Visualisation de graphes

Rapport scientifique présenté en vue de l'obtention de
l'Habilitation à Diriger des Recherches

M.	Jean-Daniel Fekete	Directeur de Recherche	**Rapporteur**
Mme	Pascale Kuntz-Cosperec	Professeur	**Rapporteur**
M.	Michael McGuffin	Professeur	**Rapporteur**

David Auber

15 octobre 2012

Table des matières

Chapitre 1

Introduction

Les travaux résumés dans ce rapport scientifique ont pour thème la visualisation d'informations. Plus précisément, la création et l'utilisation de décompositions (chevauchantes ou non) de sommets ou d'arêtes.

Après une introduction des différents concepts liés à mes recherches, je présente les axes de recherches que j'ai développés. Ces différentes recherches ont fait l'objet de publications dans des journaux et conférences internationales. Ce rapport est basé sur une sélection de 13 articles que nous avons publiés. Je les ai spécifiquement choisi car ils sont le résultat de quatre thèses que j'ai co-encadrées. Le lecteur peut se référer à ces articles pour connaître tous les détails qui sont omis dans dans ce rapport.

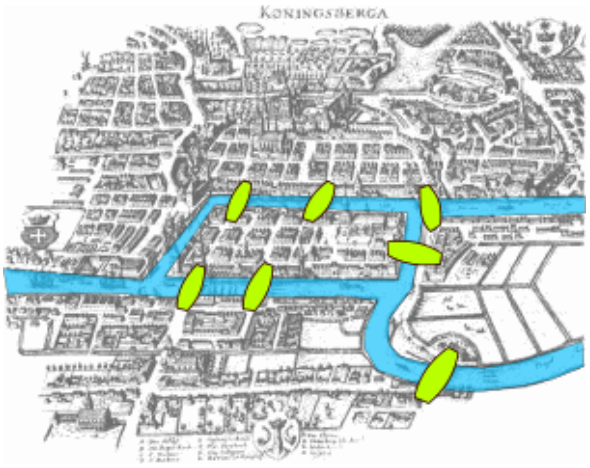


FIGURE 1.1 – La ville de Königsberg est construite autour de deux îles reliées entre elles par un pont. Six autres ponts relient la ville à l'une ou l'autre des deux îles. Le problème consiste à déterminer s'il existe un chemin permettant de passer une et une seule fois par chaque pont et de revenir à son point de départ. Le point de départ est libre de choix.

Précurseurs

Le graphe en tant qu'objet mathématique existe depuis de nombreuses années. Les premières recherches sur le sujet datent de 1736. Dans ses travaux, Euler a mis en place les bases de la théorie des graphes. Il a utilisé les graphes pour modéliser les problèmes d'optimisation ou de faisabilité. Les graphes lui ont permis de résoudre le problème des sept ponts de Königsberg. Ce problème consiste à trouver un chemin dans la ville qui passe une seule fois par chacun des sept ponts. La figure 1.1 montre une représentation visuelle de ce problème.

Au dix-neuvième siècle, on trouve des exemples d'utilisation des graphes à des fins de modélisation de la connaissance. La figure 1.2 montre deux cartes figuratives réalisées par Charles Joseph Minard. Ces cartes illustrent, en utilisant la métaphore visuelle nœuds/liens, les informations sur la campagne de Napoléon de 1812 pour la première et les flux migratoires pour la seconde.

En terme de visualisation d'informations, les travaux de Minard sont précurseurs. Ils ont été une source d'inspiration dans mes travaux de recherches. Dans le chapitre 5, nous présentons nos résultats de recherche portant sur l'*Edge Bundling*[1]. Ces résultats permettent de reproduire

1. Dans le document j'utilise beaucoup de terme anglais volontairement, je les indique en italique. L'objectif est de ne pas ajouter de la confusion. Une traduction du terme *edge bundling* peut être : regroupement en faisceau d'arêtes.

certaines des caractéristiques des cartes figuratives de Minard. Arriver à reproduire automatiquement de telles représentations graphiques reste encore un problème ouvert.

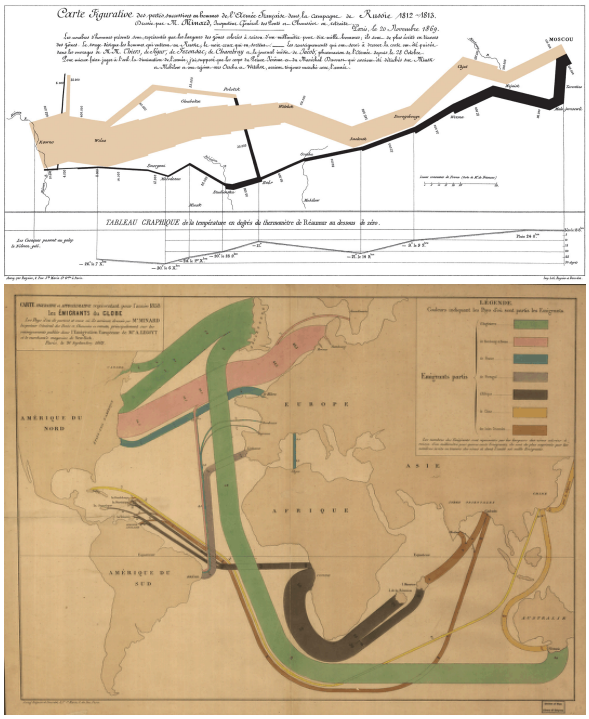


FIGURE 1.2 – (**En haut**) Carte publiée en 1869 par Charles Minard. Elle figure les pertes successives en hommes de l'armée française dans la campagne de Russie 1812-1813. (**En bas**) Carte publiée en 1862 par Charles Minard. Elle représente approximativement les flux migratoires de l'année 1858.

Graphes planaires

Il faut attendre les travaux de Tutte [94] en 1963, pour que les propriétés mathématiques des graphes soient utilisées pour automatiser leurs représentations. Dans ces travaux, Tutte a démontré que les graphes planaires intérieurement triangulés peuvent être dessinés automatiquement sans croisement d'arêtes dans le plan. Le principe de l'algorithme est de placer les sommets de la face extérieure du graphe sur un polygone simple. Ensuite, tous les sommets intérieurs sont déplacés au barycentre de leurs voisins jusqu'à ce que le système soit à l'équilibre [2]. La méthode converge vers le dessin désiré en temps polynomial. Ces travaux sont à l'origine de nombreuses recherches portant sur la représentation des graphes planaires [55, 30, 18].

Les recherches dans le domaine du dessin de graphes ont été portées par l'explosion de l'électronique à la fin du vingtième siècle. La raison est que les graphes planaires permettent de modéliser les circuits imprimés (*VLSI* [3]). De nombreux algorithmes et résultats théoriques ont été proposés sur ce sujet [32]. Ils permettent pour ces classes de graphes d'obtenir des bornes théoriques d'espace nécessaire pour la représentation des graphes, mais aussi de produire des représentations en temps linéaire.

Les travaux de Hopcroft et Tarjan [63] sur la détection et le calcul de plongement d'un graphe planaire ainsi que les travaux de De Fraysseix [30] sur le dessin de graphes planaires sont incontournables. C'est la combinaison de leurs résultats qui a permis de générer les premiers dessins de graphe planaires triangulés maximaux en temps linéaire. Une grande partie des publications sur ce sujet peut être trouvée dans les actes de la conférence *Graph Drawing*.

Bien que peu utilisée en visualisation d'informations, la théorie du dessin des graphes planaires est au centre de nombreuses de mes recherches. Le problème majeur de son application sur des problèmes réels est que les graphes rencontrés sont rarement planaires. La figure 1.3 montre le résultat obtenu par un algorithme de dessin [72] sur le réseau métabolique d'une levure. Certaines parties de ce dessin sont réalisées en utilisant des méthodes de représentation de graphes planaires. Cet exemple illustre notre approche qui consiste à décomposer les réseaux et à utiliser de manière ciblée les algorithmes adéquats en fonction des caractéristiques topologiques des clusters rencontrés. Dans les chapitres 2 et 4, je présente plusieurs utilisations de la théorie du dessin de graphes planaire pour obtenir des représentations visuelles pertinentes.

2. C'est la version triviale de l'algorithme, une méthode plus rapide consiste à résoudre directement le système d'équations linéaires.

3. Very-large-scale integration

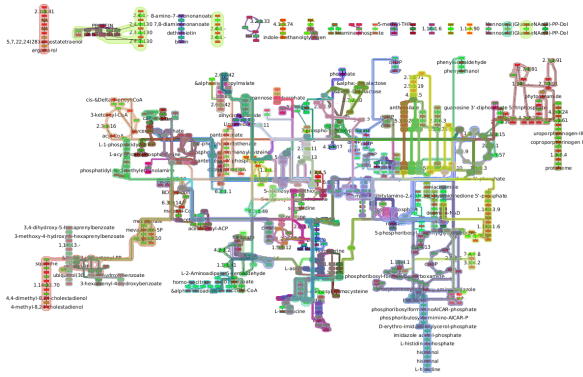


FIGURE 1.3 – Dessin automatique du réseau métabolique d'une levure obtenue avec la méthode de Lambert et al. [72]. Dans [23], la décomposition en voies métaboliques permet de construire un graphe planaire qui est ensuite dessiné avec l'algorithme de dessin planaire Mixed Model [55]

Graphes Généraux

A la fin du XXème siècle, l'automatisation de la représentation des structures relationnelles (hiérarchies, réseaux sociaux, réseaux métaboliques, etc.) explosent et de nombreux algorithmes apparaissent. Le plus célèbre et le plus utilisé de nos jours est l'algorithme de dessin de graphe par modèle de forces proposé par Eades [39].

Dans sa version basique, le principe de cet algorithme est de considérer les sommets du graphe comme des particules qui se repoussent entre elles et les arêtes du graphe comme des ressorts qui attirent les sommets entre eux. Une fois le système physique construit, l'idée est de simuler le système afin d'atteindre un état d'équilibre. En utilisant des décompositions multi-échelles des graphes ainsi que des approximations pour le calcul de forces de répulsion, les versions les plus modernes de cet algorithme permettent son utilisation sur des grands graphes. L'algorithme FM^3 [58] est un algorithme de choix, il a été le premier algorithme de ce type dont la complexité théorique en $O(n{\cdot}log(n))$ est prouvée. Pour une comparaison des approches multi-échelles pour le dessin par modèles de forces, on peut se reporter aux travaux de Hachul et al. [56].

Le plébiscite de cette approche est lié à de nombreux facteurs. Premièrement, dans sa version basique, elle est assez simple à mettre en place. Deuxièmement, les représentations sont visuellement plaisantes et troisièmement il permet de mettre en évidence les clusters d'un réseau (dans les cas simples). La figure 1.4 montre un dessin obtenu par ce genre d'algorithme sur le réseau de routeurs de Cheswick et al. [27]. La figure 1.7 montre la mise en évidence de clusters denses avec cette approche.

Dans le chapitre 2, je montre comment l'utilisation de décomposition de graphes peut permettre d'améliorer la lisibilité des représentations produites ainsi que le temps de calcul de celles-ci.

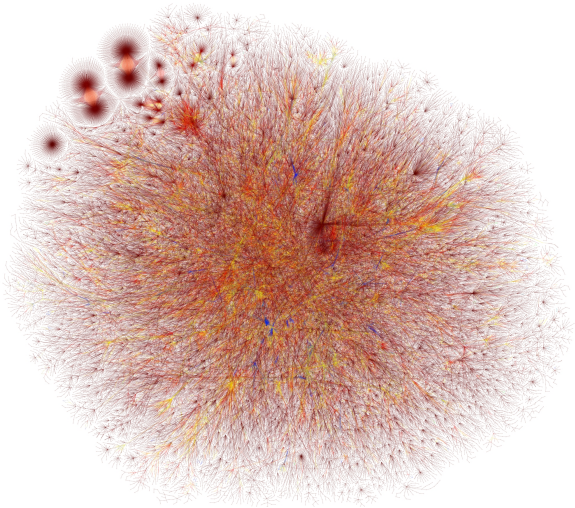


FIGURE 1.4 – Graphe représentant les interconnexions de routeurs internet à l'échelle planétaire. Ce graphe est issu de l'*Internet Mapping Project* [27]. L'objectif de ce projet était d'étudier la structure de ce graphe pour mieux comprendre les problèmes de routage et de trouver une parade contre les attaques de routeurs. Le dessin du graphe a été obtenu automatiquement en utilisant l'algorithme LGL [3]. Ce graphe contient 190384 sommets et 228354 arêtes.

Données relationnelles

L'augmentation significative de la quantité de données relationnelles a également favorisé les recherches sur les techniques de représentation et d'analyse de réseaux. Le développement des réseaux sociaux et l'augmentation du nombre de

documents au format *HTML*[4] sont deux exemples parmi d'autres. La figure 1.5 montre un exemple de jeu de données relationnelles que nous sommes amenés à analyser. Ce jeu de données représente les utilisateurs (> 650 millions) de Facebook ainsi que les relations qui existent entre eux. La détection de communautés ainsi que l'analyse de la diffusion d'informations dans ce type de réseaux sont des challenges auxquels nous sommes confrontés.

Un autre point important à signaler est la démocratisation de la notion de réseau dans le grand public. De nos jours, tout le monde connaît les concepts élémentaires de réseau et un grand nombre de personnes désirent utiliser les méthodes de visualisation d'informations et d'analyse de réseaux. Par exemple, les géographes [4] utilisent maintenant les méthodes d'analyse de réseaux pour déterminer les stratégies utilisées par les entreprises ou pour identifier les partenariats entre les villes (pays, etc.). Nos recherches en collaboration avec cette communauté d'utilisateurs nous ont permis, par exemple, d'étudier et de visualiser la topologie des réseaux aériens mondiaux.

Pour permettre l'analyse de grands corpus, les approches multi-échelles sont incontournables. Plus précisément, il faut être capable de créer des niveaux d'abstractions successifs pour permettre aux utilisateurs d'explorer interactivement les données. Par exemple, dans la figure 1.5 les auteurs ont indirectement créé une représentation multi-échelles. Les membres de Facebook qui sont placés sur le même pixel à l'écran sont implicitement agrégés et les auteurs utilisent la saturation de la couleur pour montrer le nombre d'éléments dans un agrégat. On peut se référer aux travaux de Elmqvist et Fekete [75] pour avoir une vue d'ensemble des techniques d'agrégations existantes.

Les besoins des analystes ont aussi révélé des lacunes dans les approches de partitionnement classiques. Dans les données réelles que nous récoltons, les clusters ou groupes ne sont pas clairement définis. Par exemple, dans les réseaux sociaux, les communautés partagent de nombreux membres. Pour obtenir des résultats pertinents en termes d'analyse, il faut alors utiliser des ensembles chevauchants[5].

4. HyperText Markup Language

5. Dans la suite, pour conserver la même notation que dans mes articles, j'utilise les termes *Overlapping Sets* ou bien *Fuzzy Clustering* pour dire ensemble chevauchants

Le problème est de représenter correctement ces communautés. Pour attaquer ce problème, les représentations noeuds/liens ne sont pas adaptées. Peu de résultats existent sur ce point précis qui est pourtant un challenge pour la visualisation d'informations dans les années à venir. Henri et Dwyer [79] ont proposé une méthode interactive pour visualiser ce genre de chevauchements. Les travaux de Stapleton et Rodgers [81] partagent de nombreuses similitudes avec les nôtres, je les détaille dans le chapitre 4.

La majorité des recherches développées dans les chapitres suivants portent sur la création d'agrégats (de sommets ou d'arêtes), sur les moyens de les représenter et sur leur exploration interactive. Dans le chapitre 4 nous présentons les résultats que nous avons obtenus sur la visualisation de décompositions chevauchantes.

FIGURE 1.5 – Dessin des interconnections entre les 650 millions d'utilisateurs de Facebook. Le placement des sommets est obtenu par géo localisation. La saturation de la couleur permet de percevoir le nombre d'entités et de relations affichées sur le même pixel. Les auteurs agrègent implicitement les entités proches géographiquement.

GPU vs CPU

Une autre évolution majeure est l'augmentation des capacités graphiques de nos ordinateurs et de l'augmentation de la puissance des processeurs. L'arrivé des processeurs graphiques programmables a considérablement étendu le champ des possibilités en visualisation d'informations. De nom-

breuses techniques réservées aux images statiques ou à de petits jeux de données sont maintenant utilisables sur de grand jeux de données. Ces nouvelles possibilités ont rendu obsolète une grande partie des méthodes de visualisation non interactives.

La figure 1.6 montre l'évolution de la puissance des GPU et des CPU au cours des dix dernières années. L'augmentation très rapide des performances des GPU comparée à celle des CPU laisse à penser qu'il faut privilégier les algorithmes fonctionnant sur le GPU. Cependant, la progression des CPU parallèles, et bientôt massivement parallèles, pourrait changer ce point de vue. Une certitude demeure, les nouvelles machines sont et seront de plus en plus parallèles. Ceci implique qu'il faut favoriser les méthodes compatibles avec le calcul distribué sur CPU et/ou GPU.

De nombreux algorithmes ont déjà été étudiés pour permettre leur passage sur ces nouvelles architectures multi-CPU ou multi-GPU. Par exemple, Yaniv Frishman [47] a présenté une implémentation multi-échelles d'un algorithme de dessin par modèle de forces sur le GPU. La rapidité de son implémentation permet ainsi de dessiner en temps réel de grands graphes et par la même occasion de permettre le dessin de graphes dynamiques. En 2007, nous avons proposé et expérimenté une implémentation sur le GPU d'un algorithme par modèle de forces [12]. Avec une approche naïve, nous obtenions des gains de performance de l'ordre de 10^2.

Cependant, il reste encore de nombreuses possibilités non explorées. Le réel challenge ne consiste pas à adapter les méthodes existantes à ces nouveaux matériels. Il faut plutôt repenser les visualisations ainsi que les interactions en prenant en compte le spectre complet des nouvelles possibilités qui nous sont offertes.

Dans le chapitre 3, nous présentons nos résultats sur la navigation interactive dans les grands réseaux et dans le chapitre 5, nous montrons les avantages de l'utilisation des GPU pour la génération des visualisations.

Organisation des recherches

Les cinq points développés précédemment ont été la source d'inspiration de la majorité des recherches que nous avons entreprises et que je résume dans ce manuscrit. Pour résumer, au cours de ces dix dernières années, notre objectif a été d'utiliser et d'améliorer les résultats de la théorie du dessin de graphes pour la visualisation d'informations. Nous avons aussi porté un intérêt tout particulier à l'utilisation et à l'amélioration des derniers résultats disponibles en infographie afin de permettre le passage à l'échelle de nos méthodes. Bien que notre objectif premier était et reste la modélisation des réseaux par des diagrammes noeuds/liens, les problèmes nouveaux liés aux communautés chevauchantes nous ont amenés à considérer d'autres métaphores visuelles pour modéliser les relations entre entités.

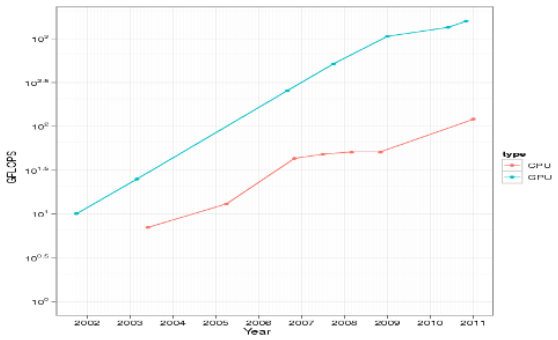


FIGURE 1.6 – Evolution de la puissance des unités graphiques et des CPU au cours des dix dernières années.

Mes travaux antérieurs portaient sur la visualisation d'informations. Cependant, le point de départ des recherches que je développe ici est le papier **"Multiscale visualization of Small World Networks"** [11] que nous avons publié en 2003. Dans cet article, nous avons posé les bases d'une méthode d'analyse de réseaux.

Cette méthode consiste à générer, à l'aide d'algorithmes de partitionnement de graphes, une décomposition multi-échelles. Nous obtenons ainsi un arbre de partitionnement de nos graphes. Nous pouvons ensuite utiliser cet arbre pour proposer différents niveaux d'abstractions à l'utilisateur et permettre à celui-ci de naviguer interactivement à différents niveaux de granularité. La figure 1.7 montre un exemple d'application de cette méthode sur un réseau d'acteurs de cinéma. Les détails techniques de cette méthode sont disponibles dans notre article [11].

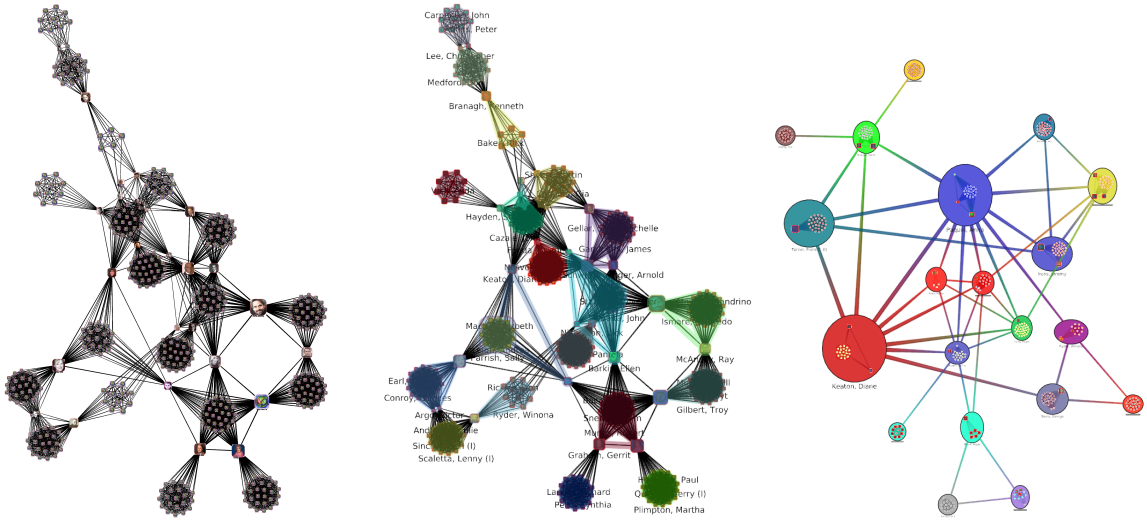


FIGURE 1.7 – Graphe extrait de l'*Internet Movie Data Base (IMDB)*. Dans ce graphe, les sommets sont des acteurs de cinéma et deux acteurs sont connectés si et seulement si ils ont joué ensemble. (**A gauche**) Dessin obtenu en utilisant l'algorithme de dessin de graphe GEM [46]. (**Au milieu**) Utilisation d'un algorithme de clustering pour rechercher récursivement des clusters dans le graphe de départ. Les enveloppes de couleur représentent le premier niveau de la hiérarchie détecté. (**A droite**) Les clusters sont récursivement agrégés en méta-sommets pour créer une représentation abstraite des données. L'utilisateur peut alors interactivement entrer ou sortir d'un méta-sommet. Cette méthode permet ainsi une navigation multi-échelles dans les données.

L'aboutissement de ces recherches n'est évidemment pas le fruit d'un travail solitaire. J'ai co-encadré 5 thèses qui ont permis de développer ces idées. La figure 1.8 donne une vue schématique du déroulement temporelle de ces thèses.

Pour approfondir le travail sur l'utilisation des décompositions pour la visualisation interactive de graphes j'ai co-encadré la thèse de **Daniel Archambault**. L'objectif de cette thèse a été de mettre en place une approche plus générale que la précédente [11] permettant d'intervertir les algorithmes de fragmentations ainsi que les algorithmes de dessin utilisés. Nous avons pour cela utilisé les caractéristiques topologiques des graphes. Les chapitres 2 et 3 présentent certains résultats de ces travaux. Cette thèse a été réalisée en collaboration avec Tamara Munzner de l'université de British Columbia. Notre objectif a été d'utiliser les compétences du LaBRI en termes d'analyse et de dessin de graphe et celle de l'université de Colombie Britannique en termes de design et d'infographie pour mettre en place des solutions efficaces sur des données concrètes.

En parallèle de la thèse de Daniel Archambault, j'ai co-encadré la thèse de **Romain Bourqui** avec Maylis Delest. Cette thèse avait pour but d'appliquer les algorithmes de dessins planaires à la visualisation de structures biologiques. Dans ce manuscrit, je ne détaille pas les aspects bio-informatiques liés à ces travaux. Je me contente de présenter dans les chapitres 2 et 3 les résultats liés à la visualisation par décomposition (des sommets ou des arêtes) ce qui est le thème de ce rapport de recherche.

Durant la thèse de Romain Bourqui, nous avons été confrontés au problème de visualisation de décompositions chevauchantes. Dans les réseaux métaboliques, on s'aperçoit que les voies métaboliques forment des clusters et ces clusters se chevauchent [6]. Sur cette problématique, j'ai co-encadré avec Guy Melançon la thèse de **Paolo Simonetto**. Cette thèse nous a permis de met-

6. Par, se chevauchent, j'entends : Deux clusters peuvent partager des éléments.

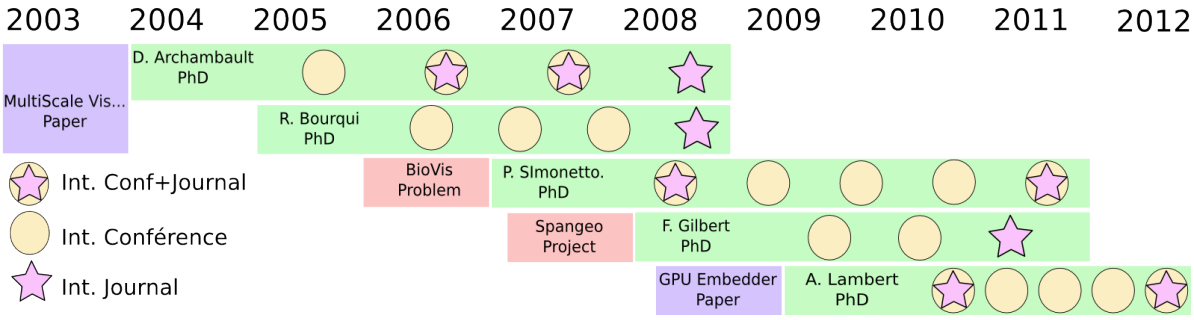


FIGURE 1.8 – Vue d'ensemble des thèses co-encadrées pendant la période 2003-2011.

tre en place un algorithme entièrement automatique pour la génération de diagramme d'Euler. Le chapitre 4 présentent les principaux résultats de ces recherches.

Un autre point que nous voulions approfondir était l'utilisation des nouvelles capacités des GPU/CPU. Pour cela, j'ai co-encadré la thèse d'**Antoine Lambert** avec Guy Melançon. Cette thèse porte sur l'utilisation du GPU en visualisation d'informations. L'objectif était continuer les recherches que j'avais commencées avec Yves Chiricota de l'université de Chicoutimi [12]. Dans ce travail, nous nous sommes attachés à mettre en place des méthodes permettant de réduire l'occlusion générée par la multitude d'arêtes dans les réseaux. Les résultats de cette thèse sont présentés dans le chapitre 5. Notre solution utilise un algorithme parallélisé sur le CPU pour générer une décomposition d'arêtes. Ensuite, nous utilisons le GPU pour générer le rendu de notre décomposition d'arêtes.

Au cours des projets dans lesquels j'ai collaboré, nous avons remarqué que la mise en forme des données représentait un verrou dans l'utilisation de nos techniques de visualisation. La thèse de **Frédéric Gilbert** que j'ai co-encadré avec Maylis Delest était sur ce sujet. L'objectif de cette thèse a été de mettre en place des méthodes de prétraitement automatiques des données pour orienter les analystes dans le choix des techniques de visualisation.

Bien que tous mes résultats soient fortement liés entre eux, j'ai découpé ces recherches selon quatre axes développés dans chacun des quatre chapitres suivants. Je termine avec une description de mes perspectives de recherches.

Chapitre 2

Diviser pour mieux dessiner

Les recherches portant sur le dessin multi-échelles de graphes se sont majoritairement concentrées sur l'accélération des performances des algorithmes existants. Les résultats obtenus permettent ainsi de représenter rapidement des graphes dépassant les cent mille éléments [50, 57, 60]. Dans cette thématique, nous avons exploré une piste un peu différente de l'utilisation des décompositions multi-échelles. L'hypothèse que nous avons développée est qu'il n'existe pas d'algorithme unique de décomposition permettant de trouver tous les clusters dans un graphe et qu'il n'existe pas non plus d'algorithme de représentation unique pour dessiner ces clusters. Partant de cette hypothèse, nous avons mis en place une méthodologie hybride permettant de combiner plusieurs algorithmes de fragmentations et aussi de combiner plusieurs algorithmes de représentation de graphes. La figure 2.1 illustre une mise en application de ce processus de représentation de graphes.

Le principe de fonctionnement des approches multi-échelles de dessin de graphes est globalement toujours le même. L'idée est d'appliquer récursivement des opérations d'agrégations de sommets pour réduire progressivement la taille du graphe de départ. Les réductions successives du graphe forment alors une hiérarchie de graphe. Le gain en termes de complexité algorithmique est obtenu en appliquant des méthodes de dessin très coûteuses sur les petits graphes (bas de la hiérarchie) et en utilisant des méthodes peu coûteuses sur les graphes de plus grande taille (haut de la hiérarchie).

Pour que ces techniques fonctionnent correctement, il est nécessaire que les graphes de chaque niveau soient représentatifs du graphe de départ. En pratique, la plupart des méthodes existantes sont basées sur des sélections de sommets pivots dans le graphe. Ces sommets sont ensuite placés en utilisant l'algorithme de dessin par modèle d'énergie de Kamada Kawaï [65] et/ou l'algorithme de dessin par modèle de forces de Fruchterman et Reingold [48]. Le gain en termes de complexité est alors obtenu en faisant varier le nombre d'itérations utilisées en fonction de la taille des graphes à dessiner.

Après un état de l'art des travaux sur ce sujet, ce chapitre présente un résumé des trois résultats majeurs que nous avons obtenus. Ces trois résultats ont été publiés dans des journaux internationaux et des conférences internationales. Ils ont été réalisé avec les deux premiers doctorants que j'ai co-encadré : **Daniel Archambault** et **Romain Bourqui**. Pour plus de détails sur ces travaux, le lecteur peut se référer aux articles suivants :

- D. Archambault, T. Munzner et D. Auber. "TopoLayout : Multilevel Graph Layout by Topological Features". Dans : *IEEE Trans. on Visualization and Computer Graphics* 13.2 (2007), p. 305–317
- D. Archambault, T. Munzner et D. Auber. "Smashing Peacocks Further : Drawing Quasi-Trees from Biconnected Components". Dans : *IEEE Trans. on Visualization and Computer Graphics (Proc. Vis/InfoVis 2006)* 12.5 (2006), p. 813–820
- Romain Bourqui et David Auber. "Large Quasi-Tree Drawing : A Neighborhood Based Approach". Dans : *Proc. of the 13th International Conference on Information Visualisation*. 2009, p. 653–660

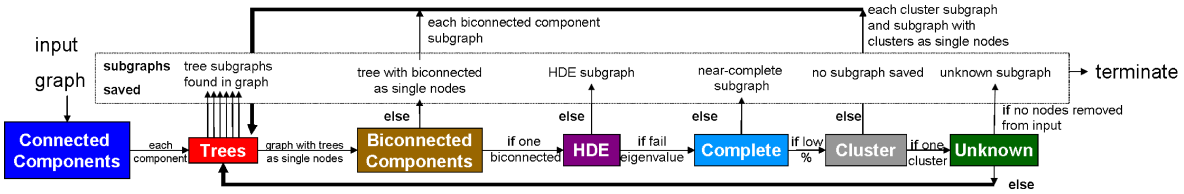


FIGURE 2.1 – Exemple de pipeline de dessin hybride. Lors de l'analyse visuelle de réseaux, l'expert cherche à détecter des motifs particuliers dans les réseaux. Le pipeline ci-dessus permet de caractériser les composantes connexes, biconnexes, les structures arborescentes, les maillages réguliers ainsi que les groupes fortement connectés. Pour chaque type détecté, l'algorithme de représentation le plus adapté est utilisé.

2.1 Etat de l'art

Au moment où nous avons effectué ces recherches, les approches multi-échelles étaient réservées aux algorithmes par modèle de forces. La grande différence entre ces techniques se trouve dans la manière dont les auteurs construisent leurs hiérarchies de graphes.

Walshaw [98], dans sa méthode, utilise une heuristique de calcul du problème de couplage maximal pour obtenir sa hiérarchie de clusters. Ce problème consiste à sélectionner le nombre maximum d'arêtes tel que deux arêtes ne soient pas incidentes au même sommet.

Dans l'algorithme HDE, Harel and Koren [60] calculent récursivement une approximation du problème des k centres. Pour cela, ils utilisent la distance théorique entre deux sommets comme distance idéale. Le problème des k centres permet d'agréger ensemble des éléments en garantissant que la distance entre tout couple de sommets est minimisée.

Dans l'algorithme GRIP, Gajer et Kobourov [50] simulent le phénomène d'agrégation des sommets en utilisant une méthode de filtrage sur le graphe original. Cette méthode consiste à construire des sous-ensembles indépendants à différentes distances. L'algorithme commence par dessiner un ensemble indépendant qui ne contient que 3 sommets (il est à la distance i_{max}). Ensuite, il raffine le dessin en ajoutant récursivement les sommets des ensembles indépendants à distance $i-1$.

Dans l'algorithme ACE, Harel and Koren [67] calculent les vecteurs propres de la matrice Laplacienne du graphe. Ces vecteurs propres permettent ensuite de calculer une projection du graphe dans des dimensions inférieures au nombre de vecteurs propres de la matrice. Dans le cas du dessin en deux dimensions, les deux vecteurs propres ayant la plus grande valeur propre sont retenus. Pour calculer rapidement les vecteurs propres de la matrice, les auteurs construisent une hiérarchie de sous-matrices de petites tailles. Sur ces sous-matrices, ils calculent les vecteurs propres en utilisant la méthode de la *power iteration*[1]. Ces vecteurs propres sont ensuite utilisés pour approximer récursivement ceux des matrices de la hiérarchie de matrice. L'algorithme s'arrête lorsque les vecteurs propres de la matrice Laplacienne du graphe original sont obtenus.

Dans l'algorithme FM^3 (Fast Multipole Multilevel Method) de Hachul et Jünger [57], le graphe est décomposé en sous-graphes appelés systèmes solaires. Ces systèmes solaires sont ensuite agrégés récursivement pour engendrer la hiérarchie. Dans cette méthode, les auteurs montrent qu'un nombre borné de sommets et d'arêtes appartient à un système solaire. Ceci permet de construire une hiérarchie bien équilibrée d'agrégats. En utilisant cette propriété les auteurs parviennent ainsi à prouver que leur algorithme permet de calculer un dessin de graphes avec une complexité $O(N \cdot log(N) + E)$. L'algorithme FM^3 est le premier dont la complexité théorique a été démontrée. Dans [56], une évaluation empirique de cet algorithme démontre clairement les avantages de celui-ci.

1. La méthode de la *Power Iteration* consiste à passer la matrice au carré jusqu'à convergence.

2.2 TopoLayout

TopoLayout [2] est un algorithme multi-échelles pour la représentation des graphes non orientés. Son principe est de dessiner les graphes en fonction des structures topologiques qu'ils contiennent. Nous utilisons le terme structure topologique pour parler d'un sous-graphe ayant une propriété particulière. Par exemple, un sous-graphe qui est un arbre.

La première étape de notre algorithme consiste à détecter récursivement les structures topologiques dans le graphe. Pour cela, nous appliquons successivement différents tests sur le graphe. Dès qu'une structure (sous-graphe) est découverte nous créons un agrégat (méta-noeud). Cette opération permet de réduire la taille du graphe car tous les sommets du sous-graphe sont agrégés en un seul sommet dans le nouveau graphe. On réitère l'opération jusqu'à ce qu'il n'y ait plus de sous-graphe à agréger. La figure 2.1 montre l'ordre de détection que nous avons retenu.

Les 6 types de structures que nous avons considérées sont les suivantes :

- **Composantes connexes :** sous-graphes possédant un chemin entre tout couple de sommets (en bleu).
- **Arbres :** sous-graphes connexes sans cycle (en rouge),
- **Composantes biconnexes :** sous-graphes dans lesquels il faut enlever au moins deux sommets pour créer deux composantes connexes (en marron).
- **Maillages :** sous-graphes dont les matrices Laplaciennes contiennent peu de vecteurs propres (violet).
- **Cliques :** sous-graphes complets (cyan).
- **Clusters :** sous-graphes avec une cohésion interne forte (gris).

La figure 2.2 montre la hiérarchie obtenue sur un petit exemple.

Pour chaque type de topologie, nous utilisons un algorithme de dessin de graphes différent. Pour les arbres, en fonction de la ramification de ceux-ci nous utilisons l'algorithme de Walker [25] ou celui de Grivet et al. [54]. Pour déterminer le niveau de ramification nous utilisons certains de nos travaux

2. Plusieurs noms de sections sont en anglais. Ils correspondent aux noms des techniques que nous avons créées.

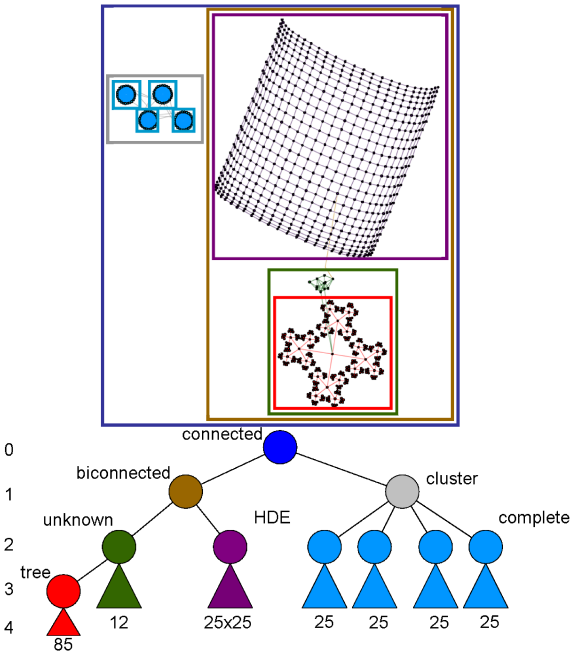


FIGURE 2.2 – Décomposition hiérarchique en fonctions des structures topologiques des clusters. Chaque type de structure est représenté par une couleur différente. (**En haut**) Le dessin final obtenu, la hiérarchie de clusters est représentée par l'imbrication de rectangles colorés. (**En bas**) Représentation arborescente de la hiérarchie. Les chiffres à gauche indique le niveau de la hiérarchie et chaque noeud est annoté par le type de structure topologique qu'il représente.

portant sur la généralisation et l'utilisation des nombres de Strahler [14].

Pour les graphes complets, nous utilisons un algorithme de dessin circulaire. Pour les maillages, nous utilisons une version de HDE [67] permettant de prendre en compte une taille de sommet variable. Pour tous les autres types, nous utilisons l'algorithme GEM [46]. Dans notre méthode, lorsque nous dessinons un graphe, le contenu de chaque méta-sommet n'est pas pris compte. Pour obtenir le dessin final, nous devons dégrouper tous les méta-sommets. Lors de cette étape, nous appliquons une technique d'optimisation locale qui consiste à faire pivoter les méta-sommets afin de réduire la longueur et le nombre de croisements des arêtes allant de l'intérieur du méta-sommet au reste

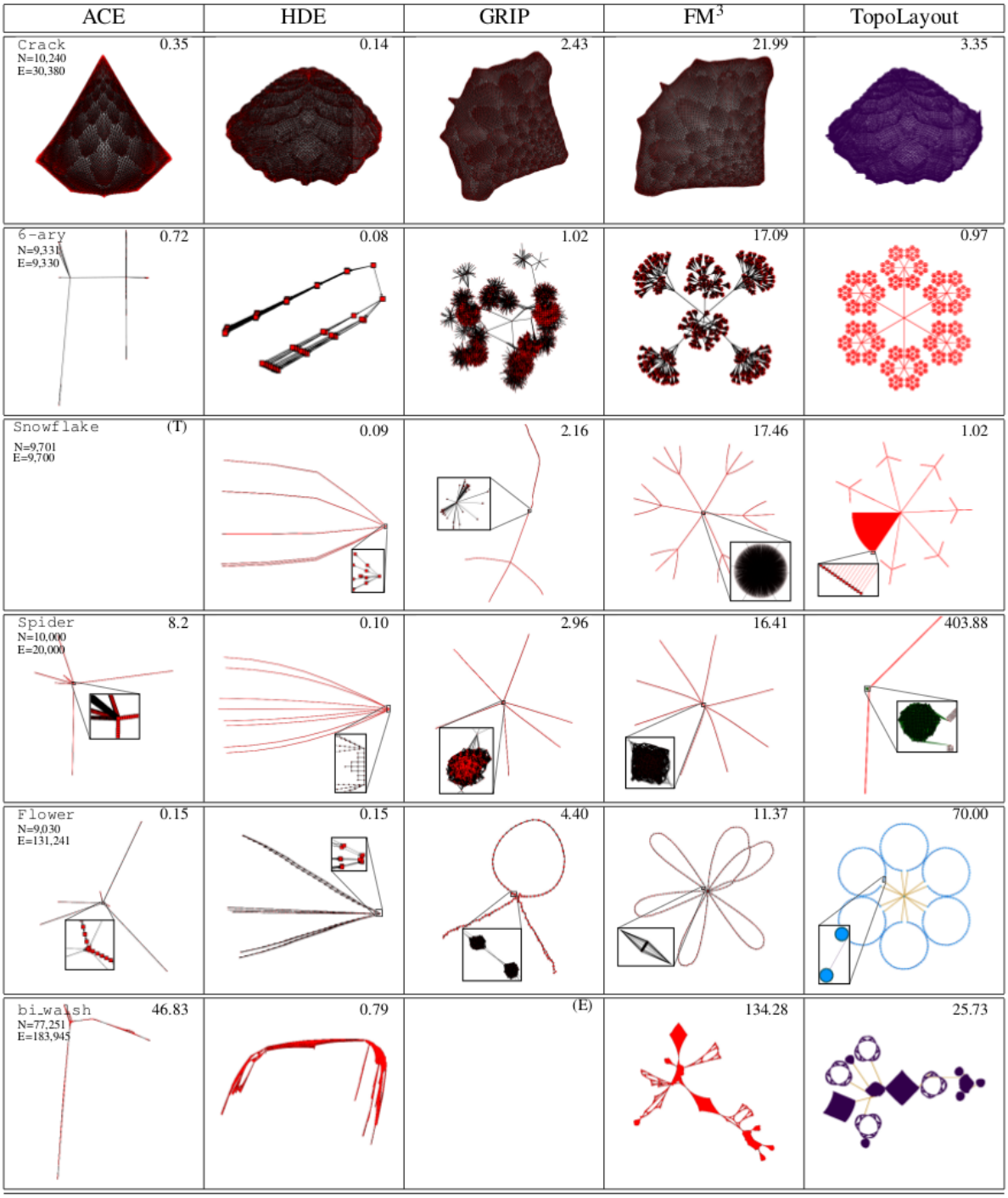


FIGURE 2.3 – Dessins de plusieurs jeux de données obtenus en utilisant ACE, HDE, GRIP, FM3 et TopoLayout. Pour toutes les lignes, un carré vide signifie que l'algorithme n'a pas réussi à générer ce dessin. Le nom du jeu de données, le nombre de sommets et le nombre d'arêtes est indiqué dans le coin en haut à gauche de chaque ligne. Le temps en seconde ou la raison de l'échec est indiqué dans le coin haut droit de chaque cellule. (T) indique qu'aucun dessin n'a été généré au bout de 4 heures de calcul. (E) indique une erreur d'exécution de l'algorithme. Les encadrés montrent un zoom sur un sous-ensemble de noeuds. Pour chaque jeu de données, tous les sous-ensembles ont approximativement la même taille.

du graphe.

La qualité et la rapidité des résultats que nous obtenons est liée au nombre de structures que nous découvrons dans le graphe. La figure 2.2 donne les différentes complexités de notre méthode. Pour évaluer l'utilisabilité de l'approche, nous avons conduit une expérimentation. La figure 2.3 compare la qualité visuelle et la rapidité de TopoLayout avec quatre autre méthodes de dessin sur différents jeux de données. TopoLayout améliore fréquemment les résultats obtenus par les autres sur ces jeux de données.

Algorithme	Complexité
Détection	
Tree	$O(N_i + E_i)$
Biconnected	$O(N_i + E_i)$
Connected	$O(N_i + E_i)$
HDE	$O(d(N_i + E_i))$
Complete	$O(1)$
Cluster	$O(r_i E_i)$
Dessin	
Bubble Tree	$O(N_i log N_i)$
Walker Tree	$O(N_i)$
Circular	$O(N_i)$
GEM	$O(N_i^3)$
HDE	$O(d(N_i log N_i + E_i))$
Améioration	
Croisement	$O(LN_iE)$
Chevauchements	$O(N_i log N_i)$

FIGURE 2.4 – Les différentes complexités des algorithmes intervenant dans notre méthode. N_i resp E_i est le nombre de sommets resp. arêtes d'un sous-graphe au niveau i. Le degré maximum d'un sommet d'un graphe au niveau i est noté r_i. d est la dimension de la matrice utilisée dans HDE. L est la hauteur de la hiérarchie.

2.3 Smashing Peacocks Further

Dans ces travaux, nous avons commencé à étudier le problème de la représentation des graphes possédant une structure topologique particulière. L'élément déclencheur de ces recherches est que nous avons rencontré fréquemment des données formant des graphes qui "ressemblent" à des arbres. Par exemple, en bio-informatique pendant l'étude des graphes d'interactions de protéines ou bien en réseau, pendant l'analyse du graphe des interconnexion de routeurs.

Notre méthode exploite la structure de ces graphes en utilisant une adaptation de la méthode mise en place dans TopoLayout. L'algorithme de clustering n'utilisera que deux niveaux de décomposition. Dans un premier temps, nous décomposons le graphe en composantes biconnexes. Le méta-graphe engendré par cet ensemble de composantes biconnexes forme alors un arbre.

Pour le dessin des composantes biconnexes nous avons amélioré l'algorithme par modèle de forces LGL [3]. Cet algorithme utilise un arbre couvrant comme point de départ pour le dessin. Notre amélioration nous permet simultanément d'améliorer les performances et les dessins produits par l'algorithme.

Pour dessiner l'arbre de composantes biconnexes, nous avons mis en place une nouvelle variante de l'algorithme RINGS [91]. RINGS donne les meilleurs résultats sur les structures arborescentes possédant des sommets internes de très forts degrés. Ce gain en lisibilité est obtenu en sacrifiant le critère de planarité du dessin. Ce critère est inapproprié sur ce type de structure.

Ce travail nous a permis de constater que de nombreux algorithmes de dessin de graphes ne permettent pas la prise en compte des tailles variables sur les sommets. De ce fait, de nombreux algorithmes ne sont pas utilisables dans notre approche "Topolayout". Une grande partie du travail réalisé ici a donc été d'améliorer l'algorithme RINGS [91] et LGL [3] pour permettre leur utilisation dans "TopoLayout".

Notre algorithme nous a permis d'obtenir des vitesses de dessin plus rapide que les algorithmes existants. Ces travaux ont aussi permis d'améliorer la qualité des dessins générés par RINGS et LGL. La figure 2.5, montre sur un exemple, les résultats que nous obtenons avec notre amélioration de LGL et avec l'algorithme hybride SPF.

2.4 Quasi-Arbres

La fréquence d'apparition des graphes de type quasi-arbre au cours de nos expérimentations nous a incité à continuer nos travaux dans cette direc-

tion. Dans ce travail, nous avons étudié plus finement les quasi-arbres pour déterminer les micro-structures topologiques dont ils étaient constitués. La figure 2.6 montre les trois micro-structures élémentaires que nous avons identifiées.

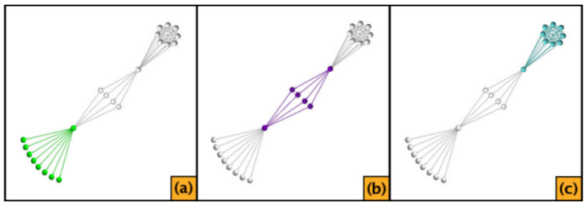


FIGURE 2.6 – Micro-structures élémentaires rencontrées dans un quasi-arbre : (**a**) l'étoile (en vert), (**b**) le diamant (en violet) et (**c**) la clique (en cyan).

La détection de ces structures peut avoir une complexité très élevée (NP dans le cas des cliques). De plus, en pratique les quasi-arbres possèdent plutôt des pseudo cliques, des pseudo diamants etc.. Pour prendre en compte ces phénomènes, nous avons mis en place un algorithme de décomposition basé sur une mesure de similarité entre les sommets. Cet indice est fortement inspiré de l'indice de Jacard [64]. Dans la formule ci-dessous, $N(u)$ représente le voisinage de u et $f(u,v)$ vaut 1 si u et v sont connectés sinon 0.

$$sim(u,v) = \frac{|(N(u) \setminus v) \cap (N(v) \setminus u)| + f(u,v)}{|(N(u) \setminus v) \cup (N(v) \setminus u)| + f(u,v)}$$

Pour calculer les clusters à partir de notre mesure, nous utilisons une approche basée sur l'algorithme DBSCAN [82]. Dans cet algorithme, les clusters de sommets sont déterminés en utilisant deux seuils ϵ et k. Si la similarité $sim(u,v)$ entre deux sommets est supérieure à ϵ alors les deux sommets ne peuvent pas appartenir au même groupe. De plus, un groupe ne peut pas posséder moins de k sommets (en pratique on utilise $k = 2$). Pour obtenir une hiérarchie de clusters, nous utilisons plusieurs seuils ϵ pour le calcul des groupes. Dans un premier temps, nous prenons un seuil faible pour former le niveau le plus bas de la hiérarchie. Ensuite, nous augmentons le seuil pour fusionner les groupes entre eux.

Pour la phase de dessin, nous utilisons une approche similaire à l'approche "TopoLayout". Un algorithme de dessin circulaire est utilisé pour les cliques et les graphes sans arêtes. Pour les graphes quotients et les autres types de structures nous utilisons un algorithme par modèle de forces LGL [3]. Une amélioration significative est apportée au dessin par l'utilisation d'une technique simple d'*edge bundling* (voir chapitre 5) dans notre algorithme.

Nous avons comparé ce nouvel algorithme avec d'autres algorithmes sur différents jeux de données réels. La figure 2.7 montre le résultat obtenu sur un graphe modélisant des protéines. Nous avons montré que, même si notre méthode n'était pas la plus efficace en termes de temps de calcul, cette technique permettait de faire ressortir plus d'informations dans la visualisation. Ce phénomène vient du fait que nous arrivons à mettre en évidence simultanément la structure arborescente du quasi-arbre et les micro-structures présentes dans celui-ci.

2.5 Conclusion

J'ai présenté dans ce chapitre trois résultats [9, 8, 21] que nous avons obtenus sur la visualisation multi-échelles de graphes. Ces trois recherches sont basées sur le même principe : décomposer le graphe en plus petits graphes puis appliquer des méthodes de dessins spécifiques. Ce chapitre montre bien la complexité de la mise en place d'algorithmes de dessins de grands graphes. D'un côté, il faut garantir une bonne complexité algorithmique et de l'autre, il faut garantir une bonne restitution de l'information dans la visualisation produite. L'approche par structure topologique que nous avons mise en place a montré que l'on pouvait trouver un compromis entre les deux.

Nos algorithmes hybrides dépendent fortement de la méthode de détection des clusters mais aussi du choix de l'algorithme de dessin à utiliser. Une caractérisation plus fine des algorithmes de dessin de graphes existant permettrait de créer pour chacun d'eux un algorithme de détection pour savoir si un sous-graphe doit ou ne doit pas être dessiné avec celui-ci. Ce problème est un problème récurrent rencontré en visualisation d'informations.

Lors de nos projets avec des académiques ou des industriels, la question à laquelle nous sommes confrontés n'est pas : quel est l'algorithme le plus

rapide exécutable sur les données ? Mais, quel est l'algorithme qui permet la meilleure analyse des données ? Sur ce point, bien que nos travaux aient donné des solutions, nous devons continuer les recherches. La création d'une taxonomie des algorithmes en fonction de la structure topologique des graphes, des données extrinsèques ainsi que de la nature de l'analyse permettrait de mieux répondre aux attentes des utilisateurs.

Les méthodes que j'ai présentées dans ce chapitre sont statiques. Dans le chapitre suivant, nous montrons qu'il est possible de les rendre plus interactives. Ces améliorations nous permettent de mieux répondre aux besoins réels des utilisateurs.

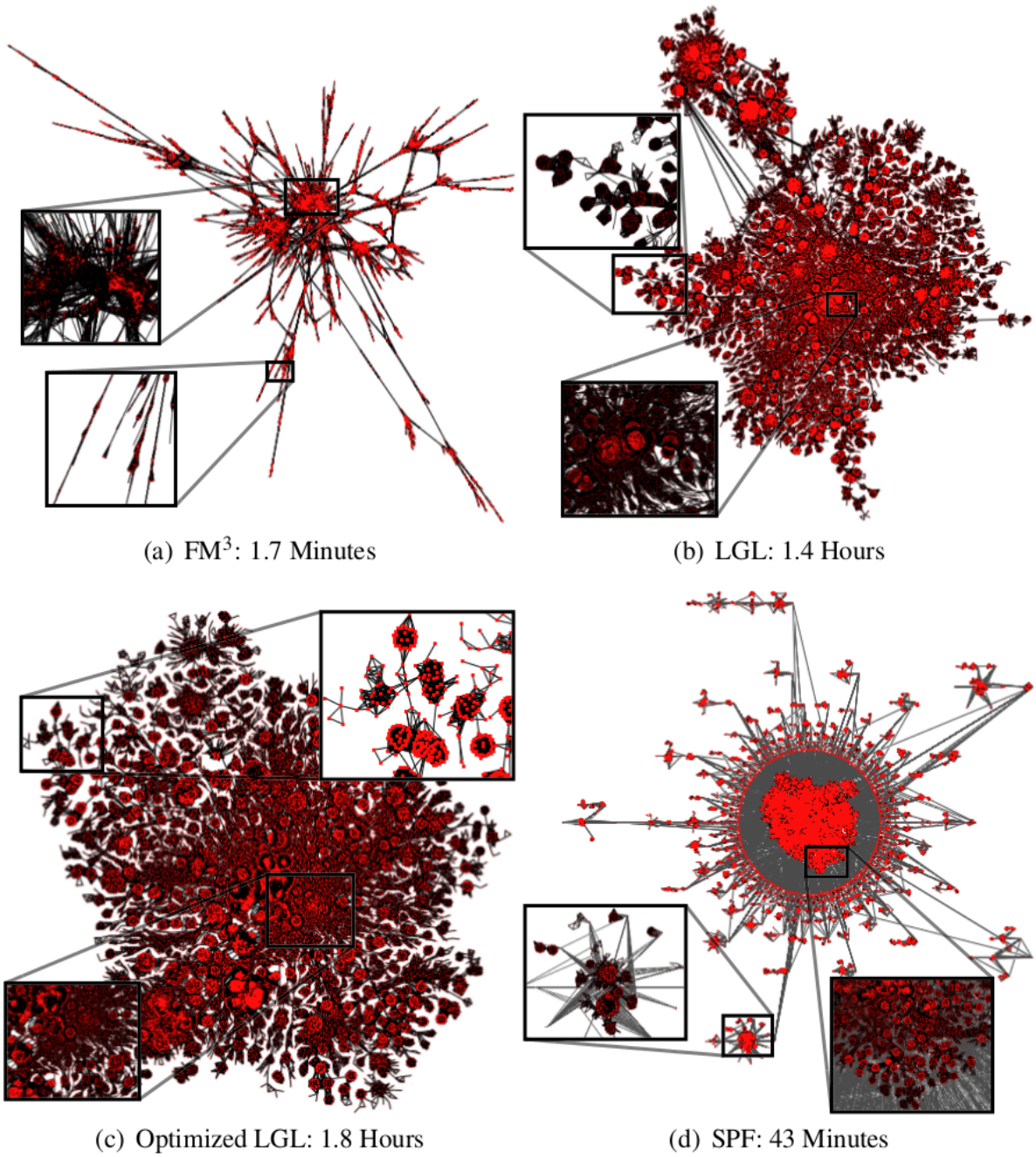


FIGURE 2.5 – Dessin du graphe d'interactions de protéines, les données proviennent du projet LGL. Le graphe contient 30,727 noeuds et 1,206,654 arêtes. (**a**) algorithme FM3, (**b**) algorithme LGL, (**c**) algorithme LGL amélioré, et (**d**) algorithme SPF, les temps de calcul sont indiqués pour chaque algorithme.

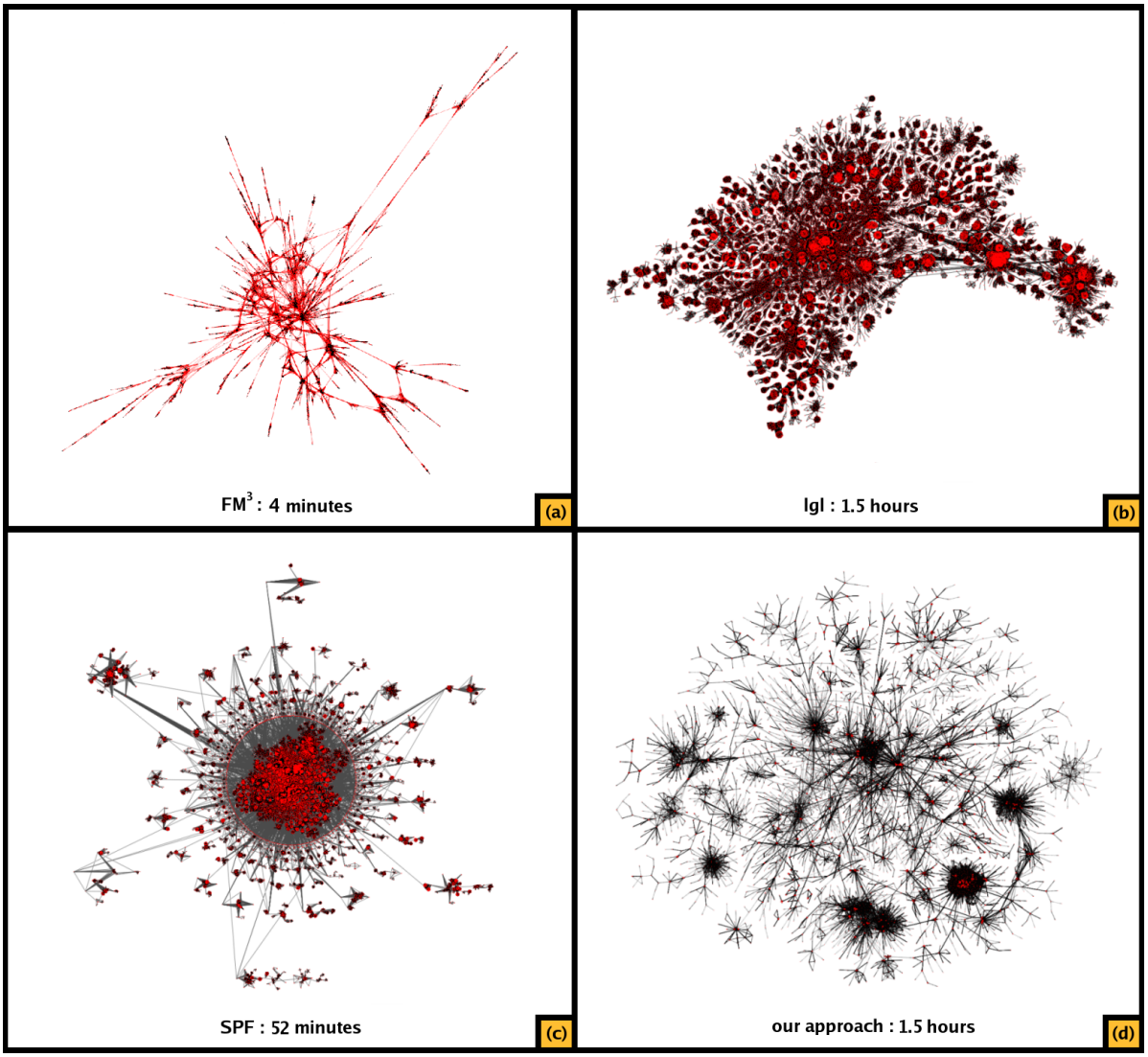


FIGURE 2.7 – Dessin du graphe d'interactions de protéines, les données proviennent du projet LGL. Le graphe contient 30,727 noeuds et 1,206,654 arêtes. (**a**) Algorithme FM3, (**b**) algorithme LGL, (**c**) algorithme SPF et (**d**) notre nouvelle méthode. Les temps de calcul sont indiqués pour chaque algorithme.

Chapitre 3

Visualiser pour mieux analyser

Au début de mes recherches, je me suis concentré principalement sur la partie algorithmique du dessin de graphe. Le chapitre 2 en fait la démonstration. Plus nos recherches ont avancé dans le domaine de la visualisation de graphes, plus nous nous sommes rendu compte que la réelle complexité du problème était de prendre en compte les spécificités des données que l'expert voudra analyser in fine. La confrontation de nos résultats aux attentes du monde industriel nous a effectivement démontré que pour obtenir l'adoption d'une technique, l'efficacité algorithmique n'apportait rien si les données d'un problème n'étaient pas prises en compte dans leur l'intégralité.

Un des résultats que nous présentons dans ce chapitre en fait la démonstration. Nous avons été contactés par un biologiste pour visualiser l'évolution d'un réseau de protéines. Dans cette application, le réseau était donné mais grâce à une expérimentation *micro-array*, nous avions aussi la mesure d'expression de chacun des gènes. Ces taux d'expression permettaient alors de calculer automatiquement un partitionnement hiérarchique des protéines. Ce genre de données remet en cause l'approche développé auparavant car nous ne pouvons plus nous appuyer sur l'arbre de fragmentation pour améliorer la complexité. Au contraire, comme je le montre plus tard, l'obligation de faire ressortir un arbre de fragmentation spécifique augmente la complexité du problème.

Une autre de nos constatations est que dans le processus d'analyse ou d'exploration visuelle des données, l'expert avance avec une méthode empirique. Nous pouvons comparer ce comportement à l'exploration d'un treillis de solutions. Dans ce cas, l'expert effectue lui-même l'élagage de branches entières qu'il ne doit pas explorer ou décide d'explorer plus en détail une branche spécifique. La stratégie "à tâtons" qu'il utilise pour effectuer ce choix n'est pas automatisable. La seule aide que l'on puisse lui apporter dans ce cas est de lui permettre d'explorer le plus rapidement possible les branches qu'il a sélectionnées.

Dans ce rapport, je fais volontairement la distinction entre visualisation de graphes et dessin de graphes. La différence n'est pas très claire. En pratique, on s'aperçoit que les papiers de dessin de graphes sont plutôt publiés dans les conférences théoriques et les papiers de visualisation de graphes dans les conférences de visualisation d'informations. De mon point de vue la différence est essentiellement liée à la prise ou la non-prise en compte des paramètres extrinsèques dans les données. Ce chapitre a pour but de donner un aperçu des techniques que nous avons mises au point pour prendre en compte les données extrinsèques et le comportement des personnes ayant recours à nos systèmes de visualisation.

Après un état de l'art, ce chapitre présente le résumé de trois résultats majeurs que nous avons obtenus. Ces résultats ont été publiés dans des journaux internationaux et des conférences internationales. Ils ont été réalisés avec les deux premiers doctorants que j'ai encadré, Daniel Archambault et Romain Bourqui. Pour plus de détails sur ces travaux le lecteur peut se référer aux articles suivants :

- Romain Bourqui, David Auber et Patrick Mary. "How to Draw Clustered Weighted Graphs using a Multilevel Force-Directed Graph Drawing Algorithm". Dans : *11th In-*

ternational Conference on Information Visualisation. 2007, p. 757–764

- D. Archambault, T. Munzner et D. Auber. "GrouseFlocks : Steerable Exploration of Graph Hierarchy Space". Dans : *IEEE Trans. on Visualization and Computer Graphics* 14.4 (2008), p. 900–913
- Daniel Archambault, Tamara Munzner et David Auber. "Tugging Graphs Faster : Efficiently Modifying Path-Preserving Hierarchies for Browsing Paths". Dans : *IEEE Transactions on Visualization and Computer Graphics* 17.3 (mai 2011), p. 276–289

3.1 Etat de l'art

Dessin de hiérarchie : Les algorithmes de dessins produisent généralement leur propre hiérarchie pour optimiser la qualité du dessin ou la complexité de l'algorithme. Dans cette section nous présentons quelques travaux proposant des méthodes pour représenter des graphes lorsque la hiérarchie de graphes est imposée par l'utilisateur. On considère alors que la hiérarchie est une donnée à part entière qu'il faut visualiser.

P. Eades et Q.-W. Feng [37] ont été les pionniers dans le domaine du dessin de graphes partitionnés. Étant donné une hiérarchie de clusters, ils ont donné deux techniques de représentation automatique. La première permet d'obtenir un dessin en deux dimensions. La deuxième est une approche multi-échelles basée sur l'utilisation de plusieurs dessins ; la troisième dimension est alors utilisée pour superposer les différents dessins.

Dans [99, 38] les auteurs ont donné chacun une méthode de dessin de graphes fragmentés basée sur un algorithme par modèle de forces. La méthode de Huang et Eades [38] permet la navigation interactive tandis que la méthode de Wang [99] utilise un système de résolution de contraintes pour améliorer la qualité du dessin final.

Granitzerand et al. [66] et Kumar et al. [68] ont présenté un algorithme par modèle de forces utilisant une approche récursive. Cet algorithme permet de prendre en compte les poids des arêtes et de dessiner des graphes fragmentés. Le principe de l'algorithme est de dessiner le graphe d'un niveau i et de calculer le diagramme de Voronoï de ce dessin. L'algorithme dessine ensuite chacun des sous clusters (niveau $i + 1$) et les repositionne à l'intérieur des cellules de Voronoï calculées au niveau i.

Données extrinsèques : Plusieurs travaux utilisent les données extrinsèques des graphes pour guider l'algorithme de visualisation. Nous donnons ici une liste non exhaustive des travaux qui ont inspirés nos recherches dans ce domaine.

Pretorius et van Wijk [78] ont décrit un système permettant de représenter dans le plan des données multi-dimensionnelles associées aux sommets et aux arêtes d'un graphe. Dans leur approche, les sommets sont positionnés dans un espace de haute dimension défini par les données extrinsèques du graphe. Une représentation en deux dimensions est ensuite obtenue par projections successives dans des espaces de dimensions inférieures. Cette opération est effectuée en utilisant une méthode d'analyse en composantes principales.

Wattenberg [101] a présenté le système PivotGraph. Ce système détermine le placement des sommets du graphe en utilisant les données extrinsèques. Il détermine des classes d'équivalence en positionnant deux sommets dans la même classe s'ils ont la même valeur pour la dimension considérée. Ces classes d'équivalence sont ensuite utilisées pour influencer l'algorithme de dessin ou simplifier la structure du graphe.

Shneiderman et Aris [84], ont permis le placement des sommets en utilisant uniquement les attributs des sommets. La topologie du graphe n'est alors plus utilisée pour le positionnement des sommets. Une technique de filtrage d'arêtes est ensuite utilisée pour permettre à l'utilisateur d'appréhender la topologie du graphe sous-jacent.

Exploration de hiérarchies : Dans les approches interactives d'exploration de hiérarchie de graphes, l'ensemble du graphe n'est pas représenté entièrement à l'écran. Les systèmes présentent une abstraction du graphe original dans lequel l'utilisateur pourra choisir interactivement les parties qu'il veut développer ou réduire. Deux techniques existent. La première utilise un dessin pré-calculé du graphe en entrée et la seconde régénère à la volée les dessins en fonction des interactions de l'utilisateur.

Les méthodes nécessitant que le dessin soit préalablement calculé sont les plus anciennes. Pour qu'elles soient efficaces, il faut que le dessin du graphe soit compatible avec la hiérarchie de graphes utilisée pendant l'interaction. Si la hiérarchie est

donnée, il faut utiliser un des algorithmes de dessin présenté ci-dessous. Sinon, il est possible de générer automatiquement une hiérarchie compatible avec le dessin du graphe. Dans [52, 83, 59] des approches de type focus plus contexte ("fisheyes") sont utilisées pour grouper ou dégrouper interactivement les clusters de la hiérarchie. Dans [2], Abello et al. utilisent une seconde représentation de type treemap [24] pour représenter et interagir avec la hiérarchie.

Un autre type de méthodes existe. Elles consistent à calculer le dessin du graphe à la volée en fonction des opérations de l'utilisateur. Ces méthodes ne demandent pas l'existence d'un dessin pré-calculé du graphe. Di Giacomo et al. [53] ont présenté un système permettant la visualisation d'un graphe et de sa hiérarchie précédemment créés par un moteur de recherche. Les auteurs utilisent un algorithme de dessin orthogonal pour représenter les abstractions du graphe. Abello et al. [1] ont présenté le système AskGraph-View, Ce système utilise plusieurs visualisations. La première permet de représenter et d'interagir avec la hiérarchie et la deuxième de visualiser le réseau. Le placement du graphe est effectué avec un algorithme par modèle de forces. L'intérêt de ce système est de garantir l'interactivité tout au long de la navigation. Pour cela, le système assure, par des opérations automatiques de groupage et de dégroupage, que la taille du graphe en cours de visualisation est constante.

Édition de hiérarchie : Peu de systèmes permettent à l'utilisateur de modifier interactivement la hiérarchie de clusters qu'il visualise. Nous en avons recensé trois.

Le système DA-TU system [38] de Huang et Eades procure une interface pour l'exploration interactive de hiérarchies de graphes en deux dimensions. Des méta-noeuds peuvent être créés et détruits. Le système DA-TU permet aussi de recalculer les dessins à la volée en utilisant une version modifiée d'un algorithme par modèle de forces.

Mes travaux avec Jourdan [13] permettent l'édition interactive de hiérarchies. Ces recherches avaient pour objectif de procurer des structures de données permettant des modifications avec une complexité linéaire en temps et en espace. Ces structures de données sont utilisées dans les travaux présentés dans ce chapitre.

Pattison et al. [76] ont présenté le système Clovis. Il permet de construire une hiérarchie de graphes en utilisant un système de requêtes sur les attributs du graphe de départ. La hiérarchie de graphes est affichée en surimpression sur le dessin du graphe de départ.

3.2 Graphes pondérés

Les recherches que nous avons menées avec Romain Bourqui et Patrick Mary portent sur le dessin de graphes ayant un arbre de fragmentation fixé et une mesure affectée à chaque arête. Elles ont été initiées lors d'une collaboration avec des biologistes. L'objectif était de voir l'évolution d'un réseau d'interactions de gènes et de protéines. Les interactions étaient pondérées et l'arbre de fragmentation était donné.

Nous nous étions donné pour objectif l'obtention d'un dessin respectant les quatre conditions suivantes :

- Interdire le chevauchement de deux groupes afin de permettre une reconnaissance aisée des groupes par les utilisateurs.
- Conserver l'inclusion des groupes de la hiérarchie pour que le dessin reflète celle générée par les algorithmes de clustering.
- Avoir un dessin dont les clusters peuvent être entourés par des enveloppes convexes.
- Respecter les poids sur les arêtes pour obtenir un dessin dont les distances euclidiennes sont corrélées avec les distances pondérées dans le graphe.

La solution que nous avons retenue est d'étendre un algorithme existant appelé GRIP [50] en intégrant dans celui-ci la gestion de distance pondérée et un système de résolution de contraintes inspiré par les travaux de Granitzer [66].

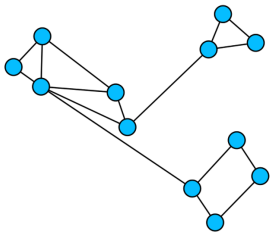
Le principe de fonctionnement de l'algorithme est le suivant : Etant donnée une hiérarchie nous commençons par dessiner le graphe quotient [1] de niveau 1. La figure 3.1 montre l'opération sur un petit exemple. Ce dessin est obtenu avec notre extension de l'algorithme GRIP [49]. Une fois le premier niveau dessiné, nous calculons le diagramme de Voronoï de ce graphe quotient. Ce diagramme nous permet de contraindre récursivement le dessin

1. Les termes : méta-graphe, *compound graph*, *nested graph*, agrégat ou encore groupe sont souvent rencontrés dans la littérature pour désigner ce concept.

de graphes contenu dans le méta-noeud. La figure 3.2 montre ce principe sur un petit exemple.

La figure 3.3 montre les résultats produits par notre méthode. D'un point de vue théorique, nous avons réussi à créer un algorithme efficace respectant toutes les contraintes que nous nous étions imposées. En pratique, comparé aux résultats obtenus avec les algorithmes classiques, notre méthode apporte de bien meilleurs résultats.

Cependant, l'interprétation de ce genre de représentations s'est avérée encore trop complexe à l'usage. Nous continuons à chercher une méthode plus efficace pour permettre une meilleur interprétation visuelle de ce type de données.



3.3 GrouseFlocks

Les recherches que nous avons menées ici portent sur l'exploration et la construction interactive de hiérarchies de graphes. Dans la plupart des travaux existants, les hiérarchies sont créées en utilisant des algorithmes de clustering ou en recherchant des structures topologiques. Dans la pratique, les graphes visualisés possèdent de nombreux attributs associés à leurs sommets et à leurs arêtes. Ces informations peuvent être utilisées pour générer un grand nombre de hiérarchies que nous pouvons alors utiliser pour générer des visualisations. L'objectif de ces recherches a été de mettre en place un système d'exploration interactive des différentes hiérarchies de graphes que l'on peut construire sur un jeu de données. La figure 3.4 montre un graphe, trois hiérarchies différentes et la visualisation de celles-ci en surimpression sur le graphe d'origine.

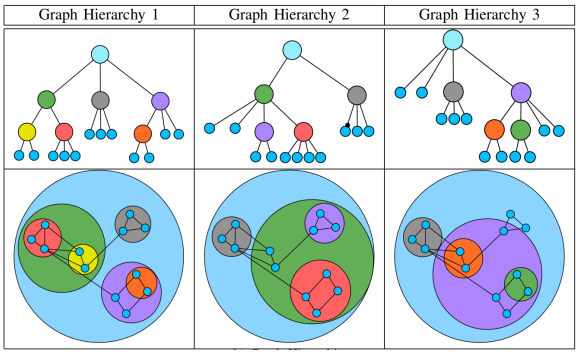


FIGURE 3.4 – Plusieurs hiérarchies en surimpression sur le même graphe. (**En haut**) Le graphe de départ sans aucune hiérarchie en surimpression. (**En bas**) Une table montrant trois hiérarchies possibles prises dans l'espace des hiérarchies que nous pouvons utiliser pour la visualisation. La première ligne de la table montre l'arbre de partition. La deuxième ligne montre la représentation de la hiérarchie en surimpression sur le graphe de départ. La hiérarchie que nous utilisons induit l'abstraction que nous allons pouvoir visualiser à l'aide des graphes quotients. Une hiérarchie unique ne permet donc pas de représenter toutes les informations pertinentes présentes dans un même jeu de données.

Pour permettre cette exploration interactive, nous avons mis en place un système de visualisation complet dédié à cette tâche. Pour cela, nous avons défini dans GrouseFlocks un ensemble d'interactions atomiques que l'utilisateur peut utiliser pour créer et modifier la hiérarchie de graphes en cours de visualisation. Ces interactions sont basées sur des sélections. Ces sélections peuvent être faites manuellement ou peuvent être générées par des requêtes sur les attributs associés aux entités du graphe.

L'analyse visuelle des hiérarchies est puissante. Cependant, l'utilisation de ce type de visualisation peut conduire à de mauvaises interprétations. Pour

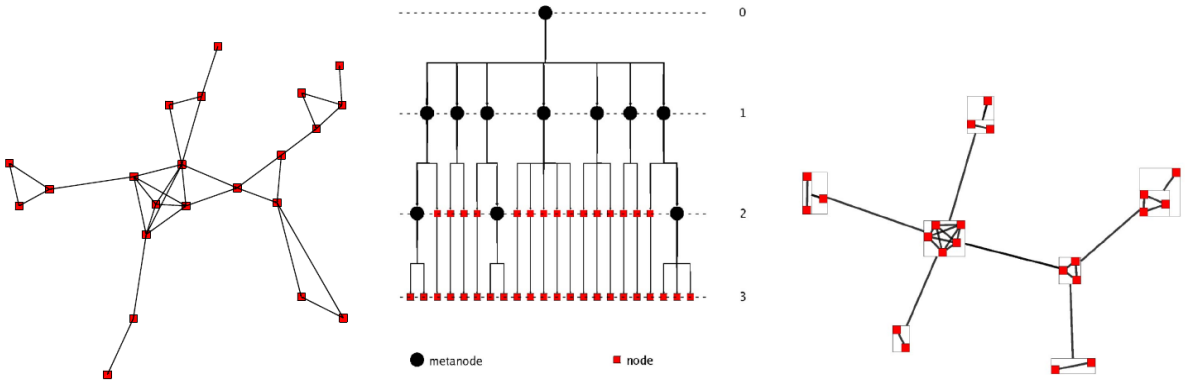


FIGURE 3.1 – Exemple d'arbre de fragmentation. (**A gauche**) Un graphe avec 22 sommets et 33 arêtes. (**Au centre**) Un arbre de fragmentation de ce graphe. Les sommets rouges de l'arbre représentent les sommets originaux du graphe et les sommets noirs des clusters. (**A droite**) Le dessin du graphe quotient obtenu en utilisant les clusters du niveau 1.

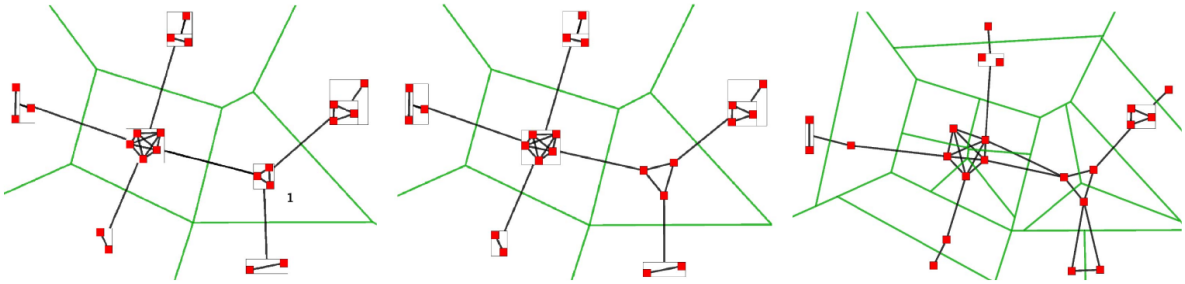


FIGURE 3.2 – (**A gauche**) En vert, le diagrame de Voronoï obtenu à partir du graphe quotient présenté dans la figure 3.1. (**Au centre**) Le résultat obtenu après le dessin du sous-graphe représenté par le méta-noeud 1 dans le graphe quotient. (**A droite**) Le résultat obtenu après le dessin du graphe quotient Q2 de la hiérarchie.

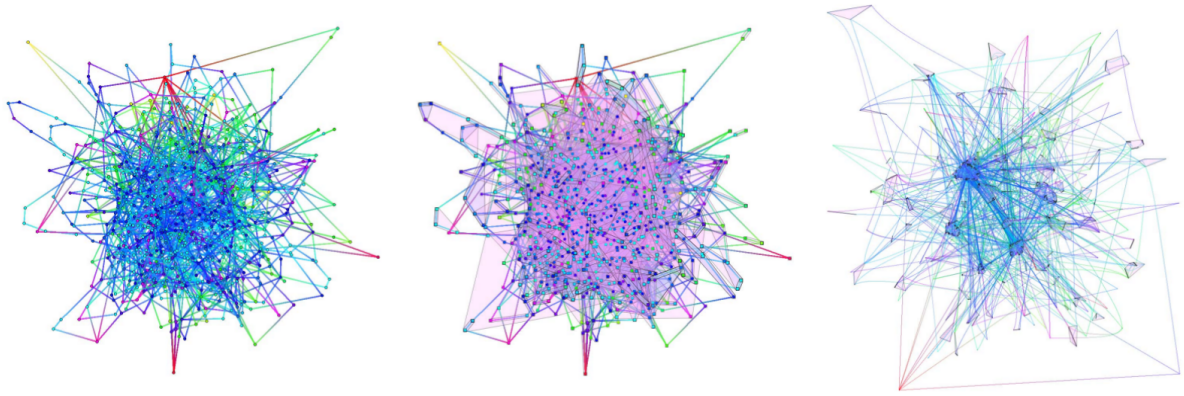


FIGURE 3.3 – (**A gauche**) Dessin d'un graphe d'interactions entre des gènes et des protéines obtenu avec un algorithme par modèle de forces. Le caractère sans échelle de ce type de graphe donne l'effet "pelote de laine" significatif pour ce genre de réseau. (**Au centre**) Les clusters obtenus par les algorithmes des biologistes sont mis en évidence en utilisant des enveloppes convexes autour des sommets appartenant aux clusters. Le chevauchement des enveloppes rend la représentation complètement inexploitable visuellement. (**A droite**) Le résultat obtenu avec notre algorithme. Le respect de toutes nos contraintes permet de percevoir les clusters mais aussi de visualiser les interconnexions entre les clusters.

améliorer la qualité des analyses obtenues par les utilisateurs avec nos systèmes, nous avons introduit le concept de *Path Preserving Hierarchies* [2] qui restreint l'espace des hiérarchies afin que les clusters respectent les deux propriétés suivantes :

- Conservation des arêtes : Une arête existe entre deux méta-noeuds m_1 et m_2 si et seulement si il existe deux feuilles f_1 et f_2 reliées par une arête dans le graphe original tel que m_1 (resp. m_2) est un parent de f_1 (resp. m_2) et $f_1 \neq f_2$.
- Conservation de la connectivité : Tout sous-graphe contenu dans un méta-noeud (ie. sous-graphe induit par toutes les feuilles accessibles par un noeud interne de la hiérarchie) doit être connexe.

Dans le système Grouseflocks, nous utilisons massivement l'algorithme de dessin que nous avons développé dans [6]. Cet algorithme, est une extension de l'algorithme "TopoLayout", il permet un dessin incrémental. Dans ce contexte, nous l'utilisons pour générer des représentations à la volée qui s'adaptent aux modifications apportées à la hiérarchie de graphes et à l'antichaîne en cours de visualisation. Cet algorithme, combiné à une structure de données adaptée, nous a permis de définir les deux interactions de haut niveau suivantes : re-fragmentation et agrégation. Ces deux interactions peuvent être effectuées interactivement sur des graphes de grandes tailles (plus de 100.000 éléments). Elles sont la base de notre système.

- Re-fragmentation [3] : Cette opération consiste à détruire récursivement tous les méta-sommets dans l'antichaîne jusqu'aux feuilles de la hiérarchie, puis de faire une fusion des éléments sélectionnées. Ainsi, cette opération permet la destruction de clusters existants. Cette opération garantit que les propriétés de "Path preserving hierarchies" sont respectées.
- Agrégation [4] : Cette opération modifie la hiérarchie en agrégeant les éléments de l'antichaîne sélectionnés. Il n'y a pas de destruction de clusters dans la hiérarchie. Cette opération garantit que les propriétés de "Path preserving hierarchies" sont respectées.

La figure 3.5 donne une vue d'ensemble et une description du système complet que nous avons implémenté à partir de nos résultats.

2. hiérarchie préservant les chemins existants dans le graphe quotient.

3. Dans nos articles, nous utilisons le terme *Reform Below Cut*

4. Dans nos articles, nous utilisons le terme *Merge At Cut*

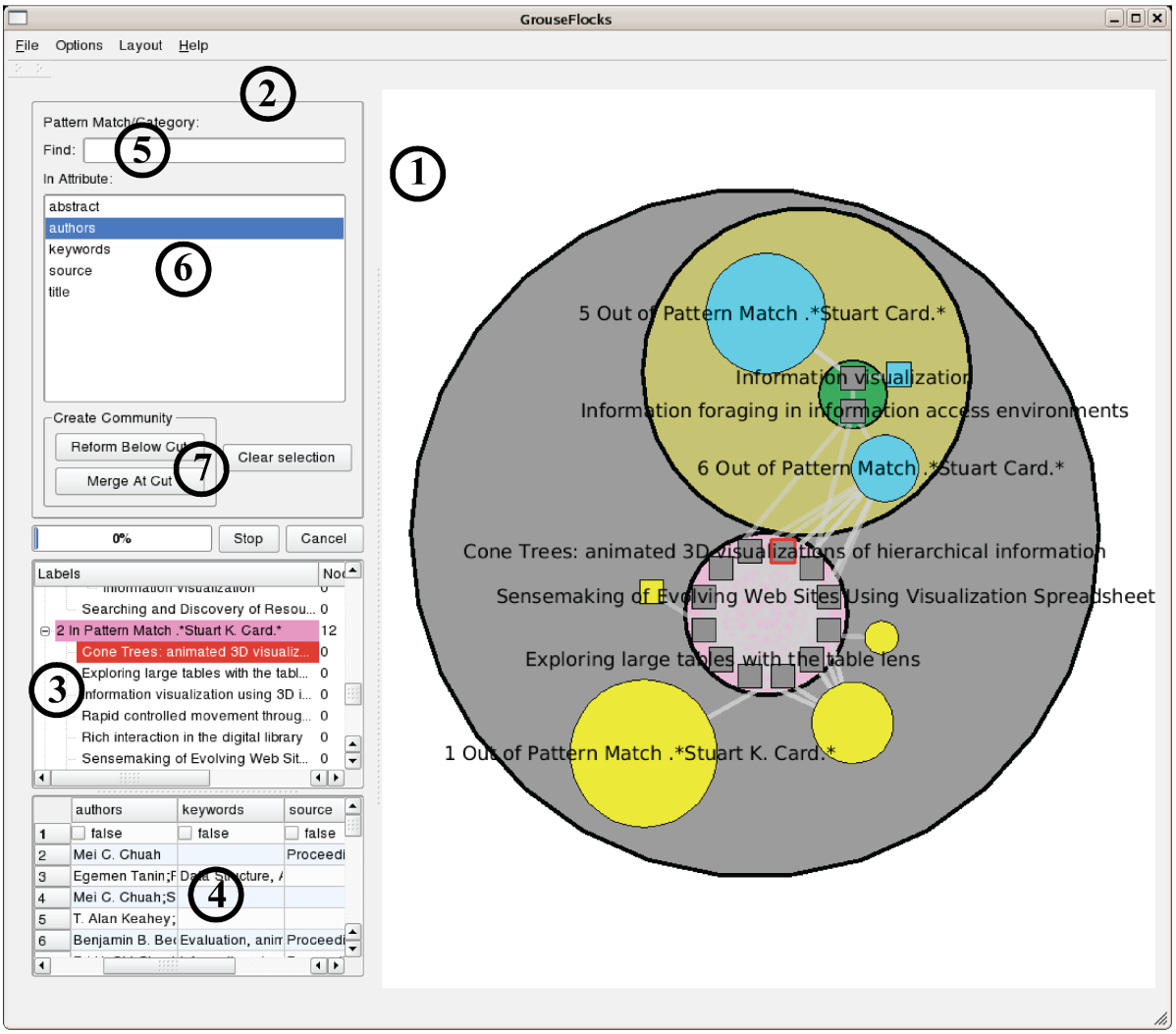


FIGURE 3.5 – Interface utilisateur du système GrouseFlocks. La vue graphe (**1**), montre le graphe quotient associé à l'antichaîne de la hiérarchie. Sur cette vue, on peut ouvrir ou fermer des méta-noeuds en double-cliquant dessus. L'algorithme de dessin incrémental est appliqué pendant l'opération. Sur la gauche (**2**), on trouve les fonctions de recherche disponibles dans GrouseFlocks. Elles permettent de sélectionner des portions du graphe en fonction des propriétés extrinsèques de celui-ci. L'arbre de fragmentation (**3**), montre la hiérarchie qui est en cours d'exploration. On peut ouvrir ou fermer des groupes à partir de cette vue. La table (**4**), montre les informations des sommets en cours de visualisation. Ces sommets peuvent être directement sélectionnés. Des expressions régulières peuvent être utilisées (**5**) pour calculer une sélection sur les attributs sélectionnés (**6**). Les boutons (**7**) permettent d'effectuer les opérations de modification de la hiérarchie en fonction de la sélection courante.

3.4 Tug Graph

L'objectif du système TugGraph a été de permettre une exploration de proche en proche d'un graphe à partir de sommets et de sous-graphes. L'idée de notre approche est de modifier une hiérarchie de graphes existante pour mettre en évidence comment un sommet ou un sous-graphe se positionne dans le réseau global. Notre méthode garantit la propriété de *Path Preserving Hierarchies* présentée dans la section précédente. Le respect de la structure originale du graphe étudié nous permet d'assurer que les abstractions que nous proposons sont intelligibles et pertinentes pour nos utilisateurs. Le système permet d'analyser de proche en proche des réseaux ayant des centaines de milliers de sommets et des millions d'arêtes. TugGraph nous permet de générer des vues de proche en proche en prenant en compte l'ensemble du réseau et ceci sans nécessiter la génération d'un dessin de l'ensemble du réseau au préalable.

TugGraph prend un graphe et une hiérarchie en entrée. S'il existe plusieurs composantes connexes dans le graphe, chaque composante est stockée dans son propre méta-noeud à la racine de la hiérarchie et le dessin est obtenu en utilisant un algorithme de *bin packing*[5]. L'interaction que nous avons développée consiste à cliquer sur un sommet de la hiérarchie affichée (graphe quotient). L'objectif est de montrer à l'utilisateur quel sont les sommets connectés à ce sommet. Ce sommet est un méta-noeud ou un sommet du graphe original. Un *tug* sur ce sommet source va sélectionner toutes les feuilles de la hiérarchie adjacentes à cette source et les placer dans un nouveau méta-noeud. La hiérarchie va être modifiée de sorte que les propriétés de *path-preserving* soient conservées. L'antichaîne affichée est ensuite modifiée pour rendre visible toutes les composantes proches de la source sélectionnée. La figure 3.6 résume l'algorithme que nous utilisons pendant un "tug". Pendant tout le processus, à chaque modification de la hiérarchie et de l'antichaîne, notre méthode de dessin incrémentale [6] est utilisée.

Nous avons effectué de nombreux cas d'études en utilisant notre technique sur des données réelles. La présentation de ces cas d'études peut être trouvée dans la version journal de nos travaux [10]. Sur chacune des expérimentations que nous avons menées (Réseau d'aéroport, réseau d'acteurs de cinéma, réseau de routeur internet) la méthode nous permet une exploration efficace.

5. Algorithme permettant de positionner des rectangles (ou boites) sans chevauchement tout en minimisant la surface nécessaire et en respectant un aspect ratio

3.5 Conclusion

Les trois travaux présentés dans ce chapitre sont révélateurs de la méthodologie à adopter dans le domaine de la visualisation d'informations. L'objectif de cette thématique est de fournir des méthodes visuelles permettant de résoudre un problème particulier. De ce fait, bien que cela semble scientifiquement moins générique, il est nécessaire de considérer toutes les dimensions et toutes les spécificités des données analysées pour mettre en place des techniques de visualisations efficaces. L'exemple présenté dans la section 3.2 sur les interactions entre protéines montre ce problème. La méthodologie d'analyse des utilisateurs experts, encore plus complexe, doit être prise en compte.

Au départ, nous avons abordé le problème de la visualisation de graphes hiérarchiques en suivant un axe purement algorithmique. Ensuite, afin d'appliquer nos résultats sur des données réelles, nous avons utilisé une approche plus expérimentale. Depuis 2003, nos travaux ont contribué à faire avancer cette thématique. Bien que le sujet soit maintenant relativement bien traité, il reste des défis à relever.

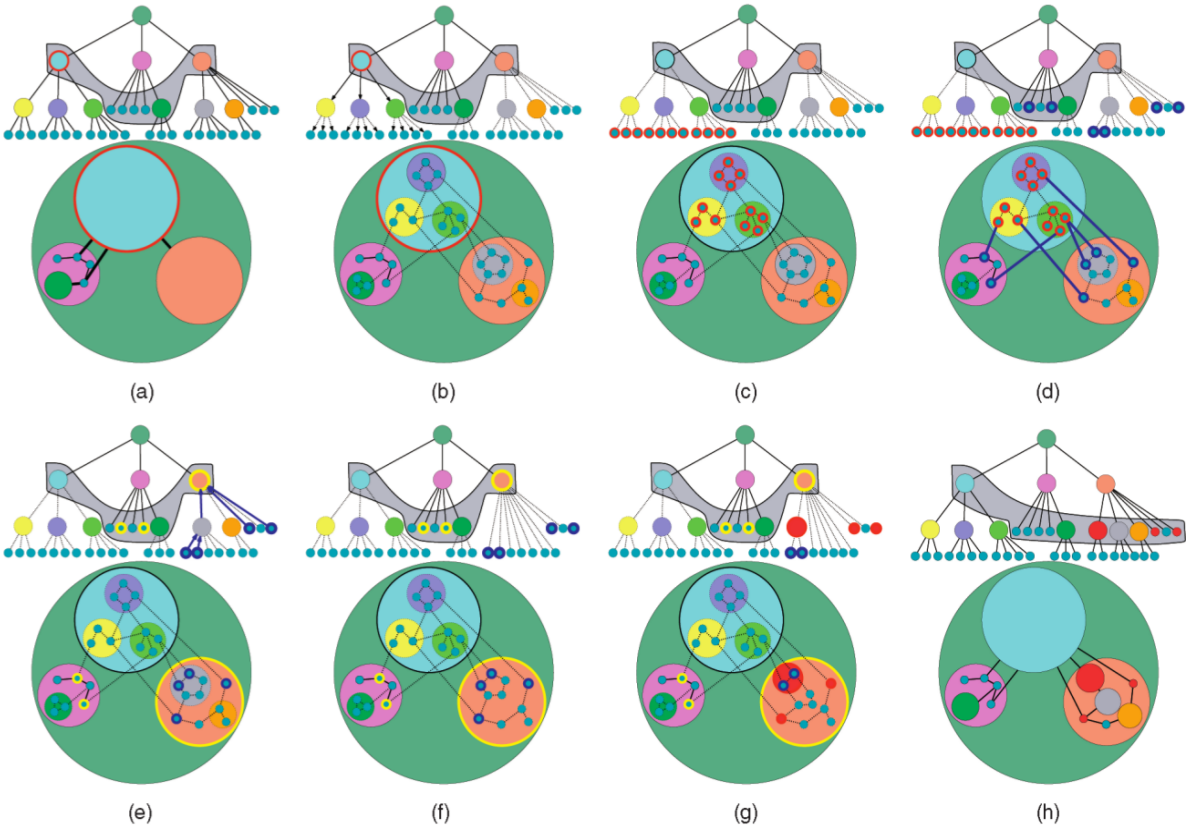


FIGURE 3.6 – Les étapes de l'algorithme de TugGraph. Pour chaque étape, la hiérarchie est représentée au-dessus et le graphe en dessous. (**a**) Les données d'entrée de l'algorithme. Le sommet en rouge est le sommet sur lequel l'utilisateur a cliqué. (**b**) La même sélection est montrée mais en affichant entièrement le graphe. Les lignes en pointillés sont utilisées pour les parties de la hiérarchie de graphes sous l'antichaîne visualisée. (**c**) Les éléments de l'ensemble source S sont entourés en rouge. (**d**) Chaque élément de l'ensemble, à proximité de S, est entouré en bleu et est noté P. Les arêtes utilisées pour créer l'ensemble P sont aussi représentées en bleu. (**e**) Les sommets de l'antichaîne contenant des éléments de P sont représentés en jaune et noté C. Les arêtes de la hiérarchie utilisées pour calculer C sont représentées en bleu sur la hiérarchie. (**f**) La hiérarchie est détruite sous tous les éléments de C. (**g**) Les composantes proches sont calculées. Ces composantes sont entourées en rouge. (**h**) La hiérarchie finale obtenue et sa visualisation à la fin de l'opération de *Tug* sur le sommet rouge présenté dans (a).

Chapitre 4

Visualisation de décompositions chevauchantes

Les progrès en visualisation de graphes, abordés dans les chapitres précédents, nous ont conduits à appliquer nos techniques sur des données de nature variée et à collaborer avec différents profils d'utilisateurs. Ces applications m'ont convaincu que la métaphore visuelle que nous utilisions était trop restrictive comparé à la complexité des problèmes et des données que nous voulions visualiser. J'ai alors commencé à élargir le spectre de mes recherches afin de traiter des représentations différentes.

Dans ce chapitre, je présente les résultats que nous avons obtenus avec mon étudiant Paolo Simonetto. Notre objectif a été de générer automatiquement des représentations pour des décompositions [1] chevauchantes. Une application typique de ces recherches est la visualisation de réseaux issus de la bio-informatique. Par exemple, les réseaux métaboliques sont composés de voies métaboliques. Ces voies forment des sous-graphes du réseau et peuvent se chevaucher. Les quatre questions que nous nous sommes posées sont : Est-il possible de représenter cette information sans mentir ? Quand est-ce que c'est possible ? Pouvons-nous le faire automatiquement ? Est-il possible de le faire rapidement [2] ?

Il existe plusieurs solutions pour représenter les décompositions chevauchantes. L'utilisation d'une représentation noeuds/liens pour les visualiser est possible. Pour cela, il suffit de considérer les décompositions chevauchantes comme des hyper-arêtes d'un hyper-graphe et de représenter l'hyper-graphe correspondant. Lorsque l'on utilise cette technique, il devient difficile de représenter simultanément les hyper-arêtes et les arêtes du graphe sans confusions. C'est pour cela que nous avons utilisé des enveloppes pour représenter les ensembles. Nous nous sommes donc tournés vers les diagrammes d'Euler. En lisant les textes originaux d'Euler, nous avons retrouvé la preuve qu'il s'était aperçu de l'efficacité de ces représentations pour l'analyse et la compréhension des imbrications et des chevauchements d'ensembles. La citation d'une partie du texte original d'Euler se trouve à la figure 4.1.

> "Ces figures rondes, ou plutôt ces espaces (car il n'importe quelle figure nous leur donnions) sont très-propres à nous faciliter nos réflexions sur cette matière, et à nous découvrir tous les mystères dont on se vante dans la logique, et qu'on y démontre avec bien de la peine, pendant que, par le moyen de ces figures, tout saute d'abord aux yeux."

FIGURE 4.1 – Euler, Lettre XXXV, "Des syllogismes, et sur leurs différentes formes, si la première proposition est universelle", 17 février 1761.

Ce type de représentation est très utilisé en pratique. Cependant, à notre connaissance, aucune méthode de génération automatique n'existait. Seul des cas particuliers ont été traités. La découverte

1. Une décomposition n'impose pas que l'intersection des clusters soit vide.

2. Par rapidement, j'entends : Avec une complexité théorique lui permettant de s'exécuter en moins de quelques minutes sur des problèmes de tailles moyennes (1000 éléments).

d'une méthode générique a été notre objectif.

Après un état de l'art sur ce sujet, ce chapitre présente un résumé des trois résultats majeurs que nous avons obtenus. Ils ont été réalisés avec mon doctorant Paolo Simonetto. Pour plus de détails sur ces travaux, le lecteur peut se référer aux articles ci-dessous :

- Paolo Simonetto et David Auber. "Visualise Undrawable Euler Diagrams". Dans : *Information Visualization*. London, UK, 2008, p. 594–599
- Paolo Simonetto, David Auber et Daniel Archambault. "Fully Automatic Visualisation of Overlapping Sets". Dans : *Computer Graphics Forum (EuroVis09)* 28.3 (2009), p. 967–974
- Paolo Simonetto et al. "ImPrEd : An Improved Force-Directed Algorithm that Prevents Nodes from Crossing Edges". Dans : *Computer Graphics -New York- Association for Computing Machinery- Forum* 30.3 (2011)

4.1 Etat de l'art

Les diagrammes d'Euler [40] sont des représentations graphiques d'intersections d'ensembles. Dans un diagramme d'Euler, un ensemble est une région du plan bornée par une courbe fermée et les intersections d'ensembles sont représentées par des chevauchements de régions. Il n'existe pas de définition plus précise reconnue par l'ensemble de la communauté. Nous avons plusieurs définitions [42, 43, 90] et les visualisations que l'on peut construire avec chacune d'elles diffèrent les unes des autres. Nous avons commencé notre travail sur ce sujet par l'introduction de notre propre type de diagrammes que nous appelons *Euler-Like diagram*. Cette étape était nécessaire à la génération de visualisations de n'importe quelle décomposition chevauchante.

Selon la définition que l'on utilise, plusieurs algorithmes existent [28, 45, 44, 95] pour générer un diagramme. Pour chacune de ces méthodes, il existe de nombreuses instances de diagrammes que l'on ne peut pas représenter. Ces cas particuliers rendent ces méthodes inutilisables sur le type de données que nous voulons analyser.

Au début de nos recherches et à notre connaissance, aucun algorithme n'existait pour générer des diagrammes d'Euler dans le cas général. Cependant, Rodgers et al. [80, 81] ont proposé une méthode en même temps que nous. Nos deux papiers ont été simultanément publié dans la même conférence. Les travaux de Rodgers et al. se basent sur une extension de ce qui avait été fait la même année par Flower et al. dans [44]. Bien que basée sur des principes similaires, notre approche diffère de celle de Rodgers et al. sur de nombreux points.

Tout d'abord, Rodgers et al. essayent d'enlever dans les diagrammes les cas particuliers telles que les croisements multiples ou les bordures concurrentes. Dans notre approche, nous utilisons des techniques de traitement d'images (des textures et des lissages de frontières) qui nous permettent de contourner ce problème. En pratique, cette solution est plus compréhensible et beaucoup plus simple à mettre en oeuvre.

Les deux approches utilisent un algorithme par modèle de forces pour améliorer le dessin original. Cependant, la structure de graphe intermédiaire que nous utilisons pour la génération des diagrammes permet d'obtenir une meilleure complexité. De plus, les travaux de Bertault [17] prouvent que la méthode de Rodgers et al. ne garantit pas un résultat correct. En utilisant une extension des résultats de Bertault que nous avons mis en place, nous pouvons prouver que notre résultat est toujours valide.

Enfin, notre méthode permet d'insérer des éléments à l'intérieur des ensembles. Dans le diagramme, nous obtenons ainsi des régions ayant une surface proportionnelle à la taille des ensembles. Ce dernier point rend notre méthode utilisable dans de nombreuses applications de visualisations.

4.2 Diagrammes d'Euler non représentables

Etant donné un ensemble d'ensembles chevauchants, il n'est pas toujours possible de le représenter avec un diagramme d'Euler. En effet, le respect des propriétés d'un diagramme d'Euler peut être contradictoire avec le respect des relations ensemblistes que l'on veut représenter. Ce problème réduit considérablement le nombre de décompositions chevauchantes que l'on peut visualiser. La conséquence est, qu'en pratique, peu

de jeux de données satisfont les critères nécessaires. De ce fait, peu de visualisations utilisent cette métaphore.

Il existe des extensions aux diagrammes d'Euler pour élargir cette classe mais aucune ne permet de prendre en compte tous les cas possibles. La figure 4.2 montre les différents types de diagrammes existants. Le but de ce travail a été de proposer une méthodologie permettant de créer des diagrammes qui ressemblent à des diagrammes d'Euler et qui fonctionnent dans tous les cas et ce, même pour les cas dégénérés.

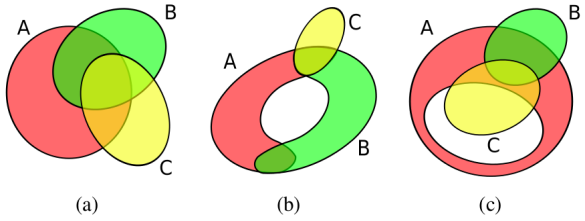


FIGURE 4.2 – Les différents types de diagramme d'Euler rencontrés dans la littérature. (**a**) Montre les diagrammes étudiés dans [45]. Ils ne permettent pas de chevauchement de courbe et de croisements multiples sur un même point. (**b**) Montre des diagrammes plus généraux étudiés dans [28]. Les croisements de plusieurs courbes en un même point sont autorisés, les bordures peuvent se chevaucher et il peut y avoir des chevauchements déconnectés dans un même ensemble. (**c**) Montre des diagrammes d'Euler étendus définis dans [95]. Des trous dans les ensembles sont maintenant autorisés.

Dans ces travaux, nous avons étudié les propriétés des jeux de données qui n'étaient pas représentables et nous avons donné toutes les bases nécessaires à l'élaboration d'un algorithme de représentation automatique. A partir d'un ensemble d'ensembles, notre approche consiste à décomposer ces ensembles en zones d'intersections. Ces zones peuvent ensuite être utilisées pour générer un graphe que nous appelons graphe d'intersections. La figure 4.3 illustre la construction de ce graphe. Une fois que ce graphe est dessiné de manière planaire, il est possible d'obtenir directement une visualisation en créant une enveloppe autour des arêtes reliant les zones appartenant à un ensemble.

Le graphe d'intersections permet d'utiliser la

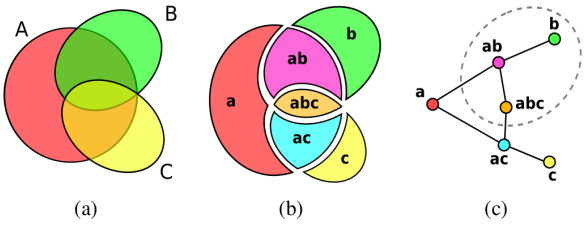
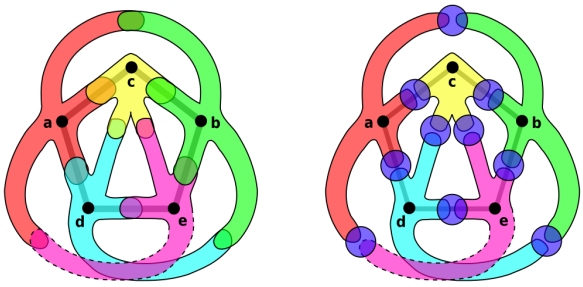


FIGURE 4.3 – Correspondance entre un diagramme d'Euler et un graphe d'intersection (**A gauche**) Diagramme original composé de trois ensembles A, B, C. (**Au centre**) Mise en évidence de chacune des zones. (**A droite**) Un graphe d'intersections que nous pouvons construire. Les lignes pointillés ne font pas partie du graphe, elles montrent comment passer du graphe au diagramme. Pour dessiner les bordures de l'ensemble B, on doit entourer les sommets b, ab, abc en coupant les arêtes (a, ab), (ac, abc) du graphe.

théorie des graphes "traditionnelle" pour détecter les situations ou la génération des diagrammes est impossible. Ces cas sont détectés en utilisant des algorithmes de tests de planarité de graphes. La figure 4.4 illustre un ensemble d'ensembles non représentable. La non planarité du graphe d'intersections impose l'apparition de zones qui se chevauchent alors qu'elles ne devraient pas.

Une grande partie de nos travaux se concentre sur la construction et la modification de ce graphe d'intersections. Le choix du graphe d'intersections n'est pas unique. Pour obtenir des représentations de qualité, il est nécessaire de le choisir correctement. Dans [86], nous avons proposé une heuristique permettant de maximiser les critères qui semblent être les plus pertinents pour l'analyse de tels diagrammes. Pour résoudre le problème présenté dans la figure 4.4, il est possible d'effectuer une passe de planarisation du graphe d'intersection. Cette opération conduit à la duplication de certains éléments dans le diagramme. Elle est obligatoire si nous ne voulons pas créer des visualisations mensongères. La figure 4.5 montre un exemple de diagramme que l'on peut obtenir en combinant tous les résultats publiés dans ce papier.



(a) (b)

FIGURE 4.4 – (**a**) Un diagramme représentant cinq ensembles qui ont deux à deux une intersection non vide. La partie en pointillé dans le diagramme montre une information mensongère. A la lecture, on peut se demander s'il y a deux intersections entre e et d. Ceci revient à couper en deux une zone. Pour avoir une lecture compréhensible d'un diagramme d'Euler, une seule région devrait représenter l'intersection entre deux ensembles. Ce n'est pas le cas dans ce diagramme. Pour remédier au problème, la seule solution est de dupliquer complètement l'ensemble e. (**b**) Un exemple de graphe ou la non-duplication de e introduit un mensonge. Dans l'exemple, les petits ensembles représentés par des cercles sont distincts. L'ensemble e et l'ensemble d ne possèdent aucune intersection. La présence d'un chevauchement indique le contraire. Seule la duplication de l'ensemble e ou d permet de résoudre ce problème.

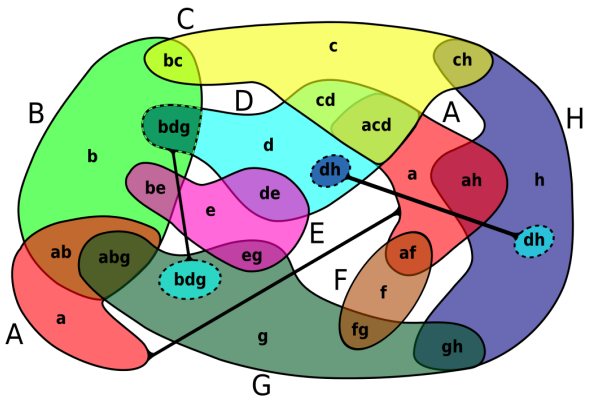


FIGURE 4.5 – Exemple de représentation de diagramme d'Euler. Le graphe d'intersections est non planaire (il contient K_5), il n'est donc pas possible de représenter ce graphe avec des diagrammes d'Euler classiques. En dupliquant des ensembles et en les connectant (pour mettre en évidence la duplication), nous arrivons à générer des diagrammes justes et compréhensibles.

4.3 Méthode automatique

Le troisième papier que nous avons publié sur ce sujet propose un algorithme complet pour la génération des visualisations que nous avons présentées dans [87, 86]. L'originalité de notre méthode est de pouvoir fonctionner sur n'importe quel ensemble d'ensembles chevauchants. Ceci nous permet de l'appliquer sur des données réelles.

Notre algorithme prend en entrée un ensemble d'ensembles. La figure 4.6 montre la visualisation d'un petit ensemble d'ensembles avec notre méthode automatique.

A partir de n'importe quel ensemble d'ensembles, nous construisons un graphe d'intersections valide en utilisant notre heuristique présentée dans [86]. Le graphe d'intersections est ensuite dessiné de manière planaire. Ce graphe servira de squelette pour la génération du diagramme final. Notre graphe d'intersections étant planaire, nous utilisons l'algorithme de De Fraysseix et al. [30] pour générer un premier dessin en temps linéaire. Pour améliorer la qualité de celui-ci, nous appliquons un post-traitement qui consiste à déplacer les positions des sommets avec un algorithme par modèle de forces [16]. Cet algorithme conserve le plongement de départ. La conservation du plongement (ordre

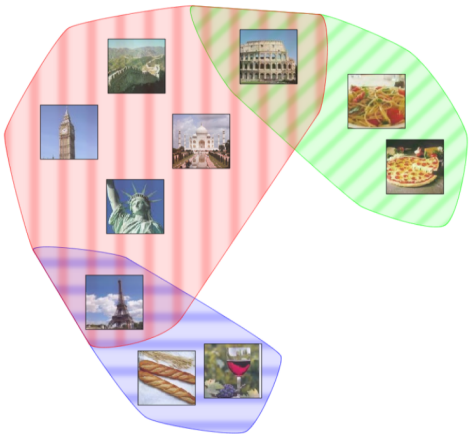


FIGURE 4.6 – Exemple de diagramme obtenu avec notre méthode en prenant en entrée uniquement le jeu de données suivant : {{Tour Eiffel, Colisée, Taj Mahal, Big Ben, Grande Muraille, Statue de la liberté}, {Vin, Baguette, Tour Eiffel}, {Colisée, Pâte, Pizza}} . L'enveloppe rouge représente les monuments, la verte les éléments associés à l'Italie et la bleue les éléments associés à la France.

des arêtes autour des sommets) nous garantit que l'on pourra générer le diagramme désiré à partir de ce squelette. Ensuite, nous construisons des régions en détourant les éléments de notre squelette. Par construction, ces régions forment elles aussi un graphe planaire que nous appelons le *grid graph*. Pour obtenir le dessin final, nous ajoutons ensuite les éléments des ensembles dans les régions correspondantes et nous appliquons de nouveau l'algorithme de Bertault [17]. La figure 4.7 montre le fonctionnement de notre algorithme sur un petit exemple.

Pour améliorer la qualité de nos diagrammes, nous utilisons des courbes de Bézier. Celles-ci nous permettent de lisser les contours et ainsi, grâce à la continuité des contours, de mieux percevoir nos régions. Nous utilisons aussi une coloration des régions en appliquant sur le graphe d'intersections l'algorithme de Welsh et Powell [102]. L'utilisation de la transparence pour le mélange des couleurs sur les zones d'intersections fonctionne mais reste difficile à comprendre quand il y a beaucoup de chevauchements. Afin d'améliorer la lisibilité de nos visualisations, nous appliquons des textures en utilisant la méthode introduite par Byelas et al. [26].

Les figures 4.8 et 4.9 montrent les résultats

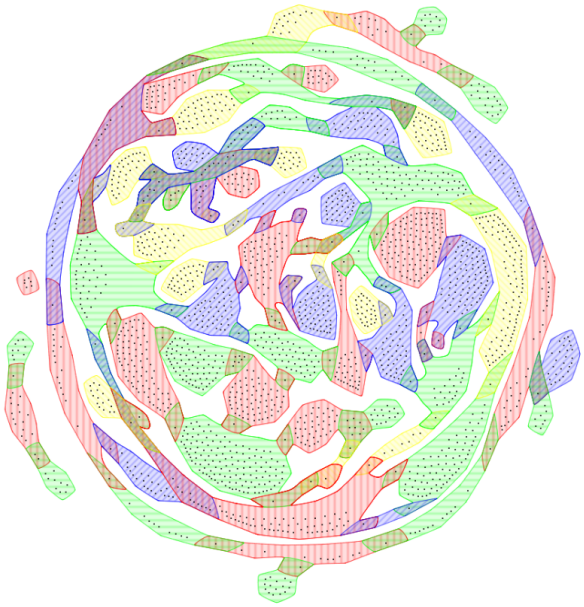
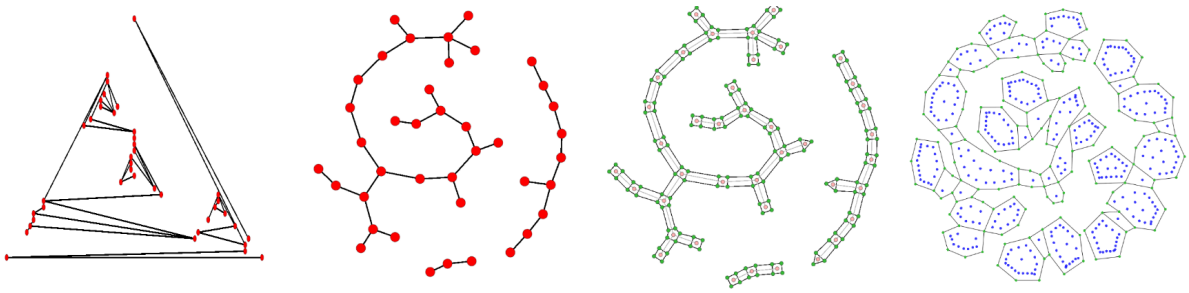


FIGURE 4.8 – Diagramme généré sur soixante des soixante-dix "meilleurs" films de l'*Internet Movie DataBase*. Pour chaque film, le casting entier a été pris en compte. Ce diagramme représente simultanément les interconnexions des castings de plus de 2000 acteurs. Même si ce diagramme est difficile à appréhender, il est lisible et compréhensible.

de notre méthode sur deux jeux de données de tailles différentes. Ils sont tous les deux extraits de l'*Internet Movie DataBase*. Les sommets sont des acteurs et nos ensembles sont des films. La visualisation de cette base de données à l'aide des diagrammes d'Euler nous permet de mettre efficacement en évidence les interconnexions entre les différents castings d'acteurs.

4.4 ImPrEd

Le dernier papier que nous avons publié sur la visualisation de diagramme d'Euler est un papier algorithmique. Notre objectif a été d'améliorer l'algorithme responsable de la complexité algorithmique de notre méthode. Comme présenté dans la figure 4.7, notre méthode utilise à plusieurs reprises l'algorithme PrEd de Bertault [16]. Cet algorithme est un algorithme par modèle de forces qui conserve l'ordonnancement des arêtes autour des som-



(a) (b) (c) (d)

FIGURE 4.7 – Exemple de fonctionnement de notre algorithme. (**a**) Dessin initial du graph d'intersections obtenu avec l'algorithme de De Fraysseix et al. [30]. Bien que théoriquement très performant, ce dessin n'est pas utilisable en pratique. (**b**) Amélioration du dessin initial obtenu par l'application, a posteriori, de l'algorithme de Bertault (PrEd) [16]. (**c**) Construction du *grid graph* par détourage du dessin du graphe d'intersections dessiné. (**d**) Les éléments des ensembles de départ sont insérés dans leurs régions respectives. L'algorithme PrEd est appliqué de nouveau pour élargir les régions en fonction de leur nombre d'éléments. Dans les quatre figures, les sommets du graphe d'intersections sont en rouge, les sommets du *grid graph* sont en vert et les éléments des ensembles sont en bleu.

(a) (b)

FIGURE 4.9 – Exemple de diagramme obtenu sur un petit nombre d'ensembles. Ici, les trois castings de la trilogie "Lord Of The Ring". (**a**) Le diagramme que nous obtenons automatiquement. "The Fellowship of the Ring" est en rouge. "The Two Towers" est en vert et "The Return of the King" est en bleu. (**b**) Les acteurs présents dans les trois films. Les miniatures des acteurs ont été ajoutées lorsque qu'elles étaient disponibles dans la base de données.

mets ainsi que les croisements dans le dessin initial. Outre la génération de diagramme d'Euler, cet algorithme a de nombreuses applications dans le domaine de la visualisation des graphes planaires. Cependant, PrEd a une complexité élevée ($O(N^3)$) et est beaucoup trop restrictif sur les modifications qu'il autorise dans le dessin initial. Ces deux restrictions nuisent à son utilisation sur des ensembles de grandes tailles et réduit notablement la qualité des visualisations que nous pouvons produire.

L'algorithme que nous avons mis en place est nommé ImPrEd pour *Improved PrEd*. Son fonctionnement est identique à celui de PrEd. A chaque itération, nous calculons une direction dans laquelle nous voulons déplacer un sommet. Ensuite, on teste si le déplacement de ce sommet à cette nouvelle position peut changer le plongement du graphe. Pour tester si le déplacement est valide, nous utilisons un découpage en secteurs angulaires comme dans PrEd. La différence entre PrEd et impPrEd est que nos secteurs angulaires permettent des mouvements beaucoup plus importants. Dans les figures 4.10 et 4.11, nous comparons les zones de déplacements autorisés par ImPrEd avec celles de PrEd. Dans ces deux figures, on considère c_v, le vecteur de collision. Deux cas sont possibles : Soit c_v coupe une arête (figure 4.10) soit il ne la coupe pas (figure 4.11).

La deuxième amélioration apportée à l'algorithme permet de réduire le nombre d'arêtes testées

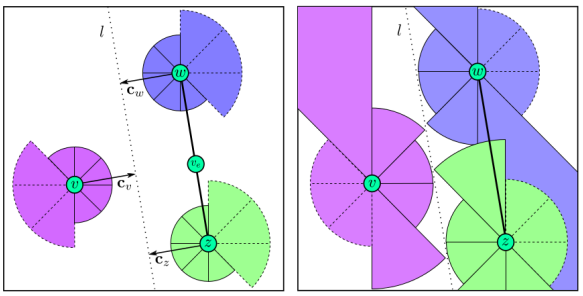


FIGURE 4.10 – Déplacement maximal autorisé lorsque le vecteur de collision coupe une arête. (**a**) Restrictions utilisées dans PrEd. (**b**) Restrictions utilisées dans ImPrEd. Les nouvelles restrictions garantissent les mêmes propriétés que celles de PrEd.

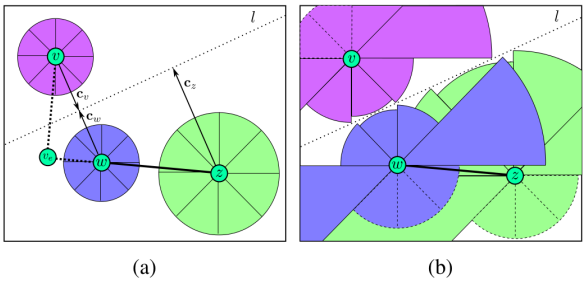


FIGURE 4.11 – Déplacement maximal autorisé lorsque que le vecteur de collision ne coupe pas une arête. (**a**) Restrictions utilisées dans PrEd. (**b**) Restrictions utilisées dans ImPrEd. Les nouvelles restrictions garantissent les mêmes propriétés que celles de PrEd.

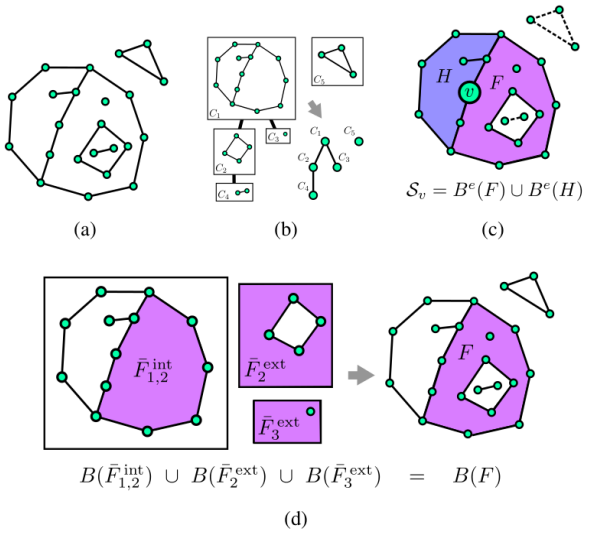


FIGURE 4.12 – Calcul des arêtes à tester pour le sommet v (**a**) Un exemple de graphe planaire. (**b**) Le graphe est décomposé en composantes connexes. Les composantes sont hiérarchisées en fonction de leurs imbrications dans le dessin. (**c**) Les arêtes à tester sont calculées en parcourant les faces F et H et en partant du sommet v. (**d**) On utilise la hiérarchie pour calculer les arêtes à tester dans les autres composantes connexes.

à chaque itération. Pour cela, nous utilisons la topologie du graphe planaire pour déterminer l'ensemble des arêtes susceptibles d'être traversées pendant le déplacement d'un sommet. C'est cette amélioration qui nous permet de réduire le temps d'exécution de l'algorithme. Cependant, nous ne pouvons pas prouver théoriquement que la complexité est meilleure. En effet, si le graphe est un arbre, il ne comporte qu'une face. Il faut donc, comme dans la version précédente de PrEd, tester toutes les arêtes du graphe. Cependant, plus le nombre de faces augmente plus notre algorithme est rapide. La figure 4.12 montre la hiérarchie de graphes que nous construisons afin de déterminer les ensembles minimaux d'arêtes à tester pour chaque sommet.

La troisième amélioration que nous avons poposée permet la prise compte de la taille des ensembles. Lors de nos expérimentations, nous nous sommes aperçus que, lorsque les enveloppes qui entourent nos ensembles ne possédent pas beaucoup d'arêtes, il n'est pas possible de les faire

"grossir". La solution que nous proposons pour résoudre ce problème est d'utiliser des arêtes flexibles. Le principe des arêtes flexibles est de couper une arête en deux (ajout d'un sommet et d'une arête) lorsqu'il y a trop de tension et de fusionner des arêtes lorsqu'il y a trop de compression. La gestion des arêtes flexibles ralentit l'algorithme. Cependant, l'amélioration de la qualité des diagrammes d'Euler que nous obtenons compense largement ce ralentissment. La figure 4.13 montre différents résultats obtenus avec PrEd, imPrEd et imPrEd avec les arêtes flexibles.

4.5 Conclusion

Cette série de travaux sur la visualisation de diagramme d'Euler a été un réel succès. Nous avons réussi à définir correctement les caractéristiques d'une visualisation de diagrammes d'Euler. Puis, nous avons proposé un premier algorithme pour générer ces visualisations. Nous avons cherché à optimiser notre méthode pour améliorer ses performances en temps de calcul et en qualité.

Comme dans le cas du dessin de graphes et de la visualisation de graphes, on peut distinguer le dessin de diagrammes d'Euler et la visualisation de diagrammes d'Euler. Bien que notre méthode soit efficace et robuste, la complexité des chevauchements que nous rencontrons dans les jeux de données réels reste très difficile à interpréter. L'exemple de la figure 4.8 le montre. La mise en place de techniques interactives pour la visualisation de ce genre de diagramme est une solution. Les travaux de Riche et al. [79] donnent une première piste.

Le couplage de la visualisation de diagrammes d'Euler avec la visualisation de graphes soulève de nombreux défis. Dans la thèse de Paolo Simonetto [85], on peut trouver quelques résultats des travaux que nous avons commencé sur ce sujet. L'aboutissement de ces recherches nous permettra de créer, sur les diagrammes d'Euler, des interactions telles que celles présentées dans le chapitre précédent.

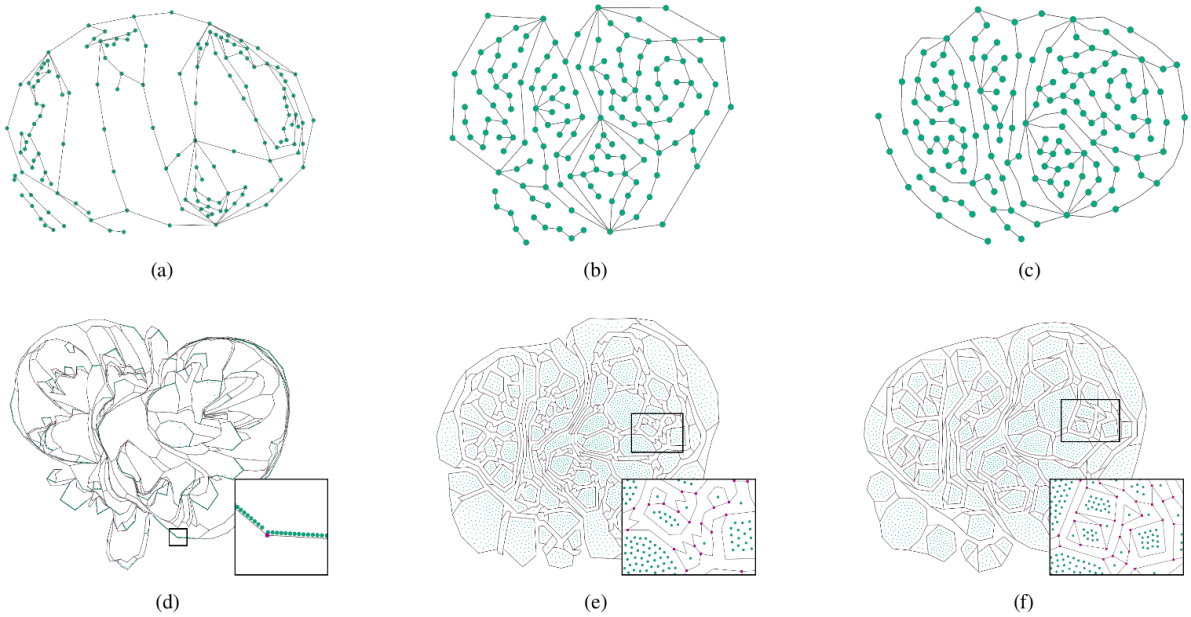


FIGURE 4.13 – Amélioration obtenue avec notre méthode dans le cadre de la génération de diagramme d'Euler. La première ligne représente le résultat obtenu après l'application de PrEd ou ImPred sur le graphe d'intersections. La deuxième ligne montre les résultats finaux avec l'insertion des éléments de chaque ensemble. (**a**, **d**) Résultats avec PrEd. (**b**,**e**) ImPrEd simple. (**c**, **f**) ImPrEd avec arêtes flexibles.

Chapitre 5

Visualisation des arêtes

Les travaux que nous présentons dans ce chapitre portent, non plus sur le placement des sommets ou la représentation de groupes de sommets, mais sur la représentation des arêtes dans un réseau.

Dans le projet Spangéo, projet interdisciplinaire informatique et géographie, j'ai constaté que les géographes manipulent, retravaillent et vérifient leurs données manuellement. Cette observation m'a fait prendre conscience que, pour supporter leurs processus d'analyse, nos techniques de visualisation devaient être encore plus interactives.

Si l'on considère qu'un système est interactif, certaines contraintes liées au monde des représentations statiques doivent être abandonnées. C'est le cas du dogme du dessin de graphe qui considère qu'il faut en priorité éviter les chevauchements d'arêtes et leurs croisements.

De ce fait, nous avons pris en compte, dans nos algorithmes de visualisation, le fait que certaines contraintes pouvaient être abandonnées grâce aux techniques d'interactions existantes.

Pour mener ces travaux, j'ai co-encadré la thèse d'Antoine Lambert. Le but de ces recherches a été de tirer parti des architectures modernes (multi CPU ou multi GPU) pour créer de nouvelles visualisations interactives. La progression de la puissance des machines permet d'effectuer des calculs en temps réel sur des ordinateurs d'entrée de gamme. Ces nouvelles possibilités doivent être utilisées pour produire des systèmes de visualisation plus performants. Par exemple, l'application de filtrage par noyaux gaussiens est quelque chose que nous pouvons faire en quelques millisecondes aujourd'hui et cela, même sur des images de plus d'un million de pixels. Certains algorithmes quadratiques se parallélisent tellement bien sur des machines GPU qu'ils peuvent devenir "linéaires" [1] si le nombre d'éléments à traiter est inférieur à quelques milliers [2]. Cette progression permet d'utiliser des méthodes quadratiques et même parfois cubiques dans nos systèmes interactifs.

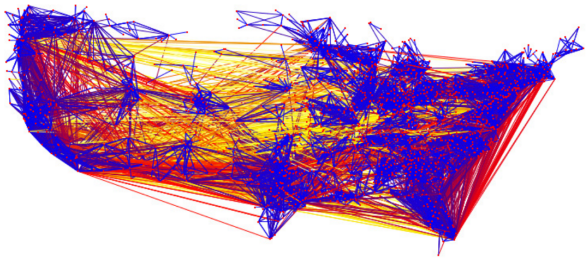


FIGURE 5.1 – Graphe de taille moyenne dessiné avec des arêtes en ligne droite. Les sommets du graphe représentent les villes principales des États Unis et les arêtes montrent les flux migratoires des citoyens américains entre ces villes. Le graphe est composé de 1715 sommets et de 9778 arêtes. Chaque sommet représente une ville et est géolocalisé. Dans cette image, les chevauchements et croisements d'arêtes rendent l'analyse visuelle très difficile.

La problématique des trois travaux que je présente ici est la suivante : étant donné un ensemble de sommets avec des positions fixées, comment créer un système interactif permettant de

1. Attention, la complexité des algorithmes ne change pas, c'est le temps d'exécution qui change. Par exemple, pour un algorithme en $O(n^2)$, parfaitement parallélisé, tant que le nombre d'éléments est inférieur au nombre de processeurs, le temps d'exécution progresse linéairement même si le nombre d'opérations effectuées reste quadratique.

2. Les machines destinées au grand public peuvent posséder plusieurs milliers d'unités de traitements par GPU.

comprendre les interconnexions dans ce réseau ? La figure 5.1 est un exemple caractéristique des réseaux que nous voulions visualiser. Pour attaquer ce problème, nous avons décidé d'explorer une technique (nouvelle à l'époque) appelée *Edge Bundling* [3]. Le regroupement en faisceaux d'arêtes consiste à fusionner des arêtes dans le dessin. En les fusionnant, on diminue le nombre de pixels nécessaires pour la représentation des arêtes et on obtient un dessin visuellement plus clair. Bien que contraire aux habitudes jusque-là utilisées en dessin de graphes, cette technique a été plébiscitée par la communauté de visualisation de graphes. La méthode de *Edge Bundling* que nous avons proposée est une méthode basée sur le routage dans les graphes planaires. Comme toutes les méthodes de *Edge Bundling*, elle nécessite des interactions pour lever les confusions introduites par les regroupements en faisceaux d'arêtes. Pour supporter ces interactions, nous avons travaillé sur des méthodes rapides de rendu de faisceaux en utilisant le GPU.

Pour plus de détails sur ces travaux, le lecteur peut se référer aux articles ci-dessous :

- Antoine Lambert, Romain Bourqui et David Auber. "Winding Roads : Routing edges into bundles". Dans : *Computer Graphics Forum* 29.3 (2010), p. 853–862
- Antoine Lambert, Romain Bourqui et David Auber. "3D Edge Bundling for Geographical Data Visualization". Dans : *Proceedings of the 14th International Conference on Information Visualization (IV'10)*. Royaume-Uni, juil. 2010, p. 329–335
- Antoine Lambert, David Auber et Guy Melançon. "Living flows : enhanced exploration of edge-bundled graphs based on GPU-intensive edge rendering". Dans : *Proceedings of the 14th International Conference on Information Visualization (IV'10)*. Royaume-Uni, juil. 2010, p. 523–530

5.1 Etat de l'art

Simplification des arêtes

Routage d'arêtes : L'un des premiers essais pour réduire l'occlusion dans les dessins de graphe a été fait par Dobkin et al. [34]. Pour augmenter la lisibilité des représentations, ils ont proposé de réduire le nombre de croisements et d'empêcher les chevauchements entre les sommets et les arêtes. Ceci permet d'utiliser des sommets de tailles variables. Dans [34], Dobkin et al. ont donné une méthode utilisant un graphe de visibilité et un algorithme de routage par la méthode des plus courts chemins pour enlever les chevauchements entre les sommets et les arêtes. La technique a été étendue par Wybrow et al. [105] pour le routage incrémental de connections. Finalement, Dwyer et Nachmanson [36] ont présenté une heuristique rapide pour calculer une approximation d'un graphe de visibilité. Ce résultat permet de réduire la complexité des approches basées sur les graphes de visibilité et rend possible le traitement des graphes de plus grandes tailles. Ces approches réduisent notablement l'occlusion dans la visualisation.

Techniques interactives : Wong et al. ont donné dans [104, 103] des techniques interactives pour enlever l'occlusion autour du centre d'intérêt de l'utilisateur. Les arêtes proches du centre d'intérêt sont déformées à la manière d'un *fish-eye* mais on conserve la position des sommets. Avec cette technique, la représentation est désambiguïsée localement mais la visualisation globale n'est pas améliorée.

Dessin de graphe par confluant : La communauté de dessin de graphes a proposé une représentation particulière appelée *Confluent Graph Drawing*. Dans un dessin par confluant, un graphe, non nécessairement planaire, est représenté sans croisement d'arêtes. Dans cette technique, les groupes d'arêtes qui se croisent sont représentés avec des lignes courbes qui se chevauchent. L'algorithme de Dickerson et al. [33] est basé sur la détection de cliques maximales et de bi-cliques (graph bipartie complet). Ensuite, les arêtes sont agrégées pour obtenir un dessin planaire de ces sous-graphes non planaires. Cette méthode donne de bons résultats mais elle ne peut pas être appliquée à tous les graphes (se référer à [33] pour plus de détails).

Partitionnement des sommets : Phan et al. [77] ont présenté une technique de génération de diagrammes de flux basée sur le partitionnement géométrique des sommets. Les arêtes sont routées le long des branches de l'arbre formé par la hiérarchie de clusters obtenue pendant le partitionnement.

3. Peut se traduire par : regroupement en faisceaux d'arêtes.

Cette idée a été également utilisée par Holten [61] pour améliorer la représentation des relations dans des données hiérarchisées. Le principal problème de ces techniques est qu'elles nécessitent un partitionnement des sommets pour router les arêtes. Cela rend ces méthodes peu flexibles dans le cas général.

Partionnement des arêtes : Gansner et Koren [51] ont donné une amélioration pour le dessin circulaire de graphe. Leur algorithme route les arêtes, à l'intérieur ou à l'extérieur du cercle formé par les sommets [4]. Les arêtes routées à l'intérieur sont agrégées ensemble en utilisant un algorithme de partitionnement. Ce partitionnement tente d'optimiser l'utilisation de l'espace. Une autre méthode, basée sur le partitionnement d'arêtes, a été introduite par Cui et al. [29]. Dans cette technique, ils utilisent une approche géométrique pour déterminer les arêtes à fusionner. L'idée principale est de construire manuellement ou automatiquement un maillage de contrôle. Ce maillage est ensuite utilisé pour déterminer les arêtes qui pourraient être fusionnées. Le choix d'agréger ou non les arêtes est déterminé par un algorithme de partitionnement qui teste l'orientation géométrique des arêtes. Un lissage est appliqué, a posteriori, afin d'éviter un effet zigzag dans le dessin. Holten et van Wijk [62] ont proposé une méthode basée sur un algorithme par modèle de forces. Cet algorithme permet d'enlever l'occlusion dans n'importe quel dessin de graphes dont les sommets sont fixés. Leur heuristique fonctionne de la manière suivante : les arêtes sont découpées en petits segments en ajoutant des sommets temporaires. Puis, une mesure de similarité est calculée entre les arêtes pour déterminer quels sont les sommets qui doivent s'attirer. Ensuit, les sommets temporaires sont connectés par des arêtes temporaires et un algorithme par modèle de forces est appliqué sur ce nouveau graphe. Cette opération va progressivement agréger les arêtes du graphe et créer le résultat attendu.

Amélioration des faisceaux

Lissage de courbes : Dans toutes les méthodes de *Edge Bundling*, les arêtes droites du graphe original sont remplacées par des arêtes courbes. L'utilisation de courbes à la place de lignes brisées permet un meilleur suivi de trajectoire par le cerveau. De plus, les dessins sont plus esthétiques. Dans [61], Holten utilise des B-Splines cubiques pour rendre ses *bundles* [5]. En utilisant ce type de courbes paramétriques, on a un contrôle local de la forme de la courbe. Cela permet à Holten d'agréger plus ou moins ses *bundles*. Zhou et al. [106] utilisent un autre type de spline : les courbes de Bézier et les courbes de Catmull-Rom. Holten et al. [62] et Cui et al. [29] ont proposé une autre méthode. Ils lissent les arêtes représentées par des lignes brisées pour les transformer en courbes.

Coloration des arêtes : Pour rendre les techniques de *Edge Bundling* plus efficaces, on peut jouer sur la couleur des arêtes ainsi que sur leur opacité. L'utilisateur va ainsi mieux percevoir le nombre d'arêtes qui ont été fusionnées. Dans [29], la couleur d'un faisceau est mis en corrélation avec la direction de l'arête originale lui correspondant. Une technique similaire est utilisée dans [61], mais la direction est encodée par un gradient de couleur allant d'une couleur fixée pour toutes les sources vers une couleur fixée pour toutes les destinations. Dans [61], l'opacité est mise en corrélation avec la longueur des arêtes. Les longues courbes sont plus transparentes que les petites. Ainsi, on conserve la visibilité des petites courbes. Dans [29], l'opacité d'un pixel est en corrélation avec le nombre de segments qui passe par lui. Une autre technique pour estimer la quantité de chevauchements a été proposée dans [62]. Le GPU est utilisé pour compter le nombre de fois qu'un pixel est dessiné. On obtient une carte de densité. Cette carte est ensuite utilisée pour faire la coloration des *bundles*.

5.2 Winding roads

L'objectif de nos recherches dans ce domaine a été de mettre en place une méthode simple et intuitive pour le regroupement d'arêtes en faisceaux. Lors de nos expérimentations de la méthode de Cui [29] [6], nous nous sommes rendu compte que plusieurs étapes étaient très compliquées à mettre en œuvre et qu'elle ne permettait pas un contrôle assez précis pour être réutilisée dans d'autres types

4. Dans les dessins circulaires, tous les sommets sont placés sur un cercle.

5. J'utilise fréquemment le terme *bundle* pour désigner un faisceaux d'arêtes.

6. A l'époque, cette méthode était la meilleure méthode disponible pour les graphes généraux

de visualisations.

Nous avont mis en place une méthode de regroupement très intuitive qui réduit considérablement l'occlusion dans les visualisations de graphes. Notre méthode est basée sur une discrétisation de l'espace. Cette discrétisation nous procure un maillage planaire. Ce maillage est construit en fonction du graphe original et il sert de support pour le re-routage de nos arêtes en *bundles*. Pour générer cette discrétisation rapidement, nous avons tout d'abord utilisé une discrétisation basée sur des *QuadTree* [41]. Bien que très efficace d'un point de vue algorithmique, cette discrétisation construit des cellules rectangulaires et par conséquent génère des faisceaux avec des changements abruptes de direction. Nous avons également expérimenté l'utilisation de diagramme de Voronoï [97]. Il sont tout aussi efficaces d'un point de vue algorithmique que les *QuadTree* [41]. Cette discrétisation pose le problème des *bundles* en zigzag. Pour remédier à ce problème, nous avons utilisé simultanément une décomposition en *QuadTree* [41] et un diagramme de Voronoï. Le *QuadTree* [41] nous sert à ajouter des points temporaires dans notre grille afin de sur-discrétiser le diagramme de Voronoï dans les régions denses du graphe. La Figure 5.2 montre un exemple de la discrétisation que nous obtenons sur le graphe du réseau aérien de l'année 2000.

A partir de la discrétisation de l'espace, nous allons pouvoir router à nouveau les arêtes du graphe original. La métaphore que nous avons voulu respecter dans le routage est la métaphore d'un réseau routier : plus il y a de chemin qui passent par une route/arête de la grille/carte routière, plus il faut que cette route/arête soit une voie rapide/un gros faisceau. Notre technique de routage consiste ainsi à pondérer les arêtes de notre discrétisation et à calculer des plus courts chemins sur cette grille. Une fois que nous obtenons les plus courts chemins, nous pouvons réévaluer la pondération de la grille de départ. Ainsi, à chaque itération de notre méthode, les arêtes seront de plus en plus agrégées. Afin de garantir des temps d'exécution rapide, nous avons proposé une version parallélisée de notre méthode permettant d'obtenir une méthode de regroupement en faisceaux d'arêtes plus rapide que toutes les autres à ce moment-là. La figure 5.3 montre le résultat que nous obtenons sur le graphe des migrations aux USA.

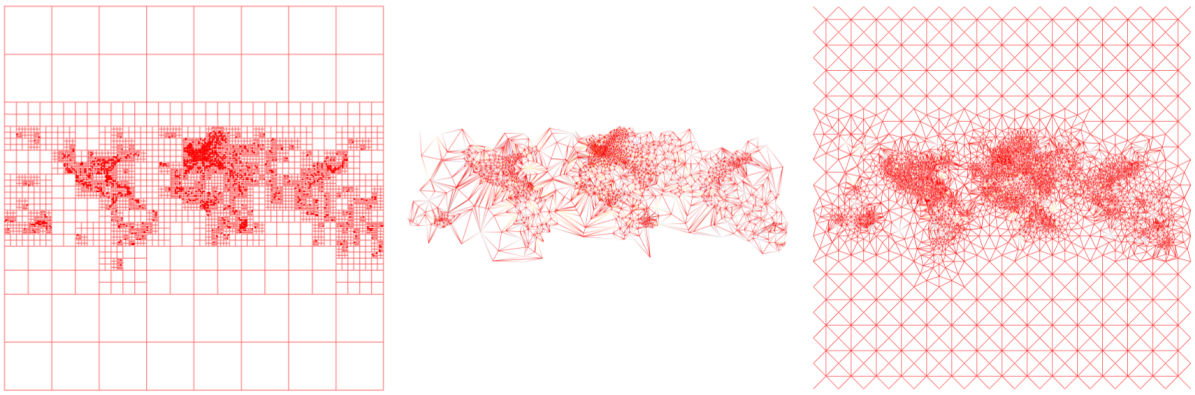
Un des gros problèmes des méthodes de *bundling* existantes est qu'elles peuvent faire passer des gros faisceaux au-dessus des zones denses en sommets. D'un point de vue de la perception, l'utilisateur a donc l'impression que des routes existent entre certains sommets alors que cela est faux. Par exemple, si l'on route des arêtes des États Unis vers l'Asie en passant au-dessus de l'Europe, l'utilisateur pensera que l'Europe est reliée aux USA et à l'Asie mais omettra peut-être que les USA sont reliés à l'Asie.

Un des points forts de notre méthode est la flexibilité offerte par l'utilisation d'une grille pour effectuer le routage. Dans nos recherches nous avons mis en évidence un cas remarquable d'utilisation de notre grille pour réduire l'occlusion dans le dessin. Pour résoudre le problème des zones denses, nous n'utilisons pas une distance euclidienne comme pondération initiale de notre grille. Nous allons plutôt favoriser l'utilisation des grandes arêtes de la grille en utilisant des distances non euclidiennes. Cette opération est effectuée en montant à la puissance toutes les distances de la grille de départ ($w = w_{init}^{\alpha}, \alpha < 1.$). La figure 5.4 montre les différents niveaux de réduction de l'occlusion que nous obtenons.

Toutes les méthodes de *bundling* utilisent une technique de coloration pour faire ressortir au mieux le nombre d'arêtes qui ont été fusionnées dans un *bundle*. Ceci permet de mieux comprendre l'image produite. Nous avons proposé une nouvelle méthode basée sur la technique du *Graph Splatting* [73] et du *Bump Mapping* [20]. Notre *Splatting* consiste à appliquer une convolution gaussienne sur tous les pixels des arêtes que nous dessinons. Cette opération nous permet de calculer la densité d'informations représentée sur chaque pixel de nos images. Nous utilisons le *Bump Mapping* pour transformer ces informations en carte d'élévation. Cela donne un effet 3D à l'image de départ. Nous avons montré que la complexité de ce genre de techniques n'est plus un frein si nous utilisons une implémentation GPU. La figure 5.5 montre le résultat de l'application de notre méthode sur le graphe des migrations aux USA.

5.3 Faisceaux d'arêtes 3D

Comme précisé dans l'introduction, les premières images qui peuvent être considérées comme du



(a) (b) (c)

FIGURE 5.2 – Trois discrétisations possibles du plan à partir d'un même graphe. Le graphe utilisé ici est celui modélisant le trafic aérien mondial de l'année 2000. Chaque sommet est un aéroport géolocalisé et chaque arête représente un vol direct entre deux aéroports. (**a**) En utilisant un *QuadTree* [41] (37395 sommets/69102 arêtes). (**b**) En utilsant un diagramme de Voronoï (4531 sommets/13558 arêtes). (**c**) En utilisant notre méthode hybride combinant le *QuadTree* et le diagramme de Voronoï (10146 sommets/30315 arêtes).

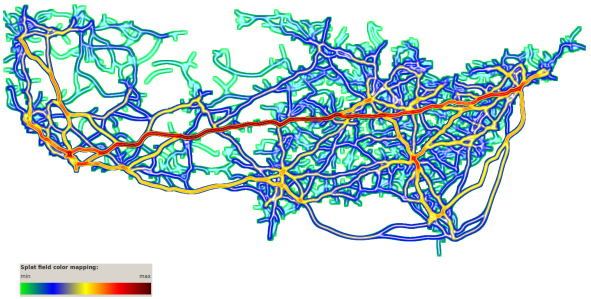


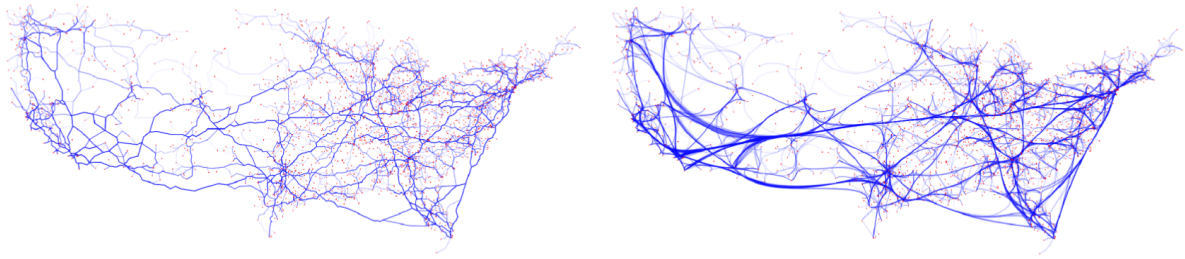
FIGURE 5.5 – Exemple de coloration obtenu sur le graphe de la figure 5.1. Le *splatting* a été obtenu en utilisant une convolution Gaussienne avec un noyau de rayon 5 et un écart type de 3. Les couleurs indiquent la densité d'arêtes dans les *bundles*. Dans cet exemple, on voit clairement que la majorité du flux migratoire est un flux est-ouest ou ouet-est.

bundling ont été produites par Charles Minard en 1862. L'objectif était de représenter des flux sur une mappemonde. Notre méthode en deux dimensions fonctionne correctement sur ce type de graphe. Cependant, la généralisation de celle-ci en trois dimensions n'est pas directe si nous voulons produire des représentations *bundlées* sur le globe. Notre objectif a donc été d'étendre nos résultats pour gérer le *bundling* sur une surface sphérique.

L'extension de notre méthode en trois dimensions est possible sans aucune modification si nous utilisons les versions 3D des méthodes que nous utilisons en 2D. Les *QuadTree* [41] deviennent alors des *OcTree* et les diagramme de Voronoï 2D sont remplacés par des diagrammes de Voronoï 3D. Le reste de notre méthode étant basée sur de l'algorithmique de graphe, aucune modification n'est nécessaire. Nous avons expérimenté cette méthode et elle fonctionne correctement pour générer des *bundles* 3D.

Le problème du *bundling* sur la sphère est différent car il faut contraindre les *bundles* à rester sur la surface de celle-ci. Dans les figures 5.6 (a) et 5.6 (d) on peut voir que les arêtes du graphe traversent la sphère. Les *bundles* générés avec notre *bundling* 3D vont aussi traverser la sphère ce qui ne produira pas la représentation escomptée.

Pour remédier à ce problème, nous avons mod-



(a) (b)

FIGURE 5.3 – Deux exemples de regroupement en faisceaux d'arêtes obtenus avec notre méthode sur le graphe des flux migratoires présenté dans la figure 5.1. (**a**) Les arêtes sont représentées avec des lignes brisées. (**b**) Les arêtes sont lissées en utilisant des courbes de Bézier. Quand les arêtes sont rendues avec des courbes de Bézier, leur forme met en évidence les flux entre différentes régions du graphes. Par exemple ici des flux est-ouest/ouest-est ou bien nord-sud/sud-nord. Cependant, l'utilisation des courbes de Béziers peut recréer des chevauchements entre les arêtes et les sommet.

ifié la méthode de génération de notre grille. Nous appliquons un prétraitement qui consiste à ajouter uniformément des sommets temporaires à l'extérieur de la sphère. L'objectif est de garantir, pour chacun de nos sites, que chacune des cellules 3D de Voronoï aura au moins un point à l'extérieur de la sphère. Cependant, dans ce cas, nous pouvons toujours obtenir des arêtes qui sont routées à l'intérieur du globe. Pour garantir que notre routage sera bien à l'extérieur du globe, nous ajoutons aussi des sommets à l'intérieur du globe. En fixant correctement les rayons intérieurs et extérieurs nous pouvons garantir que nos routages seront toujours à l'extérieur du globe. La figure 5.6 montre le résultat que nous obtenons sur la représentation du trafic aérien mondial.

Nous avons aussi amélioré notre technique de rendu GPU pour permettre son utilisation dans ce cas précis. Nous utilisons l'illumination Phong [19] pour chaque pixel de l'image en fonction des informations de densité et de la direction de l'oeil d'observation. Les couleurs finales sont obtenues à partir de la carte d'illumination et à partir de la carte de couleur (couleurs initiales de nos arêtes dans le graphe). Les figures 5.6 (c) et 5.6 (f) montrent les effets de relief que nous obtenons. Plus l'effet est fort, plus le nombre d'arêtes agrégées dans le *bundle est* important. La méthode s'exécute entièrement sur le GPU et permet une interaction fluide, ce qui est primordial pour la visualisation de données en trois dimensions.

5.4 Living flows

Le dernier papier de cette thématique a été un papier purement algorithmique et technique. L'objectif de ce travail était d'améliorer les méthodes existantes que nous avions utilisées afin de performer l'expérience visuelle de nos utilisateurs.

Notre méthode de *bundling* est rapide et elle génère un très grand nombre de points d'inflexions sur nos lignes brisées. Ces points d'inflexions sont nécessaires pour éviter les chevauchements entre les arêtes et les sommets et générer des *bundles* corrects. Comme nous l'avons vu précédemment, nous pouvons utiliser un lissage pour réduire les points d'inflexions et permettre un suivi plus aisé des *bundles*. Plusieurs techniques de lissage sont envisageables : les courbes de Béziers, les B-splines ou encore les courbes de Catmull-Rom. Le problème que nous avons rencontré pendant nos expérimentations est que la génération de ces courbes à la volée était beaucoup trop coûteuse pour permettre une interaction fluide avec nos visualisations. Cependant, il est possible de pré-calculer les lissages pour certaines interactions élémentaires telle que l'interaction *zoom and pan*. Dans le cas du *bundling*, il est nécessaire de permettre à l'utilisateur de *debundler* localement afin de retrouver l'information originale. Ce type d'interaction nécessite de déplacer les points de contrôles des courbes et par conséquent de les recalculer. Ce recalcul est très coûteux en temps lorsque les courbes possèdent un grand nombre de points de contrôle. Nous avons rencontré ce

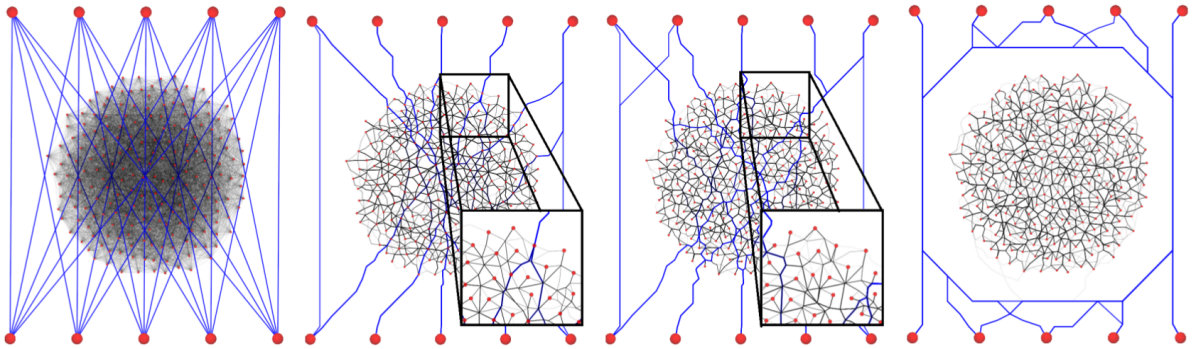


FIGURE 5.4 – Les différents types de réduction de l'occlusion que notre méthode supporte. (**a**) Un graphe d'exemple composé d'un graphe complet sur lequel nous avons superposé un graphe biparti complet. Les deux graphes forment deux composantes connexes. (**b**) Simplification par regroupement en faisceaux d'arêtes. Plusieurs arêtes peuvent emprunter le même segment de droite. Cela libère de l'espace dans le dessin et permet de mieux voir les sommets. (**c**) Réduction de l'occlusion par suppression des chevauchements entre les sommets et les arêtes. Si une arête passe sur un sommet, cela signifie forcément qu'elle connecte ce sommet. C'est une propriété capitale de notre technique de *Edge Bundling* car elle permet d'éviter beaucoup d'erreurs d'analyse. (**d**) Réduction en évitant les zones denses. Plus une arête est longue, plus on autorise de déviations de sa trajectoire pour rejoindre sa destination. Cela permet à une arête d'éviter une zone dense si elle ne connecte pas un sommet à l'intérieur de cette zone. Les avantages procurés par cette méthode sont les mêmes qu'en (c) mais à un niveau de détail différent.

phénomène lors de la mise en place de la technique d'interaction appelée *Bring And Go* introduite par Tominski et al. [92] et amélioré par Moscovich et al. [74].

Pour résoudre ce problème, nous avons proposé une approche entièrement basée sur l'utilisation du GPU pour générer nos courbes. Les gains que nous avons obtenus permettent de mettre en œuvre des techniques de *bundling* dans des systèmes interactifs. La figure 5.4 montre les gains que nous obtenons sur une carte graphique d'entrée de gamme ne possédant que 32 shaders. Les GPU plus récents sont équipés de plus de mille shaders ce qui permet d'utiliser nos méthodes sur des graphes *bundlés* ayant des dizaines de milliers d'arêtes.

Splines type	Architecture	Images/s
Bézier	CPU	0.68
Bézier	GPU	17.2
Cubic B-splines	CPU	12.79
Cubic B-splines	GPU	17.5
Catmull-Rom	CPU	6.95
Catmull-Rom	GPU	17.4

FIGURE 5.7 – Comparaison des performances entre notre méthode de rendu de Spline avec le GPU et un rendu avec le CPU lorsque l'on dessine des graphes *bundlés*. Le graphe utilisé possède 2000 arêtes. Chaque arête possède entre 4 et 87 points de contrôle. Chaque Spline a été générée avec une discrétisation de 100 points. Le CPU utilisé pour le test est un Intel(R) Core(TM) 2 Extreme CPU X9100 @ 3.06Ghz et la carte graphique est une NVidia Quadro FX 1700M possédant 32 unités de calcul (shaders).

5.5 Conclusion

Les travaux que j'ai présentés dans ce chapitre montrent une autre facette de nos recherches. Les recherches évoquées dans les chapitres précédents portaient sur le début du pipeline de visualisation. Avec ces recherches, nous avons montré qu'il était

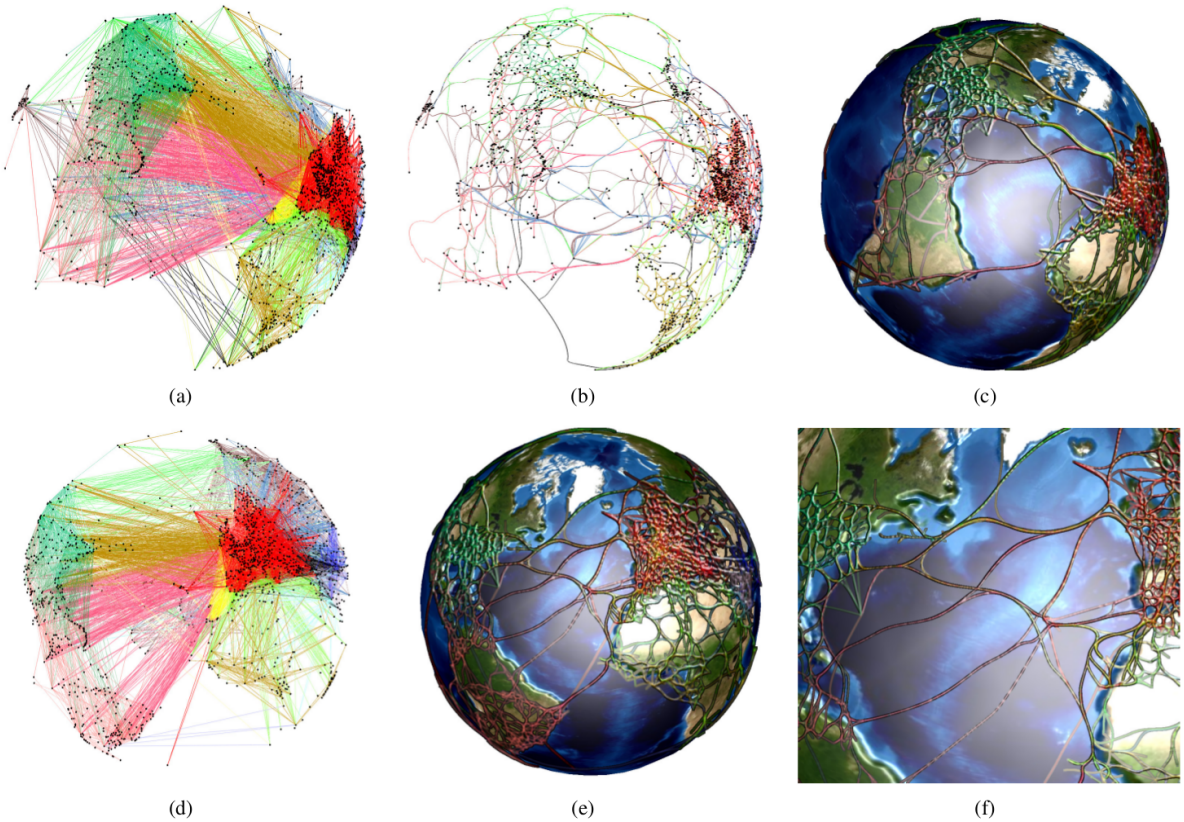


FIGURE 5.6 – (**a**) Réseau du trafic aérien mondial pour l'année 2000. Un sommet représente un aéroport géolocalisé et une arête représente un vol direct entre deux aéroports. Le graphe contient 1524 aéoroports et 16397 vols. (**b**) Le résultat que nous obtenons avec notre méthode de *bundling* 3D sphérique. Les arêtes sont rendues en utilisant des courbes splines cubiques. (**c**) Résultat que nous obtenons en appliquant notre nouvelle technique de coloration et de mise en relief par *Bump Mapping*. (**d**) Même graphe que (a) mais pour l'année 2004. Le graphe contient 1501 aéroports et 12360 vols. (**e**) Résultat obtenus en utilisant notre technique de *bundling* et de coloration ; (**f**) Une vue détaillée mettant en évidence l'efficacité du *Bump Mapping* pour faire ressortir la taille des *bundles*.

possible d'améliorer nettement la visualisation d'un graphe en post-traitement. De plus, nous avons montré que les capacités des GPU modernes permettent d'utiliser des techniques de rendu qui étaient jusque-là réservées à des applications non interactives. Ces travaux ont suscité un vif intérêt dans la communauté.

Le *bundling* soulève encore de nombreuses questions en visualisation d'informations. Il est encore difficile de comparer de manière rationnelle deux méthodes de *bundling*. Un *survey* sur le *bundling* ainsi que des évaluations rationnelles sont nécessaires pour clarifier toutes ces interrogations.

Chapitre 6

Conclusion et Perspectives

Dans ce rapport de recherche, j'ai présenté une partie des travaux que j'ai effectués depuis ma nomination au poste de maître de conférences de l'Université de Bordeaux 1. Dans chacun des thèmes que j'ai présentés : visualisation multi-échelles de graphes (cf. chapitre 2 et 3), visualisation de décompositions chevauchantes (cf. chapitre 4) et visualisation d'arêtes (cf. chapitre 5), nous avons proposé et publié des résultats significatifs.

Dans le domaine de la **visualisation multi-échelles de graphes**, nous avons proposé une nouvelle méthode basée sur la décomposition en structures topologiques [9]. Par la suite, nous avons amélioré et étendu cette méthode tant d'un point de vue algorithmique [8, 21] que d'un point de vue de l'interaction avec l'utilisateur [10].

Bien que de nets progrès aient été faits ces dernières années dans ce domaine, l'évolution des bases de données a engendré des problèmes beaucoup plus complexes que ceux que nous avons étudiés. Sur les jeux de données concrets, l'analyse de la dynamique semble être primordiale pour augmenter la pertinence des résultats des algorithmes de partitionnement de graphes. Ce dynamisme remet en cause de nombreuses parties des algorithmes. Par exemple, les méthodes de dessin multi-échelles de grands graphes ne sont absolument pas compatibles avec le dynamisme des données.

L'imprécision des méthodes de fragmentation laisse à penser que, pour de nombreuses tâches d'analyse, l'utilisateur expert muni d'outils d'édition de l'arbre de fragmentation est beaucoup plus performant qu'un algorithme automatique. Un travail important reste à faire pour permettre ce genre d'analyse sur les données actuelles. A ma connaissance, aucune recherche n'a été menée pour créer des techniques interactives d'édition d'arbre de fragmentation sur des grands graphes dynamiques.

Dans le domaine de la **visualisation de décompositions chevauchantes**, nous avons proposé la première méthode entièrement automatique [88]. Pour mettre en place cette méthode, nous avons, en premier lieu, formalisé le problème [87, 86]. Nous avons ensuite donné des outils algorithmiques pour la mise en oeuvre de cette méthode [89]. Cette thématique est en pleine évolution [80, 79]. Une des raisons qui l'explique est que les ensembles chevauchants permettent de visualiser des hyper-graphes.

Les hyper-graphes sont des structures rencontrées très régulièrement dans les graphes construits à partir des réseaux sociaux. Par exemple, si nous considérons le graphe construit à partir des publications scientifiques, nous obtenons comme entités : des articles, des auteurs et des citations entre articles. Pour construire un graphe, nous pouvons relier les auteurs à leurs articles puis relier les articles ensemble en fonction de leurs citations. Comme nous l'avons montré dans [31], ce graphe peut être analysé en construisant différents graphes ne mettant en évidence qu'un seul type d'entité à chaque fois. Cependant, il est clair qu'un article définit une relation entre tous ses auteurs et qu'une visualisation à base d'hyper-arête [1] rendrait plus efficiente l'analyse de ce type de données. L'utilisation des méthodes de visualisation de décompositions chevauchantes couplée avec une méthode d'interaction permettrait de proposer une méthode plus efficace pour l'analyse de ce type

1. Une hyper-arête modélise une relation n-aire. Elle permet ainsi de relier plusieurs entités simultanément

de données.

Dans le domaine de la **visualisation d'arêtes**, nous avons proposé une nouvelle technique d'*Edge Bundling* [71]. Tout en ayant des temps de calculs très performants, cette technique permet un contrôle beaucoup plus fin comparé aux autres méthodes [29, 62]. Nous avons ensuite étendu cette méthode au cas du *bundling* en trois dimensions [70]. Pour permettre l'utilisation de ces techniques dans des applications interactives, nous avons aussi étudié les techniques de rendu sur le GPU. Nos résultats permettent l'affichage d'un grand nombre de courbes complexes [69].

Bien que l'*Edge Bundling* suscite encore un vif intérêt, il s'agit seulement d'un outil supplémentaire dans la panoplie des techniques disponibles pour la visualisation de graphes. L'apparition de cette technique a cependant bouleversé le dogme qui consistait à toujours représenter les graphes avec des lignes droites et avec le minimum de croisements. A l'avenir, ce changement va permettre de créer des visualisations complètement différentes. Les résultats de Duncan et al. [35] sur les dessins Lombardiens ou les techniques de représentation de graphes avec des quilts [15, 5] en sont la preuve.

L'évolution des performances et des capacités graphiques de nos machines a également révolutionné les possibilités d'innovation en visualisation de graphes. Beaucoup de techniques, trop complexes pour être utilisées jusqu'à présent, peuvent être maintenant envisagées dans des systèmes interactifs. L'évolution des processeurs graphiques massivement parallèles en est une des causes. La future révolution des processeurs massivement parallèles doit, dès à présent, être anticipée dans le domaine de la visualisation d'informations. Parallèlement, les possibilités offertes par le *cloud computing* doivent être exploitées pour produire des systèmes garantissant une meilleure expérience visuelle.

6.1 Perspectives

Durand les dix dernières années, nous avons acquis une expérience significative dans le domaine de la visualisation de graphes. Cette expérience nous a conduit à réaliser des projets d'intégration de nos méthodes dans des chaînes de traitement complètes. La confrontation de nos recherches aux problèmes de nos partenaires industriels et académiques nous a permis de mieux comprendre les tâches que nos utilisateurs ont a résoudre. Je propose ici trois axes de recherches pour répondre aux grands défis soulevés par nos utilisateurs.

6.1.1 Masse de données

L'évolution de la quantité de données stockées est bien réelle. La majorité des problèmes que nous rencontrons nécessitent la mise en place de solutions basées sur des stockages externes et des architectures client-serveur. Les progrès dans le domaine du *cloud computing* ont permis de résoudre le problème de stockage de données. La visualisation d'informations reste, quant à elle trop souvent à la charge du client et ne profite donc absolument pas des capacités de calcul offertes par ces nouvelles technologies.

L'utilisation de ces nouveaux moyens demande de réinventer de nombreuses méthodes. La plupart des techniques de visualisation ont été créées pour fonctionner en possédant une connaissance globale des données à visualiser. Les algorithmes utilisés dans ces techniques ne sont donc pas adaptés à cette nouvelle technologie. Les approches multi-échelles sont cependant relativement proches des méthodes de *Map Reduce* utilisées en *cloud computing*. Il est donc fort possible que l'on puisse utiliser les résultats dans ce domaine pour traiter des données de très grandes tailles.

6.1.2 Données dynamiques hétérogènes

Comme indiqué ci-dessus, la quantité de données à traiter augmente. Cependant, dans un grand nombre d'applications, le nombre d'entités reste globalement constant. Par exemple, le nombre d'aéroports n'a pas varié de manière exponentielle ces dix dernières années. Le grand changement, c'est que nous avons maintenant, pour la grande majorité des données, l'historique complet de leur évolution. Par exemple, dans le cas des aéroports, nous avons l'évolution des lignes aériennes au cours du temps ainsi que l'évolution de la fréquentation de celles-ci. Nous appelons ces données des données dynamiques. La prise en compte de cette nouvelle

dimension (le temps) est un des défis majeurs en visualisation.

Un autre aspect important est la diversité des données que nous sommes amenés à traiter. Par exemple, pour l'étude du métabolisme d'un être vivant, il est nécessaire de croiser des données de puces ADN, des réseaux d'interactions ou encore des taxonomies construites par l'utilisateur. Cette diversité est liée à la nature même des problèmes que nos outils essayent de traiter. Dans la majorité des problèmes que nous avons rencontrés, il s'agissait d'étudier des systèmes complexes. Pour étudier ce genre de problèmes, il est nécessaire d'utiliser différentes métaphores visuelles et il est nécessaire de permettre une interconnexion cohérente de celles-ci. Les travaux de Viau et al [96] démontrent la pertinence de cette approche.

6.1.3 Visualisations esthétiques

En périphérie de la sphère scientifique, la démocratisation des techniques de la visualisation d'informations a donné lieu à la création d'une multitude de visualisations. Au premier abord, on peut considérer que ces visualisations ne sont que le reflet des travaux menés par la communauté de visualisation d'informations. Une étude plus approfondie nous montre que ces visualisations apportent souvent, de manière très empirique, un caractère esthétique aux visualisations existantes. Ces améliorations ont permis l'adoption de certaines techniques qui ne sont pourtant pas les meilleurs pour la communauté scientifique.

Ce phénomène soulève de nombreuses questions qui doivent être étudiées. Doit-on construire les systèmes les plus efficaces possibles ou les systèmes les plus agréables possibles ? Les visualisations que nous créons sont, pour la plupart, assez agressives. Les lignes brisées forment des angles abruptes, les contours forment des angles droits etc. . Il est assez facile de faire l'analogie des visualisations existantes avec les cartes géographiques proposées il y a une cinquantaine d'années. Les couleurs y étaient très saturées. De fait, on pouvait distinguer facilement les différentes régions, les zones de relief ou encore les océans. Ces cartes étaient visuellement très stressantes. On sait maintenant [93, 100], qu'il faut, en fonction de la surface d'une région, adapter le niveau de saturation pour limiter le stress généré par la visualisation. Les progrès sur ce point permettront une meilleure pénétration des résultats de recherche dans la vie publique.

La montée en puissance des ressources de nos machines accroît le degré de liberté dont nous disposons pour construire nos visualisations. Pour exploiter efficacement ces nouvelles possibilités, la collaboration avec des géographes spécialistes de la cartographie me semble indispensable. L'expertise qu'ils ont acquise en créant manuellement leurs visualisations doit être partagée avec la communauté de visualisation de graphes.

Chapitre 7

Annexe

David Auber

Candidature à l'HDR.

351 Cours de la libération
33405 Talence
+33 (0)6 09 17 33 95
+33 (0)5 40 00 28 77
david.auber@labri.fr
http://www.labri.fr/perso/auber

Curriculum Vitae

Etat civil

Nom et prénoms : Auber, David André Claude
Adresse : Le Bourg 47350 Lachapelle
Date et lieu de naissance : 15 Avril 1975 Poitiers (86)
Nationalité : Française
Situation familiale : PACS, 2 enfants

Parcours

2003–2012 **Maître de conférences section CNU 27**, *LaBRI, Université Bordeaux 1*, Délégation CNRS en 2009-2010 et INRIA 2010-2011.

1999–2002 **Doctorat en informatique**, *LaBRI*, Université Bordeaux I, *encadré par Maylis Delest*.
Sujets : Outils de visualisation de larges structures de données

1998–1999 **DEA**, *LaBRI*, Université Bordeaux I, *encadré par Nicolas Hanusse*.
sujet : Recherche sur les algorithmes de dessin de graphe planaires

depuis 2005 **Titulaire de la PEDR puis de la PES**.

Encadrements

Thèse en cours

2009–2012 **Antoine Lambert actuellement ATER**, *co-encadré à 80% avec Guy Melançon (20%)*.
Sujet : utilisation du calcul parallèle sur GPU et CPU pour la visualisation d'informations

Thèses soutenues

2004–2008 **Daniel Archambault actuellement assistant professor UK.**, *co-encadré à 50% avec Tamara Munzner (50%) de l'Université de British Columbia*.
Sujet : Feature-based graph visualization

2005–2008 **Romain Bourqui actuellement maître de conférences, Bordeaux**, *co-encadré à 80% avec Maylis Delest(20%)*, .
Sujet : Décomposition et Visualisation de graphes : Applications aux Données Biologiques

2007–2011 **Paolo Simonetto actuellement salarié en Italie.**, *co-encadré à 80% avec Guy Melançon(20%)*.
Sujet : Visualisation of Overlapping Sets and Clusters with Euler Diagrams

2008–2012 **Frédéric Gilbert, actuellement en recherche d'emploi**, *co-encadré à 80% avec Maylis Delest(20%)*.
Sujet : Modèles et méthodes pour l'analyse et la visualisation de données multidimensionnelles dynamiques

Divers

2003–2012 **Etudiants de L3 à M2**, *10 Projets Génie Log. (UB1), 4 projets de programations.(UB1)*, 2 Projets d'étude et de rercherche (UB1/ENSEIRB), 2 Stages de fin d'études (ENSEIRB), 2 Stages d'étudiants ENS (Cachan et Bretagne).

2005–2010 **Post-doc**, *Participation à l'encadrement des trois post-docs recrutés dans le thème*.

Invitations

Séminaire Dagsthul 2013: Drawing graph on the curve
Séminaire Dagsthul 2012: Putting data on the map
Séminaire Dagsthul 2011: Graph Drawing with Algorithm Engineering Methods
Keynote speaker IV 2012: Automatic generation of human made visualization (Annulé pour raison d'hospitalisation de ma fille, normalement reporté pour 2013.)

Distinctions

Seconde place au contest de visualisation d'InfoVis en 2003 et en 2004.
Bourse d'excellence: Visiting Junior Scholars Program au Peter Wall institute UBC Canada (2003)

Administration

2007–2012 **Responsable du thème Visualisation d'information**.

2011–2012 **Responsable équipe projet INRIA Gravité**, *intérim pour remplacer Guy Melançon*.

2011–2012 **Responsable commission poste MCF Mabiovis**, *Elaboration coordination des listes d'experts pour les auditions de maître de conférences.*.

2010–2012 **Membre de la commission consultative Informatique de l'UFR Maths-Info**.

2009 **Membre du comité de sélection Chaire INRIA (ENSEIRB)**.

2009 **Membre du comité de sélection Maître de conférences (IUT)**.

2006–2012 **Chef de projet pour le projet : Tulip**, *Tulip, (2 ingénieurs à temps complet depuis 2006, 4 ingénieurs à temps complet depuis le 13/10.2008*.

2007–2012 **Membre du conseil scientifique du LaBRI**.

2005–2006 **Responsable de la formation MIAGE Master 2**.

Administration de la recherche et expertise

Organisation de conférences : Graph Drawing 2013, EGC 2011, EUROVIS 2010
Chaires de session : Information Visualisation 2009, Eurovis 2010, poster d'Eurovis 2010.
Membre de Comités de programme : Graph Drawing 2010, PacificVis 2010, Eurovis 2009, 2010, Softvis 2007.
Relecteur Journals : TVCG (2007–2012), JGGA (2011), Graphic Forum (2008–2010), Information Visualization [2012]

Relecteur Conférences : SiGGraph (2012), Infovis (2004–2010), Vis (2009), Vast (2009), Eurovis (2008–2011), IHM (2004), IV (2006), Majestic (2006)
Expertise : Netherlands Organisation for Scientific Research (2011): Innovational Research Incentives Scheme , ANR (2009): International Non-Thematic
Membre du jury de 3 thèses.

Collaborations et mobilité

Collaboration avec Helen Purchase (1 mois au LaBRI)
Collaboration avec James Abello (15 jours au LaBRI).
Collaboration avec Alexandru Telea (6 mois au LaBRI)
3 mois passés à UBC, Tamara Munzner, University of British Columbia, Vancouver (Canada)
1 mois passé à UQUAC, Yves Chiricota, Université du Québec à Chicoutimi (Canada)
1 mois passé à Amsterdam CWI, Ivan Herman, national research institute for mathematics and computer science in the Netherlands.

Financements publics, leader, (247Keuro + 700Keuro soumis)

ACI jeunes chercheur : exploration et visualisation interactive de cubes de données. 2004 – 2007 75 Keuro de financement local.
ANR SPANGEO : Spatial Networks in Geography, 2006 – 2008 7,5Keuro
Projet Européen RAISME, Rapid Application Innovation for Services For Multidimensional Enterprises. CAPACITIES (Research for SMEs) 2010 – 2012 170 Keuro de financement local, leader local.
Projet REQUEST: REcursive QUEry and Scalable Technologies: investissements d'Avenir Développement de l'Economie Numérique Appel à projets « cloud computing » n°3 – Big data. 700Keuro de financement local (en cours d'évaluation), leader local.

Contrats industriels, leader (270 KEuros)

Projet OSINTLAB avec la société Thalès 100Keuro
Projet OSINTLAB v2 avec la société Thalès 55Keuro
Projet OSINTLAB v3 et LOGINTLAB V1 avec la société Thalès 115Keuro
Formation C++ pour I2S et Tulip pour thalès.

Participations significatives à des projets (min 20%)

Navigation dans les grands graphes, Ministère ACI Masse de données. 2003-2005 115Keuro
FIVE, TechLog 2006 – 2008 90 Keuro
TANGUY From Text to Arguments through Networks with Goals and User Initiative, 2009-2012, 261Keuro de financement local.
ANR Jeunes Chercheurs EVIDEN, (Exploration and Visualization of Dynamic Relational Data) 2010-2014, 243,017 Keuros de financement local.

Enseignements 1500h, 241h cours, 1256h TD

Proportion Cours/TD: 2002 (5%)...2008 (25%)...2011 (50%)
Master 2, Architecture logicielle, Cours et TD, 2011–2005.
Master 2, Algorithmes pour la bio-informatique et la visualisation, Cours et TD 2011–2008.
Licence 2, Enseignement en ligne parcours international et suivi des étudiants, 2011

Licence 2, Programmation 2, Cours et TD, 2011
Licence 3, Bibliothèque graphique IUT, Cours et TD 2011
Master 2, Génie logiciel - conduite de projet, TD 2009–2006
Miage, Systèmes d'information, architectures logiciels, TD 2008–2007
Master 1, Programmation orientée par objet, TD 2008
Master 2, Stage en entreprise 2006
Master 2, Stage C++, Cours et TD 2003–2009

Activités de recherche

Mes recherches portent sur l'analyse, la compréhension et l'exploration de grande quantité de données. Elles se placent dans un secteur de recherche en pleine effervescence : la visualisation d'informations. Cet intérêt est dû à l'augmentation permanente du nombre d'informations stockées sur des supports informatiques. D'ici les trois prochaines années, on estime que ce nombre aura doublé. L'objectif de mes travaux de recherche est l'élaboration d'outils permettant la visualisation de telles données. Plus précisément, ces recherches s'orientent vers les données pouvant être modélisées sous forme de graphes. Un des exemples connu est le graphe induit pas la structure des pages WEB. D'autres secteurs, comme celui de la bio-informatique, disposent également de données sous forme de graphes, par exemple les structures secondaires d'ARN ou les réseaux d'interaction de protéines. La figure 1 donne un exemple d'application de nos recherches sur le graphe d'interconnexion des acteurs de cinéma.

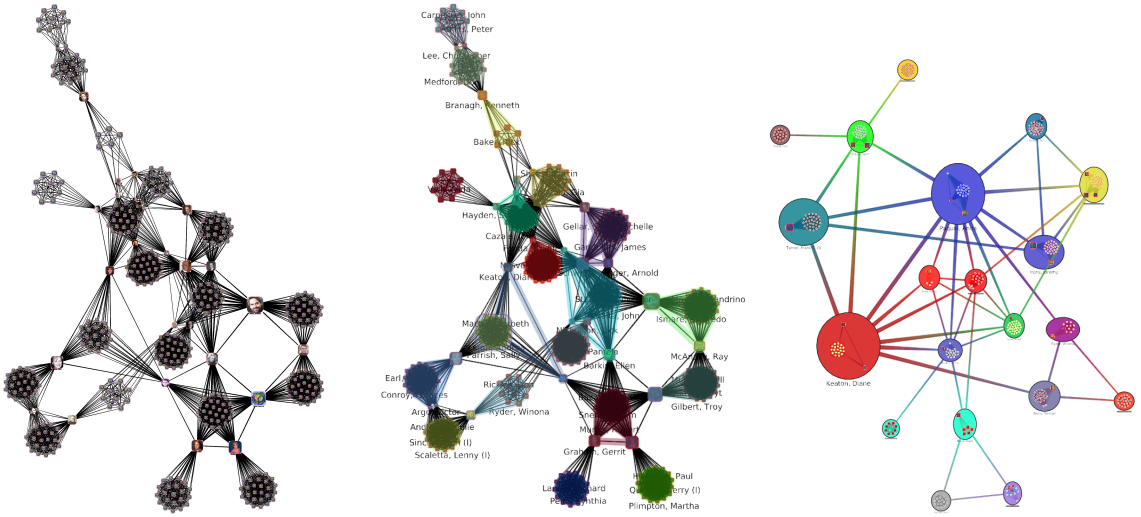


Figure 1 : Exemple de Visualisation multi échelle d'un sous graphe de "Internet Movie Data Base". Dans ce graphe les sommets sont des acteurs de cinéma et deux acteurs sont connectés si et seulement s'ils ont joué dans un film ensemble. (gauche) Un sous graphe de IMDB (milieu) Un algorithme de clustering recherche récursivement des clusters dans le graphe de départ, les enveloppes de couleur représentent le premier niveau de la hiérarchie. (droite) Les clusters sont récursivement agrégés en meta noeuds pour créer une représentation abstraite des données. L'utilisateur peut alors interactivement entrer ou sortir d'un meta noeud. Cette méthode permet ainsi une navigation multi échelle dans les données.

De nombreux résultats existent dans les domaines de recherche impliqués dans le processus de visualisation de données. Cependant, l'application directe de ces résultats sur ces jeux de données réels soulève de nouveaux problèmes pratiques et théoriques qui requièrent de nouveaux travaux de

recherche. En effet, lorsque nous nous plaçons dans le contexte de la visualisation d'informations, l'efficacité d'une méthode est liée à de multiples facteurs qui sont difficiles à prendre en compte sans une expérimentation réelle. Ces principaux facteurs sont : la taille des données (plusieurs milliers voire plusieurs millions d'éléments), le système de perception visuelle de l'être humain et les spécificités liées aux domaines d'applications.

Je travaille ainsi sur la mise en place d'outils pour la visualisation d'informations. Ces recherches se décomposent en plusieurs parties : les structures de données, les architectures logicielles, les algorithmes de fragmentation et les algorithmes de dessin. Les trois paragraphes suivants décrivent brièvement ces trois axes recherches.

Architecture logicielle

Pour permettre l'expérimentation de nouveaux outils de visualisation, mon travail s'est tout d'abord porté sur la mise au point ainsi que sur l'implémentation d'un environnement logiciel ("Framework") dédié à la visualisation de grands graphes (de l'ordre de 1.000.000 d'éléments). Dans celui-ci, un intérêt tout particulier a été porté à la gestion de la fragmentation ("clustering") ainsi qu'à l'étude du compromis, nécessaire pour les grands graphes, entre la complexité en temps et la complexité en espace des algorithmes. Le système obtenu, nommé Tulip1, a été présenté à la conférence « Graph Drawing ». Suite à cette conférence, un livre sur les logiciels de dessin de graphes a été publié. Un des chapitres [1] que j'ai rédigé est consacré aux structures de données et aux algorithmes mis en place dans Tulip. Ce logiciel permet d'expérimenter rapidement des méthodes de fragmentation de graphes, des algorithmes de dessin de graphes ainsi que des techniques de visualisation basées sur les paramètres combinatoires des graphes.

Depuis 2005, le LaBRI finance un ingénieur de recherche sur le projet Tulip. Depuis 2008, l'INRIA finance aussi un ingénieur à temps complet sur le projet. Notre équipe continue elle aussi son investissement et elle finance sur budget propre deux ingénieurs à temps complet. La qualité de plateforme et notre capacité à la faire évoluer nous a permis de proposer plusieurs projets et de les mener à bien. Par exemple, elle a permis au géographe d'utiliser les résultats théoriques basés sur la connectivité des graphes pour permettre l'étude de la structure des réseaux d'interconnexion des aéroports. Ce logiciel représente maintenant la vitrine de notre équipe, il est utilisé et téléchargé des milliers de fois chaque moi.

Algorithmes de fragmentation

Même lorsque le nombre d'informations à traiter est important, les systèmes de visualisation doivent produire une représentation interactive et intelligible. Pour respecter ces contraintes, des méthodes de simplification des objets à visualiser sont indispensables. Ils existent deux méthodes de simplifications : le filtrage ou l'agrégation. Les derniers résultats que nous avons obtenus sont essentiellement basés sur les méthodes agrégatives. Ces méthodes consistent à agréger récursivement des sous-ensembles de données pour construire une représentation multi-échelle de celles-ci. En d'autres termes, une hiérarchie d'abstractions de plus ou moins fine granularité. Pour effectuer cette opération, l'objectif de nos travaux est d'utiliser le plus possible des paramètres intrinsèques aux données (uniquement basé sur la connectivité de celles-ci). Nous avons proposé une méthode de fragmentation [37] utilisant un indice structurel comptant le nombre de cycle de longueur 3 et 4 passant par chaque arête d'un graphe. Cette méthode utilise cet indice pour filtrer les arêtes progressivement et jusqu'à ce que le graphe soit déconnecté. Chaque composante connexe représente alors un « cluster » de notre graphe. La qualité des résultats que nous obtenons est garanti par l'utilisation de la mesure

de qualité appelée MQ.

Depuis [37] nous avons largement étudié toutes les méthodes de fragmentation et dans chacune de nos méthodes nous faisons recours à cette approche. Nos derniers résultats, issu des travaux que nous avons réalisés sur l'étude des réseaux métaboliques [9], montrent cependant que l'utilisation de méthodes de fragmentation est trop restrictive pour une compréhension réelle des données. Dans [20, 4, 1] nous proposons une technique de visualisation permettant non plus de travailler sur un partitionnement des données mais sur une décomposition (l'intersection des groupes est non vide).

Algorithme de dessin de graphes

Lorsque nous manipulons des données abstraites (sans représentation dans le monde réel) ou bien lorsque la représentation réelle des données n'est pas la plus efficace d'un point de vue cognitif (ex : plan de métro) ; il faut disposer de méthodes automatiques permettant de créer une métaphore visuelle des données. Sur les données relationnelles (graphe), cette opération est un axe de recherche à part entière appelé dessin de graphes. A l'origine, cette thématique avait pour but de proposer des méthodes automatiques pour la création de circuits imprimés. L'émergence des besoins d'analyse des réseaux sociaux à largement contribué dans l'évolution de cette thématique.

Après avoir mis en place des méthodes de dessin spécifiques pour la comparaison interactive de structures secondaire d'ARN [10], nous avons travaillé sur des méthodes de représentation de réseaux métabolique [9]. Dans [2, 7] nous avons proposés des méthodes moins spécifiques au domaine d'application. Ces méthodes essayent de trouver à l'intérieur des graphes des sous-structures pour lesquelles il existe des algorithmes de dessin efficaces, une fois ces sous-structures dessinées un algorithme de dessin permet de recomposer les parties de dessin et obtenir ainsi le résultat final. Nous avons ensuite exploré un nouveau sous domaine du dessin de graphe le "Edge Bundling". Le principe de cette technique consiste à autoriser le chevauchement de certaines arêtes du graphe pour rendre le dessin plus lisible. Dans [3, 13, 14] nous avons proposé une nouvelle approche basée sur une technique de routage de graphe et de discrétisation du plan.

Publications

Mesure Microsoft — **Citations: 378, G-Index: 18, H-Index: 9**.
source : http://academic.research.microsoft.com, liste de publications vérifiée à la main

Mesure Pub or Perish — **Citations: 868, G-Index: 28, H-Index: 13** .
source : Publish or Perish, liste de publications vérifiée corrigée à la main.

Journaux Internationaux (12)

1 Paolo Simonetto, Daniel Archambault, David Auber, and Romain Bourqui. Impred: An improved force-directed algorithm that prevents nodes from crossing edges. *Comput. Graph. Forum*, 30(3):1071–1080, 2011.

2 Daniel Archambault, Tamara Munzner, and David Auber. Tugging graphs faster: Efficiently modifying path-preserving hierarchies for browsing paths. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):276–289, May 2011.

3 Antoine Lambert, Romain Bourqui, and David Auber. Winding roads: Routing edges into bundles. *Computer Graphics Forum*, 29(3):853–862, 2010.

4 Paolo Simonetto, David Auber, and Daniel Archambault. Fully automatic visualisation of overlapping sets. *Computer Graphics Forum (EuroVis09)*, 28(3):967–974, jun 2009.

5 Alexandru Telea and David Auber. Code flows: Visualizing structural evolution of source code. *Comput. Graph. Forum*, 27(3):831–838, 2008.

6 D. Archambault, T. Munzner, and D. Auber. GrouseFlocks: Steerable exploration of graph hierarchy space. *IEEE Trans. on Visualization and Computer Graphics*, 14(4):900–913, 2008.

7 D. Archambault, T. Munzner, and D. Auber. TopoLayout: Multilevel graph layout by topological features. *IEEE Trans. on Visualization and Computer Graphics*, 13(2):305–317, March/April 2007.

8 D. Archambault, T. Munzner, and D. Auber. Smashing peacocks further: Drawing quasi-trees from biconnected components. *IEEE Trans. on Visualization and Computer Graphics (Proc. Vis/InfoVis 2006)*, 12(5):813–820, Sept.-Oct. 2006.

9 Romain Bourqui, Ludovic Cottret, Vincent Lacroix, David Auber, Patrick Mary, Marie-France Sagot, and Fabien Jourdan. Metabolic network visualization eliminating node redundance and preserving metabolic pathways. *BMC Systems Biology*, 1:1–19, Jul 2007.

10 David Auber, Maylis Delest, Serge Dulucq, and Jean-Philippe Domenger. Efficient drawing and comparison of rna secondary structure. *Journal of Graph Algorithms and Applications*, 10:329–351, 2006.

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

12 David Auber and Maylis Delest. A clustering algorithm for huge trees. *Advances in Applied Mathematics*, 31:46–60, 2003.

Conférences internationales (27)

13 Antoine Lambert, Romain Bourqui, and David Auber. 3d edge bundling for geographical data visualization. In *Proceedings of the 14th International Conference on Information Visualization (IV'10)*, pages 329–335, Royaume-Uni, July 2010.

14 Antoine Lambert, David Auber, and Guy Melançon. 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, Royaume-Uni, July 2010.

15 Frederic Gilbert and David Auber. From databases to graph visualization. In *Proceedings of the 2010 14th International Conference Information Visualisation*, IV '10, pages 128–133, Washington, DC, USA, 2010. IEEE Computer Society.

16 David Auber, Patrick Mary, Morgan Mathiaut, Jonathan Dubois, Antoine Lambert, Daniel Archambault, Romain Bourqui, Bruno Pinaud, Maylis Delest, and Guy Melancon. Tulip: a scalable graph visualization framework. In *Extraction et Gestion des Connaissances (EGC) 2010*, pages 623–624, 2010.

17 Romain Bourqui and David Auber. Large quasi-tree drawing: A neighborhood based approach. In *Proc. of the 13th International Conference on Information Visualisation*, pages 653–660, 2009.

18 Paolo Simonetto and David Auber. An heuristic for the construction of intersection graphs. In *Proceedings of the 2009 13th International Conference Information Visualisation*, IV '09, pages 673–678, Washington, DC, USA, 2009. IEEE Computer Society.

19 D. Archambault, T. Munzner, and D. Auber. TugGraph: Path-preserving hierarchies for browsing proximity and paths in graphs. In *Proc. of the 2nd IEEE Pacific Visualization Symposium*, pages 113–121, 2009.

20 Paolo Simonetto and David Auber. Visualise undrawable euler diagrams. In *Proceedings of the 2008 12th International Conference Information Visualisation*, IV '08, pages 594–599, Washington, DC, USA, 2008. IEEE Computer Society.

21 D. Archambault, T. Munzner, and D. Auber. Grouse: Feature-Based and Steerable Graph Hierarchy Exploration. In Ken Museth, Torsten Möller, and Anders Ynnerman, editors, *Eurographics/ IEEE-VGTC Symposium on Visualization*, pages 67–74, Norrköping, Sweden, 2007. Eurographics Association.

22 Romain Bourqui, David Auber, and Patrick Mary. How to draw clustered weighted graphs using a multilevel force-directed graph drawing algorithm. In *11th International Conference on Information Visualisation*, pages 757–764, Jul 2007.

23 David Auber and Yves Chiricota. Improved efficiency of spring embedders: taking advantage of gpu programming. In *The Seventh IASTED International Conference on Visualization, Imaging and Image Processing*, VIIP '07, pages 169–175, Anaheim, CA, USA, 2007. ACTA Press.

24 David Auber, Noel Novelli, and Guy Melancon. Visually mining the datacube using a pixel-oriented technique. In *Proceedings of the 11th International Conference Information Visualization*, IV '07, pages 3–10, Washington, DC, USA, 2007. IEEE Computer Society.

25 Fanny Chevalier, David Auber, and Alexandru Telea. Structural analysis and visualization of c++ code evolution using syntax trees. In *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting*, IWPSE '07, pages 90–97, New York, NY, USA, 2007. ACM.

26 Noel Novelli and David Auber. Calcul et représentation efficace de cubes de données pour une visualisation orientée pixel. In *EGC'07*, pages 205–206, 2007.

27 C. Rozenblat, G. Melançon, M. Amiel, D. Auber, C. Discazeaux, A. L'Hostis, P. Langlois, and S. Larribe. Worldwide multi-level networks of cities emerging

from air traffic. In *Urban changes in different scales: systems and structures*, pages 487–502, 2006.

28 Gilles Bailly, Laurence Nigay, and David Auber. Navrna: visualization - exploration - editing of rna. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, pages 504–507, New York, NY, USA, 2006. ACM.

29 Gilles Bailly, David Auber, and Laurence Nigay. From visualization to manipulation of rna secondary and tertiary structures. In *Proceedings of the conference on Information Visualization*, IV '06, pages 107–116, Washington, DC, USA, 2006. IEEE Computer Society.

30 Romain Bourqui, David Auber, Vincent Lacroix, and Fabien Jourdan. Metabolic network visualization using constraint planar graph drawing algorithm. In *Proceedings of the conference on Information Visualization*, IV '06, pages 489–496, Washington, DC, USA, 2006. IEEE Computer Society.

31 David Auber and Fabien Jourdan. Interactive refinement of multi-scale network clusterings. In *Proc. 9th Int. Conf. on Information Visualisation (IV'05)*, pages 703–709, 2005.

32 S. Grivet, David Auber, J.-P. Domenger, and G. Melancon. Bubble tree drawing algorithm. In Springer Verlag, editor, *International Conference on Computer Vision and Graphics*, pages 633–641, 2004.

33 David Auber, Maylis Delest, and Yves Chiricota. Strahler based graph clustering using convolution. In *Proceedings of the Information Visualisation, Eighth International Conference*, IV '04, pages 44–51, Washington, DC, USA, 2004. IEEE Computer Society.

34 David Auber, Maylis Delest, Jean-Philippe Domenger, Philippe Duchon, and Jean-Marc Fédou. New strahler numbers for rooted plane trees. In *Third Colloquium on Mathematics and Computer Science Algorithms*, pages 203–215, Autriche, 2004. Birkhauser.

35 Maylis Delest, T. Munzner, David Auber, and Jean-Philippe Domenger. Exploring infovis publication history with tulip. In *Exploring InfoVis Publication History with Tulip*, pages 216–10, États-Unis, 2004.

36 G. Gainant and David Auber. Arna: Interactive comparison and alignment of rna secondary structure. In *Arna: Interactive comparison and alignment of RNA secondary structure*, Italie, 2004.

37 D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon. Multiscale visualization of small world networks. In *Proc. IEEE Symp. on Information Visualization (InfoVis'03)*, pages 75–81, 2003.

38 David Auber, Maylis Delest, Jean-Philippe Domenger, Pascal Ferraro, and Robert Strandh. Evat: Environment for visualization and analysis of trees. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 124–125, États-Unis, Oct 2003.

39 David Auber. Using strahler numbers for real time visual exploration of huge graphs. In *International Conference on Computer Vision and Graphics*, volume 1-3, pages 56–69, Tchèque, République, 2002.

Chapitres de livre (2)

40 David Auber, Guy Melancon, and Romain Bourqui. Mining networks through visual analytics: Incremental hypothesis building and validation. In Françoise Fogelman Soulié Clive Best, editor, *Mining Massive Data Sets for Security*, chapter NATO Advanced Study Institute, pages 204–211. IOS Press, Aug 2008.

41 David Auber. Tulip - a huge graph visualization framework. In Petra Mutzel and Mickael Junger, editors, *Graph Drawing Software*, Mathematics and Visualization Series. Springer Verlag, 2003.

Journaux Nationaux (1), Conférences Nationales (4), Rapports techniques (2)

42 David Auber, Daniel Archambault, Romain Bourqui, Antoine Lambert, Morgan Mathiaut, Patrick Mary, Maylis Delest, Jonathan Dubois, and Guy Mélan¸con. The tulip 3 framework: A scalable software library for information visualization applications based on relational data. Technical Report RR-7860, Jan 2012.

43 Frédéric Gilbert and David Auber. Import automatique et interactif de données dans les systèmes de visualisations. In *EGC'11*, pages 491–502, 2011.

44 David Auber, Yves Chiricota, Maylis Delest, Guy Melan¸con, Jean-Philippe Domenger, and Patrick Mary. Visualisation de graphes avec tulip : exploration interactive de grandes masses de données en appui à la fouille de données et à l'extraction de connaissances. In Gilles Venturini Monique Noirhomme-Fraiture, editor, *EGC'07: Extraction et Gestion de Connaissances*, pages 147–156, Namur, Belgique, France, Jan 2007. Cépaduès.

45 Romain Bourqui, David Auber, Vincent Lacroix, and Fabien Jourdan. Un algorithme contraint de dessin de graphe planaire pour la visualisation de réseaux métaboliques. In *Quatrième édition du congrés francophone de doctorants en STIC, MajecSTIC (2006) électronique*, France, 2006.

46 David Auber and Patrick Mary. Mise en place dun mécanisme de plugins en c++. *Programmation sous Linux*, pages 74–79, 2006.

47 Gilles Bailly, Laurence Nigay, and David Auber. 2m: un espace de conception pour i'interaction bi-manuelle. In *Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, UbiMob '05, pages 177–184, New York, NY, USA, 2005. ACM.

48 D. Archambault, T. Munzner, and D. Auber. TopoLayout: Graph layout by topological features. Technical Report TR-2005-30, University of British Columbia, December 2005.

Multiscale Visualization of Small World Networks

David Auber[1] Yves Chiricota[2] Fabien Jourdan, Guy Melançon[3]

LaBRI, Bordeaux, France Univ. Québec à Chicoutimi, Canada LIRMM, Montpellier, France

Abstract

Many networks under study in Information Visualization are "small world" networks. These networks first appeared in the study social networks and were shown to be relevant models in other application domains such as software reverse engineering and biology. Furthermore, many of these networks actually have a multiscale nature: they can be viewed as a network of groups that are themselves small world networks. We describe a metric that has been designed in order to identify the weakest edges in a small world network leading to an easy and low cost filtering procedure that breaks up a graph into smaller and highly connected components. We show how this metric can be exploited through an interactive navigation of the network based on semantic zooming. Once the network is decomposed into a hierarchy of sub-networks, a user can easily find groups and subgroups of actors and understand their dynamics.

CR Categories and Subject Descriptors: I.3.3 Computer Graphics: Picture/Image Generation - Viewing Algorithms; I.3.6 Computer Graphics: Methodology and Techniques - Interaction Techniques.

Additional Keywords: Small world networks, multiscale graphs, clustering metric, semantic zooming.

1. SMALL WORLD NETWORKS

The small world phenomenon was first identified by Milgram [13] who studied the structure of social networks. He conducted a now well known experiment, asking volunteers to deliver a letter to someone they did not personally know by passing it to one of their acquaintance they thought could help the letter reach its recipient (they knew however that the recipient was a stockbroker working in Boston). The result revealed that all letters could be delivered this way through a path consisting of six persons on average. This property is often cited as the "six degree of separation" principle [7]. The study of these networks was revived and extended to many other areas by Watts and Strogatz [17],[18].

The defining properties of small world networks rest on two structural parameters: the average path length and the clustering index of nodes. Roughly speaking, small world networks gather highly clustered subsets of nodes that are a few steps away from each other. More precisely, whilst the average path length in a small world network compares to that in a random graph (with the same number of edges), the clustering index of its nodes can be orders of magnitudes larger on average (Section 3.1).

Many important real world example networks are small world. This has been observed for neural networks by Watts [17]. Applications to this area are further discussed by Kashurinagan [10]. Adamic [1] has shown that the small world properties hold for networks of sites extracted from the web. Graphs coming from reverse software engineering provide further examples of small world networks [2]. A famous example of a small world network is obtained from the Internet movie database[1].

The figures in the table below show the small world nature of some networks. The IMDB example consists in a small part of the IMDB database of actors and films. Starting from a particular actor *X*, we extracted all other actors and actresses that played with him (films with more than 35 actors were discarded), before deleting all edges connecting these actors to the selected actor *X* (and deleting *X* as well). Two actors were then connected by an edge if they played in a movie together (but not with *X*, otherwise the graph would be complete). The final graph contained 419 actors connected with 5651 edges. This graph is illustrated in Figure 1. Groups of actors having played in a same movie define cliques (subgraphs with maximum number of edges) and appear as dense blue disks.

Graph	Clustering index	Ave Path Length	Random graph (same number of nodes and edges)	
IMDB	0.9666	3.2043	0.0243	2.6694
"Resyn assistant"	0.9518	3.2847	0.1942	1.8195
Mac OS 9	0.3875	2.8608	0.0179	3.3196
Web[2]	0.1078	3.1	2.3e-4	-
.edu sites[2]	0.156	4.062	0.0012	4.048

Table 1. Examples of small world networks with corresponding values for their clustering index and average path length. The last two columns report the same statistics for random graphs having the same number of

1. auber@labri.fr
2. ychirico@uqac.ca
3. fjourdan@lirmm.fr, Guy.Melancon@lirmm.fr

1 See the URL www.imdb.com. See also the Kevin Bacon oracle web site (www.cs.virginia.edu/oracle).

[2] Borrowed from Adamic [1]. The part of the web he studied contained 259,794 pages and site, from which he extracted 153,127 sites (leaving leaf nodes aside). The edu sites were extracted from the latter network.

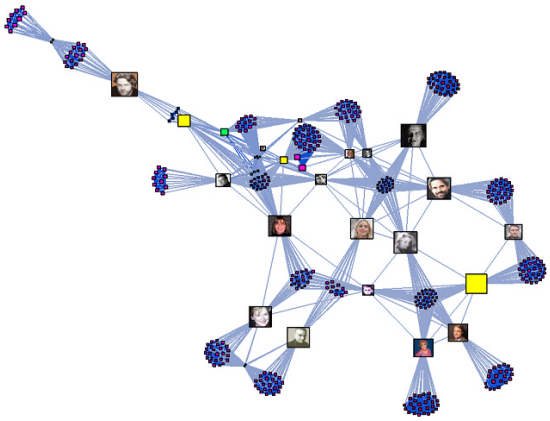


Figure 1. Force-directed layout of a subset of the IMDB network. The layout is shown for the purpose of the discussion only. The identification of relevant subgroups in the network can be done prior to visualization. A layout of the whole graph can be avoided.

nodes and edges than the example graph. In each case, the average path length of the example graph compares with that of a random graph, while its clustering index is significantly higher.

The network named "Resyn Assistant" is a graph associated with a Java API designed for the visualization of chemical components. Nodes correspond to java classes and links are induced from access of a class to attributes or methods of another class (directions are ignored). The Mac OS9 example consists in a graph where nodes correspond to header files and where edges correspond to physical inclusion of header files. Evidence showing that many graphs studied in software reverse engineering are small world networks can be found in [2].

To our knowledge, the structural properties of small world networks have not yet been fully exploited from a visualization perspective. Most of the research efforts on small world networks focus on providing theoretical models that capture the essence of the small world phenomenon. The definition of more focused classes of small networks based on the study of the degree distribution have been proposed [13]. More recently, algorithmic aspects have received much attention, with a special emphasis on the possibility of defining distributed algorithms on small world networks [11].

2. INTERACTIVE VISUALIZATION OF SMALL WORLD NETWORKS

Common tasks when dealing with a social network consist in identifying patterns in the set of connections that link the actors. These patterns often correspond to *social groups* -- collections of actors who are closely linked to one another. Alternatively, they may indicate *social positions* -- sets of actors who are linked into the total social system in similar ways [12][3]. Other more intuitive questions might be "Is there a group connected with all other groups (containing, for each other group, someone linked to it) ?", or more generally "Are the groups organized in any particular way ?", etc. Once a group has been identified, unless it consists of actors knowing each other (a clique), it might be of interest to understand how this particular group is organized, this subgroup being itself a social network, i.e. a small world network.

Most of these questions can be addressed by visual inspection, at least in a first stage. Our work focuses on the use of the small world properties of networks to support the visualization process. When exploring a small world network, the user will most probably visually ignore the highly connected components and intuitively concentrate on the network of "cliques", before digging in for more details on a specific area of the network. This scenario agrees with Schneiderman's mantra [16]. This statement is supported by the view offered in Figure 1.

The technique we present allows us to compute the decomposition of a small world network into its highly connected components prior to the visualization, and to offer the user an abstract view of the network to start with. This approach, in a sense, implements the filtering stage promoted by Schneiderman. It can bring a valuable aid to the user when dealing with larger networks.

As we shall see, the simple computation of a structural parameter on edges leads to an effective division of a small world network into its highly connected components. Applying a threshold and discarding edges with a low value divides the graph into components that can be captured using classical algorithms on graphs. Hence, based on a simple and efficient structural parameter, we are able to classify the nodes into clusters that form a coherent organization of the network into smaller components.

2.1 Multiscale networks

The examples we have studied lead us to the following observation. Most networks not only are small world, but their highly connected components themselves are small world. Hence, it seemed that small world networks can be decomposed into a hierarchy of small world networks. (Obviously, groups at the lowest level of the hierarchy might not be small world but merely consist of cliques.) This observation was made by Adamic [1] for networks induced from the web. Computations of the average path length and clustering index on a network of 153,127 web sites allowed Adamic to show that it is small world. The subset of *.edu* sites was isolated out of this network and was shown to be small world as well (see Table 1). This phenomenon has been verified on several subnetworks extracted from the IMDB and is discussed in section 4.2.

We believe that relevant information on the network can be deduced from a hierarchical decomposition into small world sub-networks. Moreover, the hierarchy can be efficiently used to navigate the network.

3. STRENGTH OF EDGES AND NETWORK DECOMPOSITION

As mentioned before, our technique requires that we compute a value for each edge of the network. The metric we define was first introduced in [2] and generalizes the clustering index for nodes introduced by Watts (see [17]). Given a node v in the network, its clustering index $c(v)$ is defined by first computing the number $r(N(v))$ of edges connecting neighbors of v and by taking the ratio:

$$c(v) = r(N(v))/ (k(k\text{-}1)/2).$$

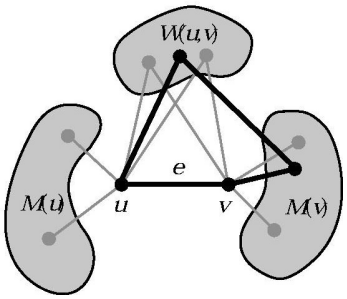
[3] The website presenting the network visualization software Inflow Software also points out at interesting tasks and areas. See http://www.orgnet.com/.

(Here, k denotes the size of v's neighborhood $N(v)$. Note that the denominator computes the number of edges in a clique of size k.)

The clustering index of a graph G is then defined by taking the average $c_G(v) = \Sigma_v\, c(v) / N$ (where N denotes the number of nodes in G).

Small world networks are those with a high clustering index while having a small average path length between nodes, in comparison with a random graph[4] with the same number of nodes and edges (see Table 1). Watts [17] discusses several models for generating graphs simultaneously satisfying these two properties. While the class of small world graphs is very large, it is certainly incorrect to say that *all* graphs occurring in Information Visualization are small world networks. Based on a set of examples we have studied, we make the assumption that many important example networks in Information Visualization are part of this class.

We now turn to the problem of defining the clustering index of an edge.



Given an edge (u, v), its strength is computed by dividing neighbors of u or v into three distinct subsets. Denote by $M(u)$ the set of neighbors of u that are *not* neighbors of v (excluding v). We define $M(v)$ similarly. Finally, let $W(u,v)$ denote the set of common neighbors to u *and* v. Write $r(A, B)$ for the number of edges linking nodes in the set A to nodes in the set B. The ratio $s(A, B) = r(A, B) / |A|\cdot|B|$ thus computes the proportion of existing edges among the set of all possible edges connecting nodes of A and B. Now, any edge connecting two of the subsets $M(u)$, $M(v)$ or $W(u, v)$ is part of a cycle of length 4 going through the edge (u, v). Note that all cycles of length 4 are captured this way. Finally, we define the ratio $|W(u, v)|/(|M(u)| + |W(u, v)| + |M(v)|)$ computing a ratio related to the proportion of cycles of length 3 containing the edge (u, v). Note that there are as many of these cycles as there are nodes in $W(u, v)$. The strength of an edge is given by computing:

$$s(M(u), W(u, v)) + s(W(u, v), M(v)) + s(W(u, v)) + s(M(u), M(v)) + |W(u, v)|/(|M(u)| + |W(u, v)| + |M(v)|)$$

(Note: we need to put $s(A) = 2r(A)/(|A|\cdot(|A|-1)$ when computing the proportion of edges connecting a set to itself).

[4] Following a uniform distribution, where each graph has the same probability of being drawn at random (among all graphs with a specified number of edges and nodes).

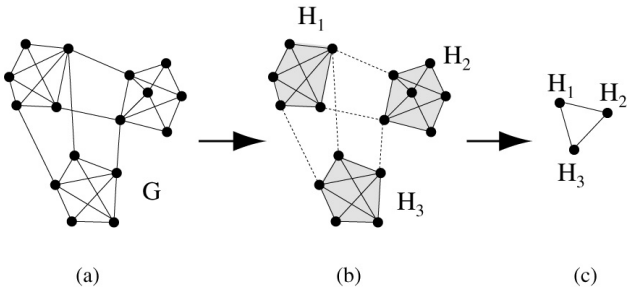


Figure 2. The clustering process.

3.1 Discarding weak edges

The strength metric of an edge can be interpreted as a measure of its contribution to the cohesion of its neighborhood. Conversely, if an edge is an isthmus connecting disjoint neighborhoods in the network then it has value 0. Hence, edges inducing weak connections between groups can be identified and *filtered out.*

This filtering process consist in removing edges having a strength value below a given threshold τ. This operation leads to a set of maximal disconnected subgraphs $\{H_1, H_2, \ldots, H_q\}$, corresponding to a clustering of the initial graph (each connected component is considered as a cluster). We then compute the *quotient graph* with respect to τ by taking the subgraphs H_i as vertices of a new and higher level graph. There is an edge between two high level nodes H_i and H_j if there exists at least one edge between a vertex of H_i and a vertex of H_j in the original graph.

Figure 2 illustrates this process. Part (a) of the figure represents a graph drawn using a force directed placement algorithm. In part (b), weak edges have been drawn using dashed lines. The resulting quotient graph is represented in part (c). This process can be applied recursively, leading to a hierarchical clustering (the clustering is one level depth in the example). Figure 3 illustrates the quotient graph resulting from the application of this technique on the example network extracted from the IMDB (see also Figure 1, which illustrate the original graph laid out using a force directed placement algorithm). Our visualization technique relies on this clustering algorithm. Technical issues related to the choice of the threshold value are addressed in section 5.

3.2 Hierarchical decomposition

This technique can be fully exploited by recursively applying the metric to each component of the quotient graph. Note that this makes sense by virtue of the multiscale nature of the network. That is, non-trivial components can be further decomposed using the same technique and this can be repeated as required (stopping conditions can be formulated in terms of their size and until the small world property holds). For example, each box in Figure 3 contains a sub-network that has been further decomposed and shows its quotient image. The center of the figure shows that the whole network is organized around four different groups of actors.

This recursive process results into a hierarchically clustered network. The hierarchy itself can actually be used to navigate the network and enter into its small world components. Hence, following a branch down the hierarchy and visually inspecting the associated sequence of quotient graphs can help a user identify the core group of actors in a part of the network.

3.3 Visual coherence

A word must be said about the layout algorithm we used in our experiments. Our technique is efficient when used in conjunction with a force-directed layout algorithm (or any other variant). This does not come as a surprise, as force-directed algorithm will naturally embed neighbor nodes close to one another. It will only succeed in separating nodes incident to an edge (u, v) if their neighborhoods are poorly connected (that is the case when the set $W(u, v)$ is almost empty, with a low value for $s(M(u), M(v))$, for instance).

Actually, force-directed layout algorithms have been used as a partitioning technique of a graph. After laying out nodes in Euclidean space, a threshold distance can be defined to determine how clusters are formed. This technique corresponds, in a sense, to what a user does intuitively when inspecting the graph and has already been used with success [7]. This is even more obvious when the graph is large as a subset of tightly connected nodes will appear as a round ball on the screen.

However, a force-directed algorithm will generally not distinguish between edges and will induce the same attractive force for all edges of the graph, weak or strong. That inconvenience can be avoided by assigning weights to edges and by taking weights into account when defining attractive forces (masses can additionally assigned to nodes). Our method implicitly takes the strength of edges into account when laying out the graph on the screen. Indeed, at each recursive step, the quotient graph is laid out using a force-directed algorithm. Now, weak edges between two clusters that have been removed to form the quotient graph are replaced by a single edge, which has a much "weaker" attractive effect.

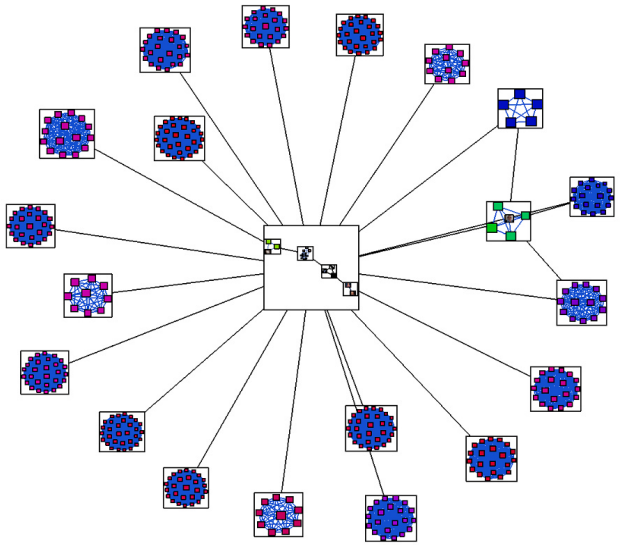


Figure 3. Quotient graph computed from the network illustrated in Figure 1. Actors are organized into more or less independent communities, all linked to a core group.

4. EXPLORATION OF THE STRUCTURE OF A SMALL WORLD NETWORK

The whole method (computation of the metric and hierarchical clustering) was implemented to provide a stand-alone visualization and exploration application. We opted to develop our tool using the Tulip library [2][4]. Tulip is able to easily scale up to tens of thousands of nodes and edges while offering real time navigation. It moreover offers a framework capable of dealing with hierarchical clusters of a graph. The strength metric and recursive clustering procedure were developed around existing Tulip plug-ins[5].

Figure 4 shows a screenshot of the application SWViz. The design focuses on the coupling of an overview of the hierarchically clustered graph (left) together with a more detailed view of a component (right). The detailed view can either show an unfolded force-directed layout, ring placement or a hierarchical view (Sugiyama) of the selected component.

4.1 Navigational coherence

The selection of a component in the overview can result in different views in the right panel depending on how it is selected. A left click results in the display of the overview graph of the selected component. A middle click will instead show the same component unfolded as a flat graph. The wheel button is used to zoom in and out.

In that case, visual coherence is maintained by showing the selected component in the exact same way in the overview panel. For instance, suppose that the user changes the layout of the selected subgraph from force-directed to ring layout in the right panel. The selected component in the left panel will then automatically be displayed in its own box using the same layout. Note that this does not disturb the overview itself, since this operation can only be applied to subnetworks lying in boxes and not to the overall graph itself.

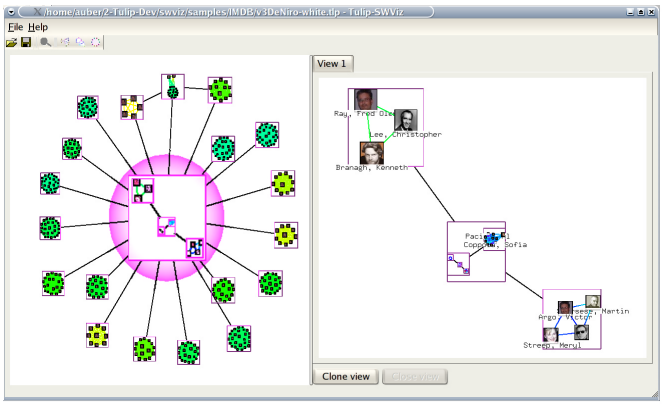


Figure 4. SWViz snapshot. An overview of the network (left) obtained from the quotien graph shows it overall structure. A selected component (pink background) is shown in greater detail in the right panel.

We now focus on specific examples and discuss how the technique was used to explore and discover structure in small world networks.

[5] Our application is made public and can be downloaded at the URL www.tulip-software.com.

4.2 Resyn Assistant API

"Resyn Assistant" is a software designed for the study of chemical components developed in Montpellier (LIRMM). Before going into a new development stage and making new design decisions, the development team decided to study the structure of the actual API. The visualization of the access graph of their API enabled the team to recover the whole history of the past development. Moreover, it offered them a new perspective on their software and showed that their code was organized around a central component (see Figure 5). Components at the periphery correspond to dedicated modules. For instance, classes related to the topology of molecules are grouped into a single cluster. The overall structure of the API is clearly illustrated in Figure 4.

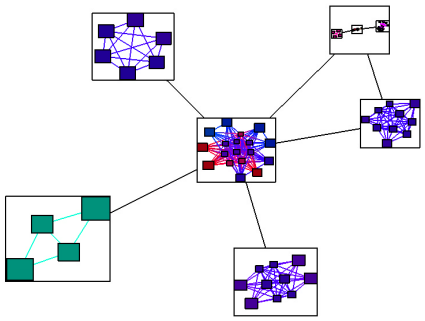


Figure 5. The hierarchical decomposition of the Resyn API shows how the software modules are related to each other.

Resyn's designers were actually able to identify an error in the design of the API. Digging into the hierarchy down to the second level, they found that classes with similar functionalities, that were expected to belong to the same subgroup, were spread out onto two different clusters. The designers plan to recover from this design error in the next release of Resyn.

4.3 IMDB subset

We will now have a closer look at the IMDB subset of actors (Figure 1). The central group in the quotient graph (Figure 3) gathers internationally renowned movie actors, such as Sharon Stone, Meryl Streep, Leonardo Di Caprio, Al Pacino. They lie at the center mainly because of their presence in numerous movies. Clusters at the periphery correspond to movies (actors having played in a same movie necessarily define a clique). This example clearly shows that the method is efficient at finding the influential actors of a small world network.

4.4 Other examples

Many other example graphs extracted from the IMDB were studied. Also, we were able to apply our method to various graphs computed from software systems (include files of Mac OS9, MFC classes, etc.). In each case the method was able to correctly identify modules or components and showed expected dependencies between them.

5. CHOICE OF THE THRESHOLD VALUE AND QUALITY MEASUREMENTS

One important question has been deferred until now. Just how one should proceed to determine whether an edge is a weak edge ? In other words, how does one choose the threshold value applied to the strength metric when filtering out edges ? A completely satisfying answer to this question requires some work and should rely on the knowledge of the distribution of values for the strength metric.

This is not enough however. Indeed, the choice of the threshold is determined by the fact that the partition induced from the filtering procedure is "good". So our initial interrogation bounces on the following question: "What is a good partition for a graph ?". Many criteria have been proposed as strategies for partitioning graphs. In many cases, the partitioning criteria can actually be rephrased as an optimization problem. That is, the "best" partition possible often minimizes a given cost function defined on the set of all possible partitions of a graph. Most of these optimization problems do not admit deterministic solutions however, and many heuristics computing an approaching solution have been proposed. (See [2], [5], [8], for instance.)

5.1 Quality of a partition

The problem of determining the quality of a partition *for a given graph* can then be answered by computing just how close a partition is to a partition with optimal cost. The answer to this question itself relies on the knowledge of the distribution of costs on the set of all possible partitions.

These considerations led us to select a specific cost function called *MQ*, first introduced as a partition cost function in the field of software reverse engineering [14]. We provide the definition of MQ, for sake of completeness (see [6]). Given a clustering $C=\{C_1, C_2, \ldots, C_n\}$ of a graph G, MQ is defined as:

$$MQ(C;G) = \frac{1}{p}\sum_{i=1}^{n} s(C_i, C_i) - \frac{1}{p(p-1)/2}\sum_{i<j} s(C_i, C_j).$$

The first term is the mean value of edge density inside each cluster. The second term is the mean value of edge density between the clusters. (See section 3 where the "s" notation is introduced.) It is worth to note that MQ is not a measure for the quality of the visualization itself. It serves as an objective measure of the quality of a clustering (among all possible clustering of the graph).

We apply MQ to automatically choose the "best" clustering for a given graph. The distribution of *MQ* values (over the set of all partitions of a given graph) is close to a gaussian distribution with $\mu = -0.2$ and $\sigma = 0.2$ (*MQ* is normalized and varies in the [-1, 1] interval). Hence, given any possible *MQ* value *c*, we are able to determine the probability for a partition to have a value at least equal to *c*. For instance, only 0.5% of all partitions of a graph have an *MQ* value above 0.315. We refer the reader to [6] for more details.

5.2 Correlating edge strength with MQ

Hence, we see that an answer to our original question of determining the optimal threshold can be based on a combined study of edge strength and *MQ*. More precisely, the selected threshold should be the one inducing a partition with highest possible *MQ* value. Figure 6 shows a curve (histogram) describing the variation of MQ with respect to strength. The optimal threshold $c = 1.6$ can be easily obtained by inspection.

5.2.1 Improving quality

Recall that once a threshold value has been selected, edges with strength below the threshold are discarded. This induces a partition of the graph into several components. In many cases however, this partition will contain several isolated vertices, which actually impoverish its *MQ* quality. This difficulty can be avoided as follows: isolated vertex with degree one in the original graph are de facto re-inserted in their unique neighbor's cluster. All other isolated vertices are momentarily grouped into a single cluster and the subgraph induced from this subset is then cut into its connected components.

We were able to observe that this simple reorganization step significantly improves the *MQ* value associated with a given threshold (by comparing with the results described in [6]). Indeed, post-processing isolated vertices enabled us to reach *MQ* values as high as 0.8 and 0.9. These values are rather exceptional since a partition has an *MQ* value at least equal to 0.75 with probability 10^{-6}. Values for Resyn and the IMDB networks are reported in Table 2 below.

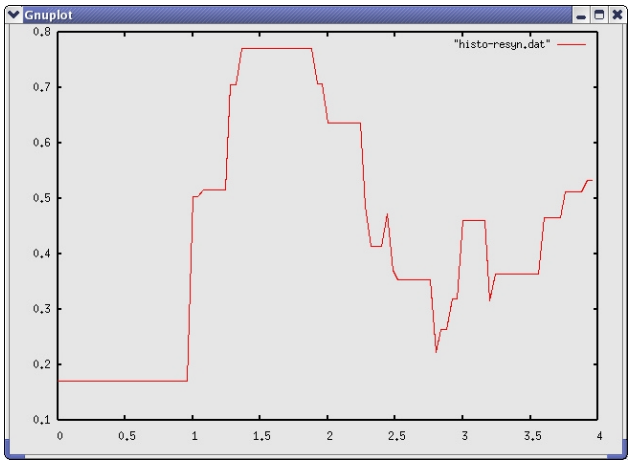


Figure 6. The curve describes the variation of *MQ* with respect to edge strength for the "Resyn Assistant" API. The "optimal" threshold approaches 1.6 with for *MQ* value close to 0.80, which is remarkably high.

Graph (and clustering)	***MQ***
Resyn top level quotient	0.771772
Resyn second level quotient (central component in Figure 5)	0.481984
IMDB example network top level quotient	0.947879
IMDB example network second level (right component in Figure 4)	0.6883

Table 2. Listing of several *MQ* values for the Resyn and IMDB examples. These scores are extremely high and exceptional, considering that *MQ* is close to a gaussian distribution with $\mu = -0.2$ and $\sigma = 0.2$.

5.2.2 A simple partitioning heuristic

Our application takes care of selecting the threshold for the user, by examining the Strength/*MQ* histogram of a graph. However, one could imagine giving the user complete control over threshold selection by mean of a range slider. This interactive approach has a low computational cost and can be seen as an alternative over other heuristics for many reasons.

- Many clustering techniques indeed aim at finding partitions having a minimal number of outer edges as possible. This is in accordance with our approach since the filtered out edges indeed correspond to those outer edges. The filtered out edges are "long range" edges, which form a minority of all edges (which can be seen as a property of small world networks).
- The application provides the user with immediate visual feedback on the partition that is computed, enabling an expert to assess of its significance, e.g.
- Its computing time is way below the usual optimization algorithms such as hill climbing or genetic algorithms used in software dedicated to graph clustering.
- But most of all, the validity of these arguments rely on the fact that our method is able to find partitions with an exceptionally high *MQ* value, which assess of its acuity.

6. CONCLUSION AND FUTURE WORK

Many other application domains must be studied, and many more data set need to be collected and analyzed in order to establish the relevance of small world networks in Information Visualization. Application domains such as biology (neural networks) and linguistics (word association) offer fertile grounds that should be investigated more closely.

We are currently extending our application with navigation techniques such as semantic fish-eye. This will allow the user to navigate the hierarchical network from and into a single panel since the fish-eye combines detailed information and overall context.

A closer examination of structural properties of small world networks such as degree distribution could help us design specific visual cues. Also, the criteria from which small world networks are defined should be revised from an information visualization perspective in order to straighten the vagueness of the statement "having a clustering index significantly higher than a random graph". Indeed, many examples we looked at and that complied best with our approach had relatively high clustering index (often up to 95%). This calls for the definition of a more focused class of small world networks that differ from some theoretical constructions and correspond better to real-life examples.

References

[1] ADAMIC, L.A. *The Small World Web*, Xerox Palo Alto Research Center.

[2] ALPERT, C.J. AND A.B. KAHNG. 1995. Recent Developments in Netlist Partitioning: A Survey. *Integration: the VLSI Journal 19*, 1-2, 1-81.

[3] AUBER, D. 2001. Tulip. In *9th International Symposium on Graph Drawing, GD 2001*. Lecture Notes in Computer Science 2265, Springer-Verlag. Vienna, Austria.

[4] AUBER, D. 2003. Tulip - A huge graph visualization framework, in *Graph Drawing Software*. P. Mutzel and M. Jünger, editors. Springer Verlag.

[5] BERKHIN, P. 2002. *Survey Of Clustering Data Mining Techniques*. Accrue Software: San Jose, CA.

[6] CHIRICOTA, Y., F. JOURDAN, AND G. MELANÇON. 2003. Software Components Capture using Graph Clustering. In: *11th IEEE International Workshop on Program Comprehension*. Portland, Oregon: IEEE / ACM.

[7] FAIRCHILD, K.M., S.E. POLTROCK, AND G.W. FURNAS. 1988. SemNet: Three-Dimensional Representation of Large Knowledge Bases. In *Cognitive Science and Its Applications for Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc.

[8] FJÄLLSTROM, P. 1998. Algorithms for graph partitioning: A survey. *Linkoping Electronic Articles in Computer and Information Science, 3*.

[9] GUARE, J. 1990. *Six degrees of Separation: A Play*. New York: Vintage Books.

[10] KASTURIRANGAN, R. 1999. *Multiple Scales in Small-World Networks*. Brain and Cognitive Science Department, MIT.

[11] KLEINBERG, J. 2000. The Small-World Phenomenon: An Algorithmic Perspective. In: *32nd ACM Symposium on Theory of Computing*.

[12] FREEMAN, L.C. 2000. Visualizing Social Networks. *Journal of Social Structures 1*, 1. `http://www2.heinz.cmu.edu/project/INSNA/joss/`.

[13] MILGRAM, S. 1967. The small world problem. *Psychology Today 1*, 61.

[14] MITCHELL, B.S., MANCORIDIS S., YIH-FARN C., GANSNER E. 1999. Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Structures. In: *International Conference on Software Maintenance, ICSM*.

[15] NEWMAN, M., WATTS D., AND STROGATZ S.H., Random graph models of social networks. *Proceedings of the National Academy of Sciences*. To appear.

[16] SHNEIDERMAN, B. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization. In *IEEE Conference on Visual Languages (VL'96)*. Boulder, CO: IEEE CS Press.

[17] WATTS, D.J. 1999. *Small Worlds*. Princeton University Press.

[18] WATTS, D.J. AND STROGATZ S.H. 1998. Collective dynamics of "small-world" networks. *Nature 393*, 440-442.

Bibliographie

[1] J. Abello, F. van Ham et N. Krishnan. "ASK-GraphView : A Large Scale Graph Visualization System". Dans : *IEEE Trans. on Visualization and Computer Graphics (Proc. Vis/InfoVis '06)* 12.5 (2006), p. 669–676.

[2] J. Abello, S. G. Kobourov et R. Yusufov. "Visualizing Large Graphs with Compound-Fisheye Views and Treemaps". Dans : *Proc. of Graph Drawing*. T. 3383. LNCS. Springer-Verlag, 2004, p. 431–441.

[3] A. T. Adai et al. "LGL : Creating a Map of Protein Function with an Algorithm for Visualizing Very Large Biological Networks". Dans : *Journal of Molecular Biology* 340.1 (2004), p. 179–190.

[4] Magali Amiel, Guy Mélançon et Céline Rozenblat. "Réseaux multi-niveaux˜ : l'exemple des échanges aériens mondiaux de passagers". Dans : *M@ppemonde* 3.79 (2005).

[5] Anastasia Bezerianos and Pierre Dragicevic and Jean-Daniel Fekete and Juhee Bae and Ben Watson. "GeneaQuilts : A System for Exploring Large Genealogies". Dans : *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), 1073–1081.

[6] D. Archambault, T. Munzner et D. Auber. "Grouse : Feature-Based and Steerable Graph Hierarchy Exploration". Dans : *Eurographics/ IEEE-VGTC Symposium on Visualization*. Sous la dir. de Ken Museth, Torsten Möller et Anders Ynnerman. Norrköping, Sweden : Eurographics Association, 2007, p. 67–74.

[7] D. Archambault, T. Munzner et D. Auber. "GrouseFlocks : Steerable Exploration of Graph Hierarchy Space". Dans : *IEEE Trans. on Visualization and Computer Graphics* 14.4 (2008), p. 900–913.

[8] D. Archambault, T. Munzner et D. Auber. "Smashing Peacocks Further : Drawing Quasi-Trees from Biconnected Components". Dans : *IEEE Trans. on Visualization and Computer Graphics (Proc. Vis/InfoVis 2006)* 12.5 (2006), p. 813–820.

[9] D. Archambault, T. Munzner et D. Auber. "TopoLayout : Multilevel Graph Layout by Topological Features". Dans : *IEEE Trans. on Visualization and Computer Graphics* 13.2 (2007), p. 305–317.

[10] Daniel Archambault, Tamara Munzner et David Auber. "Tugging Graphs Faster : Efficiently Modifying Path-Preserving Hierarchies for Browsing Paths". Dans : *IEEE Transactions on Visualization and Computer Graphics* 17.3 (mai 2011), p. 276–289.

[11] David Auber. "Tulip - A huge graph visualization framework". Dans : *Graph Drawing Software*. Sous la dir. de Petra Mutzel et Mickael Junger. Mathematics and Visualization Series. Springer Verlag, 2003.

[12] David Auber et Yves Chiricota. "Improved efficiency of spring embedders : taking advantage of GPU programming". Dans : *The Seventh IASTED International Conference on Visualization, Imaging and Image Processing*. VIIP '07. Palma de Mallorca, Spain : ACTA Press, 2007, p. 169–175. ISBN : 978-0-88986-692-8.

[13] David Auber et Fabien Jourdan. "Interactive Refinement of Multi-scale Network Clusterings". Dans : *Proc. 9th Int. Conf.*

on Information Visualisation (IV'05). 2005, p. 703–709.

[14] David Auber et al. "New Strahler numbers for rooted plane trees". Anglais. Dans : *Third Colloquium on Mathematics and Computer Science Algorithms*. Autriche : Birkhauser, 2004, p. 203–215.

[15] Juhee Bae et Benjamin Watson. "Developing and Evaluating Quilts for the Depiction of Large Layered Graphs". Dans : *IEEE Transactions on Visualization and Computer Graphics* 17.12 (déc. 2011), p. 2268–2275. issn : 1077-2626.

[16] François Bertault. "A Force-Directed Algorithm that Preserves Edge Crossing Properties". Dans : *International Symposium on Graph Drawing (GD99)*. Sous la dir. de Jan Kratochvíl. T. 1731. Lecture Notes in Computer Science. Springer, 1999, p. 351–358.

[17] François Bertault. "A Force-Directed Algorithm that Preserves Edge Crossing Properties". Dans : *Information Processing Letters* 74.1–2 (avr. 2000), p. 7–13.

[18] Therese C. Biedl et al. "Orthogonal Drawings with Few Layers". Dans : *Revised Papers from the 9th International Symposium on Graph Drawing*. GD '01. London, UK, UK : Springer-Verlag, 2002, p. 297–311. isbn : 3-540-43309-0.

[19] James F. Blinn. "Models of light reflection for computer synthesized pictures". Dans : *SIGGRAPH '77 : Proceedings of the 4th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM, 1977, p. 192–198.

[20] James F. Blinn. "Simulation of wrinkled surfaces". Dans : *SIGGRAPH '78 : Proceedings of the 5th annual conference on Computer graphics and interactive techniques*. New York, NY, USA : ACM, 1978, p. 286–292.

[21] Romain Bourqui et David Auber. "Large Quasi-Tree Drawing : A Neighborhood Based Approach". Dans : *Proc. of the 13th International Conference on Information Visualisation*. 2009, p. 653–660.

[22] Romain Bourqui, David Auber et Patrick Mary. "How to Draw Clustered Weighted Graphs using a Multilevel Force-Directed Graph Drawing Algorithm". Dans : *11th International Conference on Information Visualisation*. 2007, p. 757–764.

[23] Romain Bourqui et al. "Metabolic network visualization eliminating node redundance and preserving metabolic pathways". Anglais. Dans : *BMC Systems Biology* 1 (2007), p. 1–19.

[24] Mark Bruls, Kees Huizing et Jarke J. van Wijk. "Squarified Treemaps". Dans : *Proc. Joint Eurographics/IEEE TVCG Symp. Visualization, VisSym*. 2000, p. 33–42.

[25] C. Buchheim, M. Jünger et S. Leipert. "Improving Walker's Algorithm to Run in Linear Time". Dans : *Proc. Graph Drawing (GD'02)*. T. 2528. LNCS. Springer-Verlag, 2002, p. 344–353.

[26] Heorhiy Byelas et Alexandru Telea. "Texture-based visualization of metrics on software architectures". Dans : *SoftVis08. Proceedings of the 4th ACM symposium on Software visuallization*. Ammersee, Germany : ACM, 2008, p. 205–206.

[27] Bill Cheswick et Hal Burch. "Internet Mapping Project". `research.lumeta.com/ches/map/`.

[28] Stirling Christopher Chow. "Generating and drawing area-proportional Euler and Venn diagrams". Thèse de doct. 2007.

[29] W. Cui et al. "Geometry-Based Edge Clustering for Graph Visualization". Dans : *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), p. 1277–1284.

[30] Hubert De Fraysseix, János Pach et Richard Pollack. "How to draw a planar graph on a grid". Dans : *Combinatorica* 10.1 (1990), p. 41–51.

[31] Maylis Delest et al. "Exploring InfoVis Publication History with Tulip". Anglais. Dans : *Exploring InfoVis Publication History with Tulip*. États-Unis, 2004, p. 216–10.

[32] Josep Díaz, Jordi Petit et Maria Serna. "A survey of graph layout problems". Dans : *ACM Comput. Surv.* 34.3 (sept. 2002), p. 313–356. ISSN : 0360-0300.

[33] M. Dickerson et al. "Confluent Drawings : Visualizing Non-planar Diagrams in a Planar Way". Dans : *Proc. Graph Drawing 2003 (GD'03)*. 2003, p. 1–12.

[34] D.P Dobkin et al. "Implementing a General-Purpose Edge Router". Dans : *Proc. Graph Drawing 1997 (GD'97)*. 1998, p. 262–271.

[35] Christian A. Duncan et al. "Lombardi drawings of graphs". Dans : *Proceedings of the 18th international conference on Graph drawing*. GD'10. Konstanz, Germany : Springer-Verlag, 2011, p. 195–207. ISBN : 978-3-642-18468-0.

[36] T. Dwyer et L. Nachmanson. "Fast Edge-Routing for Large Graphs". Dans : *Proc. Graph Drawing 2009 (GD'09)*. 2010, To appear.

[37] P. Eades et Q. Feng. "Multilevel Visualization of Clustered Graphs". Dans : *Proc. Graph Drawing (GD'96)*. T. 1190. LNCS. Springer-Verlag, 1996, p. 101–112.

[38] P. Eades et M. L. Huang. "Navigating Clustered Graphs using Force-Directed Methods". Dans : *Journal of Graph Algorithms and Applications* 4.3 (2000), p. 157–181.

[39] Peter Eades. "Drawing Free Trees". Dans : *Bul. Institute for Combinatorics and its Applications* 5 (1992), p. 10–36.

[40] Leonhard Euler. *Lettres à une Princesse d'Allemagne, Letters No. 102-108*. 1761.

[41] R. A. Finkel et J. L. Bentley. "Quad trees a data structure for retrieval on composite keys". Dans : *Acta Informatica* 4.1 (1974), p. 1–9.

[42] Andrew Fish et Gem Stapleton. "Defining Euler diagrams : choices and consequences". Dans : *Euler Diagrams 2005* (2005).

[43] Andrew Fish et Gem Stapleton. "Formal Issues in Languages Based on Closed Curves". Dans : *Distributed Multimedia Systems, Knowledge Systems Institute*. 2006, p. 161–167.

[44] Jean Flower, Andrew Fish et John Howse. "Euler diagram generation". Dans : *Journal of Visual Languages and Computing* 19 (6 déc. 2008), p. 675–694.

[45] Jean Flower et John Howse. "Generating Euler Diagrams". Dans : *Proceedings of the Second International Conference on Diagrammatic Representation and Inference*. DIAGRAMS '02. London, UK : Springer-Verlag, 2002, p. 61–75.

[46] A. Frick, A. Ludwig et H. Mehldau. "A Fast Adaptive Layout Algorithm for Undirected Graphs". Dans : *Proc. Graph Drawing (GD'94)*. T. 894. LNCS. 1995, p. 388–403.

[47] Yaniv Frishman et Ayellet Tal. "Online Dynamic Graph Drawing". Dans : *IEEE Transactions on Visualization and Computer Graphics* 14.4 (2008), p. 727–740.

[48] Thomas M.J. Fruchterman et Edward M. Reingold. "Graph drawing by force-directed placement". Dans : *Software : Practice and Experience* 21.11 (nov. 1991), p. 1129–1164.

[49] P. Gajer et S. G. Kobourov. "GRIP : Graph dRawing with Intelligent Placement". Dans : *Proc. Graph Drawing 2000 (GD'00)*. 2000, p. 222–228.

[50] P. Gajer et S. G. Kobourov. "GRIP : Graph Drawing with Intelligent Placement". Dans : *Journal of Graph Algorithms and Applications* 6.3 (2002), p. 203–224.

[51] E. R. Gansner et Y. Koren. "Improved circular layouts". Dans : *Proc. Graph Drawing 2006 (GD'06)*. 2006, p. 386–398.

[52] E. R. Gansner, Y. Koren et S. C. North. "Topological Fisheye Views for Visualizing Large Graphs". Dans : *IEEE Transactions on Visualization and Computer Graphics* 11.4 (2005), p. 457–468.

[53] E. Di Giacomo et al. “Graph Visualization Techniques for Web Clustering Engines”. Dans : *IEEE Trans. on Visualization and Computer Graphics* 13.2 (2007), p. 294–304.

[54] S. Grivet et al. “Bubble Tree Drawing Algorithm”. Dans : *International Conference on Computer Vision and Graphics*. 2004, p. 633–641.

[55] Carsten Gutwenger et Petra Mutzel. “Planar Polyline Drawings with Good Angular Resolution”. Dans : *International Symposium on Graph Drawing (GD98)*. Sous la dir. de Sue Whitesides. T. 1547. Lecture Notes in Computer Science. Springer, 1998, p. 167–182.

[56] S. Hachul et M. Jünger. “An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs”. Dans : *Proc. Graph Drawing (GD’05)*. T. 3843. LNCS. Springer-Verlag, 2005, p. 235–250.

[57] S. Hachul et M. Jünger. “Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm”. Dans : *Proc. Graph Drawing 2004 (GD’04)*. 2004, p. 285–295.

[58] S. Hachul et M. Jünger. “Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm”. Dans : *Proc. Graph Drawing (GD’04)*. T. 3383. LNCS. Springer-Verlag, 2004, p. 285–295.

[59] Frank van Ham et Jarke J. van Wijk. “Interactive Visualization of Small World Graphs”. Dans : *IEEE Symposium on Information Visualization (InfoVis04)*. Sous la dir. de Matt Ward et Tamara Munzner. IEEE Computer Society, 2004, p. 199–206.

[60] D. Harel et Y. Koren. “A Fast Multi-Scale Method for Drawing Large Graphs”. Dans : *Journal of Graph Algorithms and Applications* 6 (2002), p. 179–202.

[61] D. Holten. “Hierachical Edge Bundles : Visualization of Adjacency Relations in Hierarchical Data”. Dans : *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), p. 805–812.

[62] D. Holten et J. J. van Wijk. “Force-Directed Edge Bundling for Graph Visualization”. Dans : *11th Eurographics/IEEE-VGTC Symposium on Visualization (Computer Graphics Forum ; Proceedings of EuroVis 2009)*. T. 31. 2009, p. 983–990.

[63] John Hopcroft et Robert Tarjan. “Efficient Planarity Testing”. Dans : *J. ACM* 21.4 (oct. 1974), p. 549–568. issn : 0004-5411.

[64] P. Jaccard. “Distribution de la flore alpine dans le bassin de dranses et dans quelques regions voisines”. Dans : *Bulletin de la Societe Vaudoise des Sciences Naturelles*. 37. 1901, p. 241–272.

[65] T. Kamada et S. Kawai. “An Algorithm for Drawing General Undirected Graphs”. Dans : *Information Processing Letters* 31 (1989), p. 7–15.

[66] M. Granitzerand W. Kienreich et al. “Evaluating a system for interactive exploration of large, hierarchically structured document repositories.” Dans : *INFOVIS ’04 : Proceedings of the IEEE Symposium on Information Visualization (INFOVIS’04)*. IEEE Computer Society, 2004, p. 127–134.

[67] Y. Koren et D. Harel. “Graph Drawing by High-Dimensional Embedding”. Dans : *Proc. Graph Drawing (GD’02)*. T. 2528. LNCS. Springer-Verlag, 2002, p. 207–219.

[68] G. Kumar et M. Garland. “Visual Exploration of Complex Time-Varying Graphs”. Dans : *IEEE Trans. on Visualization and Computer Graphics (Proc. Vis/InfoVis 2006)* 12.5 (2006), p. 805–812.

[69] Antoine Lambert, David Auber et Guy Melançon. “Living flows : enhanced exploration of edge-bundled graphs based on GPU-intensive edge rendering”. Dans : *Proceedings of the 14th International Conference on Information Visualization (IV’10)*. Royaume-Uni, juil. 2010, p. 523–530.

[70] Antoine Lambert, Romain Bourqui et David Auber. “3D Edge Bundling for Geographical Data Visualization”. Dans : *Proceedings of the 14th International Conference on Information Visualization (IV’10)*. Royaume-Uni, juil. 2010, p. 329–335.

[71] Antoine Lambert, Romain Bourqui et David Auber. “Winding Roads : Routing edges into bundles”. Dans : *Computer Graphics Forum* 29.3 (2010), p. 853–862.

[72] Antoine Lambert, Jonathan Dubois et Romain Bourqui. “Pathway Preserving Representation of Metabolic Networks.” Dans : *Comput. Graph. Forum* 30.3 (2011), p. 1021–1030.

[73] Robert van Liere et Wim de Leeuw. “GraphSplatting : Visualizing Graphs as Continuous Fields”. Dans : *IEEE Transactions on Visualization and Computer Graphics* 9 (2003), p. 206–212.

[74] Tomer Moscovich et al. “Topology-Aware Navigation in Large Networks”. Dans : *SIGCHI conference on Human Factors in computing systems*. Sous la dir. d’ACM Press. ACM. 2009, p. 2319–2328.

[75] Niklas Elmqvist and Jean-Daniel Fekete. “Hierarchical Aggregation for Information Visualization : Overview, Techniques, and Design Guidelines”. Dans : *IEEE Transactions on Visualization and Computer Graphics* 16.3 (2010), 439–454.

[76] T. Pattison, R. Vernik et M. Phillips. “Information Visualization using Composable Layouts and Visual Sets”. Dans : *Proc. of the 2001 Asia-Pacific Symp. on Information Visualization*. 2001, p. 1–10.

[77] D. Phan et al. “Flow Map Layout”. Dans : *Proc. of IEEE Information Visualization Symposium*. Washington, DC, USA : IEEE Computer Society, 2005, p. 219–224.

[78] A. J. Pretorius et J.J. van Wijk. “Multidimensional Visualization of Transition Systems”. Dans : *Proc. of the International Conference of Information Visualization*. 2005, p. 323–328.

[79] Nathalie Henry Riche et Tim Dwyer. “Untangling Euler Diagrams”. Dans : *IEEE Transactions on Visualization and Computer Graphics* 16.6 (nov. 2010), p. 1090–1099. issn : 1077-2626.

[80] Peter Rodgers, Leishi Zhang et Andrew Fish. “General Euler Diagram Generation”. Dans : t. 5223. Springer, sept. 2008, p. 13–27.

[81] Peter Rodgers et al. “Embedding Well-formed Euler Diagrams”. Dans : *12th International Conference on Information Visualisation* (2008), p. 585–593.

[82] Jörg Sander et al. “Density-Based Clustering in Spatial Databases : The Algorithm GDBSCAN and Its Applications”. Dans : *Data Min. Knowl. Discov.* 2.2 (juin 1998), p. 169–194. issn : 1384-5810.

[83] D. Schaffer et al. “Navigating Hierarchically Clustered Networks through Fisheye and Full-Zoom Methods”. Dans : *ACM Trans. on Computer-Human Interaction (TOCHI)* 3.2 (1996), p. 162–188.

[84] B. Shneiderman et A. Aris. “Network Visualization by Semantic Substrates”. Dans : *IEEE Trans. on Visualization an Computer Graphics (Proc. Vis/InfoVis 2006)* 12.5 (2006), p. 733–740.

[85] Paolo Simonetto. “Diagrammes d’Euler pour la visualisation de communautés et d’ensembles chevauchants”. THESE. Université Sciences et Technologies - Bordeaux I, 2011.

[86] Paolo Simonetto et David Auber. “An Heuristic for the Construction of Intersection Graphs”. Dans : *Information Visualization*. Barcelona, Espagne, 2009, p. 673–678.

[87] Paolo Simonetto et David Auber. “Visualise Undrawable Euler Diagrams”. Dans : *Information Visualization*. London, UK, 2008, p. 594–599.

[88] Paolo Simonetto, David Auber et Daniel Archambault. “Fully Automatic Visualisation of Overlapping Sets”. Dans : *Computer Graphics Forum (EuroVis09)* 28.3 (2009), p. 967–974.

[89] Paolo Simonetto et al. “ImPrEd : An Improved Force-Directed Algorithm that Prevents Nodes from Crossing Edges”. Dans : *Computer Graphics -New York- Associa-*

tion for Computing Machinery- Forum 30.3 (2011).

[90] Gem Stapleton et al. "Properties of Euler diagrams". Dans : *Electronic Communications of the EASST* 7 (2007). issn : 1863-2122.

[91] S. T. Teoh et K. Ma. "RINGS : A Technique for Visualizing Large Hierarchies". Dans : *Proc. Graph Drawing (GD'02)*. T. 2528. LNCS. 2002, p. 268–275.

[92] C. Tominski et al. "Fisheye Tree Views and Lenses for Graph Visualization". Dans : *Proc. 10th Int. Conf. on Information Visualisation (IV'06)*. 2006, p. 17–24.

[93] Edward R. Tufte. *Visual Explanations : Images and Quantities, Evidence and Narrative*. First Edition. Cheshire, Connecticut : Graphics Press, 1997, p. 156. isbn : 0961392126.

[94] W. T. Tutte. "How to Draw a Graph". Dans : *Proc. London Mathematical Society* 13 (1963), p. 743–768.

[95] Anne Verroust et Marie-Luce Viaud. "Ensuring the Drawability of Extended Euler Diagrams for up to 8 Sets". Dans : *Diagrams 2004, 3rd International Conference*. T. 2980. Lecture Notes in Computer Science. Springer, 2004, p. 128–141.

[96] Christophe Viau et al. "The FlowVizMenu and Parallel Scatterplot Matrix : Hybrid Multidimensional Visualizations for Network Exploration". Dans : *IEEE Transactions on Visualization and Computer Graphics* 16.6 (nov. 2010), p. 1100–1108. issn : 1077-2626.

[97] G.F. Voronoi. "Nouveles applications des paramètres continus à la théorie de formas quadratiques". Dans : *J Reine Angew Math* 134 (1908), p. 198–287.

[98] C. Walshaw. "A Multilevel Algorithm for Force-Directed Graph Drawing". Dans : *Journal of Graph Algorithms* 7.3 (2003), p. 253–285.

[99] I. M. X. Wang. "Generating customized layouts". Dans : *Graph Drawing* (1996), p. 504–515.

[100] Colin Ware. *Information Visualization : Perception for Design*. Morgan Kaufmann Publishers Inc., 2004. isbn : 1558608192.

[101] Martin Wattenberg. "Visual exploration of multivariate graphs". Dans : *Proceedings of the SIGCHI conference on Human Factors in computing systems*. CHI '06. New York, NY, USA : ACM, 2006, p. 811–819.

[102] D. J. Welsh et M. B Powell. "An upper Bound to the chromaticnumber of a graph and its application to timetabling problems". Dans : *The Computer journal* 10 (1967), p. 85–86.

[103] N. Wong et S. Carpendale. "Using Edge Plucking for Interactive Graph Exploration". Dans : *Proc. of IEEE Information Visualization Symposium, Poster Compendium*. Washington, DC, USA : IEEE Computer Society, 2005, p. 51–52.

[104] N. Wong, S. Carpendale et S. Greenberg. "EdgeLens : An Interactive Method for Managing Edge Congestion in Graphs". Dans : *Proc. IEEE Symp. on Information Visualization (InfoVis'03)*. Washington, DC, USA, 2003, p. 51–58.

[105] M. Wybrow, K. Marriott et P.J. Stuckey. "Incremental connector routing". Dans : *Proc. Graph Drawing 2005 (GD'05)*. 2006, p. 446–457.

[106] H. Zhou et al. "Energy-Based Hierarchical Edge Clustering of Graphs". Dans : *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*. 2008, p. 55–61.