ "Pays" $\geq$ "Ville". Mais comme nous comparons tous les couples possibles de dimensions nous obtenons aussi que "Continent" $\geq$ "Ville". Or cette arête n'apporte pas d'information supplémentaire à la taxonomie, puisque nous pouvons la retrouver grâce à la branche "Continent" $\geq$ "Pays" $\geq$ "Ville" comme le montre la Propriété 5.2. Donc nous pouvons simplifier la taxonomie en retirant toutes les arêtes de ce type. Il est même plus intéressant de détecter ce genre de situations, en évitant de comparer des dimensions qui appartiennent déjà à une même branche de la taxonomie. De cette manière, on économise plusieurs comparaisons de dimensions ainsi que des opérations de suppression de nœuds et arêtes dans le graphe.

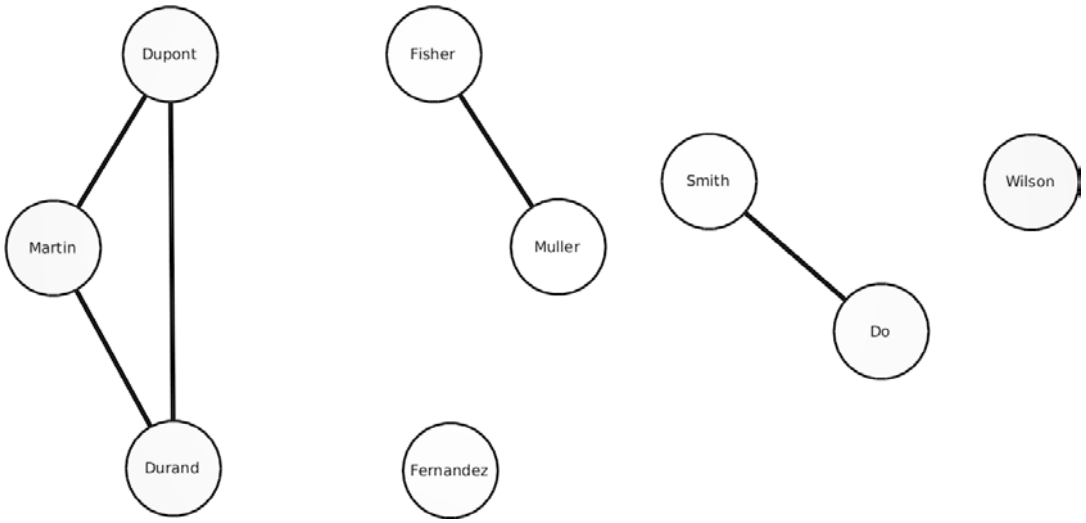
5.3.4 Complexité

En utilisant la définition ainsi que les propriétés précédentes, nous sommes capables de fournir à l'utilisateur une taxonomie des dimensions d'une table $T_{n,m}$ avec laquelle il souhaite travailler.

Dans [37], nous utilisons une méthode naïve (directement tirée de la Définition 5.1) pour comparer deux dimensions ayant une complexité en $\mathcal{O}(n^2)$. Puis étant donné qu'il faut comparer deux à deux toutes les dimensions de la table, il y aura m^2 comparaisons, ce qui donne une complexité finale en $\mathcal{O}(m^2 \times n^2)$ qui peut aussi être vue comme



(a) Graphe obtenu en utilisant comme dimension de référence la dimension "Continent" de la Table 5.1. Les personnes qui vivent sur le même continent géographique sont connectés. Il y a deux composantes connexes : celle de gauche contient les personnes vivant en Europe, celle de droite les personnes vivant en Amérique du Nord.



(b) Graphe obtenu en utilisant comme dimension de référence la dimension "Pays" de la Table 5.1. Les personnes qui résident dans le même pays sont connectés. Il y a cinq composantes connexes, une pour chaque pays : France, Allemagne, Espagne, États-Unis et Canada.

FIG. 5.3: Chacune des Figures présente le graphe obtenu à partir du choix d'une des dimensions de la table comme dimension de référence pour afficher les arêtes. Chaque valeur de la dimension de référence engendre une composante connexe. Chacune de ces composantes connexes est une clique(cf. Définition 2.41) puisque tous les sommets d'une même composante partagent la même valeur suivant la dimension prise pour référence. Étant donné que la dimension "Continent" hiérarchise la dimension "Pays" dans la taxonomie (cf. Figure 5.2(b)), chaque composante connexe du graphe de la Figure (b) est un sous-graphe induit de la composante connexe correspondante dans la Figure (a). La composante connexe "France" composée des sommets Dupont, Martin et Durand est bien un sous-graphe induit de la compopsante "Europe".

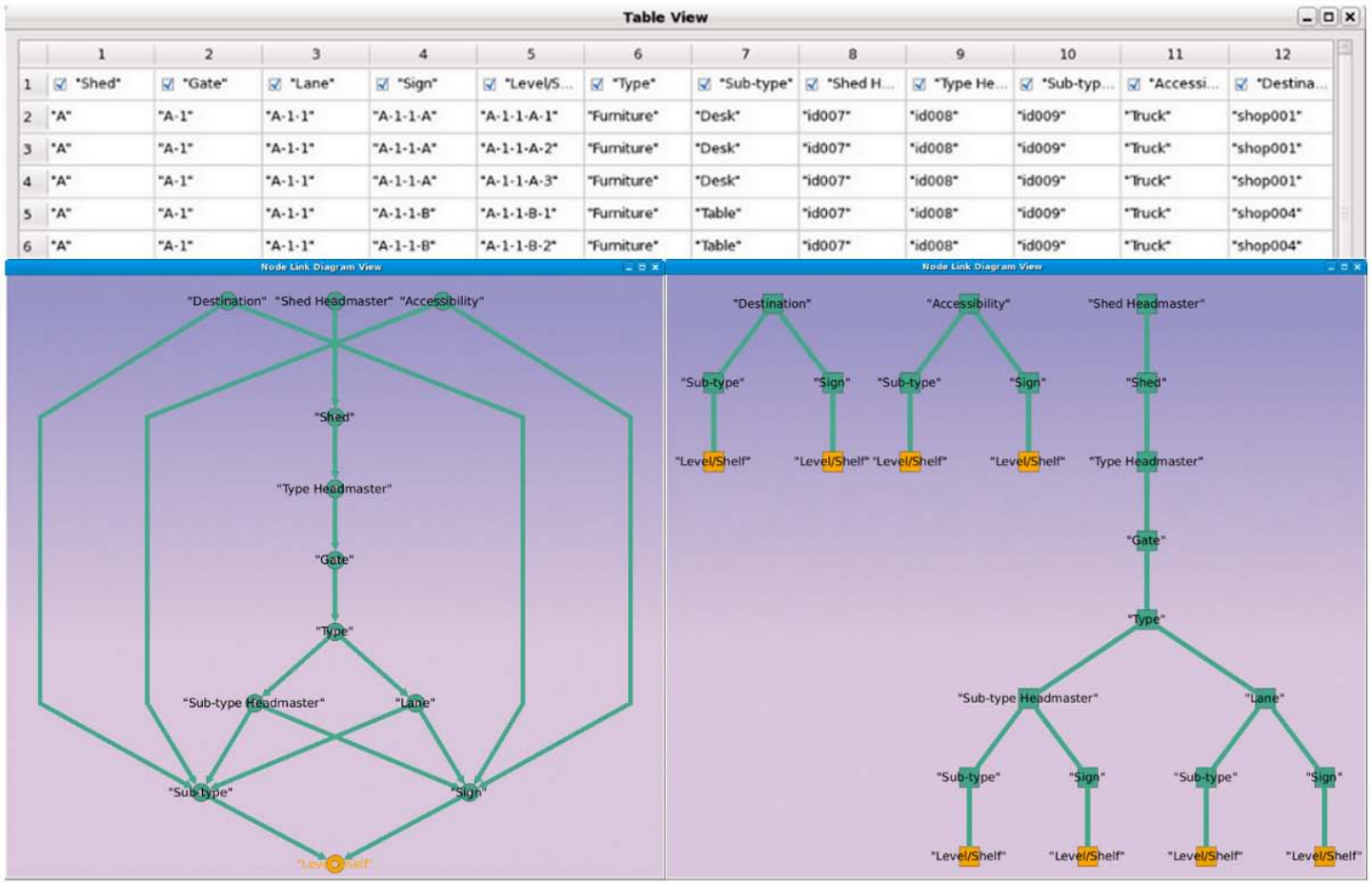


FIG. 5.4: La fenêtre intitulée "Table View" propose une vue sous forme de table des données. La première ligne contient les noms des dimensions de la table, puis chacune des autres lignes est considérée comme une entité (nœud). La fenêtre située en bas à gauche propose une vue de la taxonomie calculée à partir des dimensions de la table. Les nœuds colorés en orange représentent les dimensions pouvant être utilisées comme identifiant pour les entités. La fenêtre située en bas à droite propose une vue de la hiérarchie des dimensions obtenue en dépliant la taxonomie.

$\mathcal{O}(\min(n,m)^4)$. Or une telle complexité reste à la limite du raisonnable. Nous avons donc cherché à améliorer notre méthode afin d'en réduire la complexité. Étant donné que nous utilisions jusque-là une comparaison brute et naïve de deux dimensions et que d'autre part il n'est pas possible de réduire les nombres de comparaisons deux à deux entre les dimensions, nous nous sommes efforcés d'améliorer la complexité de la comparaison de deux dimensions.

Proposition 5.1 *La complexité de l'algorithme de construction de la taxonomie est de l'ordre de* $\mathcal{O}(m^2 \times n \log n)$.

Preuve : En appliquant des tris sur les valeurs des dimensions à comparer, il est possible de comparer deux dimensions en $\mathcal{O}(n \log n + n)$. Nous avons donc amélioré la complexité de comparaison, passant de $\mathcal{O}(n^2)$ à $\mathcal{O}(n \log n)$. Cette amélioration se reporte bien sur la complexité totale de notre méthode de construction de la taxonomie. □

Toutefois, il ne faut pas perdre de vue le fait que les complexités présentées ici sont les complexités dans les pires cas. Ainsi, dès lors que l'on commence à construire une branche dans la taxonomie, par la Propriété 5.2, il y a des comparaisons qui seront superflues ainsi en étant attentif à ce cas de figure, la complexité réelle/pratique est inférieure.

5.4 Construction des graphes à l'aide de la taxonomie

Comme nous l'avons évoqué dans la Section 5.1, nous considérons dans cette méthode que chaque ligne de la table utilisée décrit une entité. Ainsi, puisque nous souhaitons produire un graphe mettant en évidence des relations entre les entités de la table, nous allons devoir créer dans ce graphe un sommet pour chaque entité. Pour cela, nous allons utiliser une information de la taxonomie. En effet comme nous l'avons dit dans la Section 5.3.1, il est possible qu'une des dimensions ayant un degré sortant nul puisse servir à identifier les entités. Si tel est le cas, nous allons créer dans le graphe un sommet pour chaque valeur de cette dimension et l'étiqueter de cette même valeur. Si jamais il n'existe pas de dimension permettant d'identifier les entités, nous utilisons comme étiquette le numéro de la ligne de la table.

Maintenant que nous avons modélisé les entités de la table dans le graphe, nous allons décrire comment nous déterminons si deux entités sont en relation.

Afin de savoir si deux entités sont en relation, nous vérifions si elles ont la même valeur dans au moins une des dimensions. Par exemple, considérons une table rassemblant des informations sur des avions. Pour chaque avion, nous avons le nom de la compagnie à laquelle il appartient. Si deux avions ou plus ont la même valeur de compagnie, cela signifie que ces avions sont contrôlés par la même compagnie. De cette manière nous sommes capable de définir une relation d'appartenance. À chaque fois que deux entités sont en relation, elles sont reliées par une arête dans la graphe.

Toutefois, il est possible d'éviter d'avoir à comparer les différentes valeurs dans toutes les dimensions de la table. Pour cela nous allons utiliser les informations fournies par la taxonomie. Nous allons chercher s'il existe une relation entre des entités uniquement pour les dimensions les plus basses dans la taxonomie. Les dimensions les plus basses sont les dimensions les plus proches des dimensions utilisées pour identifier les entités, ce sont

elles qui apportent les informations les plus précises sur les entités. Nous nous limitons à ces dimensions car d'après la construction de la taxonomie, nous sommes sûr que si des entités sont en relation d'après une de ces dimensions, elles le seront aussi d'après les autres dimensions de la même branche de la taxonomie.

Mais si nous procédons de cette manière, il y aura un très grand nombre d'arêtes dans le graphe.

En effet, considérons l'exemple des adresses de la Table 5.1. Après avoir calculé une taxonomie des dimensions et l'avoir simplifiée, on obtient comme on peut le voir dans la Figure 5.2(b) que : "Continent" $\geq$ "Pays" $\geq$ "Ville" $\geq$ "Nom". Donc si deux personnes (lignes/entités) de la table vivent dans la même ville, par exemple "Paris", alors nécessairement elles vivent dans le même pays, la "France" et sur le même continent, l'"Europe". Il y aura donc dans le graphe trois arêtes, une pour la correspondance de la valeur de la dimension "Ville", une pour la correspondance de la valeur de la dimension "Pays" et une pour la correspondance de la valeur de la dimension "Continent". Cela illustre bien qu'il y a aura un très grand nombre d'arêtes dans les graphes qui seront générés en utilisant cette méthode.

Dans le but de limiter le nombre d'arêtes, nous avons décidé de n'ajouter des arêtes au graphe qu'en fonction des dimensions les plus basses dans la taxonomie, puis de simplement ajouter des informations sur ces arêtes. Par exemple, au lieu d'avoir trois arêtes à ajouter au graphe, dans le cas précédant nous ajouterons seulement une arête à laquelle nous ajouterons trois attributs. Un premier attribut qui indiquera que cette arête peut être considérée lors de relations suivant le "Continent", un second qui indiquera que cette arête peut être considérée lors de relations suivant le "Pays" et un troisième qui indiquera que cette arête peut être considérée lors de relations suivant la "Ville". Ainsi, le nombre total d'arêtes dans le graphe va fortement décroitre. Mais dès lors, comme chaque arête peut définir des relations pour plusieurs dimensions, il sera difficile de comprendre au travers de quelle dimension les entités observées sont en relation.

C'est pour cela que nous avons mis au point un système de filtrage décrit dans la Section 5.5.

5.5 Filtrage des graphes à l'aide de la taxonomie

Comme nous l'avons expliqué précédemment, nous ajoutons des arêtes entre toutes les entités qui partagent une même valeur dans une des dimensions de la table. Il y aura donc dans le graphe généré précédemment un grand nombre de cliques (cf. Définition 2.41). De plus, comme une arête pourra définir des relations pour plusieurs dimensions, nous devons être capable de les filtrer pour ne faire apparaitre que les plus significatives ou uniquement celles qui intéressent l'utilisateur. Pour cela, nous avons décidé de construire une hiérarchie de sous-graphes basée sur la taxonomie des dimensions que nous avons calculée. Chaque branche de la hiérarchie est un chemin dans la taxonomie des dimensions, donc au final cette hiérarchie correspondra à une vue dépliée de la taxonomie. On parle ici de hiérarchie et non d'arbre car comme il s'agit de sous-graphes, on a une relation d'inclusion entre les sous-graphes et leur prédécesseur dans la hiérarchie.

Chaque nœud est lié à un sous-graphe obtenu en filtrant les arêtes en fonction des nœuds le précédant dans la hiérarchie. Par exemple, considérons la branche "Continent" $\geq$ "Pays" $\geq$ "Ville" $\geq$ "Nom" de l'exemple sur des adresses. Le sous-graphe lié à la dimension

"Ville" contiendra toutes les arêtes qui satisfont une relation au travers de la dimension "Ville", mais ces arêtes seront aussi présentes dans les sous-graphes correspondant aux dimensions "Pays" et "Continent".

La racine de la hiérarchie est un graphe comprenant toutes les arêtes générées. Les premiers sous-graphes (les successeurs de la racine) correspondent aux dimensions les plus hautes dans la taxonomie. Il y aura dans la hiérarchie un fils pour chaque nouvelle dimension rencontrée en parcourant la taxonomie. Cela aura donc pour effet de dupliquer certains nœuds de la taxonomie comme on peut le voir dans la Figure 5.4. En effet, si un nœud de la taxonomie a pour degré entrant n, alors cela signifie qu'il existe au moins n manière de l'atteindre dans la taxonomie. Donc il y aura dans la hiérarchie au moins n branches comportant ce nœud.

Dans le système que nous avons mis en place, nous avons fait en sorte que lorsque l'utilisateur clique sur un nœud de la hiérarchie, le sous-graphe correspondant soit affiché dans une fenêtre voisine. On peut alors se demander quel est vraiment l'intérêt d'avoir créé une hiérarchie, car un tel comportement aurait pu être appliqué directement sur la taxonomie. L'avantage que présente l'utilisation de la hiérarchie réside dans la capacité à suivre plus facilement un chemin dans un arbre que dans un DAG.

La perception de l'effet de filtrage des arêtes est augmentée par le dessin de graphe que l'on donne à chaque sous-graphe. En effet, comme tous les nœuds (entités) sont présents dans chaque sous-graphe, nous avons attribué à chacun d'eux les mêmes coordonnées dans chaque dessin. Ainsi, lorsque l'utilisateur clique sur un nouveau nœud dans la hiérarchie, seules certaines arêtes apparaissent ou disparaissent. On peut se rendre compte de cet effet en observant la Figure 5.3.

5.6 Nettoyage et présélection des données

Malgré le fait que nous soyons parvenu à réduire la complexité de la construction de la taxonomie des dimensions, celle-ci reste assez élevée. Ainsi, toute manipulation de la table qui nous permettrait de réduire le nombre de lignes dans la table avant de construire une taxonomie entrainerait une diminution de la complexité. Une manipulation automatique que nous avons mis en place pour tenter de réduire le nombre de lignes dans la table consiste à repérer dans la table toutes les lignes qui apparaissent plusieurs fois. Une fois ce repérage effectué, nous ajoutons une nouvelle dimension à la table dans laquelle nous indiquons le nombre de fois où une ligne apparait dans la table, puis nous ne conservons qu'un seul exemplaire de chaque ligne. Ainsi nous pouvons réduire le nombre de lignes avec lequel nous allons travailler sans pour autant perdre l'information liée au nombre de répétitions de cette entrée de la table.

L'autre facteur apparaissant dans la complexité de la construction de la taxonomie est le nombre de comparaisons de dimensions. Ainsi, plus nous arriverons à réduire le nombre de dimensions à comparer plus la complexité de la construction diminuera. Nous avons montré qu'il est possible de réduire le nombre de dimensions à considérer en retirant les dimensions équivalentes, mais cette opération s'effectue une fois la taxonomie des dimensions calculée. Or, ici nous essayons de réduire le nombre de dimensions avant de construire la taxonomie. À l'heure actuelle, nous n'avons pas trouvé de méthode automatique performante pour cela, nous avons donc pris le parti de solliciter l'utilisateur. Nous nous sommes basé sur l'observation que bien souvent l'utilisateur est capable d'apporter

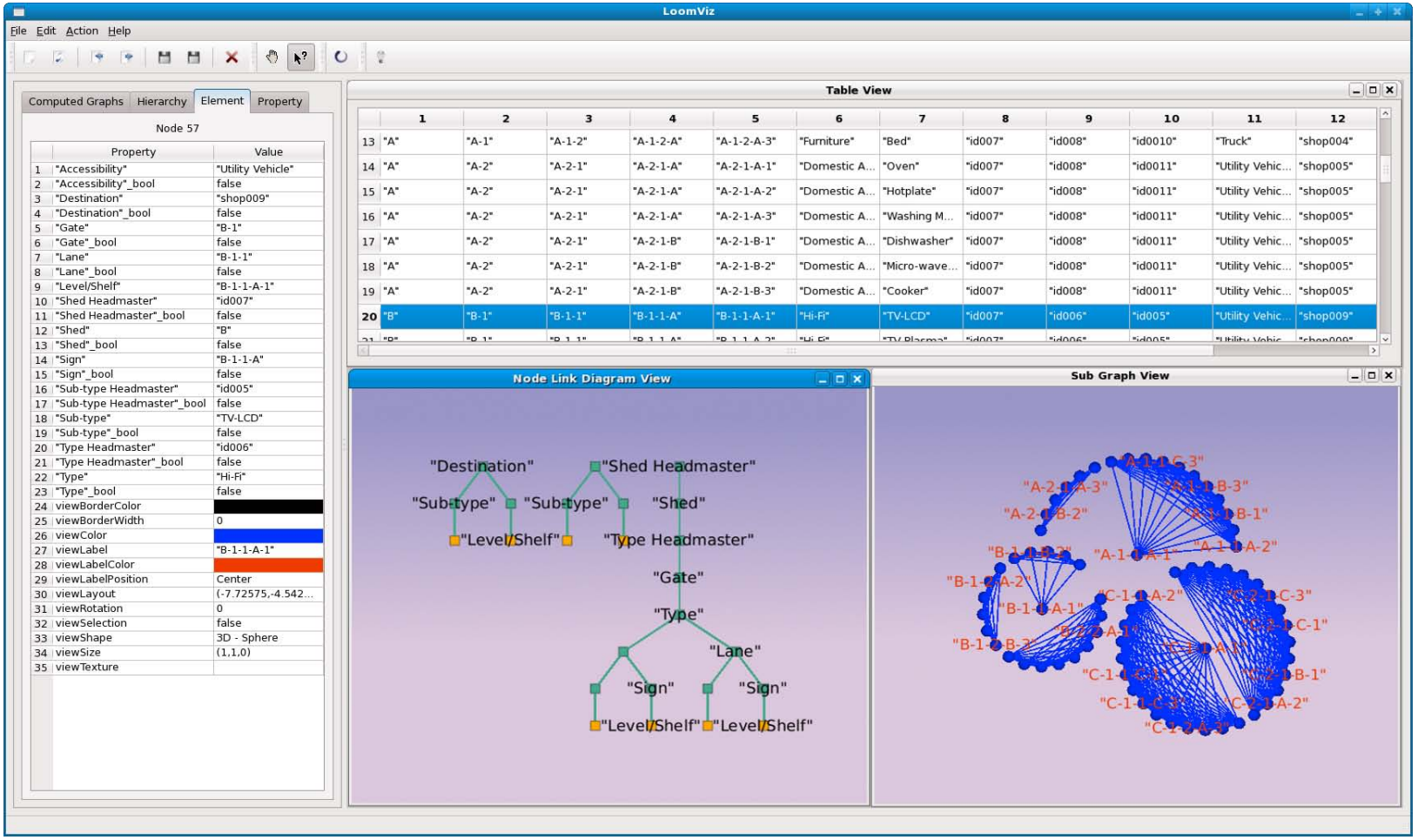
des informations supplémentaires sur les données de par l'expérience qu'il a vis à vis de ces données. Nous avons donc décidé de lui demander de sélectionner un sous-ensemble des dimensions de la table. Afin de savoir quelles sont les dimensions que l'utilisateur souhaite analyser, nous lui proposons une vue sous forme tabulaire des données. Dans cette vue, il est possible de cocher ou décocher certaines dimensions, par la suite seules les dimensions qui auront été cochées par l'utilisateur seront conservées pour la construction de la taxonomie. Le fait d'écarter certaines dimensions de la table peut aussi nous permettre de réduire à nouveau le nombre de lignes dans la table. En effet, si deux lignes de la tables diffèrent seulement par la valeur d'une dimension, et que l'utilisateur écarte cette dimension de sa sélection, alors conserver les deux lignes dans la table n'apportera pas plus d'informations pour construire la taxonomie. Au contraire comme nous l'avons souligné, cela aurait juste pour effet d'alourdir le calcul de la taxonomie.

5.7 Interactions possibles et exemple d'utilisation

Le jeu de données que nous avons utilisé pour élaborer et tester notre méthode et notre système rassemble des informations sur des produits stockés dans des entrepôts. Il s'agit d'un jeu de données que nous avons créé qui est composé de treize dimensions, certaines d'entre-elles servent à localiser les produits (”Shed”, ”Gate”, ”Lane”, ”Sign”, ”Level/Shelf”) d'autres servent à décrire les produits (”Type”, ”Sub-type”) et d'autres encore apportent des informations sur les personnes qui sont en charge de la gestion des produits (”Shed Headmaster”, ”Type Headmaster”, ”Sub-type Headmaster”). Il y a d'autres informations telles que ”Accessibility” qui définit par quel moyen les produits peuvent être extraits des entrepôts, et ”Destination” qui donne des informations sur les revendeurs qui ont stockés ces produits. Chaque ligne du jeu de données possède une valeur pour chacun de ces attributs.

La première étape de la méthode consiste à sélectionner dans la table les dimensions qui seront prises en compte pour la construction de la taxonomie et la génération des graphes. Étant donné que nous n'avons aucune connaissance particulière vis à vis de ce jeu de données, nous construisons la taxonomie qui utilise toutes les dimensions de la table. La taxonomie de dimensions que nous obtenons est celle de la Figure 5.4 et la taxonomie dépliée est la hiérarchie que l'on peut observer dans les Figure 5.4 et Figure 5.5.

Une fois la construction achevée, nous étudions la taxonomie et la hiérarchie afin de savoir quels sont les sous-graphes qu'il sera intéressant d'étudier. Toute d'abord, on peut voir que la taxonomie a généré une forêt (cf. Définition 2.39) de hiérarchies puisqu'il y a trois hiérarchies : une enracinée sur la dimension ”Destination”, une enracinée sur la dimension ”Accessibility” et une enracinée sur la dimension ”Shed Headmaster”. On peut constater aussi que celles enracinées par ”Destination” et ”Accessibility” sont très similaires. De ce fait, les sous-graphes qui seront générés à partir de ces deux hiérarchies seront similaires. Cela nous permet de conclure que nous n'aurons pas besoin d'étudier précisément la totalité des sous-graphes générés mais seulement la moité de ces sous-graphes (ceux correspondant à une des deux hiérarchies). D'autre part, la troisième partie de la hiérarchie possède une assez grande branche, qui nous indique que ces dimensions sont ordonnées. Nous savons aussi que les dimensions les plus basses fournissent des relations plus précises que celle des niveaux supérieurs, donc les dimensions ”Sub-type” et ”Sign” fournissent les relations les plus précises entres les entités, alors que la dimension ”Shed Headmaster” fournit des relations plus générales. Nous pouvons donc en conclure que le



	Property	Value
1	"Accessibility"	"Utility Vehicle"
2	"Accessibility"_bool	false
3	"Destination"	"shop009"
4	"Destination"_bool	false
5	"Gate"	"B-1"
6	"Gate"_bool	false
7	"Lane"	"B-1-1"
8	"Lane"_bool	false
9	"Level/Shelf"	"B-1-1-A-1"
10	"Shed Headmaster"	"id007"
11	"Shed Headmaster"_bool	false
12	"Shed"	"B"
13	"Shed"_bool	false
14	"Sign"	"B-1-1-A"
15	"Sign"_bool	false
16	"Sub-type Headmaster"	"id005"
17	"Sub-type Headmaster"_bool	false
18	"Sub-type"	"TV-LCD"
19	"Sub-type"_bool	false
20	"Type Headmaster"	"id006"
21	"Type Headmaster"_bool	false
22	"Type"	"Hi-Fi"
23	"Type"_bool	false
24	viewBorderColor	
25	viewBorderWidth	0
26	viewColor	
27	viewLabel	"B-1-1-A-1"
28	viewLabelColor	
29	viewLabelPosition	Center
30	viewLayout	(-7.72575,-4.542...
31	viewRotation	0
32	viewSelection	false
33	viewShape	3D - Sphere
34	viewSize	(1,1,0)
35	viewTexture	

	1	2	3	4	5	6	7	8	9	10	11	12
13	"A"	"A-1"	"A-1-2"	"A-1-2-A"	"A-1-2-A-3"	"Furniture"	"Bed"	"id007"	"id008"	"id0010"	"Truck"	"shop004"
14	"A"	"A-2"	"A-2-1"	"A-2-1-A"	"A-2-1-A-1"	"Domestic A..."	"Oven"	"id007"	"id008"	"id0011"	"Utility Vehic..."	"shop005"
15	"A"	"A-2"	"A-2-1"	"A-2-1-A"	"A-2-1-A-2"	"Domestic A..."	"Hotplate"	"id007"	"id008"	"id0011"	"Utility Vehic..."	"shop005"
16	"A"	"A-2"	"A-2-1"	"A-2-1-A"	"A-2-1-A-3"	"Domestic A..."	"Washing M..."	"id007"	"id008"	"id0011"	"Utility Vehic..."	"shop005"
17	"A"	"A-2"	"A-2-1"	"A-2-1-B"	"A-2-1-B-1"	"Domestic A..."	"Dishwasher"	"id007"	"id008"	"id0011"	"Utility Vehic..."	"shop005"
18	"A"	"A-2"	"A-2-1"	"A-2-1-B"	"A-2-1-B-2"	"Domestic A..."	"Micro-wave..."	"id007"	"id008"	"id0011"	"Utility Vehic..."	"shop005"
19	"A"	"A-2"	"A-2-1"	"A-2-1-B"	"A-2-1-B-3"	"Domestic A..."	"Cooker"	"id007"	"id008"	"id0011"	"Utility Vehic..."	"shop005"
20	"B"	"B-1"	"B-1-1"	"B-1-1-A"	"B-1-1-A-1"	"Hi-Fi"	"TV-LCD"	"id007"	"id006"	"id005"	"Utility Vehic..."	"shop009"

FIG. 5.5: Une vue globale du système. La fenêtre "Table View" affiche les données sous la forme d'une table. La fenêtre "Node Link Diagram" affiche la taxonomie dépliée (comme on peut le voir dans la Figure5.4). La fenêtre "Sub Graph View" affiche le sous-graphe correspondant au nœud sélectionné dans la hiérarchie. Sur la gauche, le table contenue dans l'onglet "Element" affiche les attributs du nœud sélectionné dans la fenêtre "Sub Graph View". La ligne correspondant à ce nœud est mise en surbrillance dans la fenêtre "Table View".

sous-graphe généré à partir de la dimension "Shed Headmaster" aura beaucoup d'arêtes mais ne fournira pas pour autant beaucoup d'informations, alors les sous-graphes générés à partir des dimensions "Sign" ou "Sub-type" auront moins d'arêtes mais sûrement plus d'informations.

La dernière chose que nous pouvons dire sur la taxonomie et la hiérarchie concerne les nœuds orange. Nous savons que ces nœuds représentent des dimensions qui peuvent être utilisées pour identifier les entités de la table, il n'est donc pas très utile d'étudier les graphes générés à partir ces dimensions. En effet si une dimension identifie les entités, cela signifie que les entités ont toutes des valeurs différentes dans cette dimension. Or, comme nous ajoutons des arêtes dans un graphe seulement si des entités ont une même valeur, les graphes générés ne posséderont pas d'arêtes.

Dans la Figure 5.5, la fenêtre "Sub Graph View" montre le graphe obtenu à partir de la dimension "Type" dimension. Dans ce graphe nous pouvons voir sept composantes connexes. Chaque composante connexe correspond à une valeur de la dimension "Type", donc chaque composante connexe est un groupe d'entités. Donc en étudiant les composantes connexes, on peut observer comment les dimensions partitionnent les entités. Dans le graphe correspondant à la dimensions "Type", il y a un groupe rassemblant les Fournitures, un autre les Appareils Ménagers, un les produits Hi-Fi, un les Ordinateurs, un autre pour les Médias, un pour le Bricolage et un dernier pour les Vêtements.

Maintenant, si on veut avoir une vue de tous les attributs d'une entité, il suffit de cliquer sur un nœud dans un sous-graphe. Comme on peut le constater dans la Figure 5.5, La ligne correspondant au nœud cliqué est mise en surbrillance dans la fenêtre intitulée "Table View". Les attributs du nœud sont aussi affichés dans l'onglet "Element" sur la gauche de l'interface graphique du système. Dans la Figure 5.5 le nœud identifié par "B-1-1-A-1" a été sélectionné. De ce fait, la ligne correspondante est mise en surbrillance dans la fenêtre "Table View" et tous ses attributs sont affichés dans l'onglet "Element".

En utilisant la hiérarchie des dimensions, il est possible de visualiser l'efficacité que procure la hiérarchisation des dimensions sur la génération des sous-graphes. En cliquant successivement sur les nœuds d'une branche de la hiérarchie, les sous-graphes correspondant seront affichés les uns après les autres. Et comme un graphe associé à une dimension est un sous-graphe du graphe associé à son nœud père dans la hiérarchie, en itérant sur les graphes nous pourrons constater comment les partitions d'entités évoluent avec la hiérarchie des dimensions. De plus, comme le positionnement des nœuds ne change pas d'un graphe à l'autre, il sera aisé de voir disparaitre des arêtes lors d'un exploration de haut en bas de la hiérarchie et d'en voir apparaitre lors d'une exploration de bas en haut.

5.8 Conclusion

Nous avons présenté une méthode pour générer de manière automatique des graphes à partir de données sous forme tabulaire. Cette méthode permet de faciliter la tache de transformation nécessaire à tout système d'Infovis basé sur les graphes. Nous décrivons un algorithme qui génère automatiquement une taxonomie de dimensions. En utilisant à la fois cette taxonomie et un outil d'exploration de graphe hiérarchique, nous sommes capables de fournir à l'utilisateur un système lui permettant d'explorer interactivement l'ensemble des transformations possibles de ses données. Nous mettons aussi en évidence l'utilité de notre solution au travers d'un cas d'étude.

Chapitre 6

Génération de graphes basée sur les interconnexions entre les dimensions

6.1 Motivation

Les motivations de ces travaux reposent sur les mêmes fondement que celles du Chapitre 5. À savoir, les facilité qu'ont les utilisateurs à collecter et stocker facilement des données, ainsi que les difficultés que représentent la transformation des données d'un format brut vers une structure de données utilisable par les systèmes de visualisation. C'est principalement cette étape de transformation qui représente une barrière empêchant l'accès des utilisateurs non experts à une riche variété de techniques de visualisation.

Une deuxième barrière est matérialisée par la multitude de transformations possible des données brutes. En effet si des données renferment une structure arborescente, il n'est pas exclu que d'autres structures soient présentes. Ces structures sont bien souvent dissimulées dans les données brutes. Il est alors primordial de fournir aux utilisateurs des méthodes pour transformer les données brutes en données utilisables par un système de visualisation. Il est aussi important que les données brutes soient analysées pour mettre en évidence toutes les structures qu'elles contiennent. Ainsi les utilisateurs novices pourront franchir les deux barrières qui se dressaient devant eux. Ils pourront alors aisément s'inscrire dans le processus d'analyse exploratoire. Grâce à l'analyse des données brutes qui aura été réalisée, ils seront même en mesure de dégager des hypothèses auxquelles ils n'avaient jamais pensé jusqu'alors. Il est donc indispensable de doter chaque système de visualisation d'un module capable d'analyser et de transformer les données brutes des utilisateurs en données utilisables par ce système.

Avant de développer des méthodes d'analyses, il est nécessaire de définir un format de données brutes. Il faut que ce format soit suffisamment générique et simple pour que les utilisateurs novices soient capables de présenter leurs données brutes dans ce format. Plutôt que de définir un format, nous avons décidé d'étudier le format le plus utilisé par les utilisateurs non experts et basé sur nos retours d'expériences avec des utilisateurs finaux, nous sommes arrivés à la conclusion que le format brut le plus couramment utilisé est sous forme tabulaire. Ainsi, c'est sous cette forme que les données initiales seront présentées dans l'algorithme que nous exposons dans cet article.

En utilisant comme données d'entrée une table, nous proposons un algorithme permettant de générer automatiquement des graphes valués à partir de n'importe quelle table.

En analysant le contenu de chaque dimension nous identifions les interconnexions entre celles-ci. Puis nous caractérisons les entités, les attributs et les relations possibles au sein des tables. Finalement, nous intégrons l'utilisateur dans le processus en lui proposant un ensemble de transformation valide. L'utilisateur a la possibilité de modifier ces transformations avant de générer les graphes correspondants.

Ce chapitre est structuré de la manière suivante : nous présentons tout d'abord le processus que suit notre méthode, puis nous en détaillions chaque étape avant d'en présenter la complexité et enfin nous concluons avec un cas d'étude.

6.2 Processus de la méthode

Dans cette méthode nous considérons, comme dans le Chapitre 5, que chaque cellule d'une table contient des données nominales. Ainsi, qu'une cellule contienne un mot, un nombre ou une date, l'information sera traitée comme une chaîne de caractères sans distinction. Par contre, nous changeons d'approche en ce qui concerne la globalité des données contenues dans une table. Au lieu de considérer chaque ligne comme une entité décrite par des attributs, nous considérons ici que les entités sont des valeurs contenues dans les lignes. Par exemple, la Table 6.1 contient des vols d'avions ayant une escale, donc plusieurs aéroports apparaissent dans chaque ligne de la table. Un utilisateur peut vouloir visualiser comment les aéroports sont inter-connectés. Ainsi le travail que nous réalisons ici se rapproche à la fois de celui réalisé dans Tableau Software [60] de par l'analyse combinatoire que l'on va mener sur les dimensions de la table et également de celui réalisé dans nodeXl [80] de par l'idée d'exploiter des listes d'arêtes. Toutefois, cette liste d'arêtes n'est pas clairement identifiée dans la table, et peut s'étendre sur plus de deux dimensions.

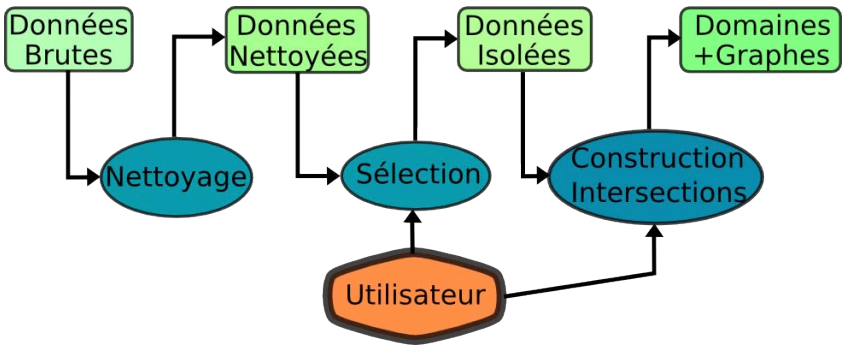


Fig. 6.1: Processus utilisé pour la détermination des intersections des dimensions d'une table. À partir des données brutes de la table, on obtient des domaines regroupant des entités qui à leur tour vont nous permettre de générer des graphes. Ce processus se décompose en trois étapes : le nettoyage, la sélection de données et la détermination des intersections des dimensions.

Si l'on se rapporte encore une fois au processus complet de visualisation présenté dans la Section 1.2, c'est à nouveau la première étape de ce processus qui va faire l'objet de toute notre attention. Comme nous l'avons fait dans le précédent algorithme, nous divisons cet algorithme en trois étapes comme présenté dans la Figure 6.1. Les deux premières sont identiques à celles du précédent algorithme (cf. Chapitre 5) et par conséquent non détaillées dans ce chapitre. La dernière étape, est celle où sont déterminés des intersections entre les valeurs des dimensions. C'est grâce à ces intersections que nous allons mettre en évidence

des domaines auxquels appartiennent les entités et où sont générés des graphes. Pour cela, dans cette étape un ensemble d'interactions avec l'utilisateur est mis en place.

Plus précisement, la détermination des domaines d'entités se décompose elle-même en plusieurs étapes, comme nous pouvons le voir dans l'Algorithme 6.1. Tout d'abord nous recherchons des intersections entre les dimensions de la table (Ligne 2), puis nous proposons les intersections trouvées à l'utilisateur afin que celui-ci puisse éventuellement les modifier (Ligne 3), et enfin une fois que l'utilisateur a validé les intersections et ses choix, nous générons les graphes entité-relation que l'utilisateur souhaite visualiser (Ligne 4 et 5).

Entrées : Une table $T_{n,m}$
Sorties : Un ensemble de domaine d'entités D, un ensemble de graphes EG
1 alphabets $\Sigma \leftarrow$ constructionAlphabet(T);
2 graphe_concept $G \leftarrow$ rechercheIntersectionDimension(Σ);
3 graphe_concept_etendu $GE \leftarrow$ validationUtilisateur(G);
4 $D \leftarrow$ constructionDomaines(GE);
5 $EG \leftarrow$ constructionGraphes(GE);

Algorithme 6.1 : Algorithme permettant de construire à partir d'une table des domaines d'entités ainsi que chaque graphe qui leur sont associés.

6.3 Recherche des intersections entre les dimensions

Nous présentons dans cette section la méthode de traitement se rapportant à la Ligne 2 de l'Algorithme 6.1. Celle-ci traite de la recherche des intersections entre les dimensions d'une table.

Afin de comparer les contenus des différentes dimensions et pour éviter d'avoir à parcourir à chaque fois toutes les valeurs d'une dimension, nous travaillons à partir des alphabets des dimensions. Pour cela, nous considérons que les valeurs d'une dimension forment un multi-ensemble (cf. Définition 2.2) et nous en extrayons un alphabet (cf. Définition 2.3).

La seconde étape consiste à déterminer s'il existe des intersections entre les dimensions. Il existe une intersection entre deux dimensions s'il existe au moins une valeur présente dans chacune des dimensions. Par exemple, les dimensions suivantes ont l'ensemble $\{A, B\}$ comme intersection : $Dim_1 = \{A, B, B, C, E, F\}$ et $Dim_2 = \{A, B, D, G, G, H\}$. En utilisant le fait que nous avons déjà construit les alphabets de chaque dimension, pour savoir s'il existe une intersection entre deux dimensions Dim_1, Dim_2, il suffit de construire un troisième alphabet $\Sigma_{Dim_1 \cup Dim_2}$ à partir de Σ_{Dim_1} et Σ_{Dim_2} les alphabets de chaque dimension. Puis grâce aux propriétés sur les ensembles, si on obtient que $|\Sigma_{Dim_1 \cup Dim_2}| < |\Sigma_{Dim_1}| + |\Sigma_{Dim_2}|$ (i.e. $\Sigma_{Dim_1} \cap \Sigma_{Dim_2} \neq \emptyset$) alors cela signifie qu'il existe une intersection entre les dimensions Dim_1, Dim_2.

Par exemple, si on souhaite déterminer s'il existe une intersection entre la dimension "Start" et la dimension "Stop" de la Table 6.1, il nous faut d'abord générer les alphabets de ces dimensions. On obtient ainsi : $\Sigma_{Start} = \{Berlin, London, NewYork\}$ et $\Sigma_{Stop} = \{London, Moscow, Paris\}$. Puis nous devons générer un troisième alphabet à partir des deux précédents, ce qui donne :
$\Sigma_{Start \cup Stop} = \{Berlin, London, Moscow, NewYork, Paris\}$.

Flight Id	Start	Stop	End
1	New York	Paris	Dublin
2	London	Moscow	Beijing
3	Berlin	London	Moscow

TAB. 6.1: Exemple d'une table contenant des informations sur des vols d'avions.

Et on peut vérifier qu'il existe bien une intersection entre ces deux dimensions puisque : $|\Sigma_{Start \cup Stop}| = 5 < |\Sigma_{Start}| + |\Sigma_{Stop}| = 3 + 3 = 6$.

En procédant par paire de dimensions, il n'est pas évident de détecter s'il existe des intersections entre plus de deux dimensions. Par exemple, considérons trois dimensions Dim_1, Dim_2, Dim_3, il est possible qu'il y ait une intersection entre Dim_1 et Dim_2, une autre entre Dim_2 et Dim_3, mais qu'il n'y en ait pas entre Dim_1 et Dim_3. Alors afin de pouvoir stocker les informations sur les intersections, nous avons mis en place un graphe de concepts (cf. Définition 2.22). Chaque dimension de la table est représentée dans le graphe par un sommet et à chaque fois que l'on trouve une intersection entre deux dimensions on ajoute une arête entre les deux sommets correspondant dans le graphe. Nous obtenons ainsi un graphe composé d'une ou plusieurs composantes connexes. Et nous sommes sûrs que toutes les dimensions appartenant à une même composante connexe possèdent des intersections. Par extension, toutes les valeurs contenues dans ces dimensions peuvent appartenir à un même domaine. Toutefois, certaines situations de polysémie peuvent engendrer des erreurs. Par exemple, le mot "échelle" a deux sens différents suivant son contexte. Il peut s'agir de l'objet de la vie quotidienne ou alors d'un moyen de mesure en géographie. Si un tel mot apparait dans deux dimensions d'une table sous deux sens différents, l'intersection qu'il créera n'aura pas réellement de sens.

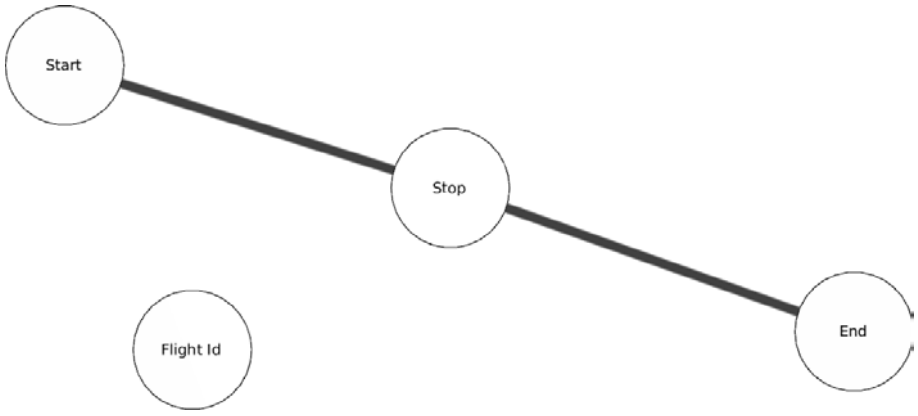


FIG. 6.2: Graphe de concepts de la Table 6.1. Chaque sommet représente une dimension de la table et les arêtes les intersections entre les dimensions.

Dans le cas de vols d'avions, chaque ligne de la table représente un vol caractérisé par les dimensions "Start", "End" et "Stop". Alors si nous trouvons des intersections entre ces dimensions, nous pouvons envisager que ces dimensions contiennent des valeurs appartenant à un même domaine, ici des aéroports. La Figure 6.2 représente le graphe de concepts de la Table 6.1.

6.4 Propositions des intersections et interactions de l'utilisateur

Maintenant que les intersections entre les dimensions ont été mises en évidence, nous allons les soumettre à l'utilisateur pour qu'il puisse éventuellement les éditer. Il s'agit là de la tâche effectuée à la Ligne 3 de l'Algorithme 6.1. Pour cela, il est intéressant que l'utilisateur intervienne, car il se peut qu'en fonction de la répartition des valeurs dans les différentes dimensions de la table, des intersections n'aient pas été détectées.

Par exemple, si on considère la Table 6.2, il n'y a une intersection qu'entre les dimensions "Stop" et "End". Pourtant, si l'utilisateur souhaite visualiser les interconnexions entre les aéroports de la table, il faut également prendre en compte la dimension "Start" afin de définir le domaine "Aéroport". Dans ce cas précis il est impossible de détecter l'inclusion de la dimension "Start" dans le domaine "Aéroport".

Flight Id	Start	Stop	End
1	New York	Paris	Berlin
2	Dublin	Moscow	Beijing
3	Madrid	London	Moscow

TAB. 6.2: Exemple de table regroupant des données sur des vols d'avions où il n'y a pas d'intersections de la dimension "Start" avec les autres dimensions de la table.

Il semble donc intéressant et indispensable pour construire des graphes, les plus précis possibles, de tirer profit des connaissances de l'utilisateur. Ainsi, si deux dimensions ne possèdent pas d'intersection alors que manifestement elles désignent des entités de même type, il est important que l'utilisateur puisse spécifier cela. C'est lors de cette étape qu'il pourra aussi corriger les erreurs provoquées par la polysémie.

Pour que l'utilisateur puisse visualiser et éditer les ensembles d'intersections, nous avons choisi de ne pas utiliser le graphe de concepts que nous avons construit lors de l'étape de recherches des intersections, car un tel graphe de concepts est complexe. Cette complexité réside dans le fait que pour modifier un domaine (une composante connexe dans le graphe de concepts) il faut que l'utilisateur manipule directement un graphe en ajoutant ou en retirant des arêtes. Nous avons alors pris en compte le fait que les données en entrée sont sous forme tabulaire, donc en conservant cette représentation le plus longtemps possible (tant que la tâche est solvable) nous évitons ainsi une surcharge cognitive.

C'est en ce basant sur cette notion de limitation de la surcharge cognitive que nous avons choisi de mettre en place une interface qui permet à l'utilisateur d'éditer les intersections précédemment trouvées. Cette interface est une table dont chaque colonne représente une dimension de la table d'origine. Les lignes de cette table représentent les ensembles d'intersections. Afin qu'on puisse visualiser dans une ligne l'ensemble qu'elle représente, nous avons doté chaque cellule de la table d'une boite cochable et chaque boite correspondant à une dimension qui apparait dans l'ensemble est cochée (cf. Figure 6.3).

Pour l'édition, l'utilisateur peut cocher ou décocher les boîtes qu'il souhaite. Ainsi il peut retirer des dimensions d'un ensemble d'intersections ou en rajouter. L'utilisateur peut aussi rajouter un nouvel ensemble d'intersections (domaine). Pour cela il lui suffit d'ajouter une ligne à la table et de cocher les dimensions qui vont définir un nouveau domaine (ensemble d'intersections). Il peut aussi nommer les différents domaines. Pour

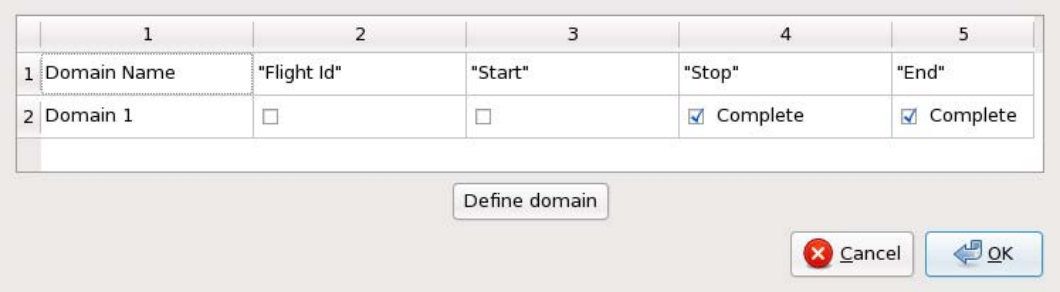


FIG. 6.3: Table permettant de visualiser les ensembles d'intersections des dimensions qui ont été trouvé lors de l'analyse de la Table 6.2. On peut voir qu'il n'y a pas d'intersection incluant la dimension "Start". Toutefois l'utilisateur va pouvoir corriger cela en cochant la boite correspondante.

cela, les cellules de la première colonne de l'interface sont éditables. Elles comportent un nom initial arbitraire de domaine que l'utilisateur peut modifier. Ce nom de domaine servira à identifier les futurs graphes construits.

Nous avons également mis en place une autre interaction qui va permettre à l'utilisateur d'agir sur la manière dont le graphe va être construit. En effet, suivant le type d'entités et le sens que l'utilisateur veut donner à sa modélisation, il est possible de distinguer deux types de construction.

Le premier peut être illustré par la construction d'un graphe du type "auteur-auteur". Considérons que la table que l'on utilise contienne des entrées bibliographiques. Chaque entrée est composée d'un identifiant de publication, du titre de la publication ainsi que des noms des auteurs. Lors de l'analyse de la table et peut-être aussi grâce à l'utilisateur, nous avons mis en évidence un domaine qui regroupe tous les auteurs se trouvant dans ces publications. Si on souhaite modéliser comment se structure la communauté des auteurs, nous allons construire un graphe dans lequel tout les auteurs qui ont publiés ensemble (apparaissent dans une même ligne de la table) sont connectés (reliés par une arête) les uns avec les autres. Ainsi un graphe complet (cf. Définition 2.40) sera créé entre tous les auteurs d'une même ligne.

Le second type de construction quant à lui peut être illustré par l'exemple des vols d'avions de la Table 6.1. Avec ce genre de table, il est tout à fait possible de construire un graphe de type "auteur-auteur" mais ici l'utilisateur peut ajouter bien plus d'informations pour la modélisation. En effet, il se trouve que les dimensions de la table sont identifiées par "Start", "Stop" et "End". On voit apparaître ici un ordre entre les dimensions, il serait intéressant de tirer parti de cette information car celle-ci réduirait considérablement le nombre d'arêtes dans le graphe et augmenterait d'autant sa clarté. De plus, un ordre sur ces dimensions retranscrirait bien la notion de trajet.

Afin de permettre à l'utilisateur de distinguer ces deux cas, nous lui permettons d'éditer le contenu des cellules contenant les boîtes cochables. Il peut y saisir une information : le terme "Complete" pour spécifier qu'il souhaite construire un graphe de type "auteur-auteur" ou un nombre correspondant à l'ordre dans lequel la dimension devra être prise en compte lors de la construction du graphe.

Si jamais une des cellules composant un domaine contient le mot clé "Complete", alors toutes les autres cellules qui décrivent le domaine doivent elles aussi comporter ce même

mot clé. Pour ce qui est de l'ordre sur les dimensions, il suffit de les numéroter de 1 à n où n est le nombre de dimensions composant le domaine.

FIG. 6.4: Interface (Table) permettant de visualiser les ensembles d'intersections des dimensions qui ont été trouvés lors de l'analyse de la Table6.1. On peut y voir que l'utilisateur a modifié le nom du domaine (cf. Figure 6.3) et a ajouté un ordre sur les dimensions qui composent le domaine des "Aéroports".

Une fois que l'utilisateur a apporté toutes les modifications qu'il souhaite aux intersections, nous construisons un nouveau graphe de concepts, que l'on appellera : "graphes de concepts étendu". Celui-ci va nous permettre de stocker les nouveaux domaines pour la suite de l'application de l'algorithme. Nous ajoutons aussi dans ce graphe les informations liées au type de construction qu'a choisi l'utilisateur pour chaque domaine que ce soit l'ordre des dimensions ou l'information "Complete". Pour cela nous stockons ces informations comme des attributs sur les sommets du graphe.

6.5 Génération des graphes entités-relation représentant les domaines

En utilisant le graphe de concepts étendu obtenu après que l'utilisateur a apporté des modifications, nous sommes en mesure de mettre en évidence des domaines (ensembles d'entités). Ces domaines sont identifiables grâce aux composantes connexes du graphe de concepts étendu. En effet chaque composante connexe représente un ensemble de dimensions possédant des intersections entres elles. S'il existe un tel ensemble, alors les valeurs contenues dans ces dimensions définissent un ensemble d'entités : un domaine. Donc pour chaque composante connexe (de taille supérieure à 1) du graphe de concepts étendu, nous sommes capable de créer un nouveau graphe ayant comme ensemble de sommets les valeurs contenues dans les dimensions de cette composante. Ce traitement correspond à la Ligne 4 de l'Algorithme 6.1. Maintenant, pour obtenir des graphes de types entité-relation, comme indiqué à la Ligne 5 de l'Algorithme 6.1 nous devons ajouter des arêtes à ce graphe.

Pour les ajouter, nous allons utiliser la structure globale de la table, ainsi que les informations ajoutées par l'utilisateur précédemment. Par définition, si des valeurs appartiennent à une même ligne de la table, alors elles décrivent toutes l'entité que représente cette ligne. On peut donc dire qu'elles sont en relation entre elles car elles définissent la ligne de la table. Par exemple, avec les vols d'avions, ce sont bien les aéroports qui définissent le vol mais réciproquement, chaque vol va définir une relation entre les aéroports qui le composent.

Donc, pour chaque ligne de la table nous allons ajouter une arête dans le graphe entre les valeurs des dimensions possédant des intersections. Mais nous allons le faire en respectant les informations ajoutées par l'utilisateur.

Par exemple avec la Table 6.1, le graphe des intersections de la Figure 6.2 et l'information "Complete" qu'aurait ajouté l'utilisateur, nous pouvons dire que le graphe que nous allons obtenir possédera une arête entre les sommets "New York" et "Paris", une seconde entre "New York" et "Dublin" ainsi qu'une troisième entre "Paris" et "Dublin". En appliquant cela à chacune des lignes de la Table 6.1, nous obtenons le graphe de la Figure 6.5 dans lequel on retrouve les interconnexions entre les aéroports que l'utilisateur souhaitait visualiser.

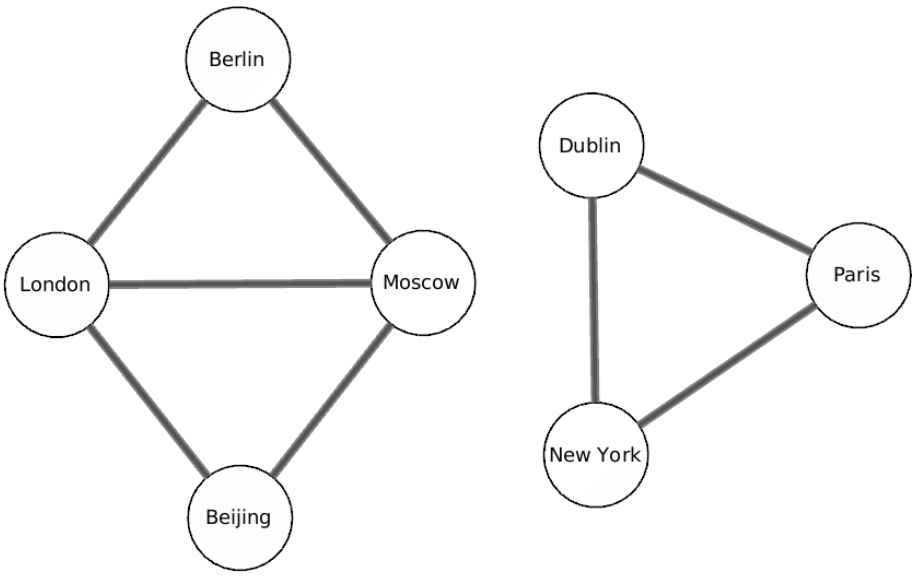


FIG. 6.5: Graphe des aéroports obtenu en tenant compte des intersections des dimensions de la Table 6.1 et en considérant que l'utilisateur a ajouté l'information "Complete".

D'un autre coté, si on considère toujours la Table 6.1, le graphe des intersections de la Figure 6.2 et les informations ajoutées par l'utilisateur dans la Figure 6.4 (l'ordre sur les dimensions), nous pouvons dire que le graphe que nous allons obtenir possédera une arête entre les sommets "New York" et "Paris" et seulement une seconde entre "Paris" et "Dublin". En appliquant cela à chacune des lignes de la Table 6.1, nous obtenons le graphe de la Figure 6.6 dans lequel on retrouve les interconnexions entre les aéroports en tenant compte des trajets.

6.6 Complexité

Notre algorithme permet à l'utilisateur d'analyser le contenu de ses données et d'en extraire des graphes entités-relation. Mais l'espace de toutes les combinaisons possibles entre les dimensions est quadratique. Il est donc important de considérer les complexités en temps que requiert notre méthode.

Nous exprimons ici les complexités en considérant que la table en entrée est composée de m dimensions et n lignes.

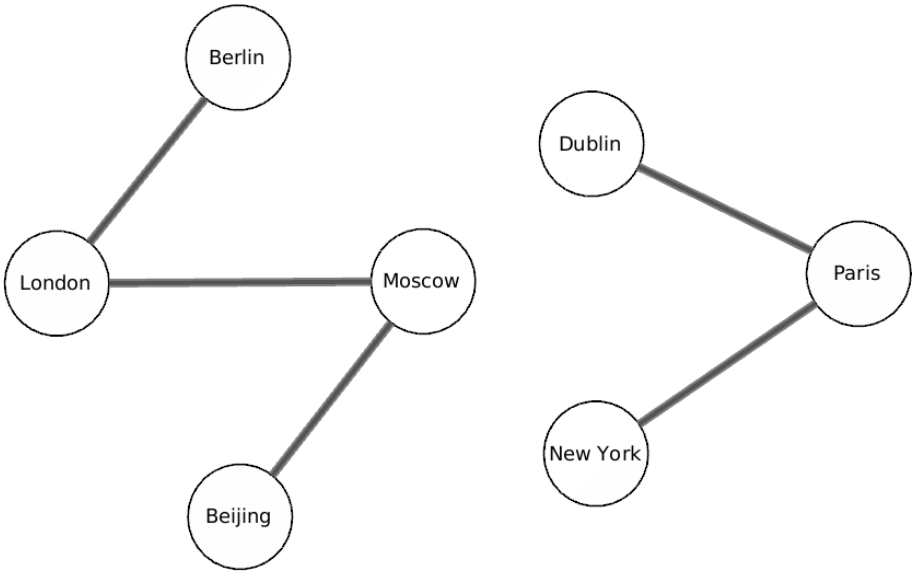


FIG. 6.6: Graphe des aéroports obtenu en tenant compte des intersections des dimensions de la Table 6.1 et en considérant l'ordre que l'utilisateur a ajouté sur les dimensions dans la Figure 6.4.

Proposition 6.1 *La complexité de l'Algorithme 6.1 est de l'ordre de* $\mathcal{O}(m^2 \times n \log n)$.

Preuve :

La construction de l'alphabet d'une dimension est en $\mathcal{O}(n \log n)$ en utilisant une structure ensembliste. Et par extension la construction de tous les alphabets est en $\mathcal{O}(m \times n \log n)$.

Pour ce qui est de la construction d'un alphabet servant à vérifier l'existence d'une intersection entre deux dimensions, on a besoin d'insérer au plus n valeurs d'un alphabet dans un autre. Donc, comme pour les alphabets associés aux dimensions de la table, cette construction est en $\mathcal{O}(n \log n)$. Mais dans une table de m dimensions il y a $m \times (m-1)$ comparaisons possibles deux à deux, on obtient donc une complexité globale en $\mathcal{O}(m(m-1) \times n \log n)$ pour la construction des intersections des dimensions. Mais cette dernière complexité considère que les alphabets initiaux de chaque dimension sont déjà construits, donc si on doit construire les premiers alphabets et les intersections, on obtient une complexité de $\mathcal{O}(m^2 \times n \log n)$.

La seconde complexité qu'il faut prendre en compte est celle de la construction d'un graphe correspondant à un ensemble d'intersections. Si cet ensemble s'étend sur l des m dimensions de la table, alors pour chaque ligne de la table on va devoir considérer chacune des l valeurs deux à deux. On obtient donc que la complexité est en $\mathcal{O}(n \times l(l-1))$.

Donc cette complexité est inférieure à celle de la construction des intersections. On peut donc dire que la complexité globale de notre méthode est de l'ordre de $\mathcal{O}(m^2 \times n \log n)$.

□

Dans la plupart des jeux de données, le nombre de dimensions est bien inférieur au nombre de lignes, donc nous pouvons dire que notre méthode est applicable en pratique malgré sa complexité quadratique.

6.7 Cas d'étude

Le jeu de données que nous utilisons ici est une extraction des publications de notre équipe de recherche à partir de la plate-forme "archive ouverte pluridisciplinaire HAL" (Hyper Article en Ligne). Nous avons extrait l'ensemble des publications réalisées par les membres de notre équipe depuis sa création en 2008. Les données de cette extraction ont été stockée dans une table. Cette table est composée de huit dimensions qui sont : "Identifiant", "Type de publication", "Titre de la publication", "Auteur 1", "Auteur 2", "Auteur 3", "Année" et "Titre ouvrage". Nous avons décidé de stocker les noms des auteurs sur trois dimensions pour simplifier la structure globale. Si jamais des publications ont moins de trois auteurs, alors nous codons dans les champs supplémentaires qu'il n'existe pas de valeurs pour ces champs. Ces champs ne seront alors pas considérés lors de l'analyse. Chaque ligne de la table décrit donc une publication.

	1	2	3	4	5	6	7	8
1	"Id"	"Published in "	"Title of the publication"	"Author 1"	"Author 2"	"Author 3"	"Year"	"Book titlie"
2	"inria-00472423"	"Journal"	"Animation, Small Multiple...	"Archambault Daniel"	"Purchase Helen "	"Pinaud Bruno"	2011	"IEEE Transactions ...
3	"inria-00516580"	"Conference"	"GVSR: an On-Line Guide f...	"Pinaud Bruno"	"Kuntz Pascale"	"UKN"	2010	"18th International ...
4	"inria-00514150"	"Conference"	"Difference Map Readabilit...	"Archambault Daniel"	"Purchase Helen "	"Pinaud Bruno"	2010	"18th International ...
5	"hal-00520906"	"Conference"	"From Databases to Graph...	"Gilbert Frederic"	"Auber David"	"UKN"	2010	"Information Visuali...
6	"hal-00495293"	"Conference"	"Living flows: enhanced e...	"Lambert Antoine"	"Auber David"	"Melancon Guy"	2010	"Information Visuali...
7	"hal-00495307"	"Conference"	"3D Edge Bundling for Geo...	"Lambert Antoine"	"Bourqui Romain"	"Auber David"	2010	"Information Visuali...
8	"hal-00495279"	"Conference"	"Winding Roads: Routing e...	"Lambert Antoine"	"Bourqui Romain"	"Auber David"	2010	"Computer Graphic...
9	"inria-00471432"	"Conference"	"The readability of Path-Pr...	"Archambault Daniel"	"Purchase Helen "	"Pinaud Bruno"	2010	"Eurovis 2010, 12th...
10	"inria-00449745"	"Conference"	"PORGY : reecriture et visu...	"Pinaud Bruno"	"Melancon Guy"	"UKN"	2010	"Extraction et Gesti...
11	"hal-00499430"	"Conference"	"Interactive Searching and...	"Koenig Pierre-Yves"	"Zaidi Faraz "	"Archambault ...	2010	"Proceedings of Gr...
12	"halshs-00466163"	"Journal"	"Ports in multi-level mariti...	"Ducruet Cesar"	"Rozenblat Celine"	"Zaidi Faraz "	2010	"Journal of Transpor...
13	"hal-00413602"	"Conference"	"Solving the Traffic and Flit...	"Simonetto Paolo"	"Koenig Pierre-Yves"	"Zaidi Faraz "	2009	"Proceedings of IEE...
14	"hal-00413602"	"Conference"	"Solving the Traffic and Flit...	"Archambault Daniel"	"Gilbert Frederic"	"Phan Quang T...	2009	"Proceedings of IEE...
15	"hal-00413602"	"Conference"	"Solving the Traffic and Flit...	"Mathiaut Morgan"	"Lambert Antoine"	"Dubois Jonath...	2009	"Proceedings of IEE...
16	"hal-00413602"	"Conference"	"Solving the Traffic and Flit...	"Sicre Ronan"	"Vieux Remi"	"Melancon Guy"	2009	"Proceedings of IEE...
17	"hal-00407269"	"Conference"	"Fully Automatic Visualisati...	"Simonetto Paolo"	"Auber David"	"Archambault ...	2009	"Computer Graphic...
18	"hal-00407218"	"Conference"	"An Heuristic for the Const...	"Simonetto Paolo"	"Auber David"	"UKN"	2009	"Information Visuali...
19	"hal-00425144"	"Conference"	"Revealing Hidden Commu...	"Zaidi Faraz "	"Sallaberry Arnaud"	"Melancon Guy"	2009	"Proceedings of the...

FIG. 6.7: Extrait de la table des publications.

La méthode que nous présentons dans cet article est implémentée à l'aide de Tulip [3]. Tulip est un logiciel libre qui fournit des outils et des composants graphiques pour la visualisations de grandes masses de données.

Lors de la première étape de notre algorithme, nous cherchons les intersections entre les dimensions de la table. Il est évident que notre méthode devrait pouvoir mettre en évidence des intersections entre les différentes dimensions regroupant les noms des auteurs des publications. Le résultat de la recherche des intersections est présenté dans la Figure 6.8, et celui-ci confirme bien les intersections attendues entre les dimensions "Auteur".

La seconde étape de notre algorithme est la génération des graphes correspondant aux intersections. Dans notre cas, étant donné qu'une seule intersection a été trouvée, un seul graphe sera généré. Le graphe que nous obtenons dans le cas des auteurs est celui de la Figure 6.9. Nous l'avons dessiné à l'aide d'un algorithme de dessin de force déjà implémenté dans Tulip.

Maintenant que nous avons importé et généré des graphes à partir de l'extraction de publication, nous pouvons entrer dans un processus d'analyse exploratoire. Ainsi si

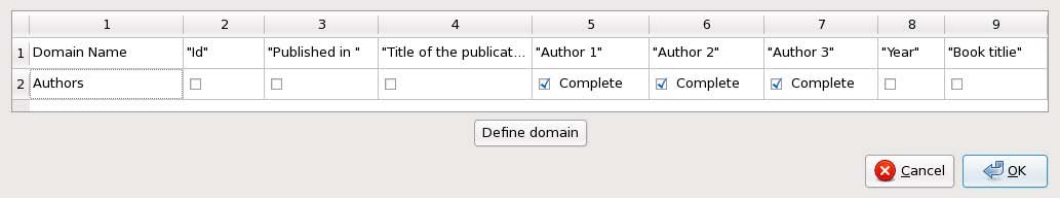


FIG. 6.8: Interface (Table) permettant de visualiser les ensembles d'intersections des dimensions qui ont été trouvés lors de l'analyse de la table regroupant les publications. On peut voir qu'il y a qu'une seule intersection incluant les dimensions "Auteur 1", "Auteur 2" et "Auteur 3".

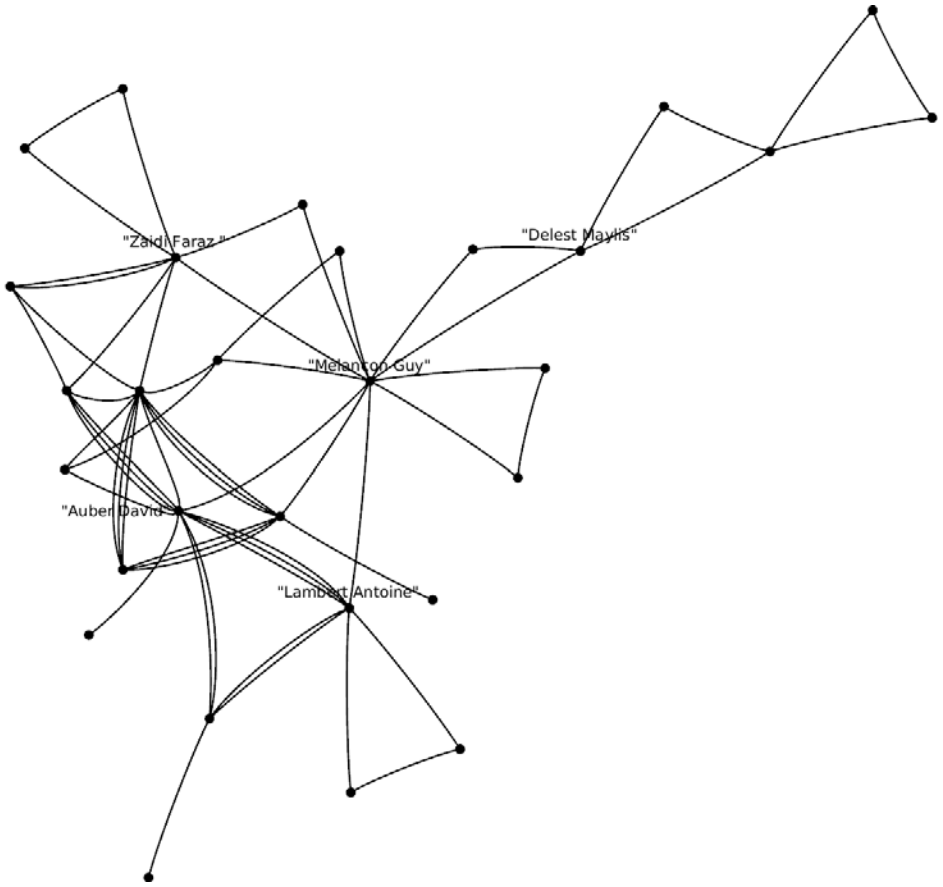


FIG. 6.9: Graphe entité-relation obtenu à partir des auteurs de la table de la Figure 6.7.

on observe le graphe obtenu, on peut voir qu'il existe des sommets particuliers qui sont des connecteurs dans le graphe. Par exemple, on peut voir dans la Figure 6.9 que "Delest Maylis" permet de connecter deux parties du graphe. On peut retrouver des rôles similaires pour "Zaidi Faraz" et "Lambert Antoine". Une autre hypothèse que ce graphe nous permet de faire est que "Auber David" et "Melancon Guy" ont des positions très centrales dans cette équipe.

6.8 Conclusion

Nous avons présenté dans ce chapitre un algorithme générant des graphes à partir d'un jeu de données sous forme tabulaire. Cet algorithme facilite la transformation des données afin de pouvoir les utiliser dans des systèmes de visualisation d'informations. Pour générer des graphes de type entité-relation nous tirons profit de l'expertise et des connaissances de l'utilisateur. Cet algorithme permet d'obtenir des graphes valués sur lesquels il est possible de procéder à une analyse exploratoire. Nous avons montré que la complexité de notre méthode reste applicable à de grands jeux de données. Nous avons aussi mis en évidence au travers d'un cas d'étude les possibilités et les facilités d'utilisation de notre algorithme.

Chapitre 7
Conclusion et perspectives

Nous avons abordés dans cette thèse deux problématiques de la visualisation d'informations. Tout d'abord, nous nous sommes intéressés au problème de l'analyse de réseaux sociaux dynamiques. Plusieurs travaux successifs sur le sujet nous ont amenés à développer notre propre système de visualisation et d'analyse de tels réseaux. Au cours de ces recherches, nous avons été confronté au grand nombre de formats existant pour stocker des données. La plupart du temps, les utilisateurs n'appartenant pas au domaine de la visualisation d'informations n'utilisent pas de format particulier pour stocker leurs données. Ils se contentent de les stocker sous forme de tables. C'est cette variété dans les données brutes qui nous a amené, dans un second temps à nous intéresser au problème de l'import de données dans le systèmes de visualisations et à la génération de graphes. Nous présentons ici les conclusion et perspectives de nos travaux menés sur ces deux axes.

7.1 Analyse de réseaux sociaux dynamiques

Les travaux que nous avons réalisés sur l'analyse de réseaux sociaux dynamiques nous ont amené à réaliser un système complet permettant de mener une analyse de bout en bout. Ce système se présente sous la forme d'une couche métier de la plateforme Tulip, ce qui la rend modifiable et personnalisable suivant les besoins de l'utilisateur. L'analyse de tels réseaux est décomposée en quatre étapes : la discrétisation du graphe, la décomposition de chacun des graphes obtenus, la détection de changements et la construction d'une hiérarchie d'influence. L'application de l'ensemble de ce processus permet de détecter les changements structuraux qui s'opèrent dans un réseau social au cours du temps. Afin de mettre en évidence ces changements, nous avons mis au point un algorithme capable de comparer deux décompositions de graphes. Ainsi, l'utilisateur est en mesure d'identifier une ou des périodes de stabilité dans le réseau, qui permettront de construire un hiérarchie d'influence du réseau. Pour construire cette hiérarchie, nous nous basons sur l'efficacité des communications au sein du réseau. Mais dans certains cas, comme par exemple les groupes terroristes, les acteurs du réseau font tout leur possible pour fausser cette efficacité et se dissimuler. Nous avons donc mis au point un second algorithme de construction de la hiérarchie afin de prendre en compte ces situations.

Nous avons mis à l'épreuve notre système sur deux jeux de données différents. Les résultats obtenus semblent satisfaisants puisque nous parvenons à détecter correctement les moments où des changements structuraux majeurs du réseau se produisent, ainsi qu'à construire une hiérarchie d'influence révélant les rôles des acteurs d"un réseau social. Mais

le système connait quelques limitations, dues au nombre élevé de graphes générés lors de l'étape de discrétisation. En effet, les performances du système dépendant fortement de l'algorithme de fragmentation utilisé, plus il y a de graphes à décomposer plus le temps de calcul sera long. Mais si on essaie de générer moins de graphes, on risque être confronté à une perte d'informations.

Les apports de cette thèse sur la problématique de l'analyse de réseaux sociaux dynamiques :

- La production d'un système complet permettant de mener de bout en bout une analyse de ces réseaux. [36]
- Un algorithme permettant de comparer des fragmentations d'un graphe. [17, 36]
- La construction d'une communauté de consensus visant à synthétiser les actions des acteurs du réseaux sur une certaine période de temps. [17, 36]
- Un algorithme permettant de construire une hiérarchie d'influence à partir d'une communauté de consensus. Ainsi qu'un mode alternatif de cette hiérarchie applicable dans le cas du contre-terrorisme. [16, 17, 36]

Perspectives :

Il y a plusieurs points que nous aimerions aborder afin d'améliorer le système (DysNAV) que nous avons proposé. À l'heure actuelle, l'étape de discrétisation découpe l'intégralité des données suivant des intervalles de temps non chevauchant. Si des changements structuraux surgissent au milieu d'un des ces intervalles, nous sommes capable de les détecter. Mais nous pouvons signaler uniquement l'intervalle de temps durant lequel le changement s'est produit et non pas l'instant exact où ils se sont produits. Il serait intéressant d'analyser les jeu de données en autorisant le fait d'avoir des intervalles de temps chevauchant. Cela permettrait de cibler plus précisément l'instant où les changement structuraux s'opèrent. Cela pourrait permettre aussi de ne pas occulter de changements. En effet, si un changement se produit à cheval sur la fin d'un intervalle et le début du suivant, nous ne pouvons pas garantir que nous le détecterons. Pour cela, nous envisageons de faire deux discrétisations pour une même valeur ϵ, une commençant à t_0 et l'autre à $t_0 + \epsilon/2$. Il faudra alors mettre au point un algorithme basé sur celui proposé dans le Chapitre 4, capable de comparer les fragmentations obtenues à partir des deux discrétisations afin de ne produire par fusion qu'une seule fragmentation contenant tout les changements.

Un autre point que nous aimerions perfectionner concerne l'analyse de la hiérarchie d'influence. Nous proposons seulement à l'utilisateur de l'observer et de l'explorer. Il serait intéressant d'ajouter des outils d'analyse égocentrique [30] qui permettraient d'avoir des informations plus précises sur un individu ou sur son rôle dans le réseau. Nous pourrions alors essayer de comparer les hiérarchies obtenues à partir de deux graphes consensus afin d'analyser l'évolution du rôle joué par un individu, ou un groupe d'individu.

Afin d'améliorer l'utilisation du système, les interactions de l'interface devront être améliorée et élargies. Pour cela, il serait intéressant de mener une évaluation sur l'utilisation du système. Par exemple, le choix d'un panneau de contrôle n'est peut être pas

le meilleur pour piloter une analyse. Ou encore, le nombre relativement élevé de fenêtres présentes simultanément à l'écran est peut être déroutant et source de confusion pour l'utilisateur. Il serait peut-être aussi intéressant d'utiliser d'autres algorithmes de dessin pour visualiser le réseau complet comme par exemple les principe de groupement d'arêtes en liasses [45] ce qui permettrait également d'ajouter des outils d'interactions pour mener une analyse, tels que une loupe grossissante,un *"fish eye"* ou un système de *"bring and go"* [55].

7.2 Import de données et génération de graphes

Dans le cadre des travaux que nous avons mené sur l'import de données et la génération de graphes, nous avons proposé deux méthodes de traitement visant à faciliter la tâche de transformation nécessaire à tout système de visualisation d'informations basé sur les graphes. La première génère une taxonomie des dimensions d'une table contenant des données. Cette taxonomie permet d'ordonner les dimensions de la table de manière à ce que l'utilisateur puisse déterminer quelles sont celles donnant des informations précises sur les entités et celles donnant des informations plus globales. À l'aide de cette taxonomie, nous sommes capables de générer plusieurs graphes qui grâce à un outil d'exploration permettront à l'utilisateur de parcourir l'ensemble des transformations possibles de ses données. Le temps de construction de cette taxonomie reste raisonnable au vue des tailles possibles que peuvent avoir les jeux de données. La seconde tire profit des connaissances que l'utilisateur a de ses données pour produire des graphes type entité-relation. Pour cela nous comparons les dimensions des tables entre elles, mais nous ne cherchons plus des différences pour les ordonner, nous cherchons des similitudes afin d'identifier des ensembles de dimensions susceptibles de contenir des domaines définissants des entités. Après avoir sollicité l'utilisateur afin qu'il valide ou modifie ces domaines, nous construisons un graphe pour chaque domaine. Par la suite, l'utilisateur pourra procéder à une analyse exploratoire de ces graphes. Le temps de construction des domaines reste tout aussi raisonnable que celui de la méthode précédente.

Les apports de cette thèse sur la problématique de l'import de données et la génération de graphes dans les systèmes de visualisation d'informations :

- La mise au point de mécanismes permettant de manipuler des données. [35, 37]
- La construction d'une taxonomie de dimensions qui amène à un ordre sur les dimensions d'une table. Cet ordre définissant la précision de l'information apportée par les attributs. [37]
- La génération de graphe à l'aide de la taxonomie. Ainsi qu'un mécanisme d'interaction permettant de naviguer simplement de graphe en graphe. [37]
- La détection automatique de domaines regroupant des entités. Ainsi qu'un mécanisme d'interactions permettant de les éditer afin de paramétrer la construction de graphes. [35]

Perspectives :

Le principal point sur lequel nous aimerions travailler à l'avenir est la combinaison des deux méthodes présentées dans les Chapitres 5 et 6. L'idée serait de produire des graphes valués profitant des apports des deux méthodes. Grâce à la méthode du Chapitre 6, nous permettons à l'utilisateur de manipuler ses données afin de produire des graphes. D'un autre coté, grâce à la taxonomie des dimensions présentée dans le Chapitre 5, nous sommes en mesure d'ordonner les dimensions de la table afin de définir un grain sur la précision des informations des dimensions. Il est alors envisageable de valuer les graphes produits en fonction du niveau de détails sélectionné dans la taxonomie par l'utilisateur.

Un autre point que nous aimerions perfectionner concerne la construction des graphes à partir des intersections de dimensions. Nous voudrions mettre en place des options de construction supplémentaires. À l'heure actuelle, l'utilisateur peut construire des graphes complets à l'aide de du mot clé *"complete"* ou alors définir un ordre sur les dimensions pour par exemple retranscrire une notion de trajet. Pourquoi ne pas lui proposer d'utiliser à la fois les deux situations ? Il pourrait définir un ordre sur certaines dimensions du domaine, et faire en sorte que les autres soient connectées à toutes comme dans le cas des graphes complets. Par exemple, dans le cas de routages dans les réseaux, il est possible qu'il existe un chemin $(A, e_{AB}, B, e_{BC}, C, e_{CD}, D)$ (cf. Définition2.31), et un sommet E pouvant aller directement vers les sommets B, C et D. Le fait de rendre possible utilisation des deux modes de construction simultanément, permet de générer des graphes à partir d'une table modélisant des routages contenant ce type de configurations.

Le dernier point que nous aimerions développer, concerne également la construction des graphes à partir des intersections de dimensions. Lorsque l'utilisateur construit un graphe en utilisant le mot clef *"complete"*, cela a pour effet de de produire une clique entre tous les sommets. Si la table en entrée contient un grand nombre de lignes, alors le graphe contiendra un grand nombre de clique. Par conséquent, le nombre d'arêtes sera d'autant pus grand et risque de compliquer l'analyse exploratoire. Nous envisageons deux solutions pour pallier ce problème. La première consiste à proposer à l'utilisateur de construire un hypergraphe [10, 11], lequel peut être visualisé efficacement grâce aux travaux de Simonetto sur les ensembles chevauchant [77, 79]. Dans ce cas, on n'utiliserait plus d'arêtes, ce qui supprimerait le problème. Mais alors il faudra se demander si une représentation en ensembles chevauchants ne requiert pas trop de temps d'apprentissage pour que l'utilisateur puisse s'en servir efficacement. La seconde solution serait d'utiliser des algorithmes de dessin utilisant les principes de regroupement des arêtes en liasses, comme nous l'avons proposé dans le cas des réseaux sociaux dynamiques. Ainsi, plusieurs arêtes ne seraient plus représentées que par un faisceau d'arêtes.

Publications

Analyse de réseaux sociaux

- Detecting structural changes and command hierarchies in dynamic social networks
  Bourqui R., Gilbert F., Simonetto P., Zaidi F., Sharan U., Jourdan F.
  Dans Proc. of the 2009 Int. Conf. on Advances in Social Network Analysis and Data Mining - Advances in Social Network Analysis and Data Mining, Grèce (2009)

- Communities and hierarchical structures in dynamic social networks : analysis and visualization
  Gilbert F., Simonetto P., Zaidi F., Jourdan F., Bourqui R.
  Social Network Analysis and Mining (2010)

Génération de graphes

- From Databases to Graph Visualization
  Gilbert F., Auber D.
  Dans Information Visualization - 2010 14th International Conference Information Visualisation, Royaume-Uni (2010)

- Import automatique et interactif de données dans les systèmes de visualisations
  Gilbert F., Auber D.
  11ème Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances EGC 2011 (2011)

Concours Vast

- VAST 2008 Challenge : Social network dynamics using cell phone call patterns
  Bourqui R., Zaidi F., Gilbert F., Sharan U., Simonetto P.
  IEEE Symposium on Visual Analytics Science and Technology, États-Unis d'Amérique (2008)

- Solving the Traffic and Flitter Challenges with Tulip
  Simonetto P., Koenig P.-Y., Zaidi F., Archambault D., Gilbert F., Phan Quang T. T., Mathiaut M., Lambert A., Dubois J., Sicre R. et al
  Dans Proceedings of IEEE Symposium on Vast 2009 - IEEE Symposium on Visual Analytics Science and Technology 2009, États-Unis d'Amérique (2009)

Théorie des graphes

- Recursively arbitrarily vertex-decomposable graphs
  F. Gilbert O. Baudon, M. Wozniak
  Discrete Mathematics, special issue 8th French Conference on Combinatorics (2010).

- Recursively arbitrarily vertex-decomposable suns
  Olivier Baudon, Frédéric Gilbert, Mariusz Woźniak
  Opuscula Math. 31/4 (2011), 533-547

Bibliographie

[1] Richard M. Adler. A dynamic social network software platform for counter-terrorism decision support. In *ISI*, pages 47–54. IEEE, 2007.

[2] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *In VLDB*, pages 81–92, 2003.

[3] D. Auber. Tulip : A huge graph visualisation framework. In P. Mutzel and M. Jünger, editors, *Graph Drawing Softwares*, Mathematics and Visualization, pages 105–126. Springer-Verlag, 2003.

[4] D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon. Multiscale Visualization of Small-World Networks. In S. C. North and T. Munzner, editors, *Proc. of IEEE Information Visualization Symposium*, pages 75–81, Seattle, USA, 2003. IEEE Computer Press.

[5] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439) :509–512, October 1999.

[6] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi : An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, 2009.

[7] Michael Baur, Marc Benkert, Ulrik Brandes, Sabine Cornelsen, Marco Gaertler, Boris Köpf, Jürgen Lerner, and Dorothea Wagner. visone - software for visual social network analysis. In *Proc. 9th Intl. Symp. Graph Drawing (GD '01),LNCS 2265*, pages 463–464. Springer-Verlag, 2002.

[8] Skye Bender-deMoll and Daniel A. McFarland. The art and science of dynamic network visualization. *JoSS : Journal of Social Structure*, Volume 7, 2005.

[9] M. Berg, O. Cheong, and M. Kreveld. *Computational geometry : algorithms and applications*. Springer, 2008.

[10] Claude Berge. *Graphes and Hypergraphes*. Monographies Universitaires de Mathématiques, n° 37. Dunod, 1970.

[11] Claude Berge. *Graphs and Hypergraphs*, volume 6. North-Holland Mathematical Library. Translated by Edward Minieka, 1973.

[12] Pavel Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[13] Jacques Bertin. *Sémiologie Graphique. Les diagrammes, les réseaux, les cartes*. EHESS, 3e édition, (1e éd. Gauthier-Villars 1967, 2e éd. 1973) 1999.

[14] Mustafa Bilgic, Louis Licamele, Lise Getoor, and Ben Shneiderman. D-dupe : An interactive tool for entity resolution in social networks. In *Graph Drawing*, pages 505–507, 2005.

[15] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling : Theory and Applications*. Springer, 2005.

[16] R. Bourqui, F. Gilbert, U. Sharan, P. Simonetto, and F. Zaidi. Social Network Dynamics using Cellphone Call Patterns. In *IEEE VisWeek VAST Challenge 2008*, 2008.

[17] Romain Bourqui, Frédéric Gilbert, Paolo Simonetto, Faraz Zaidi, Umang Sharan, and Fabien Jourdan. Detecting structural changes and command hierarchies in dynamic social networks. In *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining*, pages 83–88, 2009.

[18] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25 :163–177, 2001.

[19] Sylvain Brohée and Jacques van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7(1) :488, 2006.

[20] Christoph Buchheim, Michael Jünger, and Sebastian Leipert. Improving walker's algorithm to run in linear time. In *GD '02 : Revised Papers from the 10th International Symposium on Graph Drawing*, pages 344–353, London, UK, 2002. Springer-Verlag.

[21] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[22] Yves Chiricota, Fabien Jourdan, and Guy Melançon. Software components capture using graph clustering. In *IWPC 11th International Workshop on Program Comprehension*, pages 217–226. IEEE Computer Society, 2003.

[23] J. S. Coleman. *An Introduction to Mathematical Sociology*. Collier-Macmillan, London, UK, 1964.

[24] Jana Diesner, Terrill L. Frantz, and Kathleen M. Carley. Communication networks from the enron email corpus "it's always about the people. enron is no different". In *Comput. Math. Organ. Theory*, volume 11, pages 201–228, Hingham, MA, USA, 2005. Kluwer Academic Publishers.

[25] Reinhard Diestel. *Graph Theory*. Springer, 2010.

[26] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

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

[28] César Ducruet, Céline Rozenblat, and Faraz Zaidi. Ports in multi-level maritime networks : evidence from the atlantic (1996–2006). *Journal of Transport Geography*, 18(4) :508–518, 2010.

[29] S. Eubank, H. Guclu, V. Kumar, M. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429 :180–184, 2004.

[30] Danyel Fisher. Using egocentric networks to understand communication. *IEEE Internet Computing*, 9 :20–28, 2005.

[31] L. C. Freeman. *The Development of Social Network Analysis : A Study in the Sociology of Science*. Empirical Press, 2004.

[32] L.C Freeman. Visualizing social networks. *Journal of Social Structure*, 1(1), 2000.

[33] Michael Friendly. Extending mosaic displays : Marginal, conditional, and partial views of categorical data. *Journal Of Computational And Graphical Statistics*, 8(3) :373–395, 1999.

[34] P. Gajer and S. G. Kobourov. GRIP : Graph dRawing with Intelligent Placement. In *Proc. Graph Drawing 2000 (GD'00)*, pages 222–228, 2000.

[35] Frédéric Gilbert and David Auber. Import automatique et interactif de données dans les systèmes de visualisations. *11ème Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances EGC 2011*, pages 491–502, January 2011.

[36] Frédéric Gilbert, Paolo Simonetto, Faraz Zaidi, Fabien Jourdan, and Romain Bourqui. Communities and hierarchical structures in dynamic social networks : analysis and visualization. *Social Network Analysis and Mining*, pages 83–95, September 2010.

[37] Frédéric Gilbert and David Auber. From Databases to Graph Visualization. In *Information Visualization 2010 14th International Conference Information Visualisation*, pages 128–133, 2010.

[38] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99 :8271–8276, 2002.

[39] Peter A. Gloor, Rob Laubacher, Yan Zhao, and Scott B.C. Dynes. Temporal visualization and analysis of social networks. In *NAACSOS Conference, June 27 - 29, Pittsburgh PA, North American Association for Computational Social and Organizational Science*, page 27–29, 2004.

[40] C.M. Grinstead and J.L. Snell. *Introduction to probability*. American Mathematical Society, 1997.

[41] Stefan Hachul and Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. *Graph Drawing*, pages 285–295, 2005.

[42] Stefan Hachul and Michael Jünger. Large-graph layout with the fast multipole multilevel method. Technical report, Zentrum für Angewandte Informatik Köln, 2005.

[43] J. A. Hartigan and B. Kleiner. Mosaics for contingency tables. In *Computer Science and Statistics : Proceedings of the 13th Symposium on the Interface*. New York : Springer-Verlag, 1981.

[44] Jeffrey Heer and Danah Boyd. Vizster : Visualizing online social networks. In *INFOVIS '05 : Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, pages 32–39. IEEE Computer Society, 2005.

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

[46] Human-Computer Interaction Lab, University of Maryland. Visual Analytics Benchmark Repository. http://hcil.cs.umd.edu/localphp/hcil/vast/archive/viewbm.php.

[47] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : a review. *ACM Comput. Surv.*, 31(3) :264–323, 1999.

[48] T.J. Jankun-Kelly and Kwan-Liu Ma. Moiregraphs : Radial focus+context visualization and interaction for graphs with visual nodes. In Tamara Munzner and Stephen North, editors, *Proceedings of the 2003 IEEE Symposium on Information Visualization*, pages 59–66. IEEE Computer Society TCVG, IEEE Computer Society Press, 2003.

[49] Brian Johnson and Ben Shneiderman. Tree-maps : a space-filling approach to the visualization of hierarchical information structures. In *VIS '91 : Proceedings of the 2nd conference on Visualization '91*, pages 284–291, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[50] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[51] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *SSTD*, pages 364–381, 2005.

[52] Hyunmo Kang, Lise Getoor, and Lisa Singh. Visual analysis of dynamic group membership in temporal social networks. *SIGKDD Explor. Newsl.*, 9(2) :13–21, 2007.

[53] M. Kretzschmar and M. Morris. Measures of concurrency in networks and the spread of infectious disease. *Math. Biosci.*, 133 :165–195, 1996.

[54] J. B. Kruskal. On the shortest spanning subtree and the traveling salesman problem. In *Proc. of the American Mathematical Society*, pages 48–50, 1956.

[55] Antoine Lambert, David Auber, and Guy Mélançon. Living flows : enhanced exploration of edge-bundled graphs based on GPU-intensive edge rendering. In *Proceedings of the 14th International Conference on Information Visualization (IV'10)*, pages 523–530, 2010.

[56] Landauer. *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates, 2007.

[57] V. Latora and M. Marchiori. How Science of Complex Networks can help in developing Strategy against Terrorism. *Chaos, Solitons and Fractals*, 20 :69–75, 2004.

[58] Michael Ley. The dblp computer science bibliography. http://www.informatik.uni-trier.de/~ley/db/.

[59] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymäki, and K. Wenger. Devise : Integrated querying and visual exploration of large datasets. In *In Proceedings of ACM SIGMOD*, pages 301–312, 1997.

[60] Jock D. Mackinlay, Pat Hanrahan, and Chris Stolte. Show me : Automatic presentation for visual analysis. volume 13, pages 1137–1144, 2007.

[61] Yoshiharu Maeno and Yukio Ohsawa. Analysing covert social network foundation behind terrorism disaster. *Int. J. of Services Sciences*, 2 :125–141, 2009.

[62] N. Memon, D. L. Hicks, and H. L. Larsen. How investigative data mining can help intelligence agencies to discover dependence of nodes in terrorist networks. In *ADMA '07 : Proc. of the 3rd int. conf. on Advanced Data Mining and Applications*, pages 430–441. Springer-Verlag, 2007.

[63] N. Memon and H. L. Larsen. Practical approaches for analysis, visualization and destabilizing terrorist networks. In *ARES 06 : Proc. of the First Int. Conf. on Availability, Reliability and Security*, pages 906–913. IEEE Computer Society, 2006.

[64] Nasrullah Memon, David L. Hicks, Henrik Legind Larsen, and Muhammad Aslam Uqaili. Understanding the structure of terrorist networks. *Int. J. of Business Intelligence and Data Mining*, 2 :401–425, 2007.

[65] Charles Minard. Carte figurative des pertes successives en hommes de l'armée française dans la campagne de russie 1812-1813, 1869. http://fr.wikipedia.org/wiki/Carte_figurative.

[66] James Moody, Daniel Mcfarland, and Skye Benderdemoll. Dynamic network visualization. *American Journal of Sociology*, 110(4) :1206–1241, 2005.

[67] M. E. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Phys Rev E Stat Nonlin Soft Matter Phys*, 64(1 Pt 2), July 2001.

[68] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys Rev E Stat Nonlin Soft Matter Phys*, 69(2 Pt 2), 2004.

[69] Andreas Noack. Energy models for drawing clustered small-world graphs. Technical report, 2003.

[70] Anatol Rapoport and William J. Horvath. A study of a large sociogram. *Behavioral Science*, 6(4) :279–291, 1961.

[71] Purnamrita Sarkar and Andrew W. Moore. Dynamic social network analysis using latent space models. *SIGKDD Explor. Newsl.*, 7(2) :31–40, 2005.

[72] Satu E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1) :27–64, 2007.

[73] Visual Analytics Science and Technology. IEEE VAST 2008 CHALLENGE. 2008. http://www.cs.umd.edu/hcil/VASTchallenge08/.

[74] John P. Scott. *Social Network Analysis : A Handbook*. SAGE Publications, 2000.

[75] Zeqian Shen, Kwan-Liu Ma, and Tina Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 12(6) :1427–1439, 2006.

[76] Ben Shneiderman. The eyes have it : A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pages 336–. IEEE Computer Society, 1996.

[77] Paolo Simonetto and David Auber. Visualise undrawable euler diagrams. In *Proceedings of the 12th International Conference on Information Visualisation (IV08)*, pages 594–599, 2008.

[78] Paolo Simonetto and David Auber. An heuristic for the construction of intersection graphs. In *Proceedings of the 13th International Conference on Information Visualisation (IV09)*, pages 673–678, 2009.

[79] Paolo Simonetto, David Auber, and Daniel Archambault. Fully automatic visualisation of overlapping sets. *Computer Graphics Forum*, 28(3) :967–974, 2009.

[80] M. Smith, B. Shneiderman, N. Milic-Frayling, E. Mendes Rodrigues, V. Barash, C. Dunne, T. Capone, A. Perer, and E. Gleave. Analyzing (social media) networks with nodexl. In *C&T '09 : Proceedings of the fourth international conference on Communities and technologies*, pages 255–264, New York, NY, USA, 2009. ACM.

[81] John Snow. 1854 broad street cholera outbreak, 1854. http://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak.

[82] Robert Spence. *Information Visualization : Design for Interaction.* Pearson - Prentice Hall (2nd edition), 2007.

[83] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris : a system for query, analysis, and visualization of multidimensional databases. *Commun. ACM*, 51(11) :75–84, 2008.

[84] Roberto Tamassia. *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications) - in preparation ; manuscript available at http ://www.cs.brown.edu/˜rt/gdhandbook.* Chapman & Hall/CRC, 2008.

[85] New York Times. Obama's 2011 budget proposal : How it's spent, 2010. http://www.nytimes.com/interactive/2010/02/01/us/budget.html.

[86] R. C. Tryon. *Cluster analysis.* Edwards Brothers, Ann Arbor, Michigan, 1939.

[87] Edward R. Tufte. *The Visual Display of Quantitive Information.* Graphics Press, 1983.

[88] Edward R. Tufte. *Envisionning Information.* Graphics Press, 1990.

[89] Edward R. Tufte. *Visual Explanations.* Graphics Press, 1997.

[90] F. B. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon. Manyeyes : a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1121–1128, November 2007.

[91] Jean Villerd, Sylvie Ranwez, Michel Crampes, David Carteret, and Jean Michel Penalva. Using concept lattices for visual navigation assistance in large databases : Application to a pantent database. In *CLA*, pages 405–425, 2007.

[92] V.Martinez, G.Simari, A.Silva, and V.S.Subrahmanian. The soma terror organization portal (stop) : Social network and analytic tools for the real-time analysis of terror groups. In *First Intl. Workshop on Social Computing, Behavioral Modeling and Prediction*, pages 9–18, 2008.

[93] C. Ware. *Information Visualization : Perception for Design.* Morgan Kaufmann Publishers, 2000.

[94] Stanley Wasserman and Katherine Faust. *Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences).* Cambridge University Press, 1995.

[95] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393 :440–442, 1998.

[96] Christopher C. Yang and Tobun D. Ng. Terrorism and crime related weblog social network : Link, content analysis and information visualization. In *ISI*, pages 55–58. IEEE, 2007.

[97] Faraz Zaidi. *Analysis, Structure and Organization of Complex Networks*. PhD thesis, Université Bordeaux 1, 2010.

[98] Faraz Zaidi, Arnaud Sallaberry, and Guy Melançon. Revealing hidden community structures and identifying bridges in complex networks : An application to analyzing contents of web pages for browsing. In *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Web Intelligence (WI'09)*, pages 198–205, 2009.

Méthodes et modèles pour la visualisation de grandes masses de données multidimensionnelles nominatives dynamiques

Résumé : La visualisation d'informations est un domaine qui connait un réel intérêt depuis une dizaine d'année. Dernièrement, avec l'explosion des moyens de communication, l'analyse de réseaux sociaux fait l'objet de nombreux travaux de recherches. Nous présentons dans cette thèse des travaux sur l'analyse de réseaux sociaux dynamiques, c'est à dire que nous prenons en compte l'aspect temporel des données. Nous nous sommes particulièrement intéressé à la mise en évidence des communautés dans les réseaux et à leurs évolutions dans le temps. Nous présentons également un algorithme permettant de construire une hiérarchie d'influence qui identifie le rôle occupé par les individus au sein du réseau.

Le second axe de recherche abordé dans cette thèse traite de l'obstacle que représente la diversité des formats de stockage de données. Cette diversité complique grandement l'import de données dans les systèmes de visualisation par des utilisateurs novices. Nous proposons dans cette thèse deux méthodes permettant de manipuler des données dans le but de produire des visualisations de type nœud-lien : une basée sur la construction d'une taxonomie des dimensions des données, l'autre sur la mise en évidence de domaines d'entités. Ces méthodes mettent l'accent sur les interactions avec l'utilisateur, afin de tirer profit de ses connaissances sur les données.

Mots-clef : Analyse de données, Génération de graphes, Analyse de réseaux sociaux dynamiques.

Discipline : Informatique

Methods and model for huge amount of nominative multidimendionnal dynamic data visualization

Abstract : Since ten years, informations visualization domain knows a real interest. Recently, with the growing of communications, the research on social networks analysis becomes strongly active. In this thesis, we present results on dynamic social networks analysis. That means that we take into account the temporal aspect of data. We were particularly interested in communities extraction within networks and their evolutions through time. We present an algorithm building an influence hierarchy which identifies the roles of the actors within the network.

The second area of research approached in this thesis deals with the variety of data storage formats. This variety spoils the import data for non expert users. In this thesis, we propose two methods allowing to manipulate data in order to produce node-link diagram visualizations. The first one is based on the construction of a dimensions taxonomy of the data. The second one tries to discover entities domains. These methods focus on user's interactions, to take avantages from his data knowledge.

Keywords : Data analysis, Graph generation, Dynamic social networks analysis.

Field : Computer Science