UNIVERSITE MONTPELLIER II
SCIENCES ET TECHNIQUES DU LANGUEDOC

T H E S E

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE MONTPELLIER II

Discipline : Informatique
Formation Doctorale : Informatique
Ecole Doctorale : Information, Structures, Systèmes

présentée et soutenue publiquement

par

Jérôme Thièvre

le 12 octobre 2006

Cartographies pour la Recherche et l'Exploration de données Documentaires

---

JURY

M. Michel Chein	Président
Mme Jocelyne Nanard	Directrice de Thèse
Mme Mountaz Hascoët	Directrice adjointe
Mme Marie-Luce Viaud	Encadrante industrielle
M. Michel Beaudouin-Lafon	Rapporteur
Mme Tamara Munzner	Rapporteur

Remerciements

Résumé

Cette thèse étudie l'utilisation de diagrammes pour la recherche et l'exploration de données documentaires.

La recherche de documents implique généralement l'utilisation de requêtes booléennes dont la formulation est problématique pour un grand nombre d'utilisateurs. Les diagrammes de Venn-Euler sont des représentations ensemblistes qui peuvent être utilisées comme interface de formulation graphique de requêtes complexes. Chaque diagramme constitue aussi une cartographie de la base documentaire qui fournit des informations sur son contenu et sur la qualité des termes de recherche employés.

Ces travaux ont été réalisés à l'Institut National de l'Audiovisuel (Ina), dont la mission est d'archiver les programmes audiovisuels des grandes chaînes de télévision et de la radio. Le format non textuel du document audiovisuel implique qu'il ne s'auto-décrit pas encore. Le fonctionnement du système documentaire de l'Ina repose donc sur la création de méta-données qui seront associées à chaque document sous la forme d'une notice de description textuelle. Ces méta-données enrichissent les documents en leur associant une sémantique. Elles permettent ainsi de déduire des relations entre différents documents ou, à granularité plus fine, entre différents éléments d'information. Ces réseaux d'éléments documentaires peuvent être visualisés par des diagrammes *nœud-lien*. La visualisation de diagrammes *nœud-lien* fournit des informations précieuses sur la structure des réseaux et l'importance de leurs entités. Cette application implique d'utiliser des techniques de visualisation avancées, telles que les calculs de placement des sommets d'un graphe et le codage graphique qui utilise la couleur, la forme et la taille des objets graphiques pour représenter de l'information.

Durant cette étude, plusieurs applications de visualisation ont été conçues. Le prototype de formulation graphique de requêtes booléennes nous a permis d'évaluer l'apport des diagrammes de Venn dans ce type de tâche. Afin de pallier à un défaut de passage à l'échelle des diagrammes de Venn nous nous sommes intéressés à la construction de diagrammes d'Euler, qui permettent dans certains cas de simplifier significativement les représentations obtenues. Nos travaux sur les réseaux d'éléments documentaires ont abouti à la conception d'une API et d'un prototype de visualisation de graphes, qui inclut de nombreuses méthodes de placement, de codage graphique et de filtrage. L'association de ces méthodes permet de créer des visualisations interactives et paramétrables particulièrement utiles à l'exploration et à l'analyse visuelle des réseaux complexes.

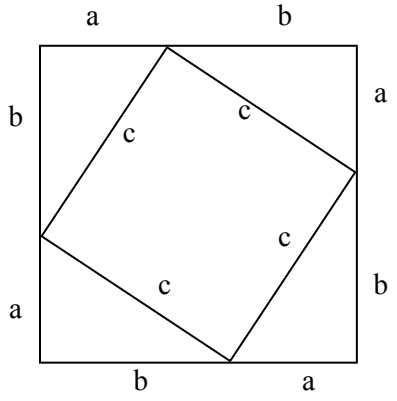
Mots-clés : diagramme de Venn, diagramme *noeud-lien*, visualisation d'information, graphe

Table des Matières

Introduction

La visualisation d'information a pour objectif de proposer des représentations graphiques interactives de structures de données abstraites afin de les rendre intelligibles. Ces représentations graphiques, ou diagrammes, doivent utiliser au mieux le plan et les indices visuels afin de communiquer de l'information.

Il convient d'abord de définir ce qu'est un diagramme et quelles sont ses propriétés. Le terme diagramme est composé des racines grecques dia, à travers et gramma, le dessin. Tout comme le dialogue est une communication par le discours, le diagramme serait une communication par le dessin. Un diagramme est une représentation visuelle d'un objet, produite dans le but de l'expliciter. Cette définition est encore un peu vague, elle englobe toute représentation visuelle. Le philosophe Deleuze précise que « le diagramme ne fonctionne jamais pour représenter un monde préexistant, il produit un nouveau type de réalité, un nouveau modèle de vérité ». Cette précision signifie-t-elle qu'un diagramme ne peut pas représenter un objet concret, du monde réel ? En fait, lorsqu'un diagramme représente un objet concret, il ne s'attache pas à le reproduire de manière réaliste, l'objectif est autre. Pour un diagramme très « concret », comme un plan d'architecte, le but n'est pas d'avoir uniquement une image du bâtiment terminé, mais de retrouver des informations utile à sa construction. Le problème ne se pose pas pour les objets abstraits. Les mathématiques, qui regorgent d'objets d'étude tout à fait abstraits, ont produits de nombreuses formes de diagrammes. La géométrie en est un exemple remarquable. La figure géométrique est un type de diagramme, elle permet non seulement de représenter spatialement un objet mathématique, un triangle par exemple, mais aussi de raisonner sur cet objet. Le théorème de Pythagore peut être facilement démontré grâce à une figure.



Cette figure est composée de 4 triangles rectangles de côtés adjacents de longueurs a et b, et d'hypoténuse c.
Cette figure montre clairement que l'aire du carré de côté c est égale à l'aire du carré (a+b) moins 4 fois l'aire du triangle a,b,c.
Soit

$$c^2 = (a+b)^2 - 4\frac{ab}{2}$$

et donc

$$c^2 = a^2 + b^2$$

Figure 1 : pythagore et diagramme

On notera d'ailleurs que bien que les figures géométriques soient le plus souvent fausses, elles permettent des raisonnements justes.

Un diagramme serait donc la représentation visuelle d'un objet, produite dans le but de comprendre cet objet et d'en déduire potentiellement de nouvelles propriétés. Le diagramme est un support de l'information et du raisonnement. A ce propos l'article de Larkin [Larkin87], dont le titre s'inspire du fameux « un bon dessin vaut mieux qu'un long discours », compare les deux représentations d'information que sont les formulations linguistiques et diagrammatiques. Il étudie

l'apport cognitif du diagramme en le comparant avec son équivalent linguistique. Lorsque tout ce qui peut se déduire d'une représentation peut aussi se déduire de l'autre, les représentations sont dites informativement équivalentes. De plus, lorsque des inférences peuvent être faites aussi facilement et rapidement sur les deux représentations, elles sont dites cognitivement équivalentes.

La puissance d'un « bon » diagramme est de faciliter l'acquisition des éléments d'information et de rendre explicite les relations qui peuvent exister entre eux. Dans le cas de la représentation linguistique, sous forme d'une liste, de tables, de phrases ou formules, le temps d'acquisition est, au mieux, proportionnel au nombre de phrases [Larkin87]. Sur un diagramme, la forme et l'agencement des symboles permettent de relier les données à utiliser conjointement. Les relations entre les différents éléments d'information sont explicites.

De manière générale, les diagrammes permettent d'élaborer une représentation mentale de données non immédiatement perceptibles dans leur ensemble : c'est un moyen d'organiser un ensemble d'objets dans un espace afin de les appréhender, les mettre en relation et éventuellement les mémoriser. Ce principe est très tôt utilisé pour synthétiser la réalité : la figure, le plan et la carte géographique sont nés. Les cartographies d'information sont virtuelles ou artificielles, dans la mesure où elles ne sont pas une représentation d'un monde ou d'une situation réelle, mais proposent de spatialiser à l'aide de certains principes, des données conceptuelles. Ainsi, elles offrent un « panorama » des données selon un point de vue défini et permettent de présenter une grande quantité d'information d'une façon structurée sur un espace réduit : elles font émerger du sens de la masse des informations représentées.

La caractéristique de la représentation cartographique par rapport à toute autre forme de présentation de données est de s'appuyer sur la perception visuelle pour faciliter la compréhension. Elle est composée d'éléments graphiques dont la forme, la position, la taille ou encore la couleur sont utilisées pour représenter les données. Ce processus d'association entre les propriétés des éléments graphiques et les propriétés des éléments de données est appelé un codage graphique. L'efficacité d'un codage est fortement liée aux caractéristiques de la perception visuelle humaine. Comme l'ont rapidement compris les géographes, l'étude et l'évaluation des systèmes de codage constituent une phase incontournable pour la mise en œuvre de cartographies efficaces. L'outil informatique offre non seulement des potentialités nouvelles au niveau du codage lui-même, mais il introduit aussi la notion d'interaction en tant qu'instrument instantané de manipulation des données. Selon le psychologue Gibson [Gibson77], il faut agir pour percevoir et percevoir pour agir. La visualisation interactive permet donc à l'utilisateur de contrôler la représentation des données dans une boucle action/perception. L'utilisateur appréhende les ressources mise à sa disposition en manipulant leur représentation en fonction de ses objectifs : il se construit instantanément une représentation mentale des propriétés à analyser. De par ces possibilités, la visualisation interactive ajoute à la fonction d'inventaire et de communication, une fonction de traitement et d'analyse de l'information.

L'Ina archive les programmes des chaînes de télévision françaises depuis 1949 et de radio depuis 1929. Les fonds comportent à ce jour plus de 3 millions de documents représentant environ 400 000 heures de vidéo et 500 000 heures de programmes audio. A chaque document, un documentaliste a associé une notice documentaire, résumant textuellement l'analyse du contenu audiovisuel. Ce contenu n'étant pas interprétable de manière univoque, il est nécessaire de faire appel à un ensemble de termes choisis, regroupés dans des listes spécifiques et dans un thésaurus. Ces notices forment ainsi

une base structurée de quelques 3 millions de documents indexés. Cette base est en constante augmentation.

Les données à représenter sont donc de plus en plus volumineuses et génèrent une complexité croissante des structures et de leurs représentations. Pour rendre ce contenu plus facilement accessible, il devient vital d'offrir aux utilisateurs les moyens de naviguer et d'analyser les données en leur permettant par exemple, de les observer à différentes échelles ou selon différents points de vue. Un des aspects de notre travail est l'élaboration de filtres interactifs permettant de simplifier les représentations tout en contrôlant la nature et la quantité des informations perdues. Le contrôle de l'utilisation des variables graphiques donne à l'utilisateur les moyens de faire émerger visuellement ou de masquer des éléments d'information pour satisfaire ses propres objectifs.

Dans ce document, nous nous sommes particulièrement intéressés à deux formes de diagrammes, les diagrammes de Venn-Euler et les diagrammes *nœud-lien*. Le document s'organise en quatre chapitres. Le premier est dédié aux diagrammes de Venn-Euler. Comment les construire et les utiliser en tant qu'interface d'interrogation. Les diagrammes de Venn-Euler sont utilisés pour représenter des ensembles et leurs intersections mutuelles. Les recherches de documents sur Internet ou dans des bases de documents produisent ce type de données, mais ils sont généralement présentés sous forme de listes de résultats qui masquent de fait leur structure ensembliste. Nous verrons que les diagrammes de Venn-Euler constituent une alternative intéressante à ces listes de résultats, car ils délivrent plus d'information contextuelle sur la pertinence des termes de recherche employés et sur la base de documents elle-même.

Le second chapitre traite des graphes en tant qu'objets théoriques. Nous donnerons des définitions et présenterons plusieurs métriques utiles à leur analyse. A fa fin de ce chapitre nous nous attacherons à étudier différentes modélisations du comportement des graphes, et notamment des graphes issus de données réelles. En effet, les technologies actuelles et la mise à disposition massive de contenu multimédia génèrent beaucoup de données de ce type. L'Internet peut être vu comme un immense graphe de pages Web se pointant les unes les autres. Le réseau des machines assurant son bon fonctionnement est aussi un graphe d'entités connectées entre elles. Les documents audiovisuels archivés par l'Ina peuvent traiter des mêmes sujets ou utiliser les mêmes images, ils peuvent aussi être vus comme des graphes dont les sommets sont des documents et les arêtes des relations de similarité entre documents.

Le troisième chapitre est dédié à la visualisation des diagrammes *nœud-lien*. Nous commencerons par une partie théorique sur la visualisation d'information au sens large. Nous nous recentrerons ensuite sur les diagrammes *nœud-lien*, en s'intéressant plus particulièrement aux méthodes de placement et au codage visuel de l'information.

Le quatrième et dernier chapitre présente la librairie de visualisation de graphes conçue durant cette thèse, ainsi que plusieurs exemples de son utilisation sur des graphes issus de données réelles, graphes du Web, réseaux bibliographiques, graphes de descripteurs des sujets des journaux télévisés.

1 Diagrammes de Venn-Euler

1.1 Qu'est-ce qu'un diagramme de Venn-Euler

Les diagrammes de Venn-Euler permettent de représenter des ensembles et leurs intersections. Prenons un exemple. Considérons un sondage effectué sur 50 personnes à propos des genres de programmes qu'ils regardent à la télévision.

Les résultats sont les suivants :

- 1 personne ne regarde aucun des genres proposés,
- 7 ne regardent que des films,
- 8 que des magazines,
- 4 que des documentaires,
- 10 ne regardent que des films et des magazines,
- 7 des films et des documentaires,
- 5 des magazines et des documentaires,
- 8 regardent films, magazines et documentaires.

Ces résultats peuvent être représenté sur un diagramme de Venn :

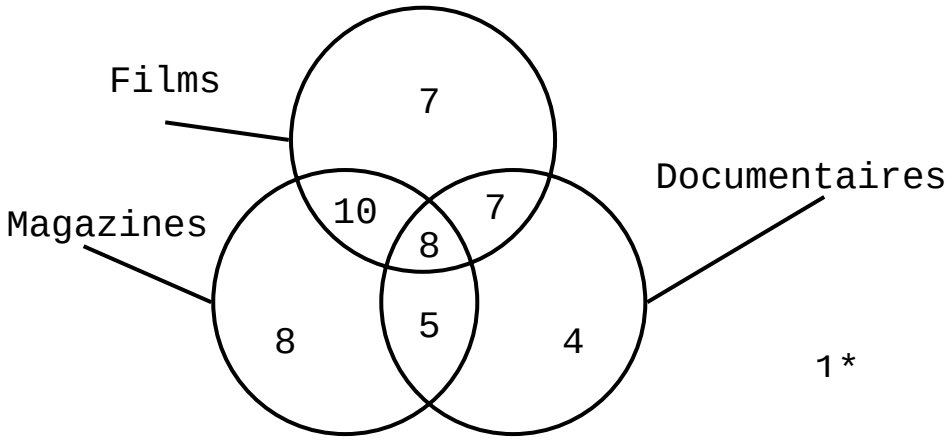


Figure 2 : diagramme de Venn aux cercles

Sous la forme linguistique, chaque catégorie est définie par une phrase, ces huit phrases sont présentées dans une liste ordonnée selon le nombre croissant de genres regardés.

Le diagramme de Venn présente lui aussi ces huit résultats, mais la représentation utilisée permet de définir chaque catégorie graphiquement sans recours à une forme linguistique. Ce diagramme de Venn est une représentation basée sur l'analogie entre un ensemble et un disque. Le plan représente l'univers, et chaque disque symbolise un ensemble. Un cercle, et plus généralement une courbe simple et fermée du plan, a la propriété de découper le plan en deux régions, une région intérieure - le disque -

et une extérieure - la région complémentaire au disque. La région intérieure symbolise l'ensemble, la région extérieure le reste de l'univers. L'analogie va plus loin car tout comme deux ensembles, deux disques peuvent posséder une intersection non vide. A l'intersection ensembliste répond l'intersection géométrique. Il est à noter ici, qu'il est important que les courbes soient simples et fermées afin que l'analogie soit respectée. Il ne serait en effet pas naturel qu'un ensemble, soit représenté par plusieurs régions disjointes du plan. Il y a en effet une notion d'unicité dans l'image qu'on se fait d'un ensemble, soit un élément lui appartient, soit non, soit on est à l'intérieur, soit à l'extérieur, et il n'y a pas de raison de considérer « l'intérieur » d'un ensemble comme plusieurs entités. Le mathématicien Cantor disait d'ailleurs d'un ensemble qu'il est « une multitude pensée comme un tout ».

Chaque catégorie est représentée par une des huit régions élémentaires du diagramme générées par l'intersection des trois cercles. Le nombre d'individu pour une catégorie est placé dans la région correspondante.

Du point de vue information, on retrouve effectivement les mêmes éléments dans les deux représentations, elles sont donc bien informativement équivalentes [Larkin87]. Du point de vue cognitif, on observe, dans le diagramme, que chacune des huit catégories est repérée par rapport aux trois ensembles étudiés, ce qui n'est pas le cas sous la forme linguistique. Il est, par exemple, beaucoup plus facile de voir sur le diagramme qu'il y a 24 personnes qui regardent des documentaires, il suffit d'additionner les quatre nombres contenus dans le cercle étiqueté « Documentaires ». Sous la forme linguistique, il est nécessaire de parcourir toute la liste pour interpréter chaque phrase afin de repérer et de dénombrer les individus. Le diagramme de Venn présente une organisation spatiale de l'information plus pertinente que la liste. La position des éléments d'information explicite la sémantique des relations qui existent entre eux. Le diagramme offre ainsi plusieurs lectures différentes de l'information. On peut y accéder par ensemble, si l'on considère successivement chaque contour, on peut retrouver aussi l'ordre de lecture offert par la liste, en considérant d'abord les régions les plus à l'extérieur du diagramme, puis les régions intérieures.

Dans la suite de ce chapitre, nous expliquons pourquoi et comment nous envisageons d'utiliser les diagrammes de Venn pour aider l'utilisateur durant le processus de recherche d'information. Nous définirons ensuite de manière plus formelle ce qu'est un diagramme de Venn-Euler. Nous nous intéresserons aussi aux problèmes théoriques et pratiques liés à ces diagrammes ; est-il toujours possible de dessiner un diagramme de Venn-Euler, comment le faire, sont-ils lisibles ? Enfin, nous présenterons notre contribution à ce domaine avec l'usage des diagrammes de Venn-Euler en recherche d'information, et la création dynamique de diagrammes d'Euler.

1.2 Motivations

Le système documentaire de l'Ina est basé sur l'indexation des notices de description des documents audiovisuels. Ces notices accompagnent les matériels audiovisuels, qu'ils soient numérisés sous forme de fichier Mpeg ou encore stockés sous forme de bandes magnétiques ou de films. Chaque notice décrit minutieusement le support qu'elles accompagne. Les documents archivés à l'Ina forment un ensemble très hétérogène, et par conséquent leurs notices présentent elles aussi des différences importantes. Deux notices peuvent être formées de champs différents. L'ensemble de ces champs permet d'offrir assez de souplesse pour s'adapter aux différents types de contenu. Les notices présentent, cependant, toujours un noyau d'information homogène. Ce noyau est constitué des informations de production qui spécifient le ou les titres du document, les dates de production et de diffusion, les intervenants les plus importants tels que les producteurs, les réalisateurs, ou encore les présentateurs ou acteurs,… Le cœur du document est lui aussi systématiquement décrit de manière synthétique par des termes descripteurs. Ces termes ou mots-clés sont issus de lexiques et de thésaurus. Ils ont donc été choisis par un collège de documentalistes comme base d'un langage minimal de représentation de l'information. Lors du processus de documentation d'un document audiovisuel, plusieurs termes sont consciencieusement choisis afin d'exprimer l'essentiel de son contenu. L'utilisation de descripteurs normés assure un accès homogène à l'information dans une base très hétérogène. Les descripteurs sont plus robustes que les données plein-texte qui peuvent générer beaucoup de bruit lors d'une recherche.

La recherche documentaire s'appuie sur cet index de notices de description. L'utilisateur exprime ses critères de recherche à travers une requête. L'exécution de cette requête ramène l'ensemble des documents dont les notices satisfont ses critères. Une requête élémentaire est définie comme un couple (Champs, Texte). Cette requête élémentaire correspond à la recherche des documents dont le champs spécifié contient le texte défini. Une requête est généralement constituée de plusieurs requêtes élémentaires articulées par des opérateurs booléens.

La fonction des opérateurs booléens dans la structure d'une requête peut être interprétée de deux façons. Avec l'approche logique ils se traduisent par la combinaison logique de plusieurs critères. Ainsi, le critère (*A ET B*) sera vérifié si les deux critères *A* et *B* sont vérifiés, alors que le critère (*A OU B*) sera vérifié si au moins un des deux critères *A* et *B* est vérifié. Enfin le critère (*NON A*) sera vérifié si le critère *A* n'est pas vérifié. Avec l'approche ensembliste, on considère alors les ensembles de documents résultats de chaque requête élémentaire. Les opérateurs booléens peuvent alors être interprétés comme effectuant des opérations ensemblistes sur les ensembles résultats. L'ensemble résultat de la requête (*A ET B*) est l'intersection des ensembles résultats des requêtes *A* et *B*. Le OU se traduit par une union des ensembles résultats, et le NON par l'opération de complémentarité.

Il a été reconnu que le langage booléen pose des problèmes à de nombreux utilisateurs [Michard82, Shneiderman98]. La sémantique des opérateurs booléens est sujette à mauvaise interprétation. L'opérateur *ET* est parfois interprété comme effectuant l'union des ensembles résultats, et l'opérateur *OU* est parfois confondu avec un *OU EXCLUSIF*. De plus la précédence des opérateurs qui définit l'ordre dans lequel ils doivent être interprétés est souvent ignoré, et par conséquent l'usage des parenthèses est souvent erroné. Nous pensons que l'utilisation de diagrammes de Venn permet d'éviter ces erreurs de formulation en proposant à l'utilisateur un mode de formulation basée sur l'approche ensembliste qui semble plus naturelle que l'approche logique purement booléenne. La formulation d'une requête booléenne peut être effectuée graphiquement, en sélectionnant une ou plusieurs régions par pointage dans un diagramme de Venn.

1.3 Définitions

De nombreuses définitions des diagrammes de Venn-Euler existent [Bowles71, Grünbaum75] et sont en générales assez complexes. C'est pourquoi je commencerai par quelques définitions relatives aux ensembles et leurs intersections. Nous verrons qu'une définition des diagrammes en découle directement.

1.3.1 Ensembles

Soit Σ un ensemble et $E = \{E_1,\ldots,E_n\}$ un ensemble de parties de Σ. Les diagrammes de Venn permettent de représenter les E_i et toutes leurs intersections potentielles.

Tout élément e de Σ peut appartenir à aucun, un, ou plusieurs des E_i. Plus précisément pour chaque E_i, on a soit e appartient à E_i soit e appartient au complémentaire de E_i dans Σ.

Donc on a :

$$\forall e \in \Sigma, \exists \left\{E_e^+, E_e^-\right\} \text{une partition de } E \text{ telle que } e \in I = \bigcap_{E_i \in E_e^+} E_i \cap \bigcap_{E_j \in E_e^-} \overline{E}_j$$

Tout ensemble I défini comme ci-dessus sera appelé intersection élémentaire de E. Soit Φ l'ensemble de ces intersections.

$$\Phi(E) = \Phi\left(\{E_1,\ldots,E_n\}\right) = \left\{ I = \bigcap_{E_i \in E^+} E_i \cap \bigcap_{E_j \in E^-} \overline{E}_j, \text{ tel que } \left\{E^+, E^-\right\} \text{ est une partition de } E \text{ et } I \neq \emptyset \right\}$$

Il y a 2^n partitions de $E = \{E_1,\ldots,E_n\}$ en deux sous-ensembles, mais certains ensembles I peuvent être vides .

Donc :

$$card\left(\Phi\left(\{E_1,\ldots,E_n\}\right)\right) \leq 2^n$$

Si l'égalité est atteinte on dit que l'ensemble des intersections élémentaires est complet, et on le note Φ_C. On remarquera de plus que, dans ce cas, le nombre d'intersections élémentaires générées par l'intersection de p E_i et (n-p) complémentaires de E_j, $i \neq j$, est égal au nombre de combinaisons de p éléments parmi n.

1.3.2 Diagrammes de Venn-Euler

Dans le plan, une courbe de Jordan est une courbe homéomorphe à un cercle. Plus concrètement, il s'agit de courbes fermées dont on peut définir un intérieur et un extérieur. Des exemples simples sont le cercle, l'ellipse, un carré ou encore les fameux patatoïdes.

Soient :

$E = \{E_1,\ldots,E_n\}$ un ensemble de parties de Σ,

$C = \{C_1, \ldots, C_n\}$, un ensemble de courbes de Jordan.
Tout point p de R^2 peut appartenir à aucun, une, ou plusieurs des régions intérieures de C_i. Plus précisément pour chaque C_i, on a soit p appartient à l'intérieur de C_i soit p appartient à l'extérieur de C_i.

On a donc :

$$\forall p \in \mathrm{R}^2, \exists \left\{C_p^+, C_p^-\right\} \text{ une partition de } C \text{ telle que } p \in R = \bigcap_{C_i \in C_p^+} \mathrm{int}(C_i) \cap \bigcap_{C_j \in C_p^-} ext(C_j)$$

Toute région R définie comme ci-dessus sera appelée région élémentaire de C. Soit Φ l'ensemble des ces régions.

$$\Phi(C) = \Phi\left(\{C_1, \ldots, C_n\}\right) = \left\{ R = \bigcap_{C_i \in C^+} \mathrm{int}(C_i) \cap \bigcap_{C_j \in C^-} ext(C_j), \text{ tel que } \left\{C^+, C^-\right\} \text{ est une partition de } C \text{ et } R \neq \emptyset \right\}$$

Il y a 2^n partitions de $C = \{C_1, \ldots, C_n\}$ en deux sous-ensembles, mais certaines régions R peuvent être vides .

Donc :

$$card\left(\Phi\left(\{C_1, \ldots, C_n\}\right)\right) \leq 2^n$$

Si l'égalité est atteinte on dit que l'ensemble des régions élémentaires est complet, et on le note Φ_C. On remarquera de plus que, dans ce cas, le nombre de régions élémentaires générées par l'intersection de p C_i et $(n\text{-}p)$ complémentaires de C_j, $i \neq j$, est égal au nombre de combinaison de p éléments parmi n.

Le couple $D = (E, C)$ est un **diagramme d'Euler** si et seulement si il existe une bijection

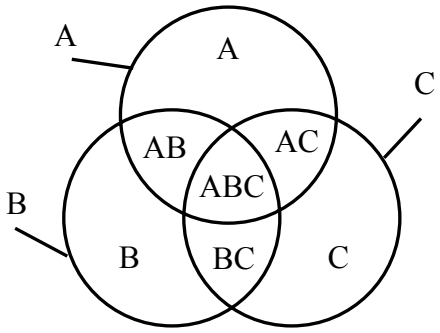
$diag : (E \cup \Phi(E)) \rightarrow R^2$ telle que :

$$\forall E_i \in E \text{ et } C_i \in C, diag\left(E_i\right) = C_i,$$

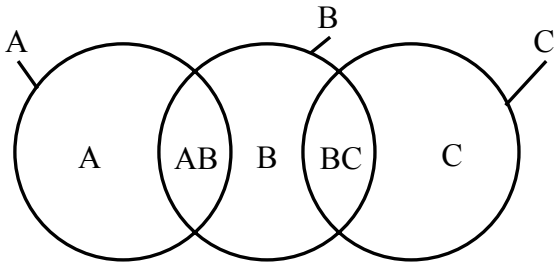
$$\forall \left\{E^+, E^-\right\} \text{ partition de } E,$$

$$\forall I = \bigcap_{E_i \in E^+} E_i \cap \bigcap_{E_j \in E^-} \overline{E}_j, R \text{ est simple et fermée, et } diag\left(I\right) = \bigcap_{E_i \in E^+} int(diag(E_i)) \cap \bigcap_{E_j \in E^-} ext(diag(E_j)).$$

Si $\Phi(C) = \Phi_C(C)$ on parle alors d'un **diagramme de Venn**. Un diagramme de Venn composé de n courbes et appelé n-diagramme de Venn et il est donc composé de 2^n régions atomiques. Un diagramme de Venn est en fait un cas particulier de diagrammes d'Euler, celui où toutes les régions élémentaires existent. La Figure 3 montre un 3-diagramme de Venn et un 3-diagramme d'Euler. Un petit point sur la notation utilisée dans les diagrammes. Chaque région élémentaire d'un diagramme de Venn-Euler peut être identifiée par l'intersection élémentaire qui lui correspond par la bijection. Cependant cette notation devient vite pesante, c'est pourquoi on préfère souvent identifier une région par la liste des ensembles (ou contours) qui la contiennent, comme on le voit sur la figure ci-dessous.



Un diagramme de Venn à 3 ensembles, il est composé de 8 régions (avec l'extérieur)



Un diagramme d'Euler à 3 ensembles, les régions AC et ABC sont absentes.

Figure 3 : différence entre diagramme de Venn et d'Euler

Définissons cette fonction d'étiquetage, elle nous servira par la suite. Soit *label :* $\Phi(C) \rightarrow L^*$ la fonction qui assigne à chaque région élémentaire du diagramme un label à partir du lexique $L = \{L_1,\ldots,L_n\}$,

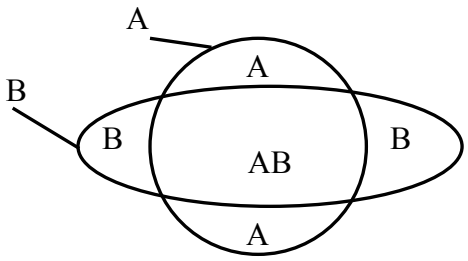
$$\forall C_i \in C, label\ (C_i) = L_i,$$

$$\forall \left\{C^+, C^-\right\} partitionde C,$$

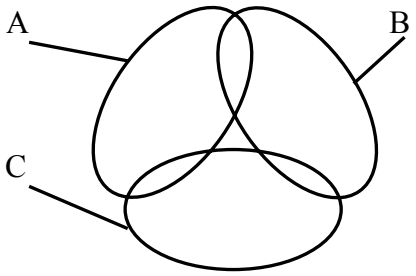
$$\forall R = \bigcap_{C_i \in C^+} int(C_i) \cap \bigcap_{C_j \in C^-} ext(C_j) \Leftrightarrow label(R) = \bigcup_{C_i \in C^+} label(C_i)$$

Un diagramme de Venn/Euler est dit **simple** si le nombre de points d'intersections des courbes est fini et si tous ces points sont simples, c'est-à-dire définis par l'intersection d'exactement deux courbes. Cela implique qu'il ne doit pas y avoir des segments de courbes superposés, et qu'exactement deux courbes passent par chaque point d'intersection.

Attention, si les régions élémentaires ou les contours ne sont pas simples, on obtient un diagramme qui n'est pas un diagramme d'Euler, voir ci-dessous.



les régions correspondant à $A \cap \overline{B}$ et $\overline{A} \cap B$ ne sont pas simples



la région correspondant à $\overline{A} \cap \overline{B} \cap \overline{C}$ n'est pas simple

Figure 4 : exemples de diagrammes invalides

1.4 Résultats théoriques et pratiques

Les diagrammes de Venn-Euler sont des objets très anciens, ils ont donc fait l'objet de nombreux travaux. La paternité de ces diagrammes revient à Leibniz, mais c'est Euler [Euler61] qui les a

popularisé en les utilisant pour la représentation des syllogismes. A ce propos l'article [Glassner03] est très instructif et fort bien illustré. Venn [Venn80] s'intéressera plus au diagramme en tant qu'objet d'étude, c'est lui qui initiera la recherche de diagramme pour représenter plus de trois ensembles. Vous trouverez de nombreuses informations quant aux diagrammes de Venn sur le site Internet de Ruskey [Ruskey01].

Dans ce paragraphe je parlerai de l'existence de diagrammes de Venn-Euler plus ou moins contraints et de la construction de tels diagrammes lorsque cela est possible. Ces diagrammes ont des spécificités géométriques qui les rendent plus ou moins lisibles. L'utilisation de diagrammes en visualisation d'information implique de se pencher sur cette question de lisibilité.

Enfin, je développerai quelques points quant à nos contributions [Thievre03, 04, 05], la représentation des résultats de recherche par diagrammes et la construction de diagramme d'Euler étendus jusqu'à huit ensembles.

1.4.1 Existence des diagrammes de Venn

Est-il possible de dessiner un n-diagramme de Venn ? Oui, et plusieurs méthodes de construction le démontrent. De nombreuses recherches ont été entreprises sur ce type de méthodes [Bowles71, Fisher88, More59, Polythress71] je ne parlerai ici que des travaux initiaux de Venn [Venn80] et ceux plus récents d'Edwards [Edwards89].

1.4.1.1 Construction incrémentale

Cette technique consiste à construire un n-diagramme de Venn à partir d'un (*n-1*)-diagramme de Venn par l'ajout d'une courbe. Si $D_{n-1} = \{C_1,\ldots,C_{n-1}\}$ est un (*n-1*)-diagramme de Venn, il faut trouver une courbe C_n telle que :

$$D_n = D_{n-1} \cup \{Cn\} = \{C_1,...,C_n\}$$

Afin de s'assurer que D_n est bien un n-diagramme de Venn simple, C_n doit bien sûr être une courbe de Jordan. Il est nécessaire et suffisant que cette courbe sépare chaque région élémentaire de D_{n-1} en exactement deux régions distinctes, en ne passant par aucun point d'intersection existant pour former les régions élémentaires de D_n.

1.4.1.2 Construction de Venn

Cette méthode incrémentale a été proposée par Venn [Venn80] dans sa recherche de diagrammes pour plus de 3 ensembles. Les courbes des quatrième et cinquième contour sont ajoutées au diagramme à trois cercles, comme illustré ci-dessous.

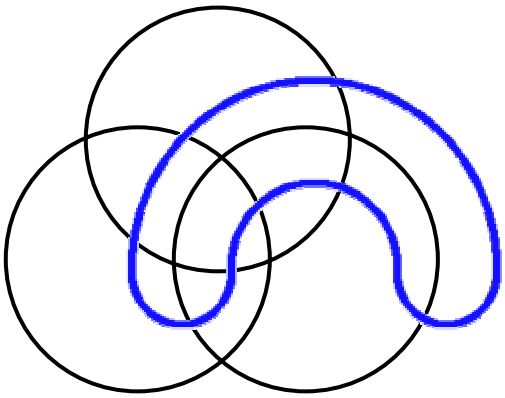


Figure 5 : 4-diagramme de Venn

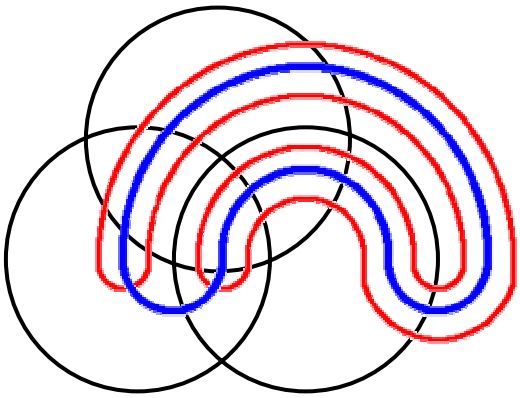


Figure 6 : 5-diagramme de Venn

On remarque que chaque courbe ajoutée vient bien séparer chaque région du diagramme précédent en exactement deux régions, on obtient donc bien des diagrammes de Venn.

Cet algorithme permet théoriquement de construire n'importe quel n-diagramme de Venn, cependant le tracé des courbes devient à chaque étape plus complexe, et bien que les diagrammes obtenus soient théoriquement valides, leur lisibilité est très médiocre.

1.4.1.3 Construction d'Edwards

Il y a une dizaine d'années Edwards [Edwards89] présente une élégante technique de construction incrémentale de n-diagrammes de Venn. C'est d'abord le choix des 3 contours initiaux qui est original. Un 2-diagramme de Venn est composé de quatre régions. Edwards choisit de dessiner ce diagramme avec deux rectangles qui sépare le plan en 4 régions rectangulaires congruentes, voir Figure 7. Le cercle qu'il choisit comme troisième contour permet de conserver la symétrie centrale du diagramme, voir Figure 8. Chaque contour suivant slalome élégamment entre l'intérieur et l'extérieur de ce cercle. La Figure 12 montre le motif utilisé pour ces contours. On remarque qu'à partir du 2-diagramme, les diagrammes possèdent une symétrie centrale, cela implique que chaque contour ajouté ensuite possède cette symétrie.

Figure 7 : Edwards pour 2 ensembles

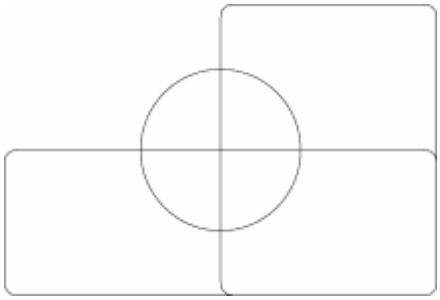


Figure 8 : Edwards pour 3 ensembles

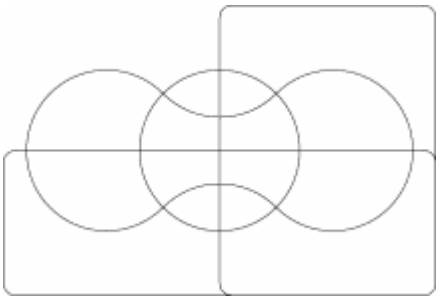


Figure 9 : Edwards pour 4 ensembles

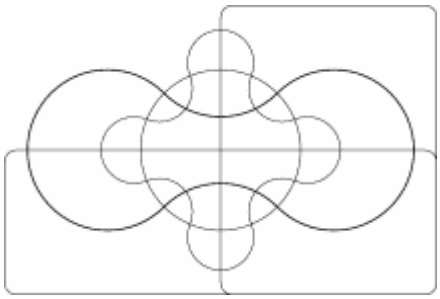


Figure 10 : Edwards pour 5 ensembles

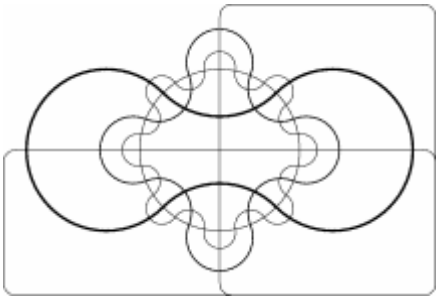


Figure 11 : Edwards pour 6 ensembles

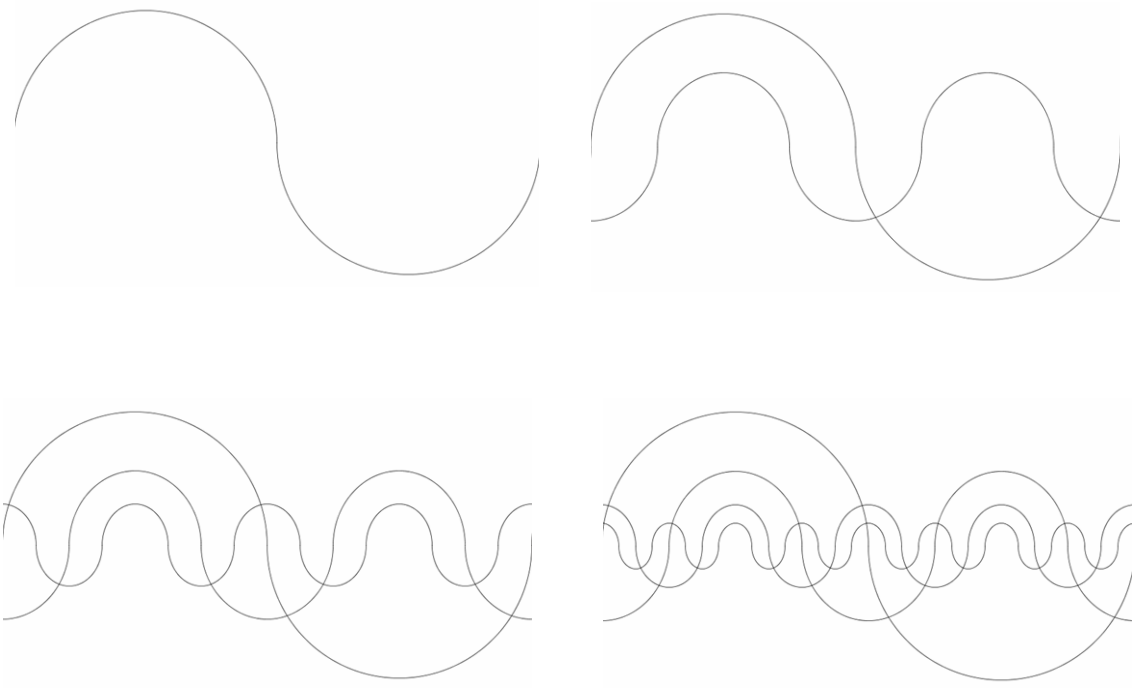


Figure 12 : évolution des courbes à partir de 4 ensembles

A partir du troisième ensemble, chaque contour est composé de demi-cercles plus ou moins déformés. Chaque contour est composé de deux fois plus de demi-cercles que le contour précédent. Cette technique proche de la construction des fractales est très facilement automatisable.

Les diagrammes obtenus par cette technique sont plus lisibles que les diagrammes construits précédemment à nombre d'ensembles égal. La symétrie des courbes améliore la lecture, leur tracé est plus facile à suivre et elles sont continues. Cependant, chaque nouvelle courbe possède deux fois plus d'arcs que la précédente et les contours deviennent vite très complexes.

Ces deux méthodes montrent qu'il possible, en pratique, de dessiner un n-diagrammes de Venn pour tout n. Les diagrammes issus de la méthode de Venn sont simples, cf. 1.3.2. Ceux d'Edwards ne le sont pas sur cet exemple, car les deux rectangles ont une partie de leurs contours superposés, mais c'est très simple à corriger.

Je m'attarde sur ce dernier point car cette propriété n'est pas compatible avec l'existence de certains diagrammes d'Euler. Par exemple si l'on veut représenter deux ensembles A et B égaux, il est nécessaire que les contours de A et B soient identiques. Nous verrons qu'il est nécessaire de relâcher certaines contraintes afin de pouvoir dessiner certains diagrammes d'Euler.

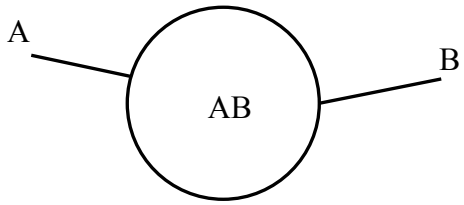


Figure 13 : diagramme d'Euler pour deux ensembles égaux, ce diagramme n'est pas simple

1.4.2 Graphe dual et Topologie

Un diagramme de Venn-Euler peut être vu comme un graphe planaire. Il suffit de considérer les intersections comme des sommets, les arêtes sont alors les parties de contours qui relient deux intersections. Ce graphe n'est pas directement utile, par contre tout dessin pouvant être transformé en graphe planaire, c'est le cas pour les diagrammes de Venn-Euler, possède un graphe dual géométrique [Harary94].

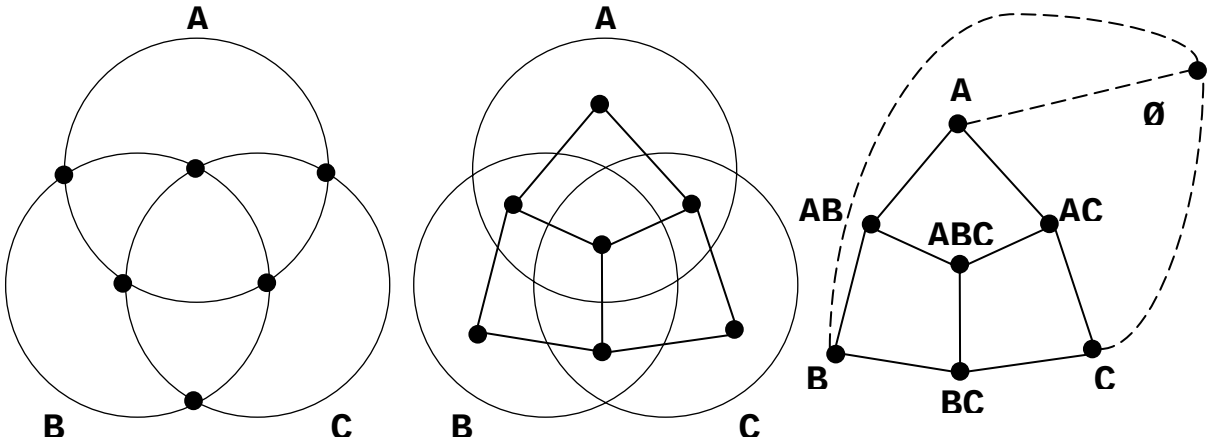


Figure 14 : un diagramme de Venn-Euler et son graphe dual

A tout diagramme de Venn-Euler on peut donc associer son graphe dual. A chaque région atomique du diagramme correspond un sommet dans le graphe dual. Si deux régions atomiques possèdent une frontière commune, alors les deux sommets correspondants dans le graphe dual sont liés par une arête. Le graphe dual obtenu ainsi est planaire. Si on ne considère pas la position des sommets du graphe dual on parle de graphe dual abstrait. On peut remarquer que le graphe dual des n-diagrammes de Venn, pour $n \leq 3$, est un hypercube de dimension n, voir ci-dessous. Le placement du graphe dual pour le 3-diagramme de Venn présenté ici n'est pas planaire, j'ai choisi le placement qui révèle la structure hypercubique (et dans ce cas cubique) du graphe. Toutefois si on place la face (C, BC, ABC, AC) à l'intérieur de la face (Ø, B, AB, A) le graphe est bien planaire.

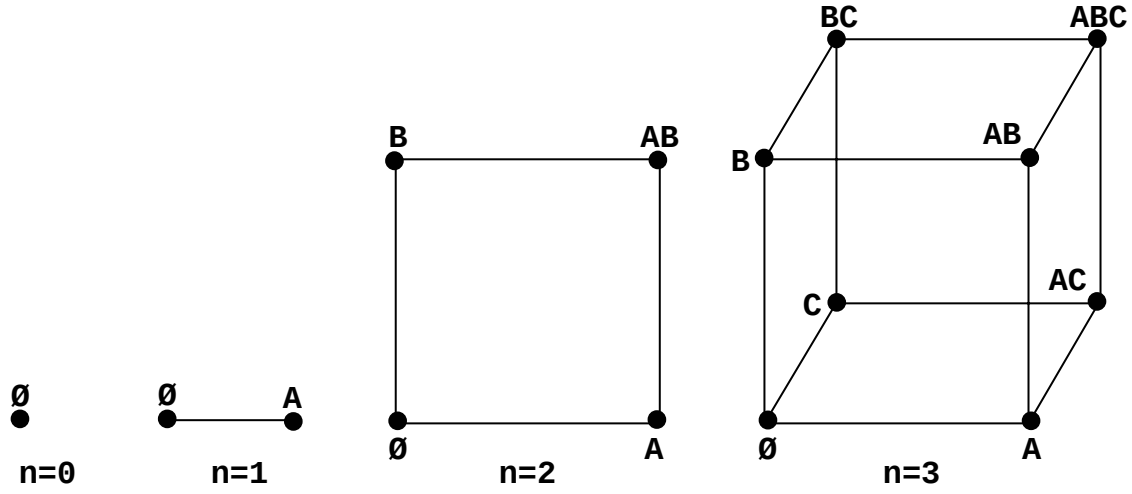


Figure 15 : Les graphes duals des n-diagranmmes de Venn sont des hypercubes

A partir de n=4, l'hypercube n'est plus planaire. Dans ce cas le graphe dual d'un n-diagramme de Venn simple est nécessairement un sous-graphe planaire de l'hypercube de dimensions n.

Le plongement dans le plan du graphe dual spécifie les propriétés topologiques d'un diagramme, c'est-à-dire qu'il définit l'ensemble des régions présentes dans le diagramme et la manière dont elles

sont placées les unes par rapport aux autres. Le plongement du graphe dual d'un diagramme est planaire. Il peut exister plusieurs plongements d'un graphe dual dans le plan qui conduisent à des diagrammes topologiquement différents

Les propriétés géométriques telles que la forme des contours et la position des points d'intersection ne sont pas spécifiées. Aussi, pour un même plongement, il est possible de construire une infinité de diagrammes géométriquement différents, mais topologiquement équivalents.

1.4.2.1 Définitions

Graphe dual

Soit $D = (E,C)$ un diagramme d'Euler. Soit $G = (S,A)$ un graphe. G est le graphe dual de D si et seulement si :

Il existe un isomorphisme de $\Phi(C)$ vers l'ensemble S des sommets. Soit $graph : \Phi(C) \rightarrow S$, la bijection qui fait correspondre à chaque région élémentaire du diagramme un sommet de S.

L'ensemble des arêtes A est défini par :

$$A = \left\{ a = (s,t) \in S \times S, R_s = graph^{-1}(s) \text{ et } R_t = graph^{-1}(t), R_s \text{ et } R_t \text{ ont une frontière commune} \right\}$$

On peut étendre la définition de la fonction *label*, afin que sommet et région correspondant partagent la même étiquette.

$$\forall R \in \Phi(C), s = graph(R) \Rightarrow label(s) = label(R),$$

Les arêtes aussi peuvent être étiquetées. On choisit l'étiquetage suivant : l'arête $a = (s,t)$ porte comme étiquette l'ensemble des lettres qui ne sont pas à la fois dans les étiquettes de s et t.

$$label((s,t)) = label(s) \cup label(t) - label(s) \cap label(t)$$

En procédant ainsi l'étiquette d'une arête permet d'identifier l'ensemble des contours du diagramme qui devront la franchir.

Connectivité

Un graphe G est dit l-connecté si tous les sommets contenant un label l constituent un sous graphe connexe de G. G est dit L-connecté s'il est l-connecté pour tous les éléments l de L.

Propriété

Si D est un diagramme d'Euler alors son graphe dual est planaire et L-connecté. Et donc, si G n'est pas planaire ou L-connecté , le diagramme correspondant n'est pas un diagramme d'Euler.

1.4.3 Construction de Diagrammes d'Euler

Soit Σ un ensemble et E = {E_1,…, E_n} un ensemble de parties de Σ. Les diagrammes d'Euler, quand ils existent, permettent de représenter les E_i et toutes leurs intersections non vides, c'est-à-dire les éléments de $\Phi(E)$ tels que :

$$\Phi(E) = \Phi(\{E_1,\ldots,E_n\}) = \left\{ I = \bigcap_{E_i \in E^+} E_i \cap \bigcap_{E_j \in E^-} \overline{E}_j, \text{ tel que } \{E^+, E^-\} \text{ est une partition de } E \text{ et } I \neq \emptyset \right\}$$

Plus précisément, si $\Phi(E) = \{Y_1,\ldots,Y_k\}, k \leq 2^n$, un diagramme d'Euler aura k régions, chacune correspondant à un Y_i.

Euler a introduit ses diagrammes en utilisant des cercles, mais tous les diagrammes ne peuvent pas être dessinés avec de tels contours, cf. 1.4.4.1. En utilisant des contours convexes mais non circulaires, il devient possible de dessiner un plus grand nombre de diagrammes.

D'autres contraintes doivent être relâchées pour étendre le nombre de diagrammes dessinables :

- les contours des Y_i peuvent avoir des parties communes non réduites à un point,
- plus de deux contours peuvent s'intersecter en un même point,
- la zone du plan correspondant à un Y_i est connexe mais peut comporter des trous : dans ce cas cette zone sera définie par plusieurs contours ; un contour externe et des contours internes bordant les trous.

Les diagrammes d'Euler étendus sont ceux qui satisfont ces conditions. Ils sont définis par :

Soit Σ un ensemble, E = {E_1,…, E_n} un ensemble de parties de Σ .
Un diagramme d'Euler étendu sur E est un triplet (L, C, R) tel que :

- L est un ensemble fini de labels (L en bijection avec $\Phi(E)$)
- C est un ensemble de courbes de Jordan étiquetées par L et vérifiant :
  - pour tout l dans L, il existe C_i dans C tel que label(C_i)=l et sign(C_i)= +,
  - si label(C_i)= label(C_k) , avec C_i différent de C_k et sign(C_i)=sign(C_k), alors C_i et C_k ne s'intersectent pas,
  - si label(C_i)= label(C_k), avec C_i différent de C_k et sign(C_i)=+, alors sign(C_k)= - et C_k est inclus dans int(C_i).

R est l'ensemble des régions correspondant à la partition planaire définie par C.
Chaque région de R est étiquetée avec un ensemble de labels de L.

L'article [Verroust04] montre que le graphe dual d'un diagramme d'Euler étendu (L, C, R) est un graphe L-connecté planaire. Dans ce même article, une démonstration constructive sur l'existence de diagrammes d'Euler étendus est donné pour tout ensemble E de cardinalité n < 9.

En effet, le dessin d'un diagramme d'Euler ne peut être assuré dans tous les cas à partir de n = 9. Comme le montre l'exemple de la Figure 16, pour lequel il n'existe pas de graphe dual L-connecté planaire et donc pas de diagramme.

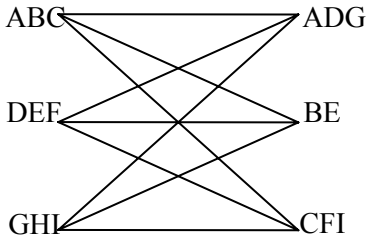


Figure 16 : graphe $K_{3,3}$

La démonstration consiste à construire progressivement un graphe dual L-connecté planaire. Ce graphe a un ensemble V de k sommets correspondant aux régions du diagramme d'Euler étendu. Ces sommets sont étiquetés par des mots de L.

Les grandes lignes de cette construction sont les suivantes :

1. On sélectionne un sous ensemble V0 de sommets tels que :
    a. l'ensemble des labels contenus dans les étiquettes des éléments de V0 est égal à L
    b. l'ensemble V0 est de cardinalité minimale
    c. la longueur des mots associés aux éléments de V0 est maximale
2. On montre que l'on peut construire un graphe G0 L-connecté planaire sur V0 quand n < 9.
3. On montre que, quand n < 9, les sommets de V n'appartenant pas à V0 peuvent être ajoutés à G0 par l'insertion progressive de 1, 2, 3 ou 4 arêtes les liant aux sommets de G0, tout en conservant la planarité et la L-connectivité du graphe résultant.

Une fois ce graphe L-connecté placé, un diagramme d'Euler étendu peut être facilement construit [Thievre05].

1.4.3.1 Dessin d'un diagramme d'Euler à partir d'un graphe L-connecté

Rappelons que chaque sommet du graphe détermine une région du diagramme, et que chaque arête doit être traversée par au moins un contour.

Un sommet et sa région associée possèdent le même label. Le label d'une arête identifie les contours qui la franchissent.

La première étape de construction est une simple triangulation du graphe L-connecté. Cette étape consiste à ajouter le nombre minimum d'arêtes pour rendre triangulaires toutes les faces internes du graphe. Sur la Figure 17, la triangulation résulte en l'ajout de l'arête (abd , cd) qui permet de transformer la face de 4 côtés (abd , ac , cd, bd) en deux faces triangulaires. La triangulation n'est pas nécessaire pour produire un diagramme, mais elle facilite l'écriture de l'algorithme de dessin en réduisant le nombre de cas à traiter. Sachant que toutes les faces internes sont des triangles, il est possible de leur appliquer exactement les mêmes opérations.

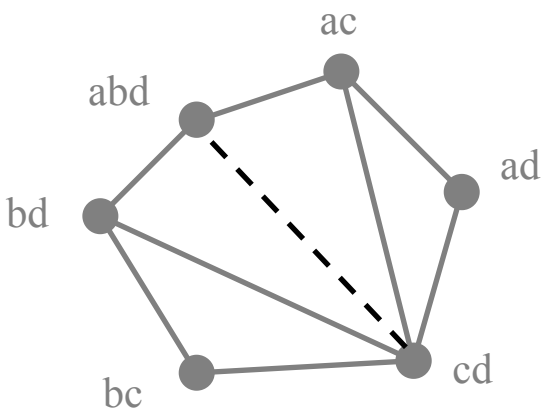


Figure 17 : triangulation du graphe dual

La construction du diagramme consiste à déterminer une région élémentaire autour de chaque sommet du graphe dual. Chaque contour C_i du diagramme est tracé sur le bord de l'union des régions élémentaires dont le label contient label(C_i).

Construction des régions

Nous avons choisi de dessiner chaque région comme un polygone centré sur son sommet. Ce polygone est défini par des points de construction. Ces points peuvent être classés en 4 types :

- type face interne,
- type arête,
- type sommet externe,
- type arête externe.

Pour chaque face interne triangulaire du graphe dual, on définit un point de construction type *face interne* placé au centre de gravité de la face. Pour chaque arête, on définit un point de construction type *arête* placé en son milieu. Sur la Figure 18, les points *face interne* sont en orange et les points *arêtes* sont en violet.

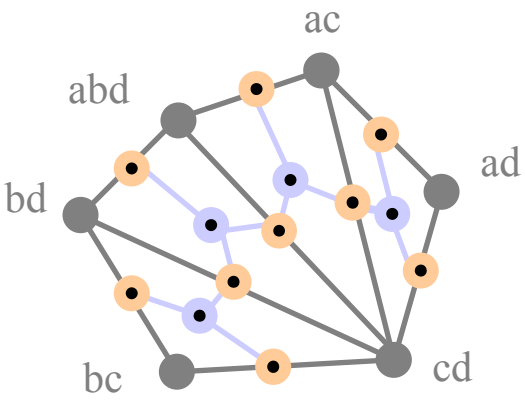


Figure 18 : points type face interne et arête

Pour construire les points type *sommet externe* on construit pour chaque arête externe, une droite qui lui est parallèle et à distance fixée. Deux droites consécutives s'intersectent en un point de type *sommet externe*. Les points type *arête externe* sont définis par le milieu de segment créé par deux points type *sommet externe* consécutifs. Sur la Figure 19, les points type *sommet exter*ne sont en rouge et les points *arête externe* sont en vert.

Ces points de construction définissent une région polygonale pour chacun des sommets du graphe dual.

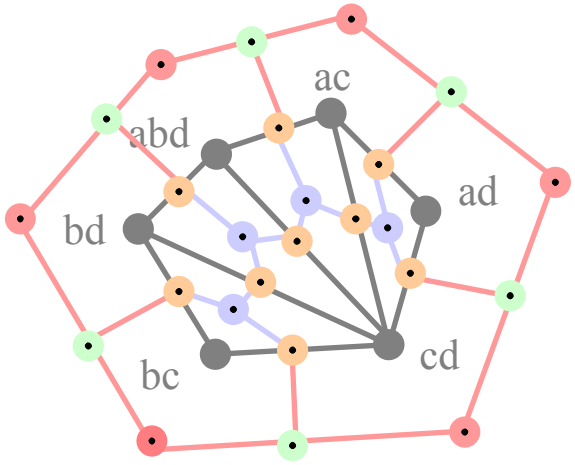


Figure 19 : points type sommet externe et arête externe

Construction des contours

Le contour étiqueté par *l* est le bord de l'union des régions dont le label contient *l*. Cependant, en procédant ainsi, deux contours peuvent se chevaucher partiellement, ce qui nuit à leur identification et dégrade la lisibilité du diagramme. Pour remédier à ce problème nous limitons le chevauchement des contours en les décalant de quelques pixels par rapport au tracé théorique. Nous assignons un décalage différent à chaque contour, suivant le principe illustré sur la Figure 20.

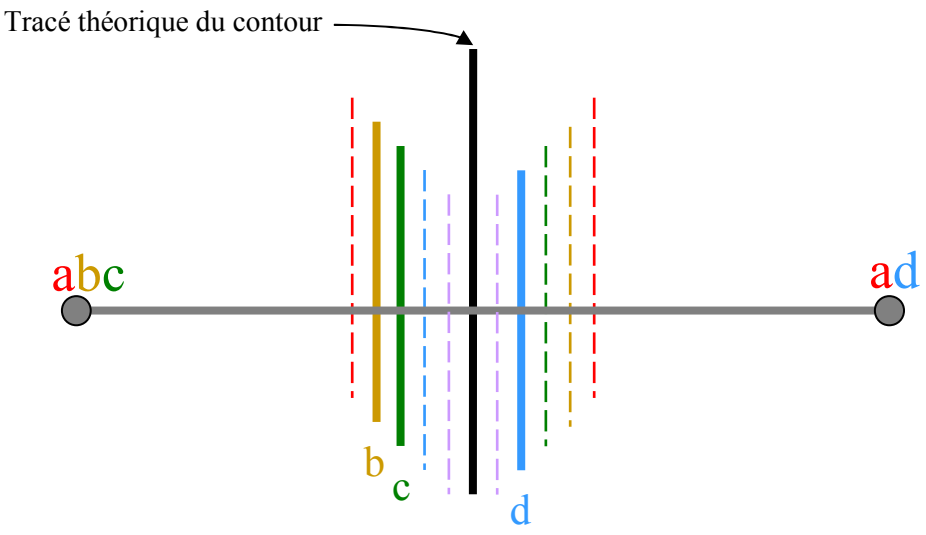


Figure 20 : stratégie de tracé des contours

Soient *u* et *v* deux sommets connectés par l'arête *(u,v)*. Cette arête possède un point de construction de type *arête* placé en son milieu. Les contours coupant cette arête doivent théoriquement passer par ce point. La stratégie utilisée pour éviter les chevauchements consiste à faire passer les contours de part et d'autre de ce point. Rappelons qu'un contour étiqueté par *l* coupe une arête si un seul des deux sommets extrémités contient *l* dans son label. Sur notre exemple ci-dessus les sommets *u* et *v* sont étiquetés respectivement par *abc* et *ad*. Cela implique que les contours *b*, *c* et *d* coupent l'arête *(u,v)*.

Nous appliquons un décalage différent à chaque contour. Sur la Figure 20 :

- le contour *a* est décalé de 5 pixels,
- le contour *b* de 4 pixels,
- le contour *c* de 3,
- le contour *d* de 2,
- et le contour *e* de 1 pixel.

Un contour coupant l'arête *(u,v)* peut être décalé selon deux directions, vers u ou vers *v*. Le contour étiqueté par *l* sera décalé vers le sommet dont l'étiquette contient *l*.

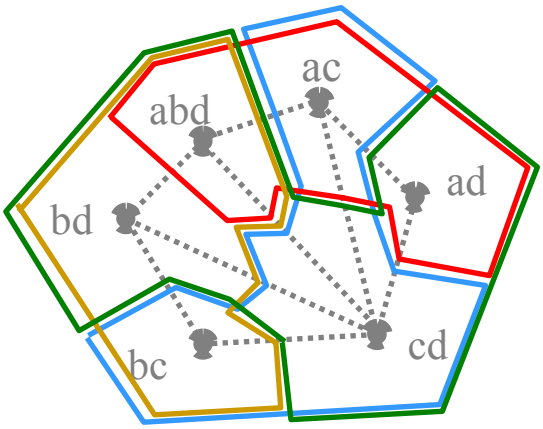


Figure 21 : dessin final du diagramme

En Figure 21, nous pouvons observer le résultat du dessin final des contours. Chaque contour est dessiné dans une couleur différente :

- le contour a est en rouge,
- le contour b est en ocre,
- le contour c est en bleu,
- le contour d est en vert.

Le prototype de construction de diagrammes d'Euler étendu est bâti au-dessus de notre API de visualisation de graphe Grapho, cf. chapitre 4. La Figure 22 est une impression d'écran du prototype, on peut y voir un graphe dual avec en surimpression le diagramme correspondant. Le prototype est interactif et permet de modifier le diagramme en manipulant les sommets du graphe dual.

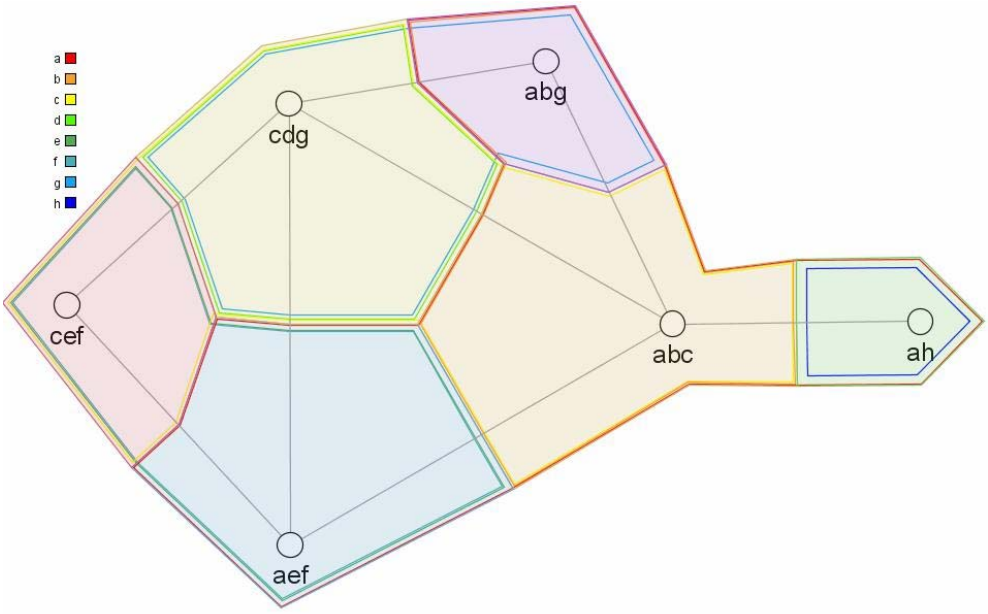


Figure 22 : 8-diagrammes d'Euler avec 6 régions non-vides

Le prototype est robuste et permet de créer des diagrammes complexes, comme cet exemple de 7-diagramme de Venn symétrique de type adelaide [Ruskey01] composé de 127 régions, cf. Figure 23.

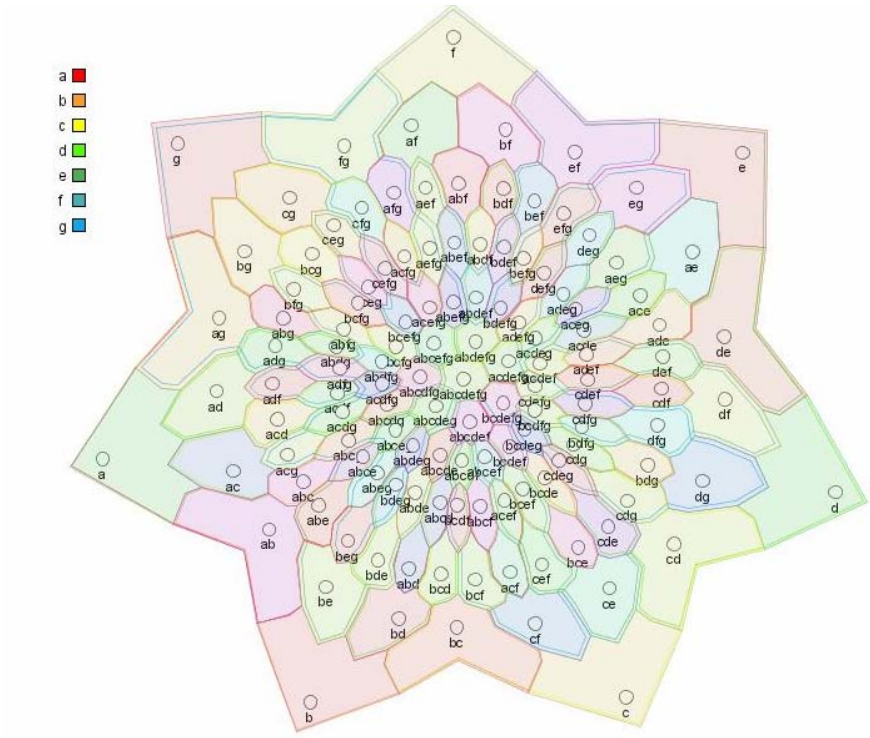


Figure 23 : 7-diagramme de Venn

1.4.4 Lisibilité des diagrammes de Venn-Euler

Un « bon » diagramme doit être permettre une acquisition rapide de l'information. Dans le cas des diagrammes de Venn-Euler, il est important de pouvoir identifier rapidement les ensembles et leurs

intersections. Cela implique que les courbes et les régions issues de leurs intersections soient facilement identifiables.

Une évaluation de la lisibilité de diagrammes d'Euler selon plusieurs critères a été réalisée [Benoy05]. Il en résulte plusieurs observations. Pour obtenir une bonne lisibilité des contours, il est nécessaire d'utiliser des courbes les plus continues possible afin de faciliter le suivi visuel du tracé. L'utilisation de formes simples facilite les processus d'identification et de mémorisation. Les cercles et ellipses sont les formes les plus facilement reconnaissables, malheureusement il n'est pas toujours possible de les utiliser. De manière générale il faut essayer de privilégier les formes les plus simples, les plus convexes et dont le polygone descripteur possède le plus petit nombre de côtés.

Dans un diagramme de Venn-Euler, plusieurs contours se trouvent juxtaposés. Le choix de contours congruents, similaires ou tout du moins de même nature peut faciliter la lecture et renforcer la cohérence du lien entre ensemble et contour. Afin de bien identifier les régions résultant de l'intersection de contours, les points d'intersection se doivent d'être assez éloignés. De même il est important à veiller que les régions n'aient pas une surface trop petite.

Beaucoup de travaux ont été entrepris pour construire des diagrammes de Venn les plus simples possibles. Citons en deux. Les travaux de Caroll sur la construction de diagrammes de Venn en utilisant des contours de même nature dont le nombre de côtés est fixé, et les travaux sur les diagrammes dits symétriques. Les travaux de Caroll constituent une bonne référence lorsque l'on souhaite utiliser les contours les plus simples pour construire un diagramme de Venn. Les diagrammes symétriques sont quant à eux intéressants à étudier car ils offrent généralement une bonne lisibilité.

1.4.4.1 Diagrammes de Venn et k-gones

Un k-gone est un polygone convexe à k côtés. Deux k-gones peuvent s'intersecter en au plus 2k points. La définition de k-gone peut être étendue ainsi, un k-gone est une courbe telle que deux k-gones s'intersectent en au plus 2k points. Avec cette définition, le cercle est un 1-gone et l'ellipse un 2-gone.

Quels n-diagrammes de Venn peut-on dessiner avec des k-gones ? Est-il, par exemple, possible de dessiner un 7-diagramme de Venn composés uniquement de triangles (ou 3-gones), c'est à ce type de questions que les travaux de Carroll [Carroll00, 05] répondent. Dans [Carroll00] la preuve de l'existence de 6-diagrammes de Venn composés de triangles est apportée par l'exemple, voir Figure 24.

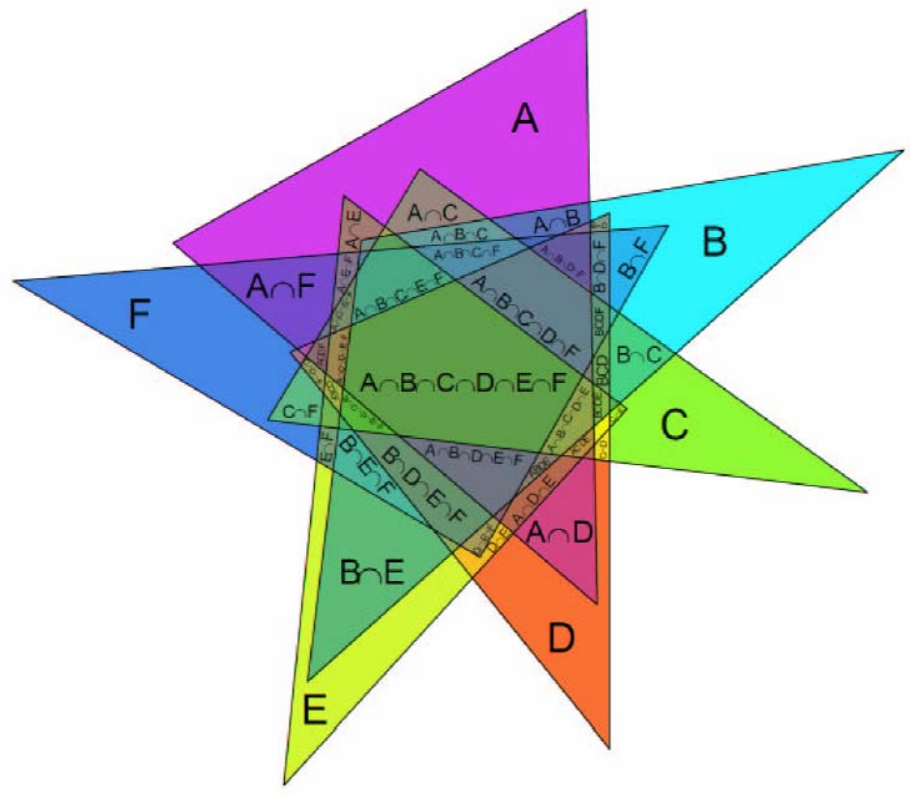


Figure 24 : un 6-diagramme de Venn composé de triangles

Dans l'article [Carroll05], les auteurs montrent que si $k \geq \left(\frac{2^n - 2 - n}{n(n-2)} \right)$ alors il est possible de construire un n-diagramme deVenn en utilisant des k-gones.

Le Tableau 1 montre les valeurs minimales de k pour n < 11.

N	1	2	3	4	5	6	7	8	9	10
k ≥	1	1	1	2	2	3	4	6	8	13

Tableau 1 : relation entre n-diagrammes de Venn et k-gones

Sur la Figure 25 un bel exemple de 7-diagramme de Venn composé de 4-gones congruents.

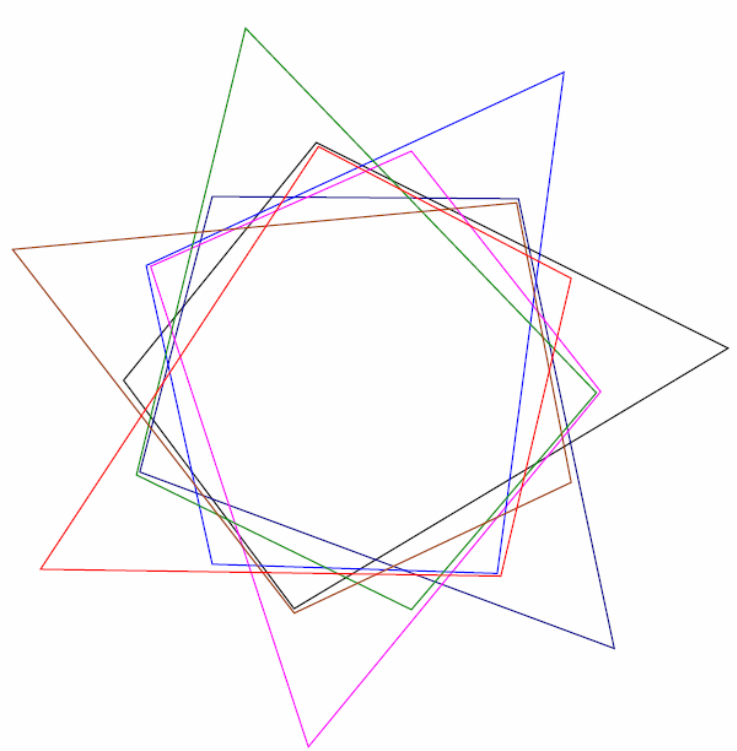


Figure 25 : un 7-diagramme de Venn composé de 4-gones.

Il est important de préciser que ces travaux fixent de nouvelles bornes théoriques à l'existence de diagramme de Venn dessinés avec des k-gones, mais ne fournissent aucune méthode de construction.

1.4.4.2 Diagrammes de Venn et Symétrie

Un diagramme de Venn est dit symétrique lorsqu'il existe un point autour duquel le diagramme peut subir des rotations d'angle $2i\pi/n$, avec i $\in \mathbb{N}$ en restant invariant. Grunbaum [Grunbaum99] a démontré que de tels diagrammes n'existent que lorsque le nombre d'ensembles est un nombre premier. Le diagramme aux trois cercles et les deux diagrammes des figures Figure 26 et Figure 27 sont les exemples les plus connus de diagrammes symétriques.

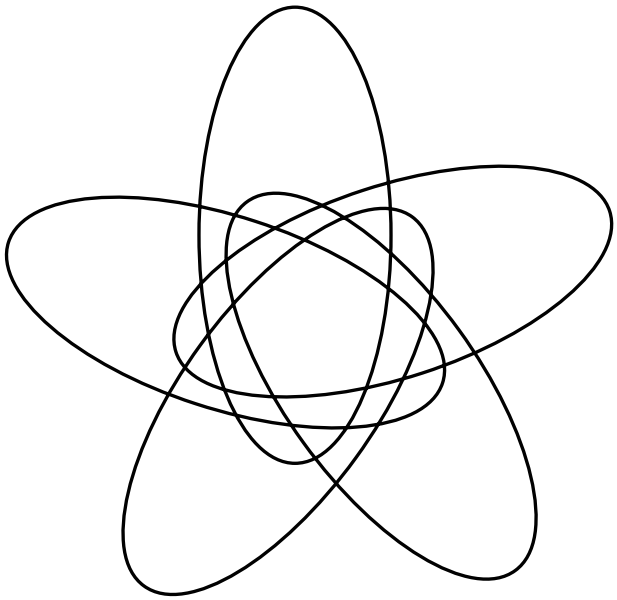


Figure 26 : 5-diagramme avec symétrie centrale

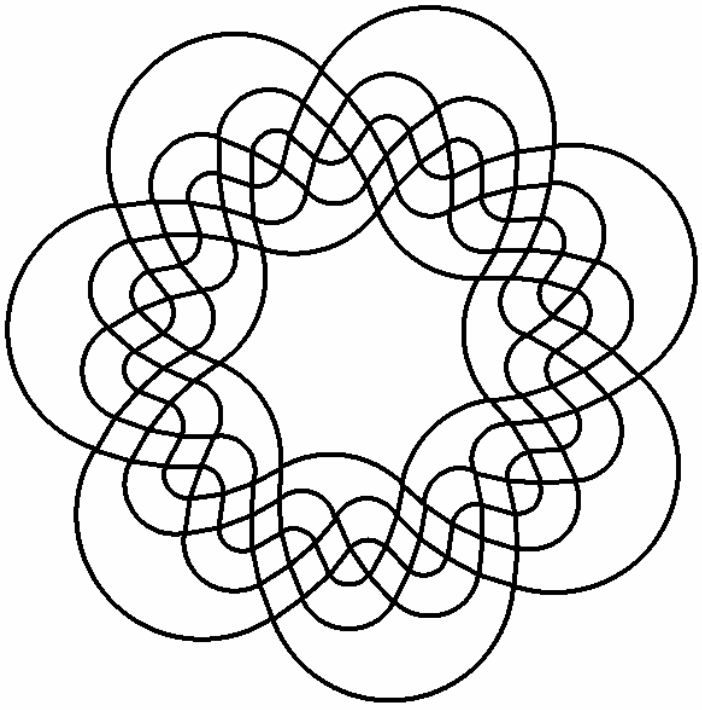


Figure 27 : 7-diagramme type adelaide

La Figure 28 montre un 4-diagramme de Venn est composé de quatre ellipses identiques et possède une symétrie axiale, cependant il n'est pas symétrique au sens défini ci-dessus

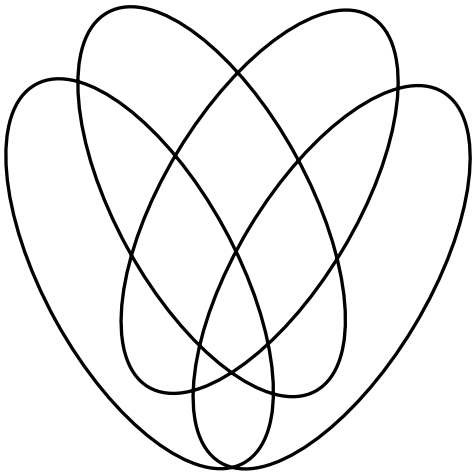


Figure 28 : 4-diagramme avec symétrie axiale

Ces exemples de diagrammes symétriques sont généralement plus faciles à comprendre que des diagrammes équivalents sans symétrie. Les contours d'un diagramme symétriques sont congruents, c'est-à-dire qu'ils sont superposables à une rotation près. Cette propriété participe à la cohérence du diagramme, et facilite la reconnaissance des contours.

1.5 Diagrammes et Recherche d'Information

1.5.1 Interfaces basées sur les Diagrammes de Venn

L'utilisation de diagrammes de Venn pour la formulation des requêtes booléennes a fait l'objet d'une évaluation en 1982 [Michard82]. Le protocole de l'évaluation est simple. L'objectif est de comparer les taux d'erreurs de la formulation textuelle et de la formulation par diagrammes de Venn. La formulation textuelle est complètement manuelle et sans assistance. La formulation graphique consiste à sélectionner des régions dans un diagramme de Venn.

Les utilisateurs doivent formuler des requêtes de même complexité selon les deux modalités, textuelle et graphique. Ces requêtes sont composées de deux ou trois termes ou ensembles élémentaires.

Lors de cette évaluation les utilisateurs ont commis quatre fois moins d'erreurs lorsqu'ils utilisaient l'interface à base de diagrammes de Venn. L'auteur conclut que la formulation par diagrammes de Venn est supérieure à la formulation textuelle, car elle rend impossible les erreurs structurelles liées au parenthésage des expressions booléennes, et permet d'éviter les problèmes liés à la mauvaise interprétation de la sémantique des opérateurs booléens. En effet, le sens des opérateurs booléens *ET* et *OU* est parfois confondu avec le sens des conjonctions de coordination *et* et *ou* du langage naturel. Ainsi l'opérateur disjonctif *OU* est fréquemment interprété comme exclusif. L'opérateur conjonctif ET est quant à lui parfois interprété comme une union plutôt que comme une intersection. Ces difficultés ont par ailleurs été maintes fois discutées [Boyles83, Anick90, Young93]

Plus récemment Spoerri [Spoerri99] a expérimenté une interface de formulation graphique de requêtes qui s'inspire et ressemble beaucoup aux diagrammes de Venn. Il a contourné la difficulté liée au dessin des diagrammes de Venn en adoptant une représentation simplifiée qu'il appelle InfoCrystal. Cette simplification consiste à transformer chaque région du diagramme en une entité graphique dont

la forme et la couleur indiquent le nombre de critères correspondant. La construction d'un InfoCrystal pour trois critères est illustrée dans le Figure 29.

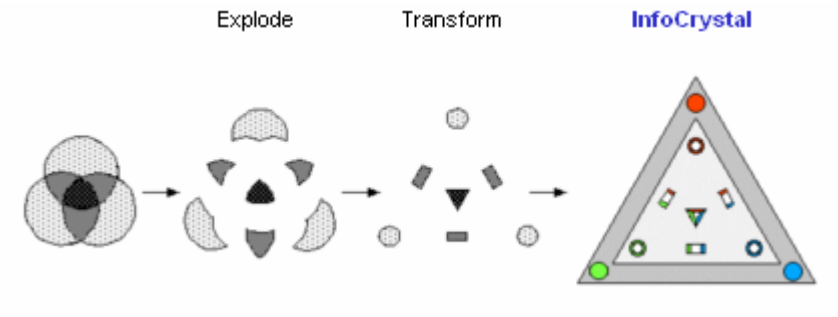


Figure 29 : du diagramme de Venn à InfoCrystal

L'InfoCrystal ressemble en fait beaucoup au graphe dual d'un diagramme de Venn. Les liens ne sont pas affichés, car la position, la forme et les couleurs de chaque entité suffisent à préciser leur fonction. L'algorithme de placement peut générer des InfoCrystals pour N critères. Les entités des N critères sont placées circulairement à intervalles réguliers. Chaque entité correspondant à l'intersection de k critères, est placée au centre de gravité des k entités. Cet algorithme est simple et parfaitement linéaire. Des entités peuvent toutefois se retrouver au centre de la figure alors que cette position est strictement réservée à l'entité correspondant à l'intersection de tous les critères. Dans ce cas l'entité qui pose problème est dédoublée et positionnée de manière appropriée, cf. Figure 30.

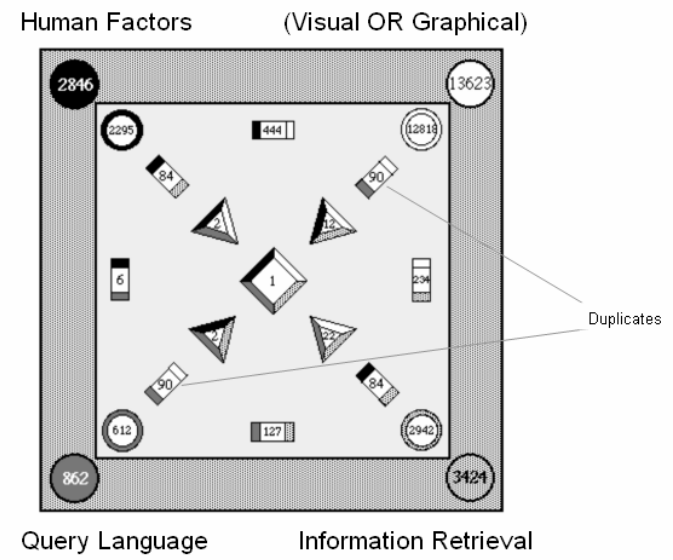


Figure 30 : certains éléments peuvent être dupliqués

Spoerri a récemment fait évoluer le design d'InfoCrystal. Renommé pour l'occasion MetaCrystal [Spoerri04], l'interface est utilisée pour visualiser les ensembles résultats de différents moteurs de recherche pour une même requête, cf. Figure 31.

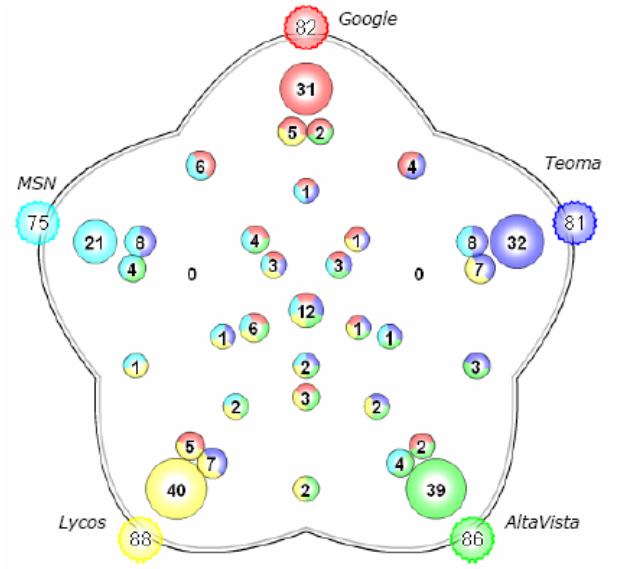


Figure 31 : MetaCrystal pour cinq ensembles résultats

Ces différents travaux démontrent le potentiel des diagrammes de Venn pour la formulation des requêtes complexes et la visualisation des résultats de recherche.

1.5.2 Applications pour l'Ina

Le mode de recherche classique associé aux bases de données documentaires est la formulation booléenne à partir des termes descripteurs. L'utilisateur dispose d'un formulaire comportant des champs correspondant aux classifications définies par la structuration documentaire (type de programme, droits, matériels…) et un champ sémantique. Il formule une requête en remplissant ces champs par un ou plusieurs termes descripteurs reliés par les opérateurs booléens classiques ET, OU et NON. Si l'expression d'une requête en langage naturel est souvent relativement simple, la formulation booléenne associée peut être complexe.

En effet, la requête booléenne est issue du système documentaire et de la stratégie élaborée pour la recherche. Prenons un exemple de requête réelle simple conduisant à une formulation documentaire non triviale : trouver une séquence de François Mitterrand avec un casque. Une requête du type (`Mitterrand ET casque`) a peu de chance de retourner des résultats, car le niveau de description en terme de contenu image n'est, en général, pas aussi précis. Aussi, la requête booléenne correspondante ressemblera-t-elle sans doute plus à l'expression suivante : `Mitterrand ET (visite ET (chantier) ET (NON (discours OU interview OU débat)))`. La stratégie consiste ici à enlever tous les documents dont la forme ne permet pas d'imaginer le port d'un casque et qui constituent la plus grande partie des documents sur Mitterrand, puis à supposer qu'une telle séquence a de fortes chances de se trouver lors d'une visite de chantier. Une étude des requêtes effectuées par les documentalistes de l'Ina montre qu'une formulation booléenne est très rarement composée de plus de 5 termes. Les documentalistes préfèrent utiliser un mode d'interrogation itératif. La requête précédente s'effectuera alors en deux temps : `(Mitterrand ET visite ET chantier)` puis l'ensemble R des résultats étant trop vaste, la requête `(R ET (NON (discours OU interview OU débat)))` permettra de réduire l'ensemble des résultats à visualiser. Cependant, même sur un nombre restreint d'éléments, l'ordre des termes et des parenthèses rend la formulation booléenne délicate à manipuler. L'utilisation de représentations ensemblistes interactives permet aux utilisateurs d'effectuer ces opérations de manière visuelle.

Notre premier prototype de visualisation ensembliste est composé de 3 zones, cf. Figure 32, une zone de formulation de la requête dans laquelle 5 champs de saisie sont proposés à l'utilisateur, une zone d'affichage du diagramme de Venn, et enfin une zone d'affichage des documents résultants. La validation des champs de saisie déclenche une séquence de requêtes dans la base de données associée afin de générer le diagramme de Venn correspondant. Le nombre de documents résultats apparaît dans chaque région et peut être associé à l'intensité de couleur de la région. L'utilisateur peut sélectionner une ou plusieurs régions du diagramme, soit pour visualiser la liste des documents qu'elles contiennent, soit pour générer le nouvel ensemble de départ d'une requête itérative. En effet, dans le cas où le nombre de documents sélectionnés est trop important pour être visualisé, l'utilisateur peut réduire l'ensemble sur lequel effectuer une nouvelle requête en utilisant sa sélection comme nouvelle base de recherche. Une formulation booléenne relative à l'ensemble des régions sélectionnées est proposée à l'utilisateur pour permettre éventuellement une vérification sémantique. Cette formulation correspond à la forme conjonctive la plus simple : elle minimise le nombre d'opérateurs. Pour amplifier la lisibilité du diagramme, une animation sous forme d'un clignotement continu de deux cycles est associée aux ensembles constituant la région sur laquelle est positionnée la souris. Ce type d'interaction utilisant une variable temporelle permet d'amplifier la lisibilité du diagramme sans le surcharger [Saulnier05]. Dans le cas d'une requête exploitant une recherche *full text,* on peut associer un score de pertinence aux documents d'une zone. Ce score compte le nombre d'occurrences du terme dans le texte. Un filtrage sur ce score permet d'ajuster le nombre de documents dans chaque région. Nous ferons remarquer ici que le score de pertinence peut être différent de la pertinence du document du point de vue de l'utilisateur.

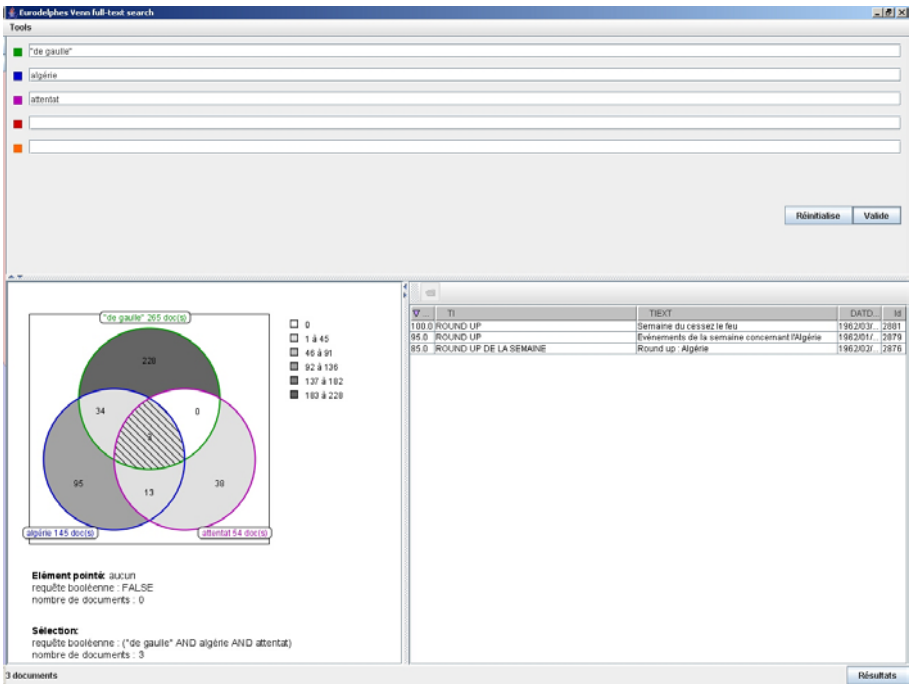


Figure 32 : Cartographie sous forme de diagramme de Venn à 3 ensembles.

Cette cartographie des résultats de requête possède une propriété intéressante : un document ne peut appartenir qu'à une seule zone du diagramme. Par conséquent lors d'une exploration des résultats dans l'ordre des zones les plus pertinentes, on ne consultera jamais deux fois un document. Par ailleurs, cette représentation permet à l'utilisateur de valider, instantanément et pour chaque champ, la pertinence des termes de la requête. Lorsqu'on utilise un langage normalisé comme le langage

documentaire, cette fonctionnalité est particulièrement intéressante puisqu'elle permet de détecter instantanément qu'un terme n'appartient pas au langage. L'affichage du nombre de documents dans chaque zone donne la représentativité de chaque terme. En cas de région cible vide, une stratégie d'exploration des zones proches peut être envisagée. Enfin, ce type de représentation propose à l'utilisateur une cartographie locale de la base consultée centrée sur son objet: elle lui permet de valider la pertinence de son contexte de recherche. Si l'utilisation de mots clés partitionne et réduit la description du document en supprimant les relations entre termes, elle permet aussi une recherche sur des associations plus vastes car non formulées au cours de l'indexation. Ce type de cartographie révèle les associations formalisées au niveau de la requête par rapport à l'ensemble de la base de documents.

1.5.3 Discussion

Notre prototype permet donc de formuler graphiquement des requêtes complexes à partir de cinq requêtes élémentaires. Cinq diagrammes de Venn sont utilisés en fonction du nombre de requêtes élémentaires saisies par l'utilisateur. Nous avons choisi d'utiliser des diagrammes symétriques car ils sont de loin les plus faciles à interpréter. Les diagrammes en question sont illustrés en Figure 33.

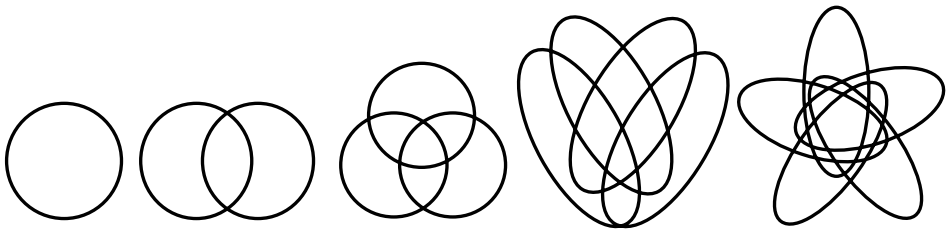


Figure 33 : les cinq diagrammes symétriques du prototype

Si nous n'avons pas évalué formellement la lisibilité de ces diagrammes, nous avons tout de même eu l'occasion de les présenter à plusieurs documentalistes qui nous ont donné leurs impressions. Tous connaissaient déjà les diagrammes constitués de un, deux et trois contours et comprenaient parfaitement leur signification. C'était par contre la première fois qu'ils observaient des diagrammes de Venn représentant plus de trois ensembles. Les deux derniers diagrammes leur ont semblé de fait plus difficiles à lire. Nous avons pu vérifier qu'ils étaient tous capable d'énoncer sans difficulté la sémantique d'une région dans les trois premiers diagrammes, alors qu'il n'en est pas de même pour les deux diagrammes auxquels ils n'étaient pas familiers. Avec le diagramme constitué de quatre contours nous avons observé que les documentalistes reconnaissaient bien la signification des régions, seulement quelques erreurs ont été commises, mais que ce travail nécessitait un temps de réflexion notablement plus important que pour les diagrammes familiers. Le dernier diagramme pose, lui, beaucoup plus de problèmes. Si les cinq régions les plus extérieures et la région centrale sont bien interprétées, celles intermédiaires sont sujettes à beaucoup d'erreurs d'interprétations et demandent un temps de réflexion très important.

Les problèmes de lisibilité des deux diagrammes les plus complexes ont, à nos yeux, plusieurs causes.

Ces diagrammes sont constitués respectivement de seize et trente-deux régions (en comptant la région extérieure), alors que les autres ne comprennent que deux, quatre et huit régions. Le nombre de régions est un facteur qui joue un rôle important dans la lecture d'un diagramme de Venn. Lire un diagramme, c'est dans un premier temps, être capable d'acquérir l'information qu'il représente. Et la

quantité d'information représentée dans un diagramme Venn est proportionnelle au nombre de ses régions. On peut donc dire qu'à lisibilité égale, le temps de lecture de différents diagrammes est une fonction croissante de leur nombre de régions.

Dans un second temps, l'information acquise doit être mémorisée pour un usage à court terme. Et là encore le nombre de régions est un facteur très lourd sur ce processus. Au cours de différentes expérimentations sur l'identification et la mémorisation immédiate de différents stimuli visuels et auditifs, Miller [Miller56] constate que le nombre d'éléments pouvant être stockés en mémoire immédiate est approximativement de sept. De nombreuses expérimentations ont confirmé ce résultat, et il est déroutant de constater que ce chiffre se trouve être la taille de nombreuses énumérations de la vie courante. Les sept jours de la semaine, les sept péchés capitaux, les sept couleurs de l'arc-en-ciel ou encore les sept merveilles du monde, … et les sept régions intérieures du 3-diagramme de Venn. Il paraît clair que cette limite est valable pour la mémorisation des différentes régions d'un diagramme de Venn. Les diagrammes de Venn pour quatre et cinq ensembles dépassent largement cette limite avec respectivement seize et trente-deux régions. La mémorisation immédiate de ces deux diagrammes est donc théoriquement impossible, ce qui peut expliquer des temps importants pris pour l'identification du sens d'une région. Au-delà de sept (ou un peu plus) régions, cette identification nécessite de réacquérir l'information représentée dans le diagramme car elle n'est jamais stockée entièrement en mémoire immédiate.

Nous pouvons aussi penser que le fait d'être familier avec un diagramme peut améliorer sa compréhension. Nous constatons en effet, que nous, qui sommes désormais habitués à manipuler ces diagrammes, avons plus de facilités à les lire maintenant. Avec le temps, nous avons appris à les utiliser. L'effet d'un apprentissage s'observe en particulier sur le diagramme à quatre ensembles. Cette hypothèse nécessiterait cependant d'être vérifiée.

Les régions des deux derniers diagrammes présentent des tailles très hétérogènes. Nous pensons que cette hétérogénéité nuit à leur lisibilité. La présence de régions dont la surface est très petite pénalise notamment le processus de suivi des contours. Cela a pour conséquence de favoriser l'apparition de points d'intersections et de segments de courbes parallèles trop rapprochés qui agissent comme de faux points d'intérêt venant interférer dans la lecture des contours. Ce problème de surface des régions a aussi des conséquences sur la cohérence du diagramme lorsque les tailles des ensembles doivent être représentées, nous développerons ce point un peu plus loin.

Le fait que le nombre de régions d'un diagramme augmente de manière exponentielle par rapport à son nombre de contours est un problème inhérent au diagramme de Venn. On ne peut pas envisager d'utiliser un diagramme de Venn pour représenter plus de quatre ou cinq ensembles. Cependant, dans notre application, nous observons que le nombre de régions représentant une intersection vide augmente lorsque le nombre d'ensembles résultats grandit. En effet, les descripteurs de type de programmes (reportage, documentaire, fiction, divertissement, …), de localisation (pays, régions, villes, …), ou encore temporels ne sont jamais utilisés simultanément pour indexer un même document. Par conséquent, une requête conjonctive (ET) de plusieurs termes différents mais du même type correspond à un ensemble résultat vide. En ne représentant pas ces intersections vides, nous pouvons diminuer significativement le nombre total de régions d'un diagramme. C'est la raison pour laquelle nous nous sommes aussi penchés sur la construction de diagrammes d'Euler.

En tant qu'utilisateurs experts du système documentaire et des techniques de recherche d'information, l'analogie entre requêtes booléennes et diagrammes de Venn est bien comprise par les documentalistes. Le logiciel de recherche utilisé aux archives présente justement un icône représentant

un diagramme de Venn pour symboliser les fonctionnalités d'opérations ensemblistes applicables à deux ensembles de résultats. Il leur a paru, d'ailleurs, intéressant d'intégrer, dans une prochaine version, des diagrammes de Venn pour étendre cette fonctionnalité à la manipulation de trois ensembles. Toutefois, ils sont très réservés sur les diagrammes pour quatre et cinq ensembles, qui présentent en plus de la complexité liée au nombre de régions, un défaut de cohérence causé par l'impossibilité de contrôler la surface de chaque région. La conséquence est qu'il est fréquent qu'une région très petite représente un ensemble très grand ou vice-versa. Leur constat est tout à fait justifié. Il est clair que la lisibilité et la visibilité d'une région dépendent beaucoup de sa surface. De plus lorsque l'on utilise des représentations surfaciques comme les diagrammes de Venn, l'aire d'un élément visuel est naturellement associée à la taille de la donnée qu'il représente, ici un nombre d'éléments. Plus prosaïquement, les régions de petites tailles sont aussi difficiles à étiqueter correctement.

2 Graphes : Théorie et Propriétés

2.1 Introduction

On peut définir simplement un graphe comme un ensemble d'entités, appelés sommets ou nœuds, et un ensemble de relations, appelées arêtes ou arcs, entre ces entités. Cette structure très générale permet de représenter un nombre important de données différentes en informatique (diagrammes de conception, graphes d'appels de méthodes, réseaux de communications Internet, ontologies et réseaux sémantiques), en géographie (réseaux de transports), en sociologie (réseaux sociaux) ou encore en électronique (circuits imprimés).

Notre intérêt pour ce type de structures provient d'un faisceau de besoins différents. En premier lieu, les notices de description associées à chaque document audiovisuel archivé par l'Ina constituent une source d'informations particulièrement intéressante. L'intérêt premier de ces notices est de permettre de retrouver efficacement les documents qu'elles décrivent. Mais elles établissent aussi des relations de proximité sémantique entre différents documents. Il est possible de considérer que les documents sont les sommets d'un graphe dont les arêtes correspondent aux termes descripteurs communs à deux documents. Ou de manière duale, de considérer que les descripteurs sont les sommets d'un graphe dont les arêtes correspondent aux documents qui sont décrits par deux mêmes descripteurs. Considérer ces données comme des graphes nous paraît offrir de nouvelles possibilités d'analyse des informations contenues dans le fond de l'Ina. C'est cette approche que nous avons choisie afin de construire des cartographies des journaux télévisés, cf.4.6.1. En second lieu, la mission de Dépôt Légal confiée à l'Ina va bientôt être étendue à l'Internet. L'Ina sera alors en charge d'archiver les sites relatifs à la culture et l'audiovisuel. L'Internet présente naturellement une structure de graphe. A granularité variable, il peut être vu comme un réseau de pages connectées les unes aux autres, comme un réseau de sites ou encore comme un réseau de communauté de sites. Troisièmement, le système documentaire utilise et produit différentes structures qu'il est intéressant de pouvoir visualiser. Le thésaurus est un lexique hiérarchisé des termes descripteurs qui est utilisé durant le processus de documentation. Il peut être très utile de proposer à des utilisateurs recherchistes d'accéder à cette structure afin de sélectionner les termes les plus pertinents pour former leurs requêtes. La thématisation est une classification thématique de documents. Elle est produite par les documentalistes afin de proposer un accès thématique hiérarchisé à la base documentaire. Ces deux structures sont arborescentes, et les arbres forment une catégorie spécifique de graphe. En dernier lieu, nos travaux sur la construction de diagrammes d'Euler, cf. chapitre 1, impliquent la création et la manipulation de graphes. Ces différents objectifs ont en point commun de nécessiter l'utilisation de graphes, et c'est pourquoi une grande partie de mes travaux de thèse ont porté sur cette structure.

Avant d'étudier les méthodes de visualisation de graphes il est important de connaître et de comprendre les caractéristiques qu'ils peuvent présenter. Ces caractéristiques peuvent être déterminées grâce aux outils mathématiques et statistiques. Nous présentons donc dans ce chapitre les principales métriques et modélisation statistiques utilisées afin d'analyser les propriétés des graphes.

2.2 Définitions et critères d'analyse

2.2.1 Définitions

2.2.1.1 Graphe

Définition 1 : Un **graphe** $G = (V, E)$ est défini par un ensemble V de **sommets** (ou **nœuds**) et un ensemble $E \subseteq V \times V$ d'**arêtes**.

Définition 2 : Un graphe est dit **non orienté** si toutes ses arêtes $e = \{u, v\}$ sont des paires non ordonnées de sommets. Les sommets u et v sont appelés extrémités de l'arête e.

Définition 3 : Un graphe est dit **orienté** si toutes ses arêtes $e = (u, v)$ sont des couples ordonnés de sommets. u est appelé sommet de départ et v sommet d'arrivée. Les arêtes d'un graphe orienté sont aussi appelées **arcs**.

Définition 4 : L'**ordre** d'un graphe est défini comme le cardinal de V, c'est-à-dire le nombre de sommets. On notera par la suite $N = |V|$ et $M = |E|$.

Définition 5 : Deux sommets liés par une arête sont dits **adjacents**. L'arête est dite **incidente** aux deux sommets.

Définition 6 : Le **degré** d'un sommet est défini comme le nombre d'arêtes incidentes à ce sommet. Pour un graphe orienté le **degré entrant** d'un sommet u est défini comme le nombre d'arcs ayant comme sommet d'arrivée u, le **degré sortant** est le nombre d'arcs ayant comme sommet de départ u.

Définition 7 : Le **voisinage** d'un sommet v est défini comme l'ensemble de ses sommets adjacents.

Définition 8 : Un **sous-graphe** de $G = (V, E)$ est un graphe $G' = (V', E')$ tel que $V' \subset V$ et E' est l'ensemble des arêtes induites par E sur V', $E' = \{(u, v) \in E \mid u, v \in V'\}$.

Définition 9 : Un **graphe partiel** de $G = (V, E)$ est un graphe $G' = (V, E')$ tel que $E' \subset E$.

Définition 10 : On appelle **boucle** une arête dont les sommets sont confondus.

Définition 11 : Deux arêtes $e = (u, v)$ et $e' = (u, v)$ sont dites **parallèles**.

Définition 12 : Un graphe est dit **simple** s'il ne comporte ni boucle, ni arêtes parallèles.

Définition 13 : Dans un graphe non orienté, une **chaîne** est constituée d'une suite de sommets adjacents. Elle est dite **simple** si elle ne contient pas deux fois la même arête, et **élémentaire** si elle ne contient pas deux fois le même sommet.

Définition 14 : Un graphe est dit **complet** lorsque toute paire de sommets est liée par une arête.

Définition 15 : Un sous-graphe complet d'ordre *n* est appelé une ***n*-clique** et est notée K_n .

Définition 16 : Dans un graphe orienté, on appelle **chemin** une chaîne orientée de sommets.

Définition 17 : Un **cycle** est une chaîne simple fermée d'un graphe non orienté, un **circuit** est un chemin simple fermé.

Définition 18 : La **distance géodésique** entre deux sommets est la longueur de la plus petite chaîne non orientée entre ces deux sommets lorsqu'elle existe, l'infini sinon. La distance entre *u* et *v* est notée $\delta(u,v)$. La distance géodésique est une vraie distance au sens mathématique, elle vérifié les propriétés de symétrie et d'inégalité triangulaire.

$$\delta(u,v) = \delta(v,u)$$
$$\delta(u,v) \leq \delta(u,w) + \delta(w,v)$$

Dans un graphe orienté, la longueur du plus petit chemin entre deux sommets est parfois appelée **distance orientée**, attention toutefois car elle ne vérifie ni la symétrie, ni l'inégalité triangulaire.

Définition 19 : Le **diamètre** d'un graphe est la distance maximale entre deux de ses sommets. Le **diamètre moyen** d'un graphe est la moyenne des distances entre deux sommets.

Définition 20 : Soit G un graphe, la relation « être lié par une chaîne » est une relation d'équivalence dont les classes sont les composantes connexes de G.

Définition 21 : G est dit connexe s'il ne contient qu'une seule composante connexe.

Définition 22 : On appelle **graphe orienté acyclique** ou **DAG** pour *directed acyclic graph* en anglais, un graphe orienté sans cycle. Un sommet sans arc entrant est appelé source, et un sommet sans arc sortant est appelé un puits.

Définition 23 : Un **graphe valué** est un graphe dont les entités peuvent avoir un **poids**, noté $\boldsymbol{\omega(v)}$ pour un sommet et $\boldsymbol{\omega(u,v)}$ pour une arête. Par convention si (u,v) n'est pas une arête $\omega(u,v) = 0$; dans un graphe non valué on considère que les arêtes ont toutes un poids égal à 1.

Définition 24 : Dans certains travaux, un graphe G est vu comme une **matrice d'adjacence**, cette matrice est notée A(G) et est définie par :

$$A(G) = \left[\omega(u,v)\right]$$

2.2.1.2 Arbre

Définition 25 : Un **arbre** est défini comme un graphe non orienté connexe acyclique.

Définition 26 : Une **forêt** est un graphe non orienté acyclique. Ses composantes connexes sont des arbres.

Définition 27 : Une **arborescence** ou **arbre enraciné** est définie par un arbre ayant un sommet particulier appelé **racine**.

Définition 28 : Dans une arborescence la **profondeur** d'un sommet est définie par sa distance à la racine.

Définition 29 : La **hauteur d'une arborescence** est définie par la longueur de la plus longue chaîne simple partant de la racine. La hauteur est aussi le rang maximal.

Définition 30 : La **hauteur d'un sommet** est définie par la hauteur de la sous arborescence ayant pour racine le sommet.

Définition 31 : Soit une arborescence de racine *r* et deux sommets *v* et *v'*. On dit que *v* a pour **descendant** *v'* si *v* appartient au plus court chemin de *r* à *v'*. Dans ce cas *v* est dit **ascendant** de *v'*. Si de plus, *v* et *v'* sont adjacents, *v'* est appelé **fils** de *v*, et *v* **père** de *v'*.

Définition 32 : Un **arbre couvrant** de G = (V, E) est un arbre construit à partir des arêtes de E qui relie tous les sommets de V.

Définition 33 : Un **arbre couvrant minimal** d'un graphe valué est un arbre couvrant qui minimise la somme des poids des arêtes.

2.2.1.3 Planarité

Un graphe est dit planaire lorsqu'on peut le dessiner sans croisement d'arêtes. Le théorème de Kuratowski [Kuratowski30] permet de définir précisément la notion de planarité.

Définition 34 : Un graphe est **planaire** si et seulement si, il ne contient pas de sous-graphes homéomorphes au graphe complet K_5 ou au graphe bipartite complet $K_{3,3}$.

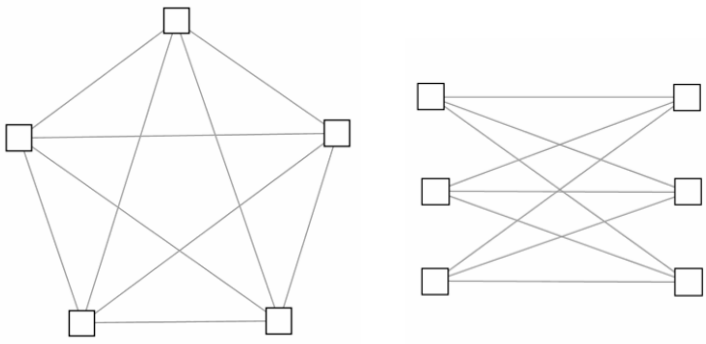


Figure 34 : K_5 et $K_{3,3}$

Les graphes K_5 et $K_{3,3}$ sont les plus petits graphes non planaires.

Si un graphe simple est planaire, il existe donc un placement de ce graphe sans croisement d'arêtes. On peut donc appliquer la formule d'Euler : $v - e + f = 2$, avec

- v le nombre de sommets,
- e le nombre d'arêtes,
- et f le nombre de faces, une face étant une région du plan délimitée par au moins trois arêtes.

Dans un graphe simple chaque face est délimitée par au moins 3 arêtes, et chaque arête appartient au plus à 2 faces, on en déduit que $v \geq 3 \Rightarrow e \leq 3v - 6$.

Cette inégalité est une condition nécessaire pour la planarité d'un graphe simple. Si cette condition n'est pas respectée le graphe ne peut pas être planaire. Si elle est vérifiée le graphe peut être planaire et il faut faire appel à d'autres techniques pour tester la planarité.

Cette formule est valide uniquement pour des graphes simples, dans le cas contraire il est nécessaire d'éliminer toutes les arêtes parallèles afin de pouvoir l'appliquer. On remarquera à ce propos qu'un graphe peut toujours être placé comme son graphe simple correspondant, il suffit de superposer ses arêtes parallèles. Cette inégalité montre aussi que les graphes planaires ont une très faible densité $\simeq \frac{6}{n}$.

Hopcroft et Tarjan [Hopcroft74] proposent un algorithme qui permet de déterminer si un graphe est planaire. Cet algorithme est linéaire et fournit lorsque le graphe est effectivement planaire un placement adéquat.

2.2.2 Critères d'analyse

De nombreuses mesures ont été définies dans le but d'étudier la structure des graphes. Les mesures d'importance permettent de quantifier le rôle des sommets et arêtes dans un graphe. Par exemple, le degré est une mesure d'importance locale d'un sommet. Les mesures de clustering concernent l'identification dans un graphe de groupes de sommets fortement corrélés.

2.2.2.1 Mesures

Il existe de nombreuses mesures d'importance des sommets dans un graphe. Nous ne considérons ici que les mesures d'importances liées à la structure d'un graphe. Les principales mesures sont basées sur la connectivité des sommets, pour le degré et les mesures de feedback, et sur les distances entre sommets, pour les mesures de centralités. Le Tableau 2 regroupe l'ensemble des définitions en fin de section.

2.2.2.1.1 Mesures locales

Les mesures dites locales sont basées sur la connectivité locale des sommets. Une des mesures d'importance pour un sommet est constituée par l'étendue de son voisinage Cette mesure de connectivité locale s'exprime simplement comme le degré du noeud pour un graphe non orienté, et les

degrés entrant et sortant pour un graphe orienté. Ces mesures ont l'avantage d'être simples et peu coûteuses à calculer. Cependant, le fait de considérer exclusivement le voisinage direct des sommets limite l'effectivité de ces mesures pour rendre compte du rôle d'un sommet par rapport à la connectivité globale du réseau.

2.2.2.1.2 Mesures de feedback

Les mesures de feedback sont, en quelque sorte, des extensions des mesures locales. Un sommet est considéré comme important lorsqu'il est lié à de nombreux sommets importants. Ces mesures sont basées sur une propagation et une accumulation des valeurs des degrés par voisinage. On peut parler de mesure d'accessibilité des sommets dans un réseau. Ces mesures sont principalement définies pour un graphe orienté. Elles s'appliquent particulièrement bien aux graphes de citations et aux graphes du Web.

La méthode générale d'accumulation des valeurs de feedback est souvent la même, la valeur de chaque sommet est initialisée à 1, puis on procède pour chaque sommet au calcul de sa nouvelle valeur par accumulation des valeurs des sommets voisins. On normalise ces valeurs sur [0,1] et on réitère l'étape précédente jusqu'à ce que les valeurs convergent. Lorsque les valeurs n'ont pas changé significativement entre deux étapes, le calcul s'arrête. Les différentes mesures de feedback sont définies par des choix de valeurs à cumulées différents.

Le ***status*** d'un sommet u a été défini [Katz53] comme le nombre de chemins qui arrivent sur u. C'est une accumulation des degrés entrants. Ainsi, un sommet est *important* s'il est pointé par des sommets *importants*. Il est à noter que la formule d'accumulation telle que définie ici ne donne pas une mesure normalisée car les valeurs ne convergent pas. Pour la normaliser, il est nécessaire de diviser toutes les valeurs par la valeur maximale à chaque étape.

La mesure ***eigenvector*** [Bonacich72] est comme son nom l'indique basée sur le calcul des valeurs propres de la matrice d'adjacence. La principale différence avec la mesure de ***status*** est la prise en compte d'un poids sur les arêtes, et une légère modification de la méthode d'accumulation. C'est donc une accumulation des degrés entrant pondérée par le poids des arêtes.

Le ***pagerank*** [Brin98] est une mesure conçue pour identifier les pages Web importantes. L'Internet est vu comme un graphe orienté dont les sommets sont les pages Web et les arcs les liens hypertextuels. Une page est considérée importante lorsqu'elle est pointée par des pages elles-mêmes importantes. Cette mesure correspond aussi à la probabilité d'atteindre une page particulière en navigant aléatoirement de pages en pages. Cette mesure est au cœur du moteur de recherche *Google*, elle est utilisée dans le calcul de pertinence des résultats de recherche.

Pour Kleinberg [Kleinberg99], les pages Web peuvent jouer deux rôles, ***hubs*** ou ***autorités***. Ces deux rôles sont duals et se renforcent mutuellement. Leurs définitions d'ailleurs sont étroitement liées. Plus une page pointe vers de bonnes ***autorités***, plus elle a une grande valeur de ***hub***. Plus une page est pointée par des bons ***hubs***, plus sa valeur d'autorité est grande. Pour Kleinberg le réseau Internet est constitué de communautés formées autour de fortes ***autorités***, et connectées entre elles par des ***hubs*** qu'on appelle plus communément des portails. Tout comme le ***pagerank***, la mesure ***d'autorité*** a été conçue pour mesurer l'accessibilité d'une page Internet.

2.2.2.1.3 Mesures de centralités

Ces mesures sont principalement issues des travaux sur les réseaux sociaux. Les réseaux sociaux sont généralement considérés comme étant des graphes non-orientés. Par exemple, la relation d'amitié entre deux personnes doit être exprimée mutuellement. Si un seul des deux acteurs considère que l'autre est son ami, alors on dit que le lien n'est pas confirmé ou partiel. Si on ne considère que les liens confirmés, le réseau est un graphe non-orienté. Dans un graphe non-orienté, la distance géodésique entre sommets est une distance au sens mathématique (**Définition 18**), les mesures de centralité sont basées sur cette notion de distance. Les notions de centralité permettent de quantifier si un acteur (un sommet) tient un rôle central dans un réseau. Plusieurs notions de centralité ont été définies.

Un sommet est central lorsque de nombreux plus courts chemins passent par lui, on parle de ***betweenness centrality*** [Anthonisse71, Freeman77, Brandes01] en anglais. Dans un réseau de communication, les sommets centraux, selon cette définition, sont des sommets où beaucoup d'information est susceptible de circuler, et constituent des goulets d'étranglement potentiels.

La ***closeness centrality*** [Beauchamp65, Sadibussi66] d'un sommet est l'inverse de la somme de ces distances aux autres sommets du graphe et l'***excentricité*** [Hage95] est l'inverse de sa distance maximale. La ***radialité*** [Valente98] est aussi une fonction décroissante de la distance moyenne.

Ces mesures sont définies pour les sommets et arêtes d'un graphe connexe. Pour des graphes non connexes, la distance entre deux sommets appartenant à deux composantes distinctes est infinie. On peut néanmoins calculer les centralités des sommets pour chacune des composantes. Dans ce cas les valeurs de centralités sont comparables uniquement entre sommets appartenant à la même composante.

2.2.2.1.4 Tableau récapitulatif

Le tableau ci-dessous donne les définitions des différentes mesures d'importance.

Mesures	Définition	Références
Mesures locales		
degré	$deg(v) = \sum_{e \in \text{edges}(v)} \omega(e)$	
degré entrant	$indeg(v) = \sum_{e \in \text{inEdges}(v)} \omega(e)$	
degré sortant	$outdeg(v) = \sum_{e \in \text{outEdges}(v)} \omega(e)$	
Mesures de feedback		
status	$c_s(v) = \alpha . \sum_{(u,v) \in \text{inEdges}(v)} \left(1 + c_s(u)\right),$ avec $\alpha = \frac{1}{\min\left(\max_{v \in V}(\text{indeg}(v)), \max_{v \in V}(\text{outdeg}(v))\right)}$	[Katz53]

vecteur propre	$c_e(v) = \frac{1}{\mu} \cdot \sum_{(u,v) \in \text{inEdges}(v)} \omega(u,v).c_e(u),$ avec μ la plus grande valeur propre de A(G)	[Bonacich72]
pagerank	$c_p(v) = \gamma \cdot \frac{1}{n} + (1-\gamma) \cdot \sum_{(u,v) \in \text{inEdges}(v)} c_e(u),$ avec $0 < \gamma < 1$	[Brin98]
authority	$c_a(v) = \frac{1}{\mu} \cdot \sum_{(u,v) \in \text{inEdges}(v)} \omega(u,v) \cdot \sum_{(u,w) \in out\,\text{Edges}(u)} \omega(u,w) \cdot c_a(w)$ avec μ la plus grande valeur propre de $A(\text{G})^T \cdot A(G)$	[Kleinberg99]
hub	$c_h(v) = \frac{1}{\mu} \cdot \sum_{(v,w) \in \text{outEdges}(v)} \omega(v,w) \cdot \sum_{(u,w) \in \text{inEdges}(u)} \omega(u,w) \cdot c_h(u)$ avec μ la plus grande valeur propre de $A(\text{G}) \cdot A(G)^T$	[Kleinberg99]
Mesures basées sur la distance		
betweenness centrality	$c_b(v) = \sum_{u \neq v \neq w \in V} \frac{\sigma(u,w\|v)}{\sigma(u,w)},$ avec $\sigma(u,w)$ et $\sigma(u,w\|v)$ resp. le nombre de plus courts chemins entre u et w, et entre u et w passant par v.	[Anthonisse71], [Freeman77], [Brandes01]
closeness centrality	$c_c(v) = \frac{1}{\sum_{u \in V} \delta(u,v)}$	[Beauchamp65], [Sadibussi66]
excentricité	$c_e(v) = \frac{1}{\max_{u \in V}(\delta(u,v))}$	[Hage95]
radialité	$c_r(v) = \frac{\sum_{u \in V}(\text{diam}(G) + 1 - \delta(u,v))}{(n-1) \cdot \text{diam}(G)}$	[Valente98]

Tableau 2 : les principales mesures d'importance

2.2.2.2 Mesures de Clustering

Le clustering est le processus d'identification de sous-graphe fortement connexe au sein d'un graphe. Ces sous-graphes sont appelés clusters ou agrégats. Les mesures suivantes sont définies pour des graphes simples. De nombreux graphes ne vérifient pas cette propriété car ils possèdent des boucles et/ou des arêtes parallèles, dans ce cas il est préférable de filtrer ces arêtes pour revenir à un graphe simple.

Définition 35 : Les mesures de **ratio** et de **densité** permettent d'évaluer la connectivité d'un cluster $C_i = (V_i, E_i)$ d'un graphe $G = (V, E)$.

$$ratio(C_i) = \frac{|E_i|}{|V_i|} \text{ et } densité(C_i) = \frac{|E_i|}{|V_i|^2}$$

La **densité** mesure, pour un graphe orienté simple, le rapport du nombre d'arêtes sur le nombre d'arêtes potentielles dans le graphe. Pour un graphe non orienté simple la mesure est différente :

$$densité_{norm}(C_i) = 2\frac{|E_i|}{|V_i|\cdot(|V_i|-1)}$$

La **densité** est nulle pour un graphe totalement déconnecté et égale à 1 pour un graphe complet.
On préfère souvent utiliser le **degré moyen** d'un graphe à la place du ratio, le degré moyen est égal à deux fois le ratio.

Définition 36 : L'ensemble des arêtes qui connectent deux clusters $C_1 = (V_1, E_1)$ et $C_2 = (V_2, E_2)$ du graphe $G = (V, E)$ est définie par :

$$\text{interEdges}(C_1, C_2) = \{(v_1, v_2) \in E \,|\, v_1 \in V_1, v_2 \in V_2\}$$

Le nombre d'arêtes inter-cluster entre C_1 et C_2 est aussi appelé **cut**(C_1,C_2).

$$cut(C_1, C_2) = |\{(v_1, v_2) \in E \,|\, v_1 \in V_1, v_2 \in V_2\}|$$

Définition 37 : Le **couplage** entre deux clusters est leur densité inter-cluster, c'est-à-dire le rapport du nombre d'arêtes qui les relient sur le nombre d'arêtes potentielles. Cette mesure de couplage est aussi appelée **node-normalized cut**.

$$couplage(C_1, C_2) = \frac{cut(C_1, C_2)}{|V_1|\cdot|V_2|}$$

Dans un graphe simple, le couplage de deux clusters est nul si les deux clusters ne sont pas connectés, et vaut 1 s'ils sont complètement connectés.

Définition 38 : La **distance intra-cluster** peut s'exprimer comme le diamètre ou le diamètre moyen du cluster.

$$\delta(C_i) = \text{diam}(C_i) \text{ et } \delta_{avg}(C_i) = \text{diam}_{avg}(C_i)$$

Définition 39 : La **distance inter-cluster** est la plus petite distance entre paires sommets de deux clusters, ou la distance moyenne entre paires de sommets de deux clusters.

$$\delta(C_1, C_2) = \min_{v_1 \in V_1, v_2 \in V_2} \left(\delta(v_1, v_2)\right) \text{ et } \delta_{avg}(C_1, C_2) = \frac{1}{|V_1| \cdot |V_2|} \cdot \sum_{v_1 \in V_1, v_2 \in V_2} \delta(v_1, v_2)$$

De nombreux indices de validité d'un clustering sont basés sur la comparaison entre distances intra et inter clusters [Boutin04] et entre les densités des clusters et leurs couplage [Noack03].

Définition 40 : L'**indice de clustering** [Watts99] d'un sommet mesure la cohésion de ses sommets voisins, c'est la densité du sous-graphe induit par le sommet et son voisinage direct.

$c(v) = densité\left(V(v)\right)$, avec $V(v)$ le sous-graphe induit par v et son voisinage.

2.2.2.3 Mesures globales

Ces mesures sont utilisées pour caractériser la topologie globale d'un graphe. Les fonctions de mesures globales sont les fonctions statistiques classiques, valeurs minimale, maximale, moyenne et médiane. Les fonctions de distributions comme l'écart absolu à la moyenne, et l'écart type permettent quant à elles d'étudier la répartition des mesures locales.

Par exemple la distribution des degrés permet de reconnaître des types de graphes différents, comme les graphes petits mondes, ou encore les graphes sans échelle, cf. 2.2.3.1. La distribution des mesures de centralités permet quant à elle de déterminer la centralisation d'un graphe. Un graphe est dit centralisé lorsque la distribution de la centralité suit une loi normale de faible variance. Dans ce cas la courbe des distributions prend l'allure d'une gaussienne très resserrée. Cela correspond à un graphe dont les sommets sont proches du centre.

Le diamètre mesure l'étendue d'un graphe, et la densité exprime sa connectivité. Ces deux mesures sont souvent liées. Lorsque l'on ajoute des arêtes à un graphe, sa densité augmente et son diamètre ne peut que diminuer.

Enfin l'indice de clustering d'un graphe est la moyenne des indices de clustering des sommets, il permet de mesurer les chances de trouver des clusters dans un graphe.

Ces mesures sont reprises dans le paragraphe suivant, notamment dans l'étude des graphes généraux, cf. 2.2.3.1.

2.2.3 Analyse des Graphes

L'analyse d'un graphe consiste à déceler ses propriétés particulières. Certains types de graphes sont bien identifiés comme les arbres, les treillis, les graphes orientés acycliques et possèdent des propriétés structurelles qui obéissent à des règles strictes et facilement identifiables. Ces propriétés sont acquises le plus souvent par construction et sont utilisées dans de nombreux domaines.

La planarité est une propriété du plongement du graphe dans le plan : un graphe est planaire lorsqu'il peut être dessiné sans croisement d'arêtes. C'est une propriété particulièrement intéressante à exploiter en visualisation, mais elle correspond à des contraintes extrêmement fortes sur la structure du graphe et sur son plongement dans le plan. Peu de graphes réels sont planaires.

La grande majorité des graphes que l'on est amené à considérer à travers les différentes activités humaines, comme les réseaux sociaux, les réseaux de communications, les graphes du Web, ne possèdent aucune des propriétés précédentes. Ces graphes sont généralement complexes et relativement denses, et il est difficile d'analyser leur structure. C'est pourquoi de nombreuses études statistiques ont été réalisées sur ces graphes complexes, afin de déterminer leur nature. Ces travaux ont conduit à l'émergence de plusieurs modèles, les graphes aléatoires, petits mondes, sans échelle et arborés.

2.2.3.1 Graphes généraux

2.2.3.1.1 Graphes aléatoires

Les travaux d'Albert [Albert02] sont une grande source d'information sur l'étude statistique des graphes aléatoires. Les travaux sur la théorie des graphes aléatoires ont été initiés par Erdõs et Rényi [Erdos59]. L'idée est de construire des graphes de manière contrôlée afin d'étudier l'émergence de propriétés particulières. Un graphe aléatoire est un graphe dont les arêtes sont distribuées aléatoirement. Il existe deux méthodes pour construire un graphe aléatoire :
Soit on considère que chaque paire de sommets a une même probabilité de connexion p d'être connectée par une arête. Si p est égal à 0, il n'y a aucune arête et le graphe est donc totalement déconnecté, si p est égal à 1 alors il y a $\frac{N(N-1)}{2}$ arêtes et le graphe est complet. L'algorithme de construction consiste, pour chaque paire de sommets, à effectuer un tirage aléatoire d'un nombre sur [0,1[. Si ce nombre est strictement inférieur à p, alors on connecte les deux sommets. La complexité de cet algorithme est $\theta(N^2)$. Dans ce cas, le nombre d'arêtes est variable, et son espérance mathématique est $E(M) = p \cdot \frac{N(N-1)}{2}$.
L'alternative est de considérer que le nombre d'arêtes est fixé. Pour chacune des arêtes, on sélectionne de manière aléatoire ses deux sommets. Dans ce cas $M = p \cdot \frac{N(N-1)}{2}$, et la complexité de cet algorithme est $\theta(M)$.

La probabilité de connexion p que deux sommets quelconques soient liés par une arête dans un graphe aléatoire est égale à sa densité.

La structure d'un graphe aléatoire $G_{N,p}$ dépend de deux paramètres, son nombre de sommets N, et sa densité ou probabilité de connexion que l'on continuera à noter p dans ce paragraphe. Lorsque l'on augmente progressivement la probabilité de connexion d'un graphe contenant un nombre de sommets fixé, on observe l'apparition de différentes propriétés caractérisant l'émergence de certains types de sous-graphes. On peut alors se demander à partir de quelle densité un graphe peut contenir des arbres ou des cliques ou encore quand devient-il connexe. Si on exprime la probabilité de connexion d'un graphe d'ordre N par $p(N) = cN^z, z \in]-\infty, 0]$, on peut calculer les probabilités à partir desquelles un graphe aléatoire contient des sous-graphes particuliers [Bollobas85]:

- les arbres d'ordre k apparaissent à partir de $p(N) = cN^{-k/(k-1)}$,
- les cycles d'ordre k à partir de $p(N) = cN^{-1}$,
- les cliques d'ordre k à partir de $p(N) = cN^{-2/(k-1)}$.

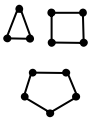
z	$-\infty$	-2	$-3/2$	$-4/3$	-1	$-2/3$	$-1/2$
							

Tableau 3 : apparition de type de sous-graphes en fonction de z

Les graphes aléatoires ont en commun d'avoir une distribution des degrés qui suit une loi normale (cf. Figure 35), un faible diamètre, et un faible indice de clustering. De plus le degré moyen est proche de l'espérance mathématique des degrés, soit $\langle \text{degre} \rangle \simeq pN$. On peut démontrer que le diamètre d'un graphe aléatoire peut s'écrire $diam = c \cdot \frac{\log(N)}{\log(\langle \text{degree} \rangle)} = c \cdot \frac{\log(N)}{\log(pN)}$, on en déduit les propriétés suivantes :

- pour $\langle \text{degre} \rangle < 1$, le graphe est composé d'arbres isolés et son diamètre est celui d'un arbre,
- pour $\langle \text{degre} \rangle > 1$, le graphe contient un cluster géant,
- pour $\langle \text{degre} \rangle \geq \log(N)$, le graphe est totalement connecté et son diamètre est $\frac{\log(N)}{\log(\langle \text{degre} \rangle)}$.

Enfin, par construction les graphes aléatoires ont un faible indice de clustering car la densité est uniforme, la probabilité que deux voisins d'un sommet v soient connectés est la même que pour toute autre paire de sommets. Plus précisément, dans un graphe aléatoire l'espérance mathématique de l'indice de clustering est égale à la densité du graphe.

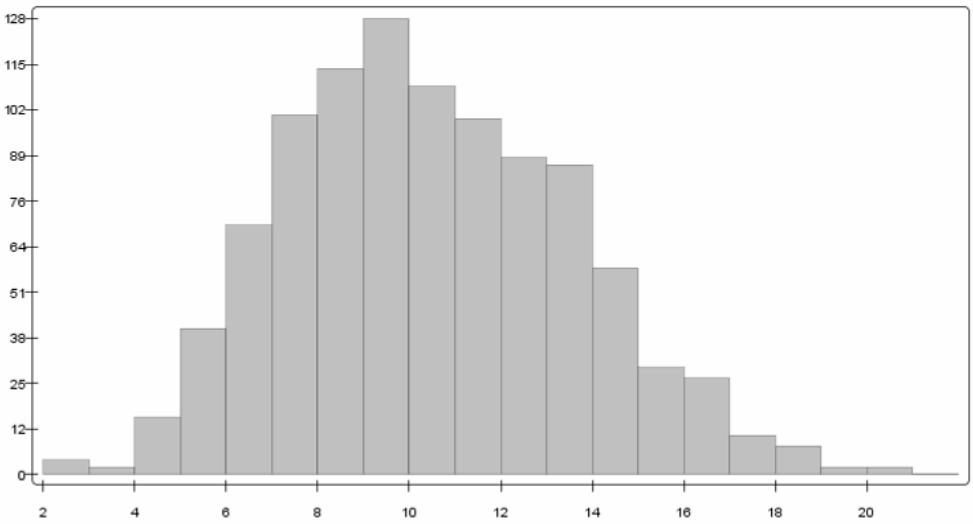


Figure 35 : distribution normale des degrés d'un graphe aléatoire

Les propriétés des graphes aléatoires ont été étudiées avec attention. Et même si les graphes construits à partir de données réelles ne se comportent pas comme des graphes aléatoires [Barabasi99], leur distribution des degrés suivant plutôt une loi de puissance, ils sont intéressants à utiliser comme objet de comparaison. On peut tirer un certain nombre de propriétés de la comparaison d'un graphe avec le graphe aléatoire de même ordre et densité $G_{N,p}$, avec $p = densité(G)$. Pour évaluer la possibilité de trouver des clusters dans un graphe on, peut par exemple comparer son indice de clustering avec celui du graphe aléatoire de même ordre et densité.

De nombreuses études ont été réalisées sur l'analyse des graphes réels durant cette décennie. Il en résulte que sur des données de types différents et de tailles différentes telles que les graphes du Web, les réseaux Internet, les réseaux sociaux, les réseaux biologiques ou bien encore des graphes linguistiques, on observe des propriétés similaires. Ces graphes ont une distribution des degrés qui suit une loi de puissance, un diamètre proche du diamètre d'un graphe aléatoire comparable, et un fort indice de clustering. Ces résultats ont amenés les chercheurs à imaginer de nouveaux modèles pour représenter les graphes réels.

2.2.3.1.2 Graphes petits mondes

La propriété de petit monde est connue depuis longtemps dans les réseaux sociaux. Milgram [Milgram67] a mené une étude sur les réseaux de connaissance. Un réseau de connaissance est un graphe de personnes, deux sommets sont connectés si les deux personnes sont capables de se reconnaître mutuellement. Cette étude conclut que malgré la taille importante du réseau de connaissance de la population américaine, la distance entre deux individus est au maximum 6. La première caractéristique d'un graphe petit monde est d'avoir un diamètre faible par rapport au nombre de sommets.

Dans un réseau social, on constate souvent que les voisins d'un sommet ont de grandes chances d'être voisins entre eux. L'adage « les amis de mes amis sont mes amis » prend corps, un groupe d'amis formant une clique où toutes les paires de sommets sont liées. Cette caractéristique se traduit par un fort coefficient de clustering.

Définition 41 : Un graphe est dit **petit monde** si son diamètre est faible et si son indice de clustering est élevé.

Nous avons vu que le diamètre d'un graphe aléatoire est faible, il s'exprime en $\log(N)$. Généralement on dit qu'un graphe a un fort indice de clustering si ce dernier est grand devant l'indice de clustering d'un graphe aléatoire de même ordre et densité.

De nombreux travaux ont été effectués sur cet effet petit monde, notamment sur la modélisation de graphes semi-aléatoires ayant cette propriété. Le modèle le plus connu [Watts98] permet d'obtenir un graphe dont on a fixé le nombre de sommets et d'arêtes, ainsi que le coefficient de clustering. Les graphes sont générés par interpolation entre un graphe aléatoire et un treillis régulier, c'est-à-dire une grille multidimensionnelle.

2.2.3.1.3 Graphes sans échelle

La majorité des graphes issus des activités humaines ont une distribution des degrés qui suit une loi de puissance, la majorité des sommets ont un degré très faible, et quelques sommets ont un fort

degré. Dans ce cas la distribution n'est pas centrée sur la moyenne, c'est pourquoi on dit de ces graphes qu'ils sont sans échelle.

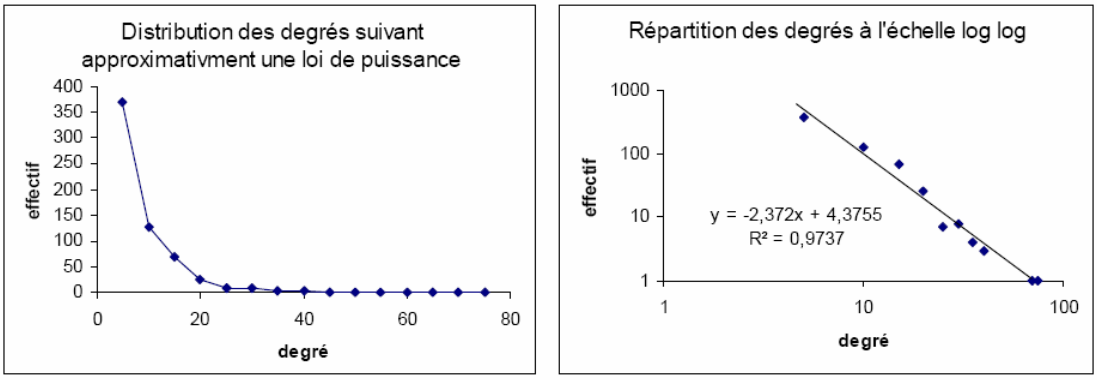


Figure 36 : distribution des degrés des graphes sans échelle

Cette propriété semble être intrinsèquement liée à la manière dont évoluent ces graphes réels dans le temps. Par exemple, le graphe du Web est une entité qui grossit perpétuellement. A tout moment de nouveaux sites se créent, et pointent vers d'autres sites. Les sites ayant une grande notoriété (authority) sont bien sûr plus souvent pointés que les autres, et ont donc beaucoup de chance de voir leur notoriété et leur degré augmenter. Ce processus d'évolution est observé sur de nombreux types de graphes sociaux.

2.2.3.1.4 Graphes arborés

Un graphe est dit **arboré** [Boutin06] lorsque sa structure est proche de celle d'un arbre. Peu de graphes réels présentent cette propriété. Les arbres ont une faible densité et leur diamètre est grand, cela les rend facilement visualisables. Les graphes réels ne présentent que très rarement une structure arborée, mais il existe une technique de filtrage qui permet de transformer un graphe en graphe arboré par interpolation entre le graphe et son arbre couvrant.

L'algorithme consiste à construire un arbre couvrant du graphe, ou une forêt d'arbres couvrants si le graphe n'est pas connexe. Le choix de l'arbre couvrant peut être guidé par un attribut numérique des arêtes ou des sommets. Une première technique est basée sur la maximisation de la centralité des arêtes [Kim04]. L'algorithme de Kruskal [Kruskal56] permet de trouver cet arbre couvrant avec une complexité en $\Theta(M \cdot \log(N))$. La seconde technique [Boutin06] consiste à sélectionner de préférences les arêtes des sommets de forts degrés ou de fortes centralités, cette technique est de complexité linéaire. Ces deux techniques donnent des arbres couvrants différents qui partagent cependant les mêmes caractéristiques. Si le graphe d'origine est sans échelle les arbres couvrants sont aussi sans échelle.

Une amélioration importante apportée par Boutin est de pouvoir réinsérer progressivement les arêtes filtrées. On peut en effet calculer la longueur λ de chaque arête comme la distance entre ses deux sommets dans l'arbre couvrant. On peut alors réinsérer progressivement les arêtes selon leur λ croissant.

Définition 42 : Soit G un graphe connexe et T un arbre couvrant de G. On appelle G_T^λ le graphe partiel de G dont on a supprimé les arêtes de longueurs supérieures à λ. Lorsque λ est faible le graphe est dit **arboré**.

Cette technique permet d'obtenir un graphe arboré, qui peut être vu comme une interpolation entre l'arbre couvrant et le graphe d'origine. Ce filtrage a une propriété intéressante, si le graphe d'origine est petit-monde, le graphe arboré $G_T^\lambda, \lambda \geq 2$ est aussi petit-monde. De manière générale, cette technique de filtrage permet de supprimer un grand nombre d'arêtes tout en conservant les clusters. L'indice de clustering des sommets diminue de manière homogène dans tout le graphe. La distribution des indices de clustering est globalement conservée.

3 Visualisation de Graphes

3.1 Généralités

La visualisation de graphes consiste à appliquer les techniques de visualisation d'information aux graphes dans le but d'analyser leurs structures et propriétés. Dans la pratique, les graphes que l'on manipule ne sont pas des objets théoriques, ils sont construits à partir de données existantes. Prenons, par exemple, le graphe de collaboration des auteurs du LIRMM. Chaque auteur est représenté par un sommet du graphe, et lorsque deux auteurs ont écrit un article ensemble, une arête relie leurs sommets respectifs. Un auteur est identifié par son nom et son prénom, et est rattaché à un des départements du laboratoire. Un article est défini par son titre, son lieu et sa date de parution et est associé à l'ensemble des arêtes qui connectent ses auteurs deux à deux.

Visualiser un graphe c'est d'abord choisir une représentation. Deux types de représentation peuvent être adoptés. Le diagramme *nœud-lien* qui consiste à représenter une arête comme un lien. Dans ce type de représentation chaque sommet est symbolisé par une entité graphique et une arête est représentée par une courbe qui connecte deux entités graphiques. La représentation matricielle consiste quant à elle à représenter un graphe d'ordre N par une matrice $N \times N$. Chaque sommet est symbolisé par une ligne et une colonne de la matrice. Chaque cellule représente une arête potentielle du graphe. La cellule de la matrice qui est au croisement de la ligne du sommet *u* et de la colonne du sommet *v* symbolise la ou les arêtes partant de *u* vers *v*, dans le cas d'un graphe orienté, ou la ou les arêtes entre u et v dans le cas non orienté. S'il y a plusieurs arêtes entre deux sommets, la cellule de la matrice pourra contenir le nombre d'arêtes.

La représentation matricielle a été un des objets d'étude du cartographe Bertin [Bertin67] qui a étudié les moyens de manipuler ces matrices afin de les rendre plus lisibles. Les manipulations en question consistent généralement en un réordonnancement des lignes et colonnes. Cette représentation est souvent délaissée au profit des diagrammes *nœud-lien*, qui offrent généralement une meilleure lisibilité. Il apparaît cependant que pour des graphes très denses et pour des tâches spécifiques, comme l'analyse des liens, la représentation matricielle puisse être avantageuse [VanHam03] [Ghoniem04].

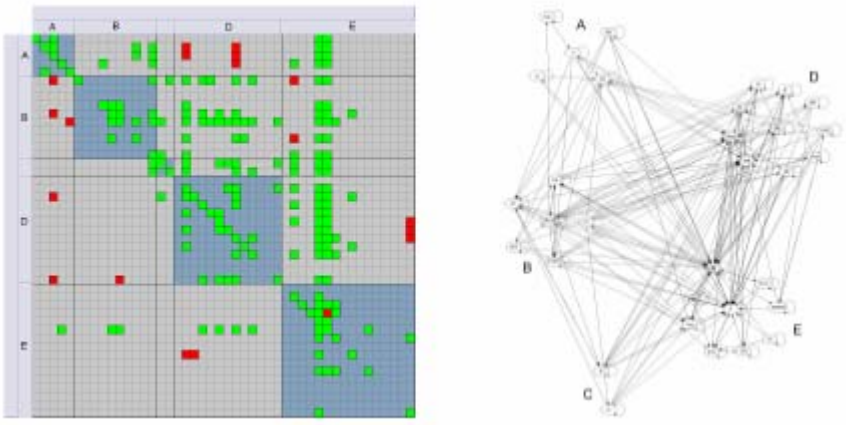
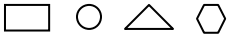
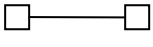
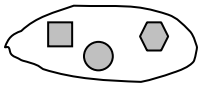
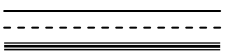


Figure 37 : matrice et diagramme nœuds-liens

Nous nous intéresserons principalement aux diagrammes *nœud-lien*, dont le concept intuitif et naturel semble offrir beaucoup de souplesse. Plusieurs techniques doivent être maîtrisées afin d'être capable de produire un diagramme *nœud-lien* qui révèlent véritablement des informations intéressantes et lisibles sur la structure d'un graphe. Le placement ou plongement d'un graphe est reconnue comme étant la méthode principale. Elle consiste à assigner une position à chaque sommet. Le placement des éléments visuels dans un diagramme ou une cartographie donne une représentation des distances entre les différents objets représentés. Les distances géométriques entre les éléments d'une cartographie sont perçues de manière préattentive, elles sont lues et interprétées très rapidement par notre cerveau. Pour que la représentation soit pleinement effective, les distances dans la cartographie doivent être réellement représentatives de distances dans les données. Nous verrons que les méthodes de placement basées sur les modèles de force ou d'énergie génèrent de tels plongements pour les diagrammes *nœud-lien*. La position est en fait une des caractéristiques des entités graphiques. Ces caractéristiques, appelées variables graphiques, ont été étudiées dans le contexte de la cartographie et de la visualisation d'information. A la base d'un véritable langage, les variables graphiques permettent d'enrichir significativement les représentations en communiquant différents type d'information. Ce chapitre présente donc un état de l'art sur l'utilisation des différentes variable graphique afin de transmettre de l'information. Nous nous intéresserons plus particulièrement à l'utilisation de la couleur en visualisation d'information et aux méthodes de placement spécifiques aux diagrammes *nœud-lien*.

3.2 Le langage graphique

L'utilisation de signes élémentaires dans les représentations graphiques constitue un véritable langage. Ce langage graphique a été étudié au début du XXième siècle par des psychologues allemands [Koffka35]. Leurs travaux sont connus sous le nom de Théorie de la Gestalt, et rassemblent l'ensemble des règles de la grammaire visuelle. Les éléments les plus communs de cette grammaire sont la forme, la couleur, la taille, la connectivité, l'inclusion et le placement, la Figure 38 [Ware04] illustre ces différentes règles.

Code graphique	Exemple	Sémantique
Contour fermé, région		Objet, entité
Forme du contour		Attribut qualitatif de l'entité
Couleur d'une région		Attribut qualitatif de l'entité
Taille de la région		Attribut numérique de l'entité
Segment entre deux régions		Relation entre entités
Contour englobant des régions		Inclusion d'entités
Style de tracé de segment		Attribut qualitatif de la de relation

Epaisseur du segment		Attribut numérique de la relation
Proximité		Groupement d'entités

Figure 38 : le langage graphique

La représentation de graphes par diagrammes *nœud-lien* se base principalement sur l'utilisation de deux règles de cette grammaire :

- la représentation des sommets par la région intérieure d'un contour fermé que nous nommerons entité graphique,
- la représentation des arêtes par un segment de courbe entre deux entités graphiques.

La lisibilité d'un diagramme *nœud-lien* dépend de plusieurs facteurs. La connectivité induite par les liens entre sommets peut être plus ou moins lisible selon la position des sommets, cf. Figure 39, et dans une moindre mesure selon la méthode utilisée pour le tracé des liens.

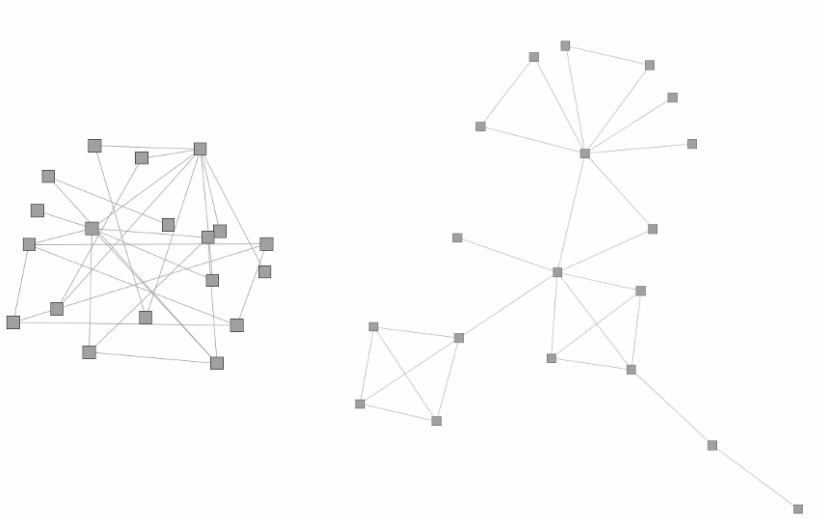


Figure 39 : certains placements assurent une meilleure lisibilité de la connectivité d'un diagramme

Dans le cas de graphes denses, dont la structure est très complexe le placement peut être utilisé non pas pour améliorer la lisibilité de la connectivité mais pour faire apparaître des clusters de sommets fortement corrélés, cf. Figure 40.

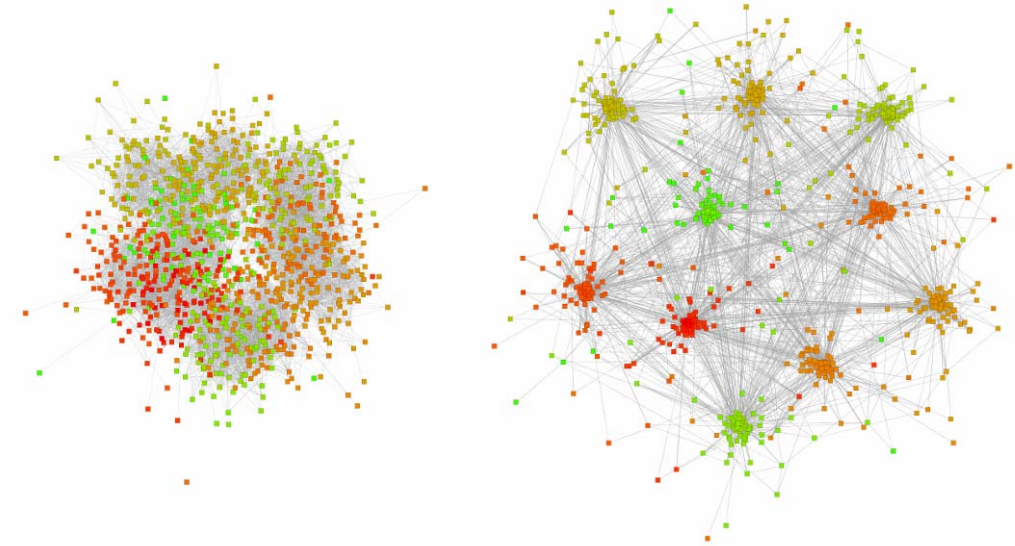


Figure 40 : le placement peut révéler des groupes de sommets fortement corrélés

Le placement d'un graphe est le paramètre le plus important pour s'assurer d'une bonne lisibilité et parvenir ainsi à en construire une représentation intelligible. Cependant, la plupart des règles de la théorie de la Gestalt peuvent être utilisées pour enrichir la représentation. Ainsi la variation des tailles de sommets peut être utile lorsque on veut représenter un attribut numérique, la couleur quant à elle est souvent utilisée pour représenter une catégorie ou une étiquette.

Avant d'étudier ces différentes options, il est important de s'intéresser aux types d'attributs qui peuvent être attachés aux entités d'un graphe.

3.3 Types d'attributs

Comme on l'a vu auparavant on peut associer aux sommets et arêtes d'un graphe différentes mesures qui permettent de déterminer leur rôle au sein du graphe. Ces attributs sont dits endogènes car ils sont définis par la structure même du graphe. D'autres types d'attributs peuvent être associés aux entités d'un graphe, ces attributs sont liés à la nature et l'origine du graphe. Par exemple, dans un réseau social, les sommets sont des personnes caractérisées par leurs nom, âge, fonction, etc. Une arête qui représente une relation entre deux personnes peut être définies par la nature de la relation (amicale, professionnelle,…), elle peut être valuée de manière à préciser la force de la relation. Ces attributs ne dépendent pas de la structure du graphe, ils sont dits exogènes.

Qu'ils soient endogènes ou exogènes les attributs peuvent être de trois types. Si on peut seulement dire de deux attributs qu'ils sont égaux ou différents, on parle d'attribut qualitatif. Ce type d'attribut permet principalement d'identifier de manière unique les objets, comme le nom d'une personne, ou de regrouper des objets différents sous une même étiquette, comme le service dont dépend un employé. Si on peut calculer la distance entre deux attributs, on dit qu'ils sont de type quantitatif. Toutes les valeurs numériques sont quantitatives, les dates le sont aussi. Enfin si on peut ordonner des attributs mais qu'il n'est pas possible de mesurer les distances qui les séparent, on parle d'attributs ordonnables. Tous les attributs quantitatifs sont aussi ordonnables. Une liste de chaînes de caractères peut être triée selon l'ordre alphanumérique afin de rendre plus facile la recherche d'un élément particulier, mais il n'y a pas de notion de distance naturelle entre deux chaînes. Les données ordonnables non numériques sont toutefois assez rares.

En visualisation d'information, il est très commun d'utiliser ce que l'on appelle un codage graphique, qui associe une variable graphique à un attribut. Ces notions de codage et de variables graphiques ont fait l'objet de nombreuses recherches, nous allons nous intéresser plus particulièrement aux travaux de Bertin et à ses travaux sur la sémiologie graphique.

3.4 La Sémiologie Graphique

3.4.1 Généralités

Les principes de la sémiologie graphique ont été exposés par Jacques Bertin dans l'ouvrage de référence "Sémiologie Graphique" publié en 1967 [Bertin67]. Ses principes ont été édictés à partir de règles sur la cognition: comment l'œil humain perçoit-il les signes graphiques et comment le cerveau transcrit-il cette perception en information ? Ces conclusions reposent principalement sur l'expérience cartographique de Bertin.

La sémiologie graphique est un ensemble de règles qui définissent un langage graphique dont :

- l'ensemble des signes graphiques élémentaires (points, lignes, surfaces) est l'alphabet
- les variables visuelles constituent le vocabulaire
- les règles de perception visuelle sont la syntaxe

Son but est de transmettre un ensemble d'information sous la forme d'une représentation graphique facilement lisible et mémorisable. Pour cela, la représentation doit utiliser des codes intelligibles par la perception visuelle. Elle utilise en effet les propriétés de l'image pour faire apparaître les relations de ressemblance et d'ordre entre les données.

La graphique s'applique à un ensemble de données préalablement définies sous forme de tableau La théorie matricielle de la graphique prend en compte :

- la correspondance entre le tableau des données et l'image,
- le niveau de perception, élémentaire ou d'ensemble, requis par l'objectif visé

Les informations à transmettre peuvent appartenir à trois types généraux :

- les informations qualitatives : une liste de noms communs ou de noms propre, ...
- les informations ordonnées,
- les informations quantitatives : mesures, proportions, ...

Le choix des variables visuelles, qu'on appelle codage graphique, est un des éléments qui contribue non seulement à la lisibilité du document mais aussi à son intelligibilité. En effet, toutes les variables ne possèdent pas la même aptitude à exprimer les mêmes informations. La forme, par exemple, est difficilement utilisable pour le codage d'un attribut numérique. Pour ces raisons, les représentations graphiques doivent utiliser des codes intelligibles par la perception visuelle. C'est dans cet objectif que Bertin a distingué 6 variables visuelles (taille, valeur, couleur, grain, orientation et forme) pour représenter l'information sur deux dimensions du plan (X,Y).

Ces variables visuelles ont des propriétés et sont transcrites selon trois formes d'implantation :

- ponctuelle,
- linéaire,
- zonale ou surfacique.

Dans le cas des diagrammes *nœud-lien,* les sommets sont représentés par des entités en implantation ponctuelle, et les arêtes par des segments en implantation linéaire. En cartographie traditionnelle, la représentation des données géographiques implique l'utilisation d'entités dans les trois types d'implantation. Les pays et régions sont en implantation surfacique, les villes et lieux en implantation ponctuelle (tout du moins à petite échelle) et les voies de transports et cours d'eau en implantation linéaire.

3.4.2 Variables visuelles

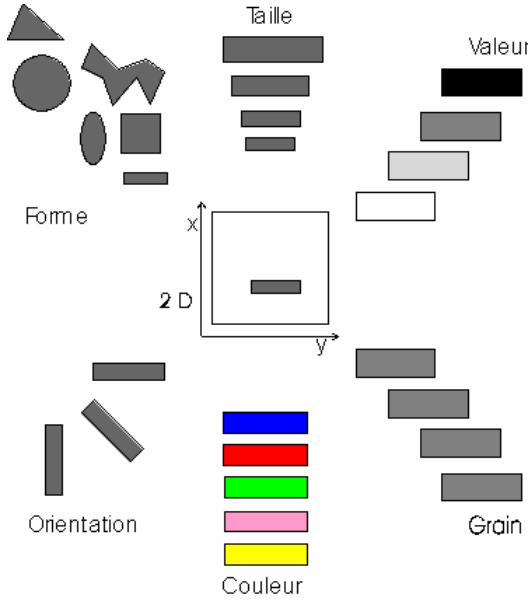


Figure 41 : les variables graphiques de Bertin

Bertin sépare la notion actuelle de couleur en deux variables, couleur et valeur. Il est plus pertinent de parler de teinte et d'intensité. La variable de couleur de Bertin correspond à l'usage de teintes différentes d'intensité égale. La variable de valeur correspond à l'utilisation d'échelle de gris, on peut l'étendre aux échelles à teinte constante et intensité croissante.

La variation de grain est très mal représentée sur la Figure 41. Le grain est une texture dont on peut faire varier la taille et la fréquence du motif. Les textures les plus courantes sont les pointillés, les hachures ou quadrillages, comme sur la Figure 42.

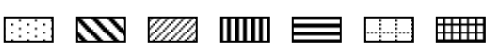


Figure 42 : Exemples de types de grain

La variation de grain est une variation dans la taille et la fréquence d'apparition du motif primitif de la texture. En imprimerie, la densité d'un grain est définie par la quantité d'encre nécessaire à sa réalisation. Plus généralement, la densité d'un grain est proportionnelle à l'aire des zones noircies (dans le cas d'un grain noir). La variation de grain peut se faire à densité constante comme sur la

Figure 43 ou être accompagnée d'une variation d'intensité, cf. Figure 44. On constate aussi que le grain peut être utilisé dans les trois types d'implantation.

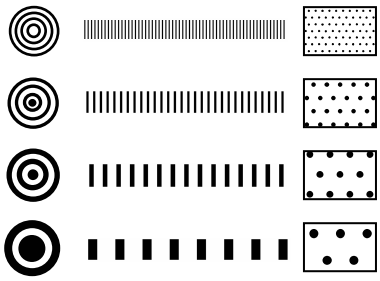


Figure 43 : Exemples de variation du grain à densité constante en implantation ponctuelle, linéaire et surfacique

Figure 44 : grain de densité croissante

Tout cela ressemble beaucoup à la théorie de la Gestalt. Là où Bertin va plus loin, c'est dans l'analyse de la perception de ces variables graphiques. Il définit les quatre propriétés perceptives, sélection, ordre, quantité et association, que peuvent présenter les variables visuelles.

3.4.3 Propriétés perceptives

3.4.3.1 Sélection ≠

Une variable est dite sélective lorsqu'elle permet d'isoler de manière préattentive les familles d'entités partageant la même valeur. Par exemple, dans la Figure 45, il est possible d'isoler de manière instantanée l'ensemble des entités bleues des autres. La teinte permet de sélectionner visuellement une catégorie d'objets, c'est donc une variable sélective.

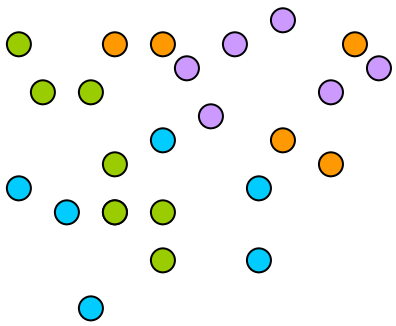


Figure 45 : la couleur permet d'isoler préattentivement différentes catégories

D'après Bertin, la forme n'est pas une variable sélective. Elle peut permettre d'isoler différentes catégories, mais pas de manière préattentive, cf. Figure 46.

Les variables sélectives sont utiles pour représenter des attributs qualitatifs. Elles permettent de repérer efficacement différentes catégories.

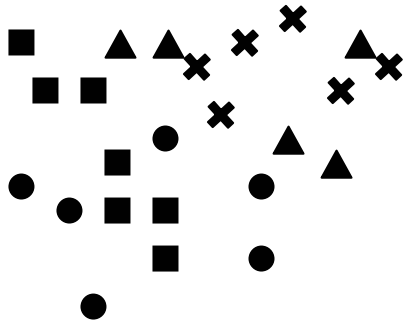


Figure 46 : la forme permet d'isoler des catégories, mais pas de manière préattentive

3.4.3.2 Ordre O

Une variable est dite ordonnée lorsque ses différentes valeurs sont perçues comme naturellement ordonnées. Par exemple, le gris est reconnu comme un intermédiaire entre le blanc et le noir. La taille est elle-aussi une variable ordonnée, cf. Figure 47.

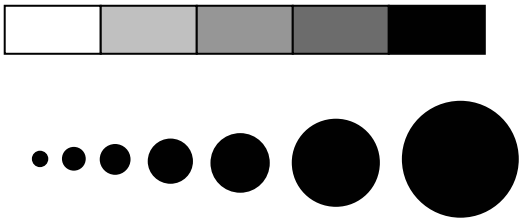


Figure 47 : taille et intensité sont des variables ordonnées

Par contre, la forme n'est en aucun cas une variable ordonnée, cf. Figure 48. La teinte est souvent utilisée comme variable ordonnée en visualisation d'information, notamment avec les échelles de teintes spectrales comme les échelles bleu-violet-rouge ou vert-jaune-rouge. Toutefois Bertin la considère comme une variable non ordonnée. Cela peut s'expliquer par le fait que la couleur est sous-utilisée en cartographie à l'époque de Bertin. De plus, s'il est vrai que certaines échelles de teintes semblent ordonnées, l'ordre qu'elles induisent est généralement artificiel, et susceptible d'être mal interpréter, cf. Figure 48.

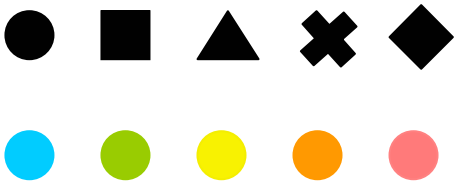


Figure 48 : couleur et forme ne sont pas ordonnées

Les variables ordonnées sont naturellement utilisées pour représenter des attributs ordonnables.

3.4.3.3 Quantité Q

Une variable est dite quantitative lorsqu'elle permet de d'évaluer la distance entre deux de ses valeurs. La position est une variable naturellement quantitative, tout comme la taille. La Figure 49 illustre l'utilisation de la position et de la taille comme variables quantitatives. On peut évaluer approximativement le rapport des longueurs des segments sur la figure du haut. La figure du bas présente une variation de taille en implantation ponctuelle. La méthode la plus exacte est d'utiliser la surface de la tâche comme taille. La surface doit donc être proportionnelle à la taille de l'entité.

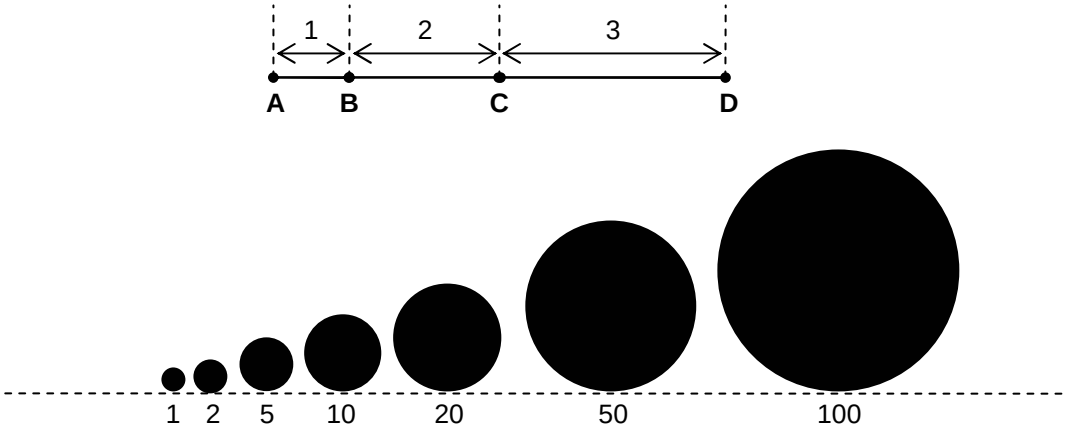


Figure 49 : position et taille sont les seules variables quantitatives

Les variables quantitatives sont utilisées pour représenter des attributs quantitatifs.

3.4.3.4 Association ≡

Enfin Bertin a défini une dernière propriété perceptive, l'association. Une variable est associative si ces différentes valeurs ne modifient pas la visibilité des objets. Deux phénomènes peuvent modifier la visibilité d'une entité graphique, sa taille et son intensité. Plus un objet est grand ou intense plus il sera visible. On en conclut rapidement que toutes les variables sont associatives sauf la taille et l'intensité.

Une variable associative doit être utilisée lorsqu'on veut s'assurer que les entités visuelles aient une visibilité équivalente.

3.4.3.5 Récapitulatif

Le Tableau 4 énonce et illustre l'ensemble des propriétés des différentes variables.

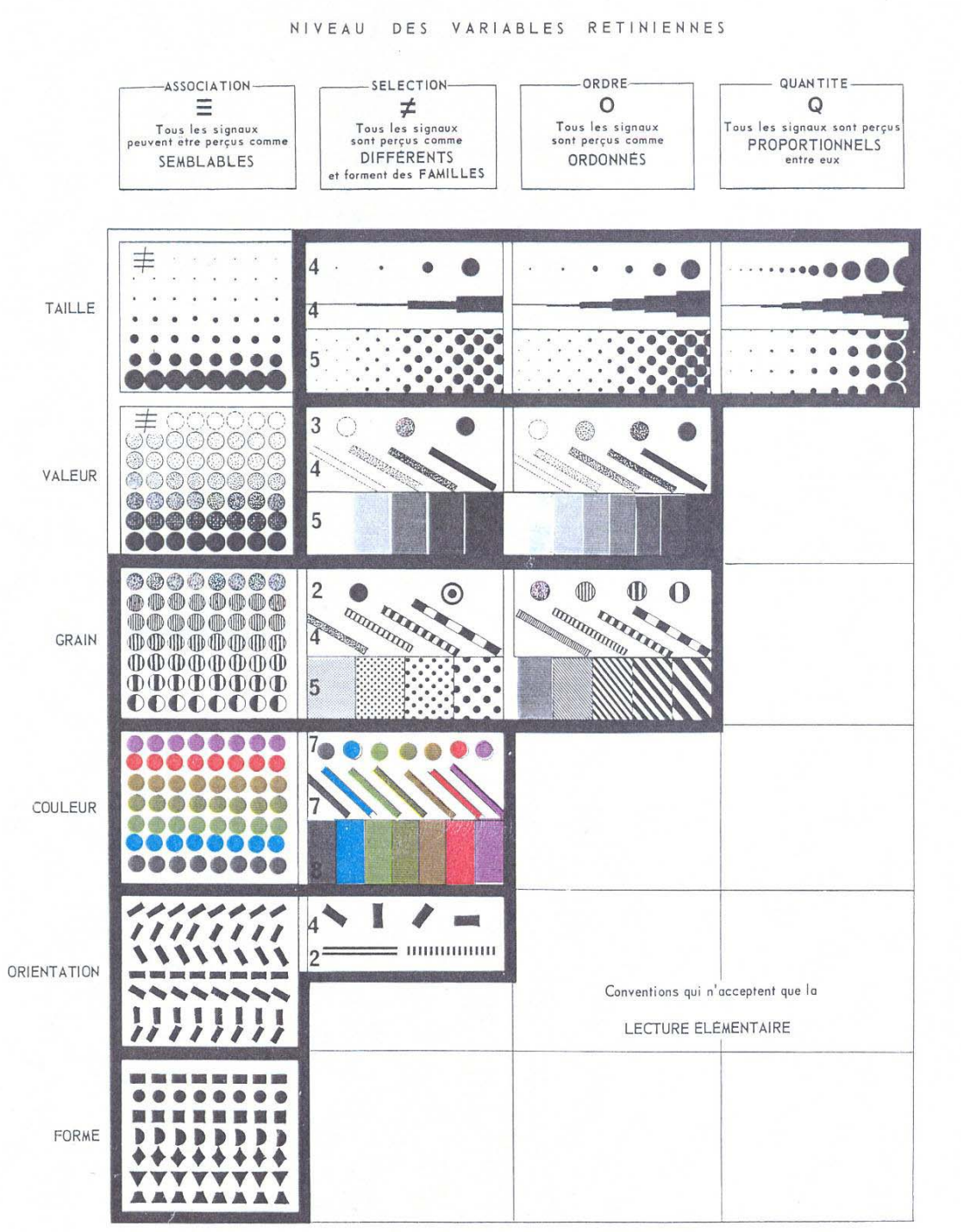


Tableau 4 : tableau des propriétés des variables graphiques (extrait de [Bertin67])

Dans les trois dernières colonnes, chaque variable est mise en contexte dans les trois implantations. Dans la seconde colonne, qui correspond à la propriété de sélection, le nombre accolé à chaque exemple précise le nombre maximal de valeurs qu'il est possible d'utiliser pour une sélection préattentive. Au delà de ce seuil la sélection s'approche d'une lecture élémentaire de l'image.

Les travaux de Bertin constituent une bonne référence pour le choix d'un codage graphique. On peut tout de même lui reprochait de sous estimer les possibilités qu'offrent la couleur pour le codage des données ordonnables et numériques. L'utilisation du grain est tombée en désuétude depuis l'avènement de la visualisation assistée par ordinateur. Cependant, dans un registre assez proche on peut noter l'utilisation des textures en visualisation d'information [Ware95]. L'orientation est utilisée pour des applications très stéréotypées comme la visualisation de champs magnétiques ou de forces, ou encore de données météorologiques de types courants marins ou déplacements de masses d'air. Dans la pratique cette variable est assez contraignante à utiliser. Elle interdit l'utilisation de la forme, et est perturbée par l'utilisation de la taille.

Nous avons donc choisi de porter notre attention sur la couleur qui correspond chez Bertin aux deux variables teintes et intensité et sur la taille qui est la variable de prédilection pour le codage de données numériques. La couleur fait l'objet d'une section importante où seront introduits les modèles de couleurs les plus utilisés, et présentés les résultats des principales expériences sur son utilisation pour le codage de données qualitatives et numériques. Suit une section plus courte sur la variable de la taille. Le placement fera quant à lui l'objet d'une étude plus approfondie sur les méthodes spécifiques aux diagrammes *nœud-lien*, cf. 3.7 et 3.8. La section suivante donne quelques recommandations sur la manière d'associer les valeurs d'une variable graphique aux valeurs d'un attribut numérique.

3.4.4 Codage d'attributs numériques et distribution des données

Nous avons vu que les attributs des données peuvent être associés à plusieurs variables graphiques. Cet appariement entre les données et la représentation graphique est appelé codage. Le codage est une fonction de l'espace des valeurs d'un attribut vers l'espace des valeurs d'une variable graphique.

3.4.4.1 Codage linéaire

Un codage est dit linéaire lorsqu'une variable graphique est liée linéairement à un attribut de données numérique. Considérons le codage d'un attribut numérique des sommets par leur taille. Dans ce cas simple, l'utilisateur choisi une taille minimale et une taille maximale pour coder respectivement la valeur minimale et la valeur maximale de l'attribut. Cela revient donc simplement à corréler linéairement deux intervalles numériques. Soient $V_{\min}$ la valeur minimum de l'attribut et $V_{\max}$ sa valeur maximale. Soient $T_{\min}$ et $T_{\max}$ les tailles minimales et maximales. La taille T d'un sommet dont la valeur d'attribut est V se calcule simplement par interpolation linéaire :

$$T(V) = T_{\min} + \frac{V - V_{\min}}{V_{\max} - V_{\min}} \cdot \left(T_{\max} - T_{\min}\right)$$

L'avantage de cette méthode est de permettre de comparer les valeurs d'attributs des différentes entités en comparant leur taille. Ce codage conserve les différences, la différence de taille entre deux sommets est proportionnelle à la différence de leurs attributs. De plus, si $T(0) = 0$ alors T est proportionnel à V et dans ce cas le rapport des tailles est aussi proportionnel au rapport des attributs.

L'efficacité de cette méthode dépend de la distribution des valeurs de l'attribut codé. Pour que ce codage fonctionne bien il est souhaitable les objets de grandes tailles soient rares afin d'éviter une

saturation de l'affichage. Ce codage fonctionne particulièrement bien avec les distributions de valeurs en loi de puissance décroissante. Dans ce cas, beaucoup d'entités auront une petite taille et quelques-unes une grande. Le cas le pire serait une loi de puissance croissante. Avec une distribution en loi normale, l'efficacité de ce codage peut être mis à mal par des valeurs trop extrêmes, par exemple une loi normale avec un intervalle de valeurs très grand par rapport à l'écart-type. Dans ces cas limites, il est souvent utiles de ne pas prendre en compte les valeurs les plus extrêmes lors du codage.

Si la distribution est très resserrée, il peut être intéressant d'appliquer une puissance supérieure à 1 aux valeurs avant le codage. Et au contraire lorsque la distribution est trop étendue une puissance inférieure à 1 peut être envisagée. Enfin dans des cas extrêmes les fonctions exponentielle et logarithmique peuvent aussi s'appliquer.

3.4.4.2 Codage par quantiles

Les q-quantiles sont les (q-1) valeurs qui séparent une population en q sous populations de tailles approximativement égales. Par exemple, la médiane est le seul 2-quantile, elle sépare une population en deux parties égales.

L'identification des q-quantiles d'une distribution permet donc de séparer la population en q classes de tailles comparables. Cette méthode peut être utilisée lors d'un codage, chaque classe se verra attribuer une taille ou une couleur différente. Il est important de comprendre que cette méthode dépend beaucoup plus de la répartition que des valeurs elles-mêmes.

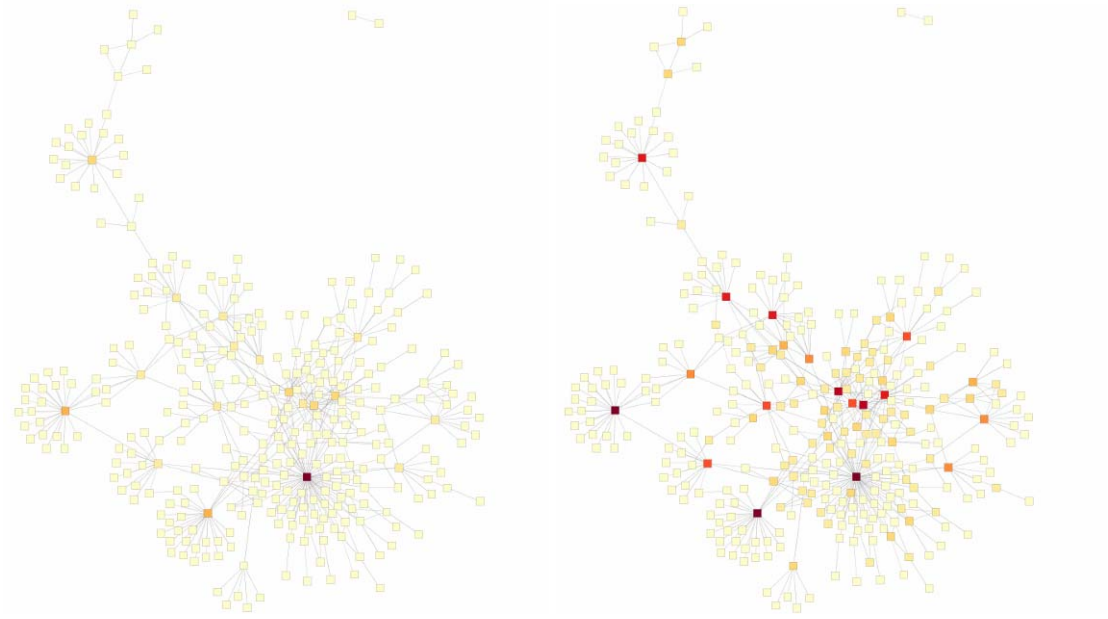


Figure 50 : codage linéaire et codage par quantiles d'une distribution en loi de puissance

3.5 Couleur

Rappelons qu'en informatique la couleur est modélisée de plusieurs façons. Les différents modèles de couleur définissent la manière de coder numériquement la couleur, par effet de bord ils définissent aussi comment se comporte le mélange de plusieurs couleurs. L'étude des différents modèles de couleurs donne un éclairage intéressant sur les différentes manières d'envisager l'utilisation des couleurs en visualisation d'information et laisse entrevoir la complexité du processus de perception humaine de la couleur.

3.5.1 Les modèles RGB et CMY

Le modèle RGB est le plus couramment utilisé, il définit une couleur comme la composition des trois teintes primaires rouge vert et bleu à différentes proportions. Une couleur est notée **`RGB(r,g,b)`** avec r,g et b des entiers bornés sur [0,255]. On représente souvent l'espace des couleurs du modèle RGB, dans un cube où chaque dimension représente une teinte primaire, cf. Figure 51.

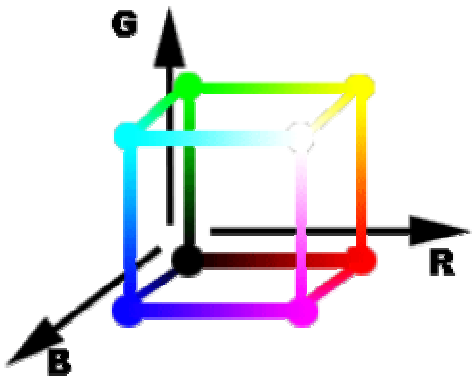


Figure 51 : l'espace RGB

La composition des teintes primaires se fait de manière additive. Si toutes les teintes sont nulles, **`RGB(0,0,0)`**, la couleur est noire, si toutes les teintes sont totalement présentes, **`RGB(255,255,255)`**, on obtient du blanc. Si les teintes primaires sont proportionnellement égales on obtient une couleur sur l'échelle des gris.

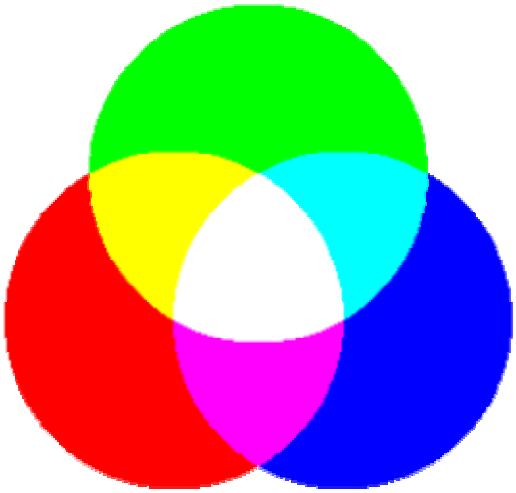


Figure 52 : composition additive des teintes primaires en RGB

Le modèle RGB correspond physiquement au fonctionnement des moniteurs et téléviseurs. Chaque pixel de l'écran est composé de trois diodes rouge, bleue et verte. Un utilisateur perçoit ces trois diodes comme un seul point, le pixel. En faisant varier l'intensité de chaque diode on obtient des couleurs différentes.

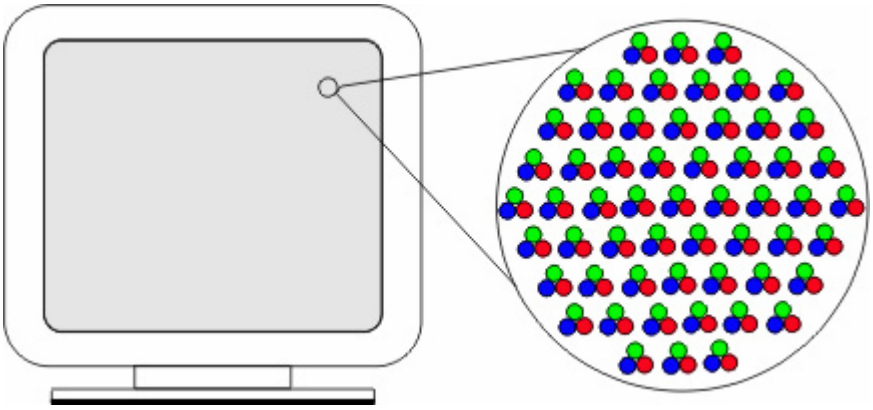


Figure 53 : Chaque pixel est composé de trois diodes RGB.

Le modèle RGB est étroitement lié au fonctionnement des écrans, c'est la raison pour laquelle il s'avère assez peu intuitif à manipuler. En particulier la composition additive des teintes primaires est assez déroutante pour le non-spécialiste. Nous sommes nombreux à avoir déjà manipulé une palette de peintre, et à s'être déjà essayé à mélanger des couleurs pour en obtenir une nouvelle. Dans ce cas le mélange est soustractif, plus on mélange de couleurs plus on tend vers un marron très sombre, presque noir, c'est le comportement exactement inverse du modèle RGB. Le modèle CMY soustractif est utilisé pour l'impression, il décompose lui aussi chaque couleur en trois composantes cyan, magenta, et jaune, ce sont les trois composantes secondaires du modèle RGB.

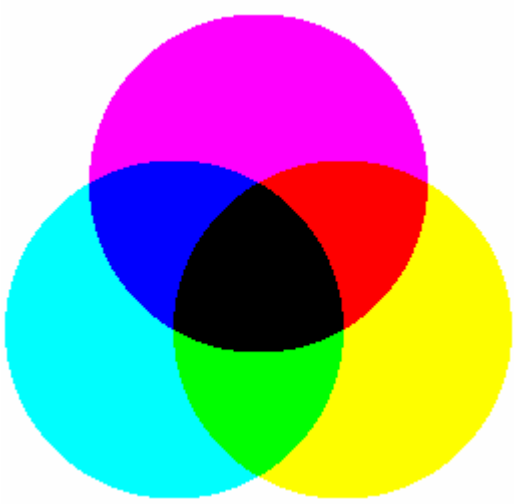


Figure 54 : composition soustractive des teintes primaires en CMY

Même si le mélange de couleurs dans le modèle CMY est plus proche du mélange de couleurs sur une palette de peintre, la couleur résultant du mélange de deux couleurs reste très difficilement prévisible pour un utilisateur. Un modèle basé sur la palette du peintre a récemment été proposé afin de résoudre ces problèmes [Gosset04], toutefois si les auteurs donnent leurs recommandations, ils n'ont pas implémenté ce modèle, qui reste donc virtuel pour l'instant.

3.5.2 Le modèle HSB

Un autre modèle de couleur est populaire dans les applications de traitement d'image, le modèle HSB. Si chaque couleur est encore une fois décomposée en trois composantes, la comparaison avec les modèles précédents s'arrête là. Chaque couleur est ici définie par sa teinte, sa saturation et sa brillance. En anglais cela donne *Hue*, *Saturation* et *Brightness*. La teinte correspond à ce qu'on appelle communément et de manière abusive la couleur, elle peut être rouge, bleue, orange, etc. Au niveau physique, chaque teinte correspond à une lumière de longueur d'onde différente. L'arc-en-ciel est un bon exemple de palette de couleurs de teintes différentes. La saturation d'une couleur définit l'intensité avec laquelle s'exprime la teinte, si elle est nulle on obtient une couleur de l'échelle de gris, si elle est maximale la teinte est dite pure. Enfin la brillance définit la quantité de lumière que renvoie la couleur. Si la brillance est nulle la couleur ne renvoie une quantité de lumière nulle, elle est noire. On peut représenter l'espace HSB dans un cône comme ci-dessous.

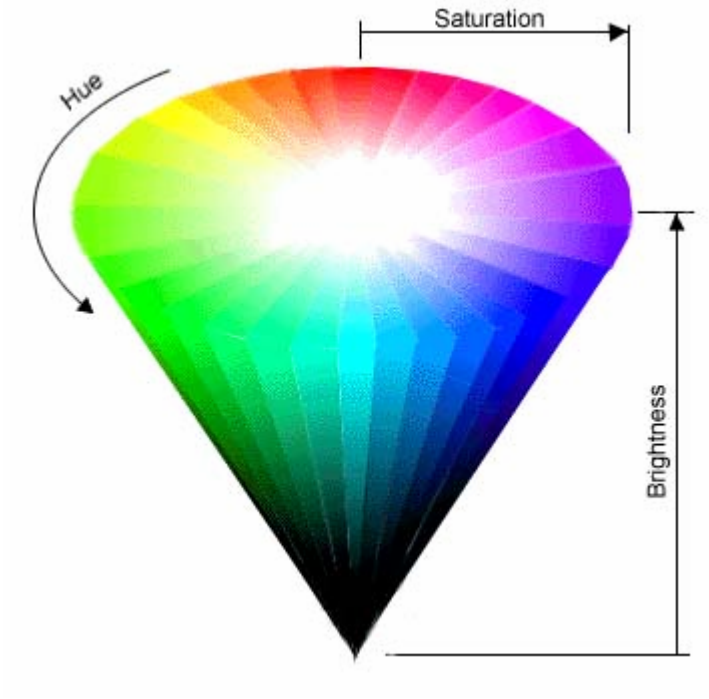


Figure 55 : l'espace HSB

Comme on le voit ci-dessus l'ensemble des teintes décrit un cycle, c'est pourquoi la teinte est généralement codée par une valeur d'angle réelle sur [0,360[, la saturation et la brillance sont codées comme des réels sur [0,1]. L'intérêt majeur de ce modèle est de séparer le chromatisme, défini par la teinte et sa saturation, de la lumière définie par la brillance. Les teintes progressent de manière prévisible en reflétant la décomposition spectrale de la lumière visible du rouge vers le violet, comme chacun a pu l'observer sur un arc-en-ciel.

3.5.3 Couleur et Perception

Ces modèles souffrent tous de problèmes liés à la perception des couleurs par l'œil humain. Ainsi les brillances des couleurs primaires des modèles RGB ou CMY ne sont pas perceptivement équivalentes. La couleur **`RGB(0,255,0)`** est un vert très clair et très brillant comparé aux couleurs **`RGB(255,0,0)`** ou **`RGB(0,0,255`**), qui donnent respectivement un rouge criard et un bleu plutôt foncé.

Il est important de revenir sur la définition de la brillance qu'il ne faut pas confondre avec luminance et luminosité. Les définitions suivantes sont celles adoptées par la *Compagnie Internationale de l'Éclairage* ou *CIE*. La brillance est une mesure de l'intensité lumineuse d'une couleur. Elle a été définie avant l'arrivée des matériels de mesures précis, et elle dépend donc du système visuel humain, elle mesure en fait l'intensité lumineuse apparente d'une couleur. A partir de 1948, la brillance est abandonnée car trop subjective, on lui préfère la luminance qui correspond réellement à l'intensité lumineuse mesurable. La luminance est la mesure de référence de l'intensité lumineuse. En 1976, la CIE réintroduit une mesure de l'intensité lumineuse apparente sous le nom de luminosité, la relation entre luminosité et luminance est exponentielle, cf. Figure 56. Lorsque l'on baisse la puissance de la source lumineuse de moitié la luminosité ne diminue que d'un quart.

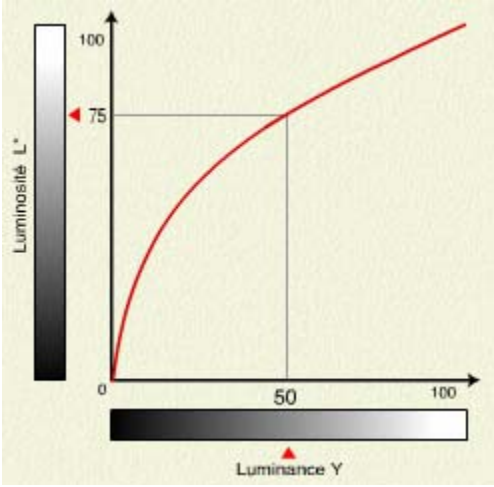


Figure 56 : relation entre luminance et luminosité

Dans le modèle HSB, le terme brillance est mal employé, car cette composante est en fait la luminance de la couleur. Avec HSB on constate qu'à luminance et saturation égales, deux couleurs de teintes différentes n'ont pas la même luminosité apparente. Les teintes du jaune au cyan paraissent plus lumineuses car l'œil humain est plus sensible à leurs longueurs d'onde, cf. Figure 57.

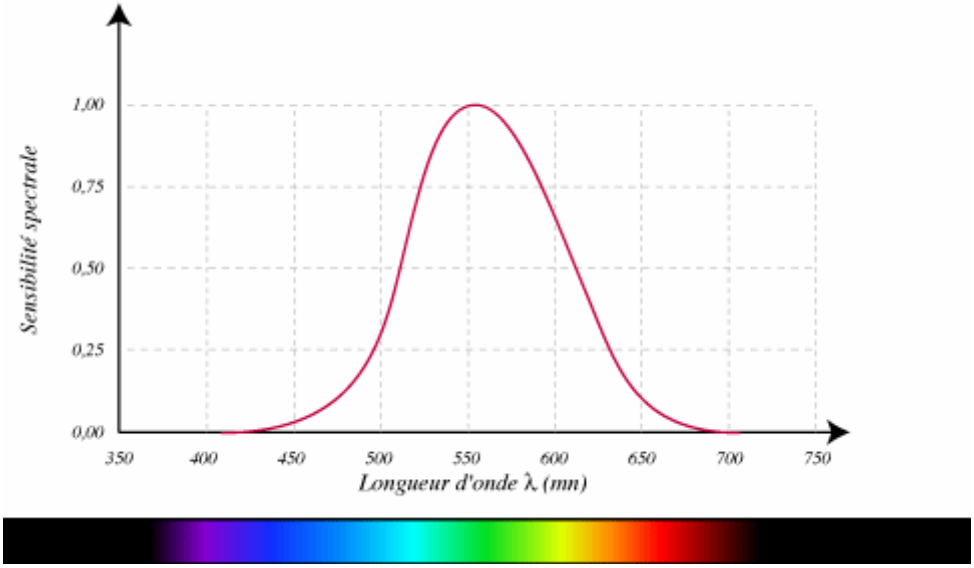


Figure 57 : sensibilité de l'œil aux différentes teintes

L'œil humain n'est pas sensible de la même manière aux différentes couleurs, cela s'explique physiquement par la présence dans l'œil de capteurs de types différents. Les bâtonnets sont sensibles à la luminosité, et les cônes aux couleurs. Il existe trois types de cônes L, M et S sensibles respectivement aux teintes rouge, verte et bleue. La perception de la couleur est un processus complexe qui résulte de l'excitation de ces différents capteurs. De manière générale on constate qu'à luminosité moyenne, l'œil est plus sensible aux teintes vertes ou rouges qu'aux teintes bleues. A forte luminosité la sensibilité aux teintes vertes diminuent, il devient difficile de distinguer deux couleurs vertes voisines.

En visualisation d'information, il est particulièrement important de pouvoir déterminer si deux couleurs différentes sont proches ou distantes en teintes, intensité et luminosité. Ainsi lorsque l'on

choisit des couleurs pour le codage de données qualitatives, il est important qu'elles soient facilement identifiables et il est tout aussi important qu'elles aient une visibilité équivalente. Pour le codage de données numériques, on choisit généralement des couleurs de teintes voisines, dont on fait varier l'intensité du clair au foncé, cette variation d'intensité doit pouvoir être contrôlée. Le modèle de couleur idéal en visualisation d'information est un modèle où la distance théorique entre deux couleurs correspond à la distance effectivement perçue par l'œil humain. De tels modèles, dits perceptifs, ont été élaborés par la C*ommission Internationale d'Eclairage*.

3.5.4 Les modèles CIE XYZ, et L*a*b*

La *CIE* qui a créé la plupart des standards colorimétriques a défini le modèle **CIE XYZ** comme une extension du modèle **RGB**. Les composantes X, Y et Z en sont respectivement les valeurs en rouge, vert et bleu. Cela ressemble jusqu'ici beaucoup à du **RGB**. La différence entre les deux modèles, c'est l'espace de couleurs qu'ils définissent. Le gamut montre l'espace de couleur d'un modèle, la figure ci-dessous présente le gamut du modèle **CIE XYZ**, le triangle qui le recouvre partiellement est le gamut du modèle **RGB**.

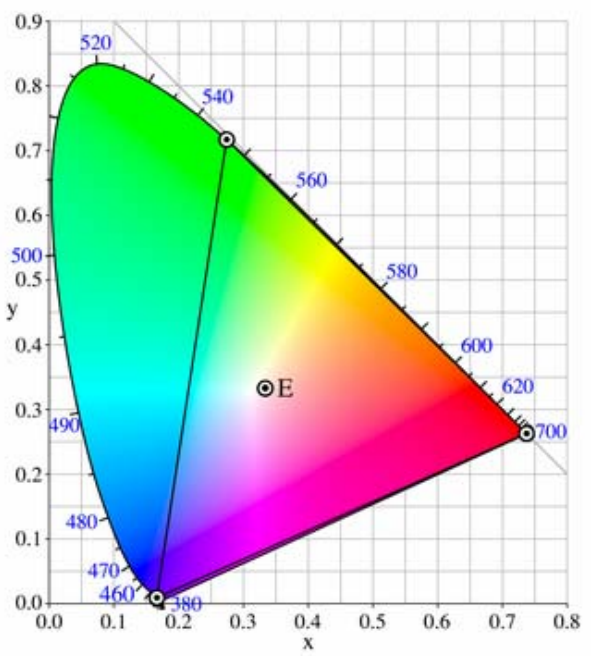


Figure 58 : Gamut RGB et CIEXYZ

De plus **CIE XYZ** est basé sur la modélisation d'un observateur humain standard. Cela signifie que le modèle prend en compte les particularités de l'œil humain. De part la sensibilité exacerbée de l'œil humain pour le vert la luminance d'une couleur est souvent considérée égale à sa composante Y dans le modèle **XYZ**. Le modèle **CIE XYZ** permet donc de contrôler la luminance d'une couleur, mais pas sa luminosité. Ce modèle n'est donc pas totalement perceptif.

C'est pourquoi le modèle **CIE L*a*b*** a été créé. Il partage le même espace de couleur que le modèle **CIE XYZ** et peut donc être vu comme une alternative. Ce modèle code toujours une couleur en trois composantes L*, a* et b*. L* est la composante de luminosité, elle varie de 0 à 1. Les composantes a* et b* sont bornées toutes deux sur [-1,1]. Ces deux composantes forment deux axes qui représente chacun deux teintes complémentaires. L'axe a* code les teintes des verts vers les

rouges, et l'axe b* les teintes des bleus vers les jaunes, cf. Figure 59. Avec ce modèle, il est théoriquement possible de calculer une distance, entre deux couleurs, qui correspond effectivement à la distance perçue par un utilisateur.

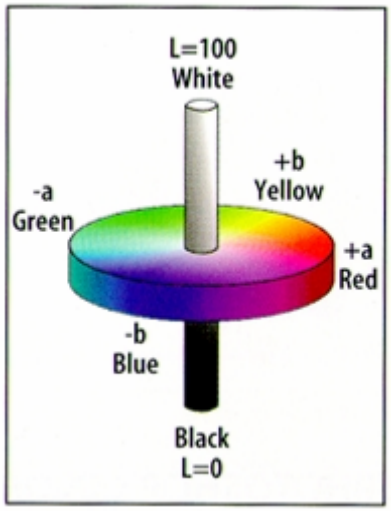


Figure 59 : le modèle CIE L*a*b*

3.5.5 Utilisation de la couleur en visualisation d'information

Les ordinateurs offrent maintenant depuis longtemps des écrans de très grande résolution permettant d'utiliser des dizaines de millions de couleurs. Cette avancée technologique a permis d'envisager l'utilisation massive de la couleur en visualisation, et a donc été l'objet de nombreux travaux et expérimentations.

3.5.5.1 Nombre de couleurs et préattention

Healey a mené une étude [Healey96] sur l'utilisation de la couleur en visualisation d'information. L'objectif principal était de déterminer combien de couleurs peuvent être différenciées de manière préattentive. Les couleurs sont générées à partir du modèle L*uv qui permet de contrôler leur luminosité tout comme le modèle L*a*b*. Les couleurs sont choisies de manière à avoir la même luminosité, la distance entre les teintes de chaque paire de couleurs est fixée afin d'assurer qu'elles soient effectivement identifiables. L'expérimentation consiste à identifier le seul objet portant la couleur cible parmi un grand nombre d'objets distracteurs portant d'autres couleurs. L'étude a été réalisée avec 3, 5, 7 et 9 couleurs différentes.

Jusqu'à 5 couleurs la durée du processus d'identification ne dépend ni du nombre d'objets affichés, ni du choix des différentes couleurs, ce qui est caractéristique d'un processus préattentif. Avec 7 et 9 couleurs, le choix de la couleur cible devient déterminant, l'identification reste préattentive pour certaines, mais se rapproche d'une recherche linéaire pour d'autres. Les couleurs cibles qui donnent les plus mauvais résultats sont celles proches du vert.

Cette étude montre qu'il est possible d'utiliser près d'une dizaine de couleurs simultanément en conservant des caractéristiques préattentives, et que plus on utilise de couleurs différentes plus le choix de ces dernières est déterminant.

3.5.5.2 Échelles de couleurs

Une échelle de couleurs est une liste ordonnée de couleurs sélectionnées qui définit l'ensemble des valeurs que peut prendre la variable de couleur lors d'un codage. La Figure 60 montre un exemple d'échelle de couleur pour des données qualitatives. Chaque couleur est utilisée pour coder une catégorie, ici différents sports.

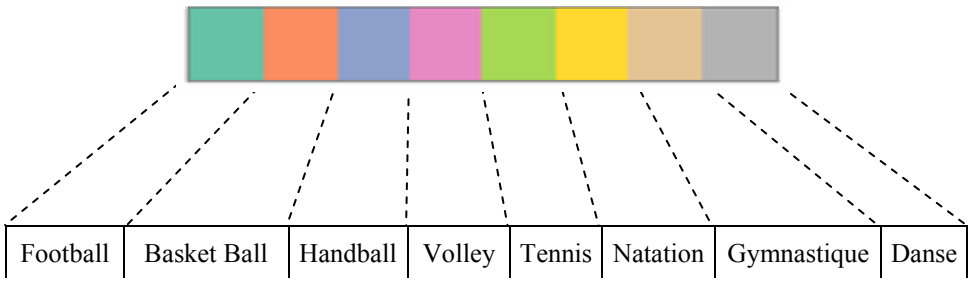


Figure 60 : exemple échelle de couleur

Rogowitz et Treinish [Rogowitz96] ont mené plusieurs expériences permettant de déterminer les caractéristiques idéales d'une échelle de couleurs. Leur objectif est entre autre d'être capables de générer automatiquement une échelle de couleur en fonction du type de données à coder, de la fréquence spatiale des valeurs dans l'affichage et de la tâche à réaliser [Bergamn95]. Ils ont implémenté un composant logiciel qui obéit à des règles de construction définies sous forme d'une taxonomie présentée dans le Tableau 5.

La première colonne définit quatre types de données :

- Ratio : donnée numérique où le rapport entre valeurs est significatif,
- Interval : donnée numérique où la différence (ou distance) entre valeurs est significative,
- Ordinal : donnée ordonnable,
- Nominal : donnée qualitative.

La seconde colonne spécifie la fréquence spatiale qui peut être faible ou élevée. Une fréquence spatiale élevée se traduit par une image dont la granularité est très fine. Les auteurs conseillent dans ce cas d'utiliser une échelle de couleurs dont la luminance varie de manière croissante. Lorsque la fréquence spatiale est faible, les couleurs de l'échelle sont à luminance constante, mais à saturation croissante. Ces règles de construction sont précisées dans la troisième colonne (isomorphic). La quatrième colonne indique le nombre de couleurs à utiliser dans le cas d'une tâche de segmentation. La dernière colonne spécifie le choix de couleurs afin d'augmenter la visibilité d'un sous-ensemble de valeurs.

Des collections d'échelles de couleur ont été de nombreuses fois définies dans la littérature. Celle proposée par Brewer [Brewer94] fait référence. Chaque échelle a fait l'objet de plusieurs expérimentations afin d'évaluer sa lisibilité, mais aussi de déterminer les médias avec lesquelles elle est compatible (moniteur, plasma, LCD, impression, projection,…).

Data Type	Spatial Frequency	Representation Task		
		Isomorphic	Segmentation	Highlighting
Ratio (true zero)	Low	*Luminance:* uniform *Hue:* opponent or complementary pairs *Saturation:* monotonically increasing from gray	Even number of segments Many segments OK	Larger range for highlighted features
	High	*Luminance:* monotonically increasing *Hue:* opponent or complementary pairs *Saturation:* monotonically increasing from gray	Even number of segments Fewer segments	Smaller range for highlighted features
Interval	Low	*Luminance:* uniform *Hue:* opponent pairs *Saturation:* monotonically increasing from gray	Many segments OK	Larger range for highlighted features
	High	*Luminance:* monotonically increasing *Hue:* uniform or small hue variation *Saturation:* monotonically decreasing	Fewer segments	Smaller range for highlighted features
Ordinal	Low	*Luminance:* uniform *Hue:* variation around hue circle *Saturation:* monotonically decreasing	Fewer segments	Increase luminance of highlighted area
	High	*Luminance:* monotonically increasing *Hue:* variation around hue circle *Saturation:* uniform	More segments	Increase saturation of highlighted area
Nominal	Low	*Luminance:* uniform *Hue:* variation around hue circle *Saturation:* uniform	Fewer segments than 7	Increase luminance or saturation of highlighted area
	High			

Tableau 5 : Taxonomie des échelles de couleurs, extrait de [Rogowitz96]

3.5.5.3 Données qualitatives

Avec des données qualitatives la couleur ne peut être utilisée que dans un objectif de différenciation. Lorsque l'on considère deux objets, ils sont égaux ou différents. Il n'y pas de relation d'ordre entre eux, ils ont tous la même importance et les couleurs utilisées doivent être de même visibilité, et donc avoir la même luminosité, de préférence moyenne. Reste le choix des teintes. Afin de ne pas donner d'impression d'ordre entre les objets il faut choisir un ensemble de teintes deux à deux distinctes, l'idéal étant de contrôler la distance minimale possible entre deux teintes.

Cette sélection de couleurs reste possible tant que le nombre de valeurs de nos données qualitatives reste faible, au maximum une petite dizaine. Dans le cas contraire, il devient impossible d'assurer que les couleurs seront deux à deux différentiables assez rapidement, dans ce cas un codage reste possible mais risque d'être difficile à lire.

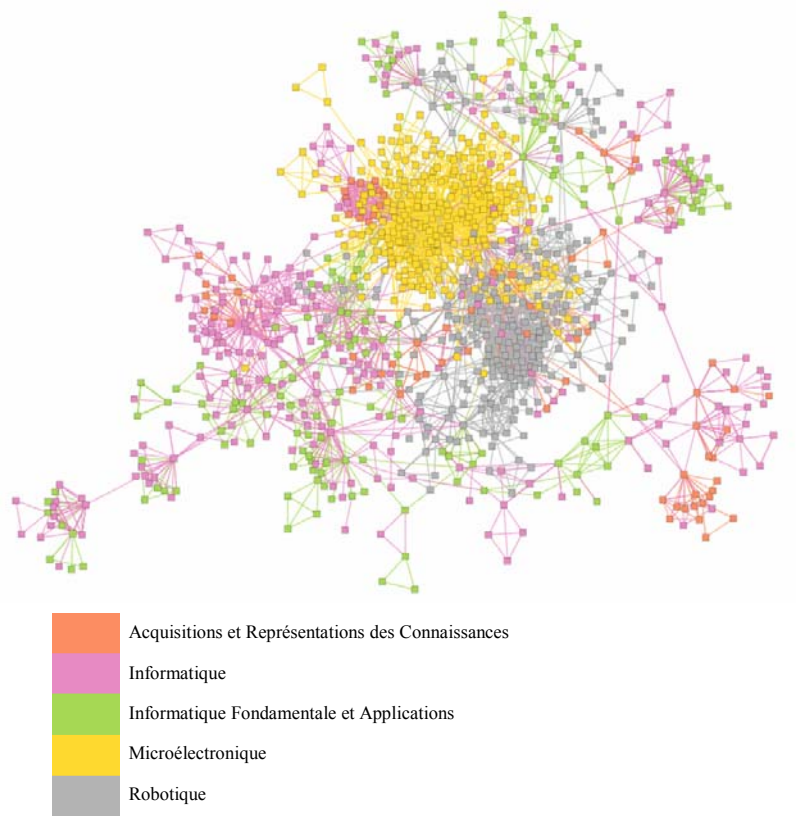


Figure 61 : codage couleur pour données qualitatives

La Figure 61 représente le graphe des co-auteurs du LIRMM, la couleur code le département auquel appartient l'auteur. Les couleurs utilisées pour ce codage proviennent des échelles élaborées par Brewer.

3.5.5.4 Données ordonnables

Pour les données ordonnables la couleur est utilisée pour révéler l'ordre des données. Dans ce cas on utilise un ensemble de couleurs issues le plus souvent de la même teinte, mais dont l'intensité varie. L'échelle des gris est couramment utilisée, cette échelle peut-être généralisée aux teintes bien saturées. L'important est d'associer la plus petite valeur au blanc et la plus grande à la couleur la plus sombre ou la plus saturée, les couleurs intermédiaires peuvent être interpolées linéairement dans un modèle perceptif, comme XYZ ou L*a*b*, cf. Figure 62.

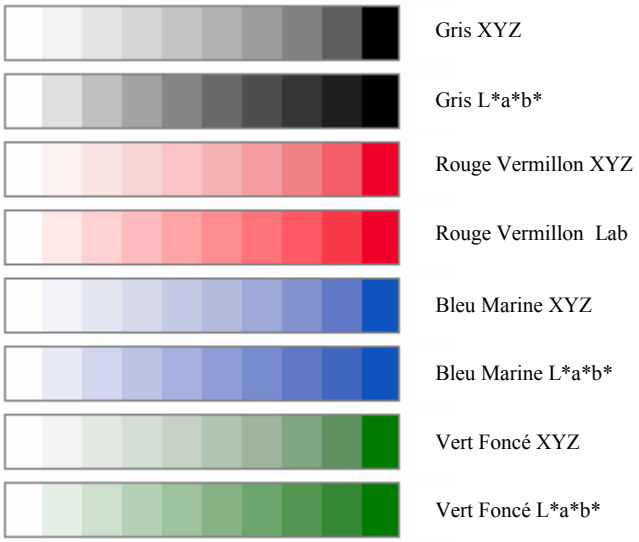


Figure 62 : Echelles de couleurs à luminosité décroissante, 10 teintes

Les échelles de couleurs de Brewer pour les données ordonnables présentent une réelle alternative aux échelles de couleurs interpolées. Ces échelles sont construites avec des couleurs dont l'intensité et la teinte varie. L'utilisation de plusieurs teintes peut en effet rendre les différentes couleurs plus facilement séparables, cf. Figure 63.

Figure 63 : Echelles Brewer à luminosité décroissante et à teinte variable, 9 teintes

3.5.5.5 Données numériques

Les données numériques sont ordonnables, et il est possible de calculer une distance entre deux données. Les échelles pour données ordonnables peuvent être utilisées mais il est alors recommandé d'utiliser un modèle perceptif basé sur la luminosité comme L*a*b*. Dans ce cas la couleur permet d'évaluer grossièrement la valeur représentée par une couleur. Il est tout de même indispensable d'accompagner une échelle de couleur de sa légende qui précisera les intervalles de valeurs correspondant à chaque couleur de l'échelle.

Les échelles de couleurs définies précédemment pour les données ordonnables conviennent aussi aux données numériques. Il toutefois nécessaire de déterminer si le rapport entre deux valeurs est significatif. Dans ce cas il est important d'associer la première couleur de l'échelle aux valeurs nulles, et cela même si les données ne contiennent pas de telles valeurs.

Brewer propose d'utiliser des échelles de couleurs divergentes pour les données numériques. Une échelle divergente peut être défini comme deux échelles ordonnables mises bout à bout, la première étant décroissante en intensité et la seconde croissante.

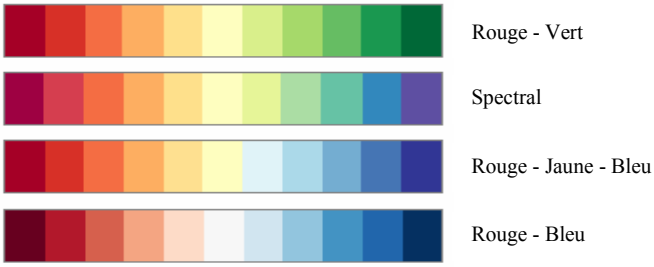


Figure 64 : Echelles de couleurs divergentes pour données numériques, Brewer, 11 teintes

Encore une fois, si les rapports entre valeurs sont importants il faudra prendre soin d'associer la couleur centrale avec la valeur nulle. Si seules les distances ont un sens, la couleur centrale pourra être associée à la valeur centrale des données. Moyenne et médiane sont de bonnes mesures de valeur centrale.

3.6 Taille

La taille des entités est typiquement utilisée pour le codage des attributs ordonnables et numériques. L'œil est sensible aux régions connexes d'une image, et plus l'aire de ces régions est grande, plus elles sont repérables rapidement [Ware04]. La taille peut être déclinée selon deux des implantations que définit Bertin :

- implantation ponctuelle, elle est définie comme la surface de la région représentant le point.
- implantation linéaire, elle est définie comme l'épaisseur d'une courbe.

Les sommets présentent une implantation ponctuelle, la taille d'un sommet est donc définie comme sa surface. Pour les arêtes dont l'implantation est linéaire, elle correspond à l'épaisseur de la courbe.

La taille est une variable sélective et quantitative. Cependant, d'après le Tableau 4 d'organisation des variables graphiques de Bertin, la taille est sélective lorsqu'on se limite à quatre valeurs de tailles différentes, au delà l'identification des catégories se rapproche d'une lecture linéaire de l'affichage. Il y a donc un choix à faire entre la précision d'un codage quantitatif et la possibilité d'une sélection préattentive.

Si la taille est utilisée pour coder des données numériques, il est recommandé d'utiliser une méthode de codage linéaire, cf. 3.4.4, afin que les différences entre tailles soient cohérentes avec les différences entre valeurs.

3.7 Placement des arbres

Un arbre est un type de graphe particulier. Les arbres sont des graphes connexes acycliques dont le nombre d'arêtes est minimal. Si une seule arête est enlevé à un arbre il n'est plus connexe. Cette structure ressemble effectivement à un arbre ou une plante, elle possède une racine d'où peut partir des branches qui elles même peuvent se séparer en sous branches, etc. Les feuilles sont les éléments finaux des branches. Cette structure est utilisée pour décrire de nombreux objets différents. Les systèmes de

fichiers informatiques basés sur l'imbrication de répertoires sont des arbres dont les feuilles sont des fichiers. Le découpage géographique des pays en régions, départements, communes, par exemple pour la France est aussi un arbre. De nombreuses classifications sont hiérarchiques, l'Open Directory qui répertorie les sites Internet et les classes de manière thématique, l'équivalent existe à l'Ina, la thématisation des documents audiovisuels, enfin les scientifiques ont produit de nombreuses classifications hiérarchiques, du monde vivant avec la taxinomie linnéenne qui décompose les entités vivantes en genre, espèce, sous-espèce, etc.

Le placement des nœuds d'un arbre obéit à des conventions qui dépendent du domaine d'application. Par exemple dans la hiérarchie géographique qui organise le territoire français, chaque niveau correspond à un type d'entités différent. Au premier niveau on trouve les régions, au second les département, ensuite les cantons et les communes. Dans ce cas, il est important que le placement de l'arbre permette d'identifier ces différents niveaux. De manière générale, un placement d'arbre doit obéir aux principales contraintes "esthétiques" suivantes [Battista94] :

- les nœuds sont placés sur des lignes horizontales ou verticales correspondant à leur profondeur,
- l'espace entre deux nœuds voisins de même profondeur doit être fixe et minimale,
- la largeur du dessin doit être aussi petite que possible,
- les symétries doivent apparaître, si deux sous-arbres sont identiques ils leurs dessins doivent l'être aussi.

Des contraintes additionnelles sont parfois mentionnées :

- un nœud père doit être centré par rapport à ses nœuds enfants,
- un nœud père soit être centré par rapport à ses descendants,
- l'ordre des enfants doit être respecté

Tous ces critères ne sont pas réalisables simultanément, par exemple le respect de l'ordre des enfants n'est pas compatible avec l'affichage de toutes les symétries, car dans ce cas il est parfois nécessaire de réordonner les nœuds.

Les paragraphes suivants décrivent les techniques de placements standards d'arbres, les placements classique et radial respectent les contraintes principales.

3.7.1 Placement classique

Cette méthode de placement [Reingold81] fut la première à répondre aux principales contraintes "esthétiques". L'algorithme original a été amélioré par la suite [Walker90] passant d'une complexité quadratique à linéaire tout en conservant le même résultat. L'orientation du dessin, c'est à dire la direction dans laquelle se déploie les branches, est paramétrable, même si culturellement les orientations horizontale, de gauche à droite ou verticale, de haut en bas sont les plus utilisées, tout du moins en occident où elles respectent alors le sens d'écriture. En orientation verticale, les sommets à profondeur égale sont placés sur la même ligne horizontale, les distances entre nœuds de la même fratrie et nœuds du même niveau sont des paramètres, l'espacement entre niveau également. Les nœuds père peuvent être centrés par rapport à leurs enfants ou descendants, les alignements à gauche ou à droite sont possibles bien que plus exotiques.

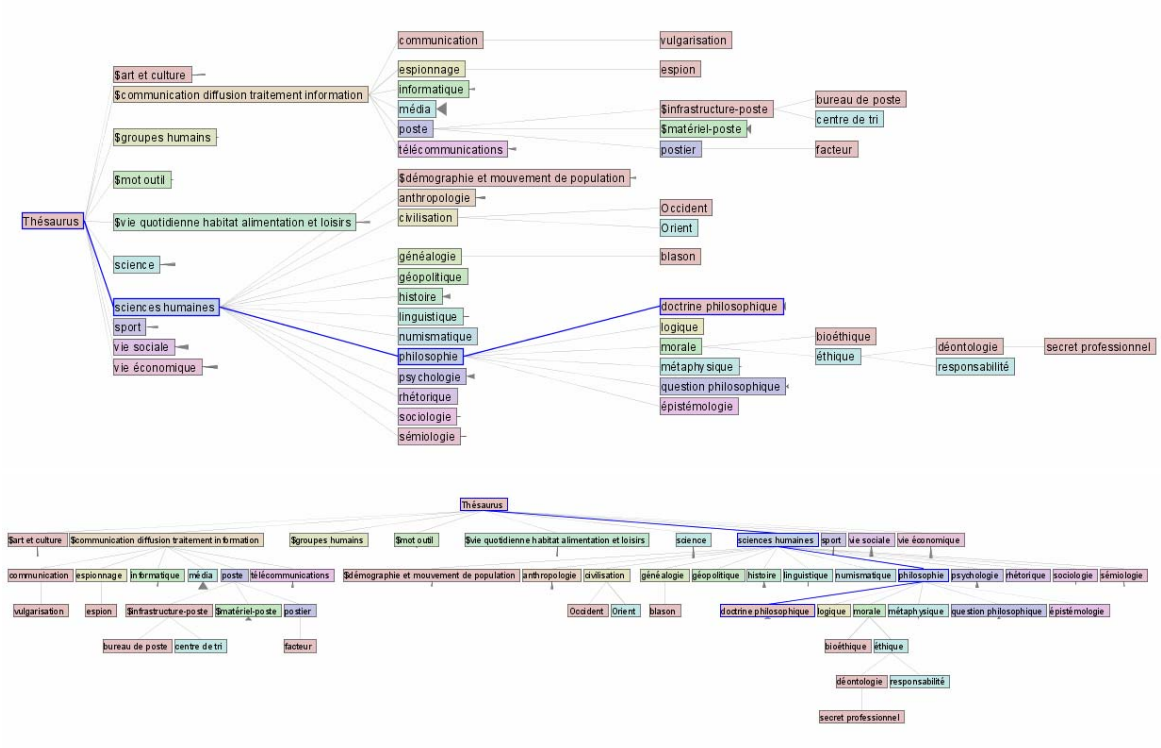


Figure 65 : Placement de Reingold vertical et horizontal

Cet algorithme possède de bonnes qualités, la largeur du dessin, distance séparant la feuille la plus à droite de celle la plus à gauche pour un dessin horizontal, est minimale. Le type même du placement se prête bien au paradigme d'interaction d'ouverture/fermeture des nœuds comme moyen d'exploration. Enfin la complexité linéaire de l'algorithme permet effectivement d'explorer l'arbre interactivement, avec des temps de calcul du placement très courts.

Son défaut majeur, c'est toutefois d'échouer à donner une représentation pertinente des arbres grands et larges. Dans ce cas le dessin est beaucoup plus large que haut, ce qui correspond effectivement à la structure de l'arbre, a un aspect très aplati et incompréhensible

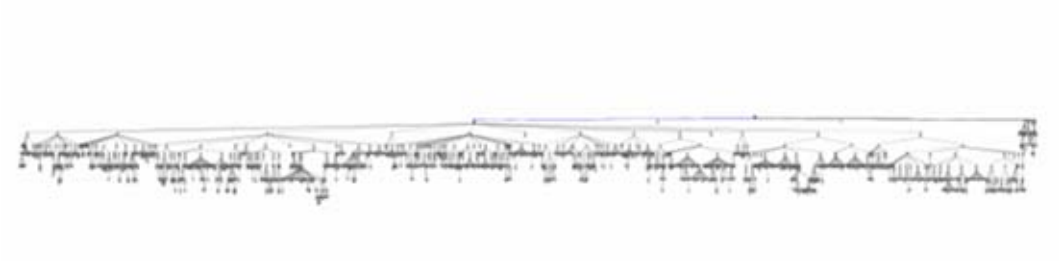


Figure 66 : Mauvaise lisibilité des grands arbres

3.7.2 Placement radial

Le placement radial [Eades92] est une variante en coordonnée radiale de l'algorithme vu précédemment, la racine est ici placée au centre du dessin, les nœuds de l'arbre sont placés sur des cercles concentriques dont le rayon indique la profondeur. Chaque sous-arbre dispose d'un secteur angulaire calculé afin d'offrir suffisamment de place pour toutes ses branches.

Figure 67 : Placement radial

Cette méthode est similaire à un placement de Reingold en coordonnées polaires. Elle est encore moins efficace sur les grands arbres larges. Si le nombre de nœuds situés à une même profondeur est grand, le rayon du cercle sur lequel ils doivent être placés doit être lui aussi suffisamment grand, ce qui conduit à deux choix. Soit la distance entre les cercles de deux niveaux consécutif est fixée, ce qui est une contrainte commune, et dans ce cas les nœuds d'un même niveau peuvent se chevaucher, soit le rayon de chaque niveau peut être ajusté afin d'éviter le chevauchement mais dans ce cas l'utilisation de l'espace risque d'être très médiocre.

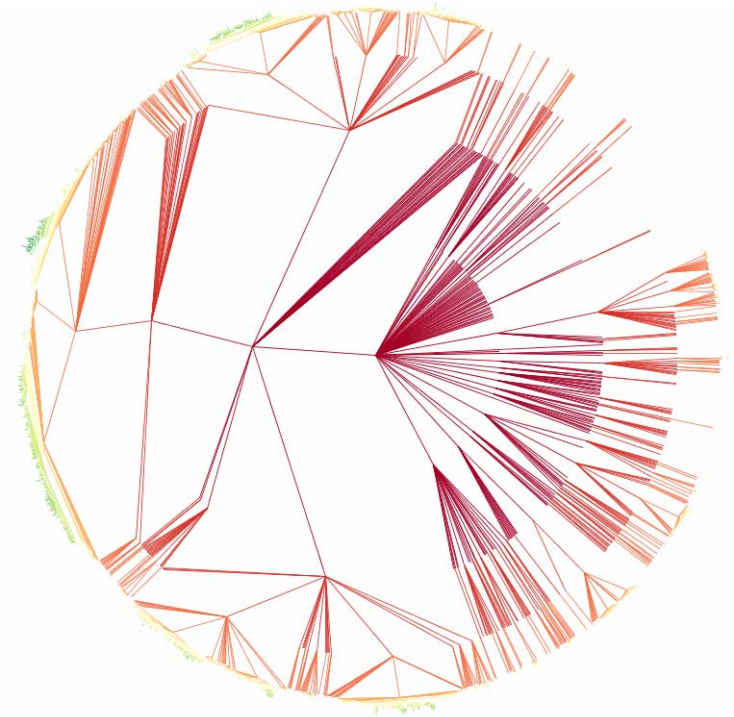


Figure 68 : Mauvaise utilisation de l'espace pour les grands arbres

3.7.3 Placement par modèle d'énergie

Les algorithmes de placement par modèles d'énergie, cf. 3.8.2, développés pour l'affichage des graphes généraux, donnent des résultats intéressants avec les arbres. La Figure 69 présente le même arbre que la Figure 68 mais placé par le modèle de Fruchterman-Reignold. Ils révèlent la structure des grands arbres plus efficacement que les techniques de placement spécifiques aux arbres. Par contre les temps de calcul ne sont pas comparables. Les algorithmes dédiés aux arbres ont une complexité linéaire au nombre de nœuds alors que la complexité des modèles de forces oscillent entre $\Theta(N \cdot \log N)$ et $\Theta(N^2)$.

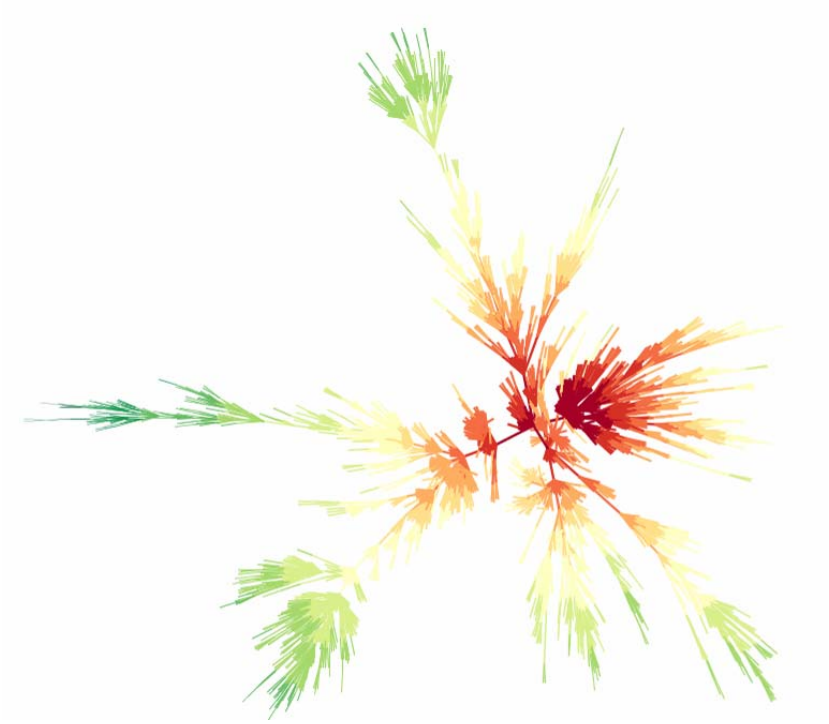


Figure 69 : Grand arbre placé par un modèle d'énergie

3.7.4 Treemaps

Abandonnons le temps de ce paragraphe les diagrammes *nœud-lien*, les treemaps n'en font pas partie. La relation hiérarchique est représentée ici par l'inclusion des régions les unes dans les autres, et non plus par un lien. Cette convention de représentation est la même que pour les diagrammes de Venn-Euler, mais ici il n'y a que des inclusions, les intersections partielles sont impossibles. Chaque nœud est symbolisé par un rectangle dont la surface permet de coder une information, comme la taille des fichiers et répertoires dans le cas d'un système de fichiers. L'avantage de cette technique est d'utiliser efficacement la surface d'affichage, c'est pourquoi on parle souvent de technique de « space-filling ».

L'algorithme original, "slice-and-dice" [Johnson91], consiste à séparer l'écran en rectangles dont la surface est donnée en entrée, en alternant entre découpage horizontal et vertical à mesure que l'on s'enfonce dans la hiérarchie. Cette alternance du découpage devait permettre de mieux identifier l'arborescence. Cette technique a toutefois un défaut important, les rectangles peuvent être très longs, très larges ou plutôt carrés. Et il est difficile de comparer la surface de deux rectangles de ratios largeur/longueur différents.

L'utilisation de régions plus « carrées », dont le ratio est proche de 1, peut résoudre ce problème de perception des aires. Les algorithmes "squarified" [Bruls00] et "cluster" réordonne les nœuds afin de pouvoir effectuer un pavage de régions les plus carrées possible. Si la lisibilité de chaque région est améliorée, deux défauts font jour. L'algorithme modifie l'ordre des nœuds. C'est un problème lorsque cet ordre est important. C'est aussi un problème si on souhaite modifier interactivement la surface des nœuds pour représenter une autre information. Dans ce cas les positions des nœuds peut changer radicalement d'un placement à l'autre ce qui empêche l'utilisateur de correctement mémoriser la structure de l'arbre.

Pour remédier à ces défauts les concepteurs des treemaps originaux proposent plusieurs nouvelles techniques de pavage qui conservent l'ordre des nœuds. Une étude avec des utilisateurs est réalisée sur trois types de pavage, "squarified", "ordered quantum", et "strip", et deux critères de qualité, le temps mis pour trouver un nœud particulier, et la préférence de l'utilisateur. Sur ces deux critères, l'algorithme "strip" obtient de loin les meilleurs résultats, suivi par "ordered quantum" et "squarified".

La critique majeure que l'on peut faire à ces techniques est la mauvaise lisibilité de la structure, la reconnaissance d'un chemin n'est pas évidente. Les labels posent problème car il n'y a tout simplement pas de place disponible pour en afficher sur un nœud qui n'est pas une feuille. Toute la place étant occupée par sa descendance. Par contre cette technique est vraiment très efficace pour la visualisation des attributs numériques associés aux nœuds, avec l'utilisation habile de la surface et de la couleur des nœuds. Cette application de visualisation des marchés boursiers en est un très bon exemple [Wattenberg98].

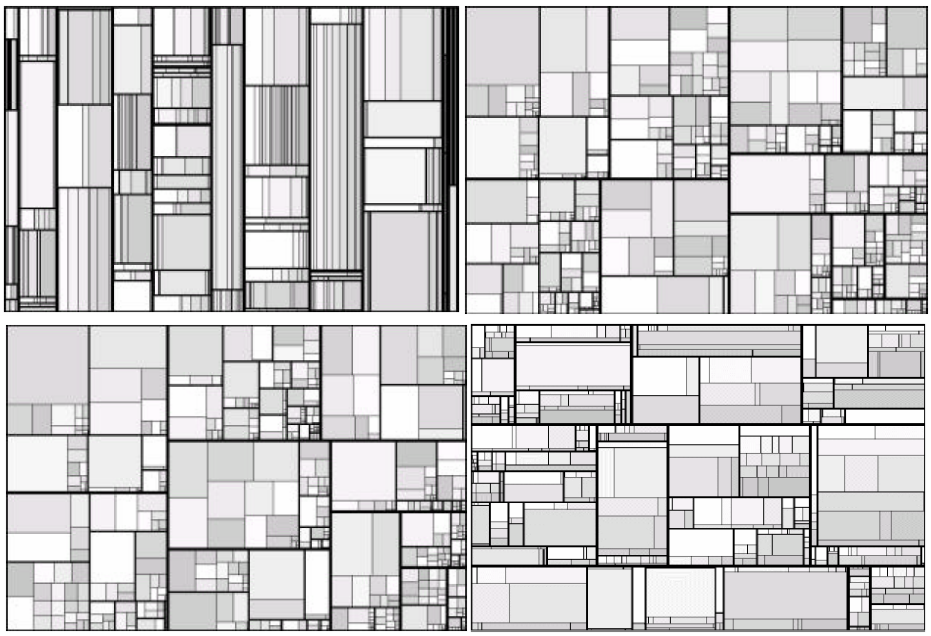


Figure 70 : slice-and-dice, cluster, squarified, et strip.

Plusieurs travaux ont été réalisés sur des treemaps utilisant des régions non rectangulaires. Une démonstration de Pad [Perlin93] [Bederson96] contenait une version circulaire des treemaps permettant de naviguer dans un site Internet. J'ai moi-même réalisé une telle version pour l'exploration de la thématisation des fonds de l'Ina [Thievre04]. Une société commercialise un logiciel de recherche sur Internet dont les résultats sont présentés dans une treemap circulaire [Grokker04]. Cette technique n'est plus vraiment du space-filling, car l'espace est utilisé beaucoup moins

efficacement qu'avec les treemaps rectangulaires. Par contre l'imbrication imparfaite de cercles dans un cercle rend l'identification des enfants et descendants d'un nœud bien plus facile.
Les Voronoi treemaps [Balzer05] sont comme le nom l'indique des treemaps composées de régions convexes. Les dessins générés sont très esthétiques, rappelant la complexité des structures organiques, mais il n'est pas évident que la lisibilité soit améliorée, aucune étude comparative ne permet pour l'instant de se prononcer.

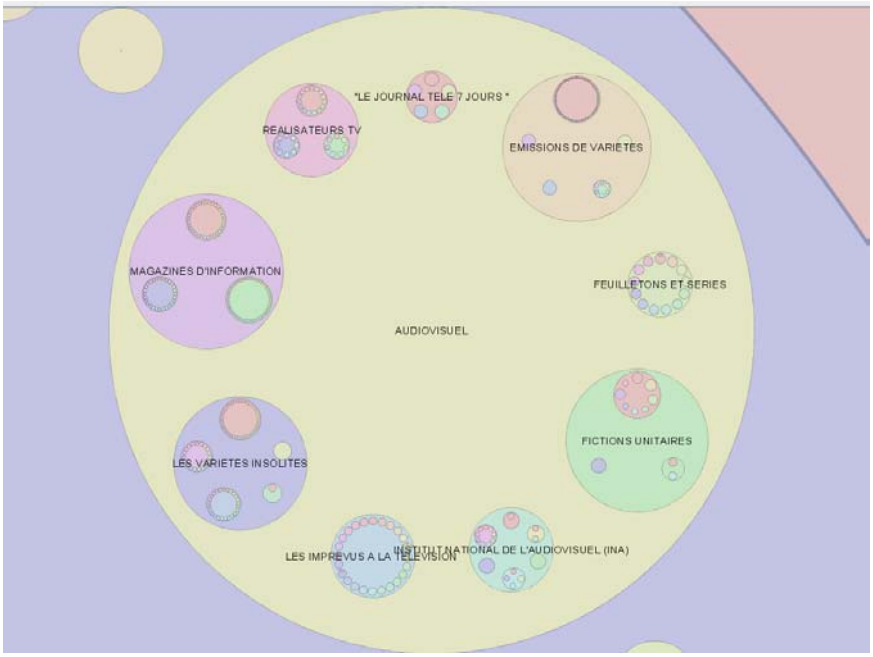


Figure 71 : Treemap circulaire

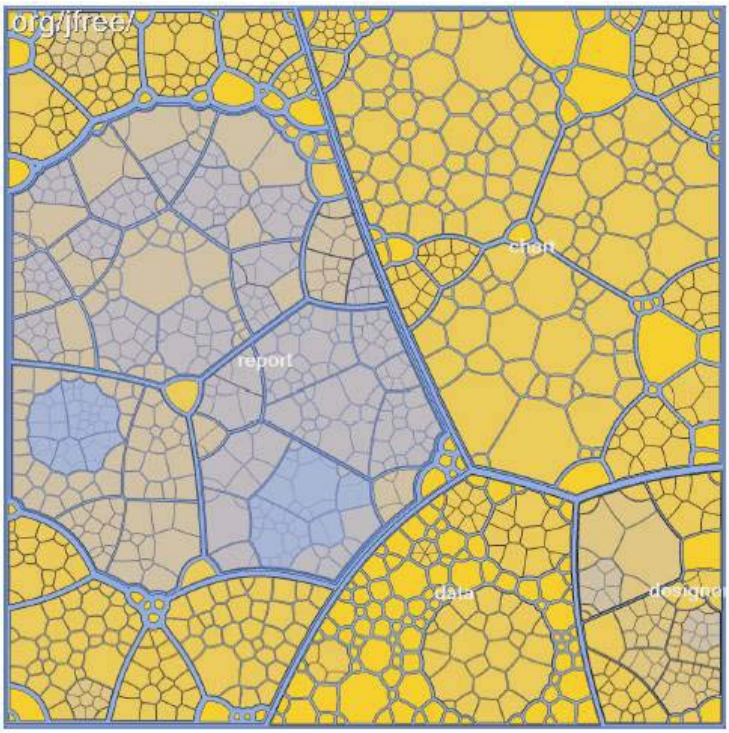


Figure 72 : Treemap Voronoi

3.8 Placements des graphes généraux

Les algorithmes de placement par modèle d'énergie ou de force sont les plus couramment utilisés pour la visualisation des graphes généraux. Lorsque les graphes sont modérément complexes, ils fournissent de bons résultats. Nous verrons que pour des graphes plus denses ce type d'algorithme peut être adapté pour l'identification de clusters.

Les algorithmes de placement d'arbre sont parfois utilisés pour la visualisation de graphes, dans ce cas le graphe est placé comme son arbre couvrant. C'est le cas du placement radial.

3.8.1 Placement radial : Bull Eye

Les visualisations de type bull-eyes permettent tout de même l'analyse structurelle d'un graphe. Dans le cas du placement radial avec focus, il permet l'évaluation exacte de la distance entre le focus et un autre sommet. Dans le cas du placement radial par centralité [Brandes04], il permet de repérer les sommets les plus « importants » du graphe. Les figures ci-dessous illustrent ces deux techniques de placements, le graphe de gauche est placé de manière radiale avec focus. Le niveau d'un sommet dans le dessin correspond à sa distance géodésique au focus. Le graphe de droite est placé sur une cible, aussi appelée bull-eye, la position d'un sommet est fonction de sa centralité.

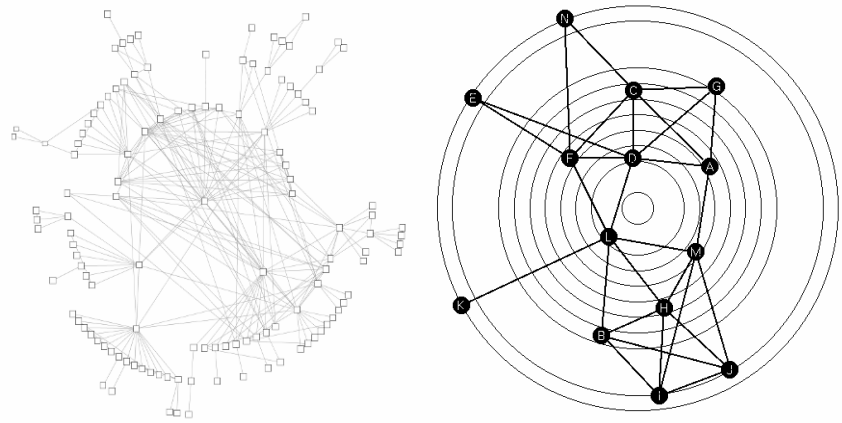


Figure 73 : radial avec focus et radial par centralités

3.8.2 Placements par modèle d'énergie

Ces techniques sont toutes basées sur l'analogie entre un graphe et un système masse-ressort. Chaque sommet possède une masse, et chaque arête est considérée comme un ressort. Un ressort est caractérisé par sa longueur au repos, et son coefficient de raideur. Les sommets se repoussent les uns des autres comme des particules électriques, et chaque arête contraint ses deux sommets à rester à une distance proche de la longueur du ressort au repos. Trouver un placement pour le graphe revient à trouver une position d'équilibre du système masse-ressort, c'est-à-dire à trouver une configuration où l'énergie du système est minimale.

Les différents algorithmes procèdent de manière similaire. On détermine une position initiale pour les sommets et à chaque itération on recalcule leur position jusqu'à tendre vers un état de stabilité. Les différences entre les algorithmes dépendent du choix des forces de répulsions et d'attractions, et de la manière dont on minimise l'énergie du système.

Ces techniques diffèrent par le modèle d'énergie ou de force considéré. Force et énergie sont deux notions qui permettent de représenter les mêmes classes de problèmes. L'énergie d'un système masse-ressort dépend des forces qui s'exercent. Plus précisément, elle est égale à la somme des travaux des forces qui s'exercent sur chaque masse :

$$E = \sum_{v \in V} W(v), \text{ avec } W(v) = \int \overrightarrow{F_v}.\overrightarrow{dx_v}$$

et $\overrightarrow{F_v}$ la somme des forces qui s'exercent sur le sommet v pendant son déplacement infinitésimale $\overrightarrow{dx_v}$.

Chaque sommet est lié à ses voisins par des arêtes considérées comme élastiques (comme un ressort), et repoussé par l'ensemble des sommets du graphe. On considère alors qu'il y a une force de tension entre deux nœuds voisins et une force de répulsion entre tout couple de nœuds.

On peut définir deux fonctions T et F qui simulent respectivement la tension de l'arête entre deux voisins et la répulsion entre deux sommets quelconques.

$$\mathrm{T} : E \to \mathbb{R}^2 \text{ et } \mathrm{F} : V \times V \to \mathbb{R}^2$$

Si on considère un modèle de force naturel, dans le sens où la force exercée par une arête suit la loi de Hook, et deux sommets se repoussent comme deux particules électriques le feraient, on obtient les formules suivantes :

$$\vec{\mathrm{T}}(u,v) = k.(|uv| - \ell_0).\frac{\overrightarrow{uv}}{|uv|},\ (u,v) \in E \text{ et } \vec{\mathrm{F}}(u,v) = g.m_u.m_v.\frac{\overrightarrow{uv}}{|uv|^3},(u,v) \in V \times V$$

k est le coefficient de raideur de l'arête et ℓ_0 sa longueur au repos. g est la constante de gravitation, négative pour une force de répulsion et m_u et m_v les masses ou charges respectives de u et v. Ces deux fonctions dépendent de la distance entre u et v, comme pour les autres modèles de forces. C'est pourquoi on peut simplifier l'expression des forces en considérant les fonctions $\mathrm{t}(x)$ et $\mathrm{f}(x)$, $\mathrm{t} : \mathbb{R} \to \mathbb{R}$ et $\mathrm{f} : \mathbb{R} \to \mathbb{R}$ pour respectivement la tension d'une arête de longueur x et la répulsion entre deux sommets à distance x.

$$t(x) = k.(x - \ell_0) \text{ et } f(x) = \frac{g.m_u.m_v}{x^2}$$

3.8.2.1 Eades

Eades [Eades84] fut le premier à appliquer ces techniques pour la visualisation de graphes. Mais son implémentation a le défaut de commettre trop d'approximations, en particulier de ne pas respecter la loi de Hook pour le calcul des tensions des ressorts, et d'ignorer les interactions entre les sommets qui ne sont pas liés par une arête. Cette dernière approximation permettait d'ailleurs de simplifier grandement la complexité du calcul. Car la complexité est le défaut majeur de ces algorithmes, comme chaque sommet exerce une force de répulsion sur tous les autres, la complexité est de l'ordre du carré

du nombre de nœuds, notée $\Theta(N^2)$. En ne considérant que la force qui existe entre deux sommets liés par une arête, la complexité devient proportionnelle au nombre d'arêtes, notée $\Theta(M)$. Mais en faisant cette approximation, on autorise des nœuds dont la distance théorique dans le graphe est grande à être placés côte à côte.

$$\mathrm{t}(x) = \begin{cases} k.\log(1 + x - \ell_0) \text{ pour } x \geq \ell_0 \\ -k.\log(1 + \ell_0 - x) \text{ pour } x \leq \ell_0 \end{cases} \text{ et } \mathrm{f}(x) = \frac{g.m_u.m_v}{x^2}$$

3.8.2.2 Kamada-Kawai

Kamada et Kawai [Kamada89] reprennent le principe en respectent la loi de Hook. Ils ne considèrent pas directement de force de répulsion, mais estime qu'entre chaque paire de sommets il y a un ressort dont la longueur est proportionnelle à leur distance théorique dans le graphe. La complexité est donc en $\Theta(N^2)$.

$$|\vec{\mathrm{T}}(u,v)| = \mathrm{t}(x), \forall (u,v) \in V^2.$$
$$\mathrm{t}(x) = k.(x - \delta(u,v)), \text{ avec } \delta(u,v) \text{ la distance géodésique entre u et v.}$$

3.8.2.3 Davidson-Harel

Davidson et Harel [Davidson89] utilisent le recuit simulé (simulated annealing) comme algorithme de minimisation. Cette technique générale d'optimisation consiste à minimiser les valeurs d'une fonction d'énergie. Cette fonction est définie afin de prendre en compte de nouvelles contraintes de placement, comme le nombre de croisements. Cette technique est toutefois pénalisée par sa lenteur.

$$\mathrm{t}(x) = k.(x - \ell_0) \text{ et } \mathrm{f}(x) = \frac{g.m_u.m_v}{x^3}$$

L'originalité du modèle d'énergie de Davidson est de gérer une composante qui pénalise le croisement de deux arêtes. Cette composante est ignorée dans la formule ci-dessus car elle ne peut pas être exprimée comme une fonction de la distance entre deux sommets. L'algorithme donne cependant de meilleurs résultats quand cette composante est désactivée. De manière générale, il est toujours possible d'ajouter des contraintes à un modèle d'énergie, ils sont très flexibles, cependant la majorité des expérimentations qui ont été entreprises en ce sens n'ont pas donné de résultats satisfaisants. L'augmentation du nombre de contraintes abaisse la qualité des placements obtenus, surtout quand les contraintes sont mutuellement destructrices, comme c'est le cas pour l'uniformisation des longueurs d'arêtes et la minimisation des croisements.

3.8.2.4 Ftruchterman-Reingold

Fruchterman et Reingold [Fruchterman91] émettent les deux principes qui régissent leur méthode :

- La distance géométrique entre deux sommets liés par une arête doit être petite,

- La distance géométrique entre deux sommets non liés par une arête ne doit pas être trop petite.

Le deuxième principe les conduit naturellement à introduire une force de répulsion entre les sommets. Ils considèrent une force de répulsion inversement proportionnelle au carré de la distance qui sépare les deux sommets. La force de répulsion entre deux sommets lointains est très faible, et peut être négligée. Cela permet d'optimiser l'algorithme en s'approchant d'une complexité en $\Theta(N\log(N))$ [Barnes86]. Pour ce faire, il est nécessaire d'indexer spatialement les sommets du graphe afin d'identifier à moindre coût les sommets qui se trouvent dans le voisinage d'un sommet particulier. Cet index est une grille qui découpe le plan en cellule. Chaque sommet est assigné à une cellule de la grille, pour trouver les voisins d'un sommet, il suffit de visiter les cases voisines. A chaque changement de position d'un sommet, il doit être réassigné à sa nouvelle case.

$$\mathrm{t}(x)=\frac{x^2}{\ell_0} \text{ et } \mathrm{f}(x)=\frac{-{\ell_0}^2}{x}$$

3.8.2.5 Evaluation des modèles d'énergie standards

L'objectif de ces différents modèles est d'obtenir un placement où la distance entre deux nœuds voisins est proche de la longueur au repos de leurs arêtes. Si on considère le cas très simple d'un graphe de deux sommets reliés par une arête de longueur idéale ℓ_0, il faut que la force d'attraction et la force de répulsion s'annule lorsque la distance entre les deux nœuds est ℓ_0. C'est le cas de tous ces modèles lorsque l'on choisit bien les valeurs des différentes constantes. On peut le vérifier sur le modèle de Fruchterman :

$$\mathrm{t}(\ell_0)+\mathrm{f}(\ell_0)=\frac{{\ell_0}^2}{\ell_0}-\frac{{\ell_0}^2}{\ell_0}=0$$

Les modèles de forces relatés ici semblent tous donner des résultats satisfaisants sur des graphes de petite taille (|V|<1000) et dont le rapport nombre d'arêtes sur nombre de sommets est faible (<5). La qualité d'un placement peut être mesuré selon plusieurs critères. On peut par exemple comparer les distances géométriques entre sommets par rapport aux distances idéales en calculant la déviation moyenne des longueurs des arêtes.

$$\mathrm{EdgeDev}\big(G(V,E)\big)=\frac{1}{|E|}\sum_{e\in E}|\,l(e)-\ell_0(e)\,|$$

Indépendamment de ces critères numériques, on peut vouloir évaluer la lisibilité d'un graphe. L'évaluation réalisée auprès d'utilisateurs par Purchase [Purchase97] a permis d'identifier des critères de lisibilité. Il en résulte que le placement d'un graphe est plus lisible principalement lorsque le nombre de croisements d'arêtes est faible, et lorsque le tracé des arêtes est simple. La symétrie semble être importante même si l'expérience n'est pas concluante pour ce critère. Cette expérimentation a été

réalisée sur des graphes très simples de quelques dizaines de nœuds, et peu denses. Cela rend ces critères difficilement applicables à des graphes plus complexes.

3.8.2.6 Noack : méthodes de clustering visuel

Node LinLog

Un nouveau modèle d'énergie a été défini récemment pour obtenir un placement qui révèle le clustering d'un graphe [Noack03]. Ce modèle définit l'énergie d'attraction et de répulsion comme :

$$E_t(x) = a \cdot x \text{ et } E_f(x) = b \cdot \ln(x)$$

L'énergie d'une arête est linéaire à sa longueur, et l'énergie de répulsion entre deux sommets est en logarithme de leur distance, d'où le nom du modèle, LinLog. Si on exprime ce modèle en terme de force on obtient :

$$\mathrm{t}(x) = k \text{ et } \mathrm{f}(x) = \frac{g.m_u.m_v}{x}$$

Les modèles de forces classiques tendent à uniformiser les longueurs d'arêtes, avec le modèle Node LinLog l'objectif est d'obtenir que la distance entre deux groupes de sommets soit liée à leur couplage. La mesure de couplage, connu sous le nom *de node-normalized cut* est définie en 2.2.2.2. Je rappelle tout de même ici les définitions mathématiques.

C_1 et C_2 étant deux clusters d'un même graphe $G = (V,E)$, le $cut(C_1,C_2)$ est le nombre d'arêtes entre C_1 et C_2 :

$$cut(C_1, C_2) = \left|\left\{(v_1, v_2) \in E \,\middle|\, v_1 \in V_1, v_2 \in V_2\right\}\right|$$

Le *cut* normalisé par rapport aux nombres de sommets est fréquemment utilisé comme mesure du couplage entre deux clusters. Cette mesure revient à calculer le rapport du nombre observé d'arêtes inter-clusters sur le nombre potentiel d'arêtes inter-clusters.

$$nodeNormalizedCut(C_1, C_2) = \frac{cut(C_1, C_2)}{|V_1| \cdot |V_2|}$$

L'auteur démontre que le modèle LinLog assure un placement où la distance entre deux clusters est inversement proportionnelle à leur couplage. Soit :

$$dist(C_1, C_2) = \frac{k}{nodeNormalizedCut(C_1, C_2)}$$

Ce modèle est adapté à la visualisation de graphes denses possédant un fort indice de clustering. De manière empirique, on constate que pour obtenir des placements pertinents avec ce modèle, il est nécessaire que la densité intra-cluster soit grande devant le couplage ou densité inter-cluster. La section 3.8.2.8 fournit des précisions sur ces ordres de grandeurs.

Edge LinLog

Dans un second article [Noack05], l'auteur revient sur le choix du *node-normalized cut* comme mesure de couplage. Il montre que cette mesure est biaisée lorsque les tailles de clusters sont significativement différentes. Il affirme d'ailleurs qu'il en est de même pour les mesures de *conductance* et *d'expansion*, qui sont elles aussi couramment utilisées comme mesure de couplage. Son choix se porte sur le *edge-normalizerd cut* qui élimine ce biais et constitue ainsi une mesure plus robuste :

$$edgeNormalizedCut(C_1, C_2) = \frac{cut(C_1, C_2)}{deg(V_1) \cdot deg(V_2)}$$

avec $deg(V)$ la somme des degrés des sommets dans V.

Ce changement de mesure de couplage a pour effet de modifier la balance entre répulsion et attraction. Cela conduit notamment à une meilleure séparation des clusters de petites tailles. En effet, la présence de petits clusters étaient souvent masquée par un cluster voisin beaucoup plus grand. Ce cluster exerçait alors une répulsion tellement grande sur ses petits voisins qu'ils se trouvaient totalement dispersés.

L'expression des forces pour le modèle Edge LinLog devient :

$$\mathrm{t}(u,v) = k \text{ et } \mathrm{f}(u,v) = \deg(u) \cdot \deg(v) \cdot \frac{g.m_u.m_v}{|\overrightarrow{uv}|}$$

3.8.2.7 Lisibilité des placements par modèles d'énergie

On peur différencier les modèles classiques des modèle de clustering visuel. Les modèles classiques ont un comportement globalement similaire, les différences sur les placements obtenus restent locales comme on peut le constater sur quelques exemples.

Figure 74 : Modèle Naturel – arbre et graphe

Figure 75 : Modèle d'Eades – arbre et graphe

Figure 76 : Modèle de Fruchterman-Reingold – arbre et graphe

Pour des graphes de faible densité les modèles classiques fonctionnent bien, ils produisent des représentations qui apportent véritablement de l'information sur la structure de ces graphes. On constate qu'avec le modèle de Eades les placements présentent une grande homogénéité au niveau des longueurs d'arêtes, contrairement au modèle FR.

Pour les graphes denses toutefois les résultats sont très médiocres. On peut le constater sur la Figure 77, le même graphe est placé par le modèle de Fruchterman-Reingold et par le modèle de

Noack. Ce graphe dense est composé de deux clusters qui eux-mêmes sont constitués de quatre clusters. Les modèles classiques sont incapables de produire un placement qui permet de conclure à l'existence de ces clusters.

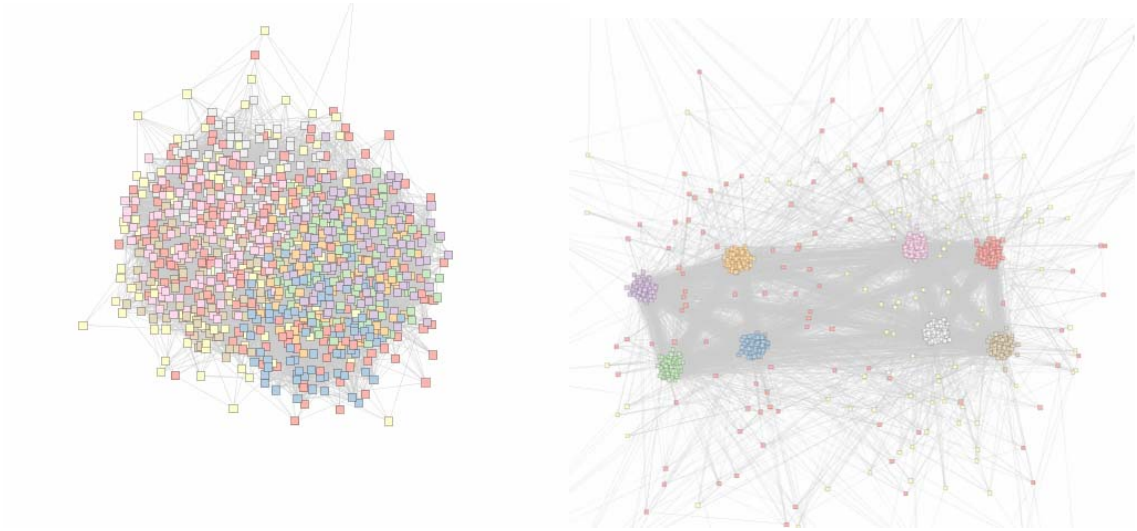


Figure 77 : graphe dense composé de clusters – FR et Node LinLog

Par contre le modèle de Noack ne donne aucun résultat lorsque le graphe est peu dense. Pour un arbre, il produit par exemple des résultats aussi médiocres qu'un placement aléatoire. C'est le comportement normal de ce modèle, où la distance entre sommets est inversement corrélée à leur couplage. Dans un arbre, la densité globale et les densités locales (indices de clustering) sont presque nulles, donc les distances sont très grandes.

3.8.2.8 Analyse des placements obtenus par le modèle Edge LinLog

Nous souhaitons mieux comprendre les propriétés du modèle de Force Edge LinLog. L'auteur affirme que la distance entre deux clusters C_1 et C_2 d'un graphe est inversement proportionnel à leur mesure de *edge-normalized cut*. Je rappelle que cette mesure permet de quantifier le couplage ou densité inter-clusters entre deux clusters.

Tout d'abord, pourquoi ce comportement est-il intéressant ? Lorsqu'un graphe est globalement dense, son diamètre moyen est faible par rapport au nombre de sommets. Les distances géodésiques entre sommets sont faibles. Dans ce cas, la distance n'est plus une métrique significative, c'est la raison pour laquelle les modèles classiques produisent des placements très compactes comme sur l'exemple de la Figure 77. Par contre, même si un graphe est globalement dense, les densités locales ne sont pas homogènes dans tout le graphe. Un placement qui révèle les groupes de densités différentes est donc une solution intéressante. Dans un tel placement il est intuitif d'avoir une corrélation inverse entre les distances géométriques et les densités locales. C'est bien ce que propose les modèles LinLog, et plus particulièrement Edge LinLog, qui génèrent des placements où les distances géométriques sont inversement proportionnelles aux densités locales.

L'auteur donne une preuve simplifiée du comportement du modèle Edge LinLog. Il assure que la distance entre deux clusters est inversement proportionnelle à leur couplage. Cette propriété nous semble toutefois incomplète car elle ne prend en compte que le couplage ou densité inter-cluster. La densité intra-cluster joue forcément aussi un rôle et nous souhaitons comprendre lequel. Nous

proposons donc une petite expérimentation afin de déterminer plus précisément les propriétés du placement Edge LinLog.

Construction des graphes aléatoires

Afin de pouvoir analyser le comportement de l'algorithme de placement, nous devons être capable de construire des graphes dont nous maîtrisons la structure. Nous proposons une adaptation du modèle de graphe aléatoire [Erdös59] afin de générer des graphes aléatoires composés de clusters de taille et de densité paramétrables.

Rappelons qu'un graphe aléatoire est caractérisé par deux paramètres, sa taille et sa densité. Notre modèle de graphes clusterisés aléatoires permet lui aussi de spécifier le graphe et chacun de ses clusters par ces deux paramètres. Dans ce modèle le graphe peut être composé de clusters, et chaque cluster peut aussi contenir des sous-clusters. Cette structure multi-échelle est décrite par une hiérarchie de clusters. La Figure 78 montre la fenêtre de construction des graphes aléatoires. Dans cet exemple nous créons un graphe de 500 sommets et de densité globale 0.0002. Ce graphe est composé de 4 clusters de taille 100 et de densité 0.06. Deux de ces clusters sont eux-mêmes constitués de trois sous-clusters de taille 20 et de densité 0.6. Sur la Figure 79, ce graphe est placé avec la méthode Edge LinLog. Un codage couleur est utilisé pour visualiser les différents clusters.

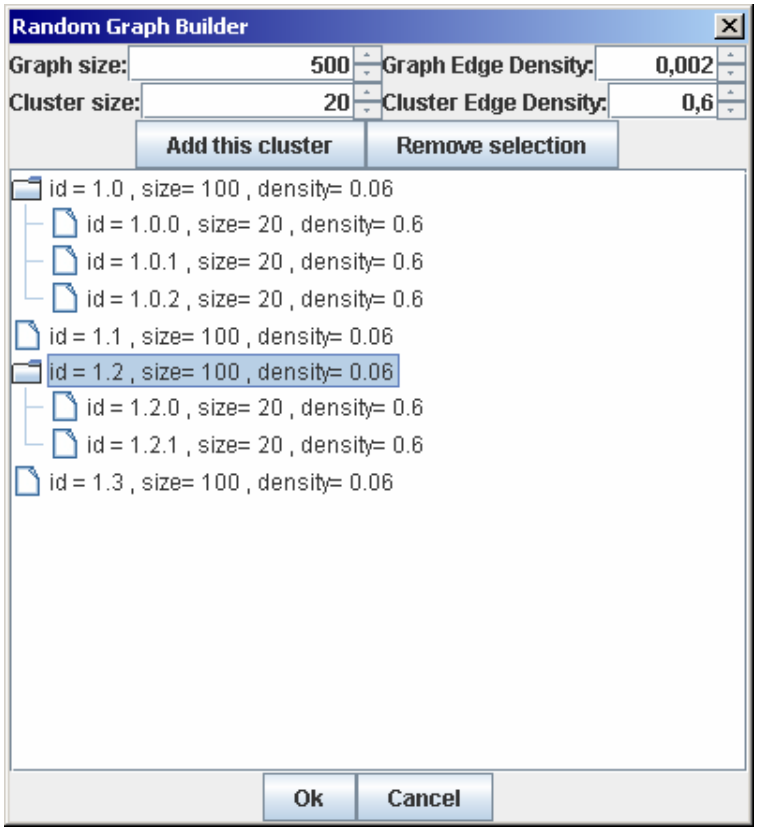


Figure 78 : création d'une hiérarchie de clusters

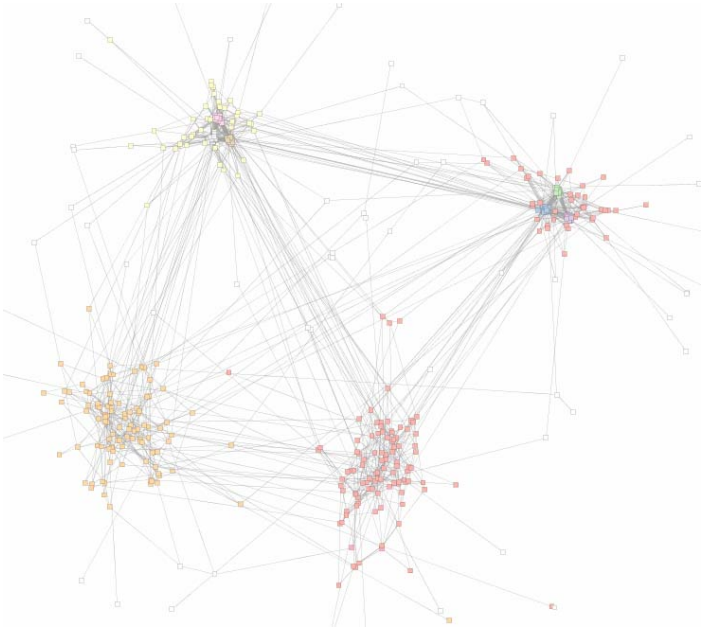


Figure 79 : affichage d'un graphe clusterisé aléatoire avec Edge LinLog

Ce modèle est une simple adaptation du modèle de graphe aléatoire, mais il nous permet de générer des graphes dont nous connaissons précisément les structures globale et locales. C'est avec ce modèle que nous avons réalisé les deux expériences suivantes.

Expérience 1 : Mesure de corrélation entre densités et longueurs d'arêtes

Dans cette expérience nous souhaitons mieux comprendre la relation entre les distances obtenues dans le placement et les densités locales. Nous construisons un graphe très simple pour cette expérience. Ce graphe est constitué de deux clusters de 100 sommets. Nous faisons varier deux paramètres, la densité intra-cluster et la densité inter-cluster. Pour chaque mesure nous plaçons le graphe avec le modèle Edge LinLog puis nous calculons la longueur moyenne des arêtes intra et inter cluster.

Nous constatons dans un premier temps que pour une densité intra-cluster fixée, la longueur moyenne des arêtes inter-cluster est inversement proportionnelle à la densité inter-cluster ou couplage. De même, pour une densité inter-cluster fixée, la longueur moyenne des arêtes intra-cluster est inversement proportionnelle à la densité intra-cluster. Ces résultats confirme donc les propriétés décrites dans l'article de Noack [Noack03].

Nous remarquons aussi que la densité intra-cluster influe sur la distance inter-cluster, et que la densité inter-cluster influe sur la distance intra-cluster. Cela suggère une corrélation plus complexe entre ces variables.

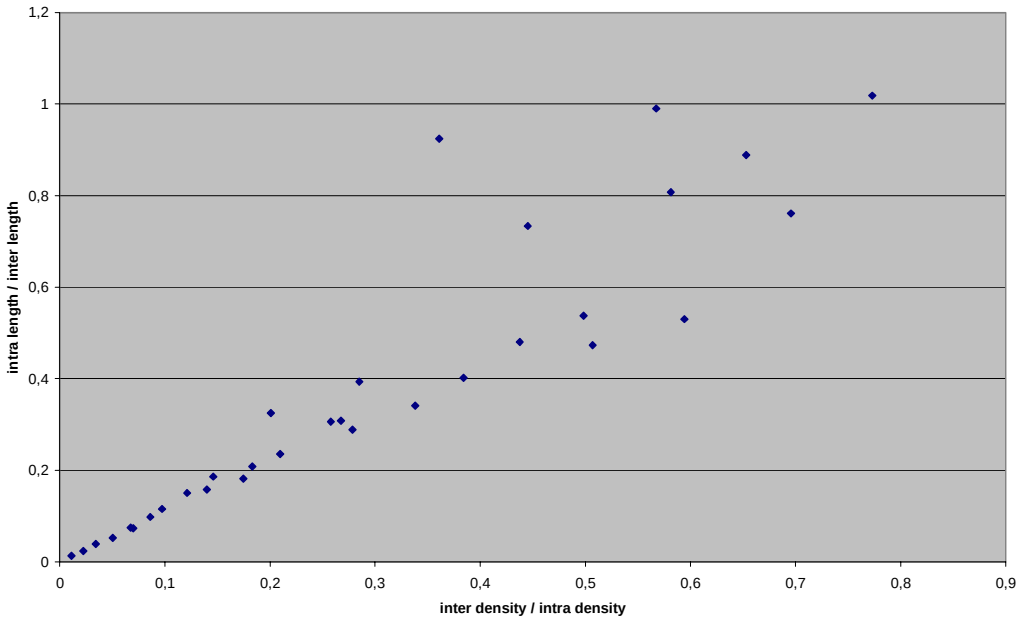


Figure 80 : corrélation entre densité et distance

On observe une corrélation linéaire (cf. Figure 80) entre rapport intra/inter des longueurs d'arêtes et le rapport inter/intra des densités. Le rapport intra/inter des longueurs est inversement proportionnel au rapport intra/inter des densités.

On peut conclure que cette expérience confirme les caractéristiques attendues du modèle Edge LinLog, et permet de comprendre son fonctionnement de manière plus précise. La corrélation observée entre distance et densité est une généralisation de la relation avancée jusqu'ici par l'auteur.

Relativisons tout de même ces résultats, le graphe choisi pour l'expérience est très simple, et il est vraisemblable que nous n'obtiendrions pas des résultats d'aussi bonne qualité pour des graphes dont la structure est plus complexe. Que se passe-t-il lorsque les clusters sont plus nombreux et de taille différentes, par exemple. Nous pouvons penser que les caractéristiques du placement s'éloignent un peu de cet idéal.

Expérience 2 : taille des clusters et distances

Dans cette seconde expérience nous souhaitons justement mesurer le rôle de la taille de clusters à densité égale. On considère donc cinq graphes aléatoires de densité fixée à d=0.05 et de taille 100, 200, 300, 400 et 500. Pour chacun de ces 5 graphes, nous effectuons un placement avec le modèle Edge LinLog puis nous calculons la longueur moyenne et l'écart-type de leurs arêtes. Nous obtenons les résultats du Tableau 6.

Taille des clusters	Longueur moyenne	Ecart-type longueur
100	126,43	116,56
200	125,97	95,74
300	125,37	84,11
400	125,61	80,84
500	125,37	77,85

Tableau 6 : relation entre la taille des clusters et les longueurs d'arêtes intra-clusters

On constate logiquement que la longueur moyenne des arêtes intra-cluster ne dépend pas de la taille du cluster. Par contre, plus la taille des clusters augmente plus la distribution des longueurs est homogène.

Sur la Figure 81 sont illustrés les placements obtenus pour les 5 graphes de l'expérience. Plus on augmente le nombre de sommets, plus le plongement du cluster est compact. Cela s'explique par la meilleure homogénéité des longueurs d'arêtes dans le cas des clusters de grande taille, mais aussi par l'augmentation quadratique du nombre d'arêtes par rapport au nombre de sommets. En effet, à densité égale un graphe de kN sommets contient près de k^2 fois plus d'arêtes qu'un graphe de N sommets.

Figure 81 : placement Edge LinLog pour des clusters de tailles respectives : 100, 200, 300, 400 et 500 sommets

4 Grapho : API et prototype de visualisation de graphes

4.1 Introductsion

Nous avons souhaité mettre en pratique les méthodes d'analyse et de visualisation de graphes présentées dans les chapitres 2 et 3. Nous avons donc développé Grapho, une API de visualisation de graphes intégrant la plupart des techniques présentées. Cette API propose donc les fonctions standard de visualisation de graphes. Chaque élément d'un graphe peut posséder des attributs qui peuvent être visualisés. Plusieurs algorithmes de placements sont disponibles dont les placements de Fruchterman-Reingold et de Novack. L'API supporte le filtrage massif des sommets et arêtes, le filtrage peut être basé sur les attributs des entités ou sur la structure du graphe. Plusieurs métriques peuvent être calculées, comme les centralités ou l'indice de clustering. La couleur et la taille des sommets et arêtes peuvent être paramétrées pour représenter des attributs de données. Ces paramétrages ou codages permettent d'obtenir des points de vue différents sur un même graphe.

Un prototype de visualisation a été réalisé avec l'API, il donne accès à toutes les fonctionnalités actuellement implémentées dans l'API. Il offre deux vues d'un graphe, une vue tabulaire des sommets et arêtes particulièrement utile pour l'analyse des attributs et le filtrage, et une vue diagramme *nœud-lien* destinée à présenter la topologie du réseau.

Avant de présenter en détail les divers aspects de l'API Grapho, nous allons passer en revue plusieurs API de visualisation de graphes existantes.

4.2 API de visualisation de graphes

4.2.1 Pajek

Pajek [Batagelj98] est un outil d'analyse de graphe. Bien qu'il propose des visualisations de graphe sous forme de diagramme *nœud-lien*, on ne peut pas parler de visualisations interactives. Les diagrammes sont très peu interactifs, et les fonctionnalités, bien que nombreuses, sont uniquement accessibles par le biais de fenêtres de dialogue. Pajek souffre de son âge, l'affichage est de mauvaise qualité et l'utilisation des différents fonctionnalités de clustering et de layout sont très peu intuitives. L'interface se bloque systématiquement durant un calcul.

4.2.2 Tulip

Tulip [Auber03] est plus récent. C'est une API de visualisation de graphe de haute performance écrite en C++. Tulip a été conçu afin de pouvoir visualiser des graphes très volumineux, de plus d'un million d'entités. La structure de graphe utilisée dans Tulip a été spécialement conçue pour limiter l'occupation mémoire. Tulip permet aussi de créer plusieurs vues d'un même graphe, et cela sans démultiplier l'occupation mémoire. Ce que l'on peut reprocher à Tulip, c'est finalement sa structure qui se révèle assez complexe à utiliser, et le manque d'interactivité des applications réalisées avec l'API.

4.2.3 InfoVis Toolkit

Infovis Toolkit [Fekete04] est une API de visualisation d'information en Java. Elle permet de construire de nombreuses visualisations différentes comme des treemaps, des diagrammes *nœud-lien*,

des tables et matrices. Les différents types de données correspondant à ces visualisations sont unifiés dans une seule structure, la table. Cette unification des données permet d'obtenir facilement des vues différentes d'une même donnée. Le toolkit propose de plus des fonctions d'interactions évoluées comme le fisheye. Le codage graphique des données est paramétrable grâce à des composants graphiques spécialisés. Cette API est généraliste, on ne trouve donc pas de fonctionnalité spécifiques aux graphes comme les calcul de métriques ou les méthodes de clustering.

4.2.4 Prefuse

Prefuse [Heer05] est aussi une API Java de visualisation d'information. Très bien conçue, cette API est particulièrement facile à utiliser. Tout comme l'Infovis Toolkit, elle se veut généraliste en proposant de nombreux types de visualisations, comme les treemaps et les diagrammes *nœud-lien*. L'affichage est de très bonne qualité et l'animation des processus est particulièrement bien conçue. Les transitions entre différents placements ou filtrages peuvent être animés afin de préserver la continuité de la représentation. Ces animations sont du plus bel effet. Enfin, l'API propose une très bonne implémentation d'algorithme de placement par modèle de forces. L'algorithme repose sur l'indexation spatiale des sommets grâce à l'utilisation de quadtree, ce qui permet d'obtenir une complexité en $\Theta(N \cdot \log N)$. Le modèle de force de l'algorithme est paramétrable, et il est possible de créer par héritage ses propres modèles. Par contre, il n'y a pas de calculs de métriques spécifiques aux graphes et pas de méthodes de clustering.

4.2.5 Guess

Guess [Adar06] est le successeur de Zoomgraph. C'est une API d'exploration de graphes. La structure de graphe provient de JUNG qui est une API d'analyse de graphes. JUNG propose la quasi-totalité des calculs de métriques existantes pour les graphes. L'affichage de Guess par défaut est implémenté en Piccolo. L'originalité de ce projet est de pouvoir piloter la visualisation grâce à un langage de script dérivé de Python. Ce langage permet d'accéder à toutes les fonctionnalités de l'API, comme le filtrage, le codage graphique ou encore le placement utilisé. Il permet aussi de programmer de nouvelles fonctionnalités.

Le langage de script est une bonne idée, cela permet effectivement à des utilisateurs expérimentés d'effectuer des actions complexes. Les fonctions les plus utilisées devraient cependant être plus accessibles, les sliders et autres éléments interactifs ont tout de même été conçus pour remplacer l'usage systématique de la console. De plus on peut émettre quelques doutes quant à la faisabilité de calculs complexes avec le langage de script, particulièrement au niveau des performances. Il y a d'ailleurs un souci de performance lors des calculs, l'affichage se bloque systématiquement, de plus les différents algorithmes (placement et métrique) semblent particulièrement lents.

4.3 Format de stockage : GraphML

GraphML est un format de stockage de graphes en XML. Il est issu de la communauté du dessin de graphe et a été proposé à l'occasion de l'édition 2001 de la conférence Graph Drawing. C'est un format de stockage très facile à utiliser, il est lisible et intuitif. Il permet de décrire la structure de graphes orientés ou non, d'hypergraphes et de graphes hiérarchiques. Il est facilement extensible, ce qui permet d'ajouter des informations spécifiques à une application, comme des attributs typés pour les sommets et arêtes. Il profite de plus de l'existence de parsers XML dans tous les langages de programmation, ce qui le rend facile à lire et à écrire.

Nous allons voir plus en détail ce langage, mais commençons par en présenter les bases avec un exemple simple.

4.3.1 Exemple minimaliste

La Figure 82 représente un graphe très simple. Les sommets sont identifiés par une chaîne de caractères. La Figure 83 représente le fichier GraphML du même graphe.

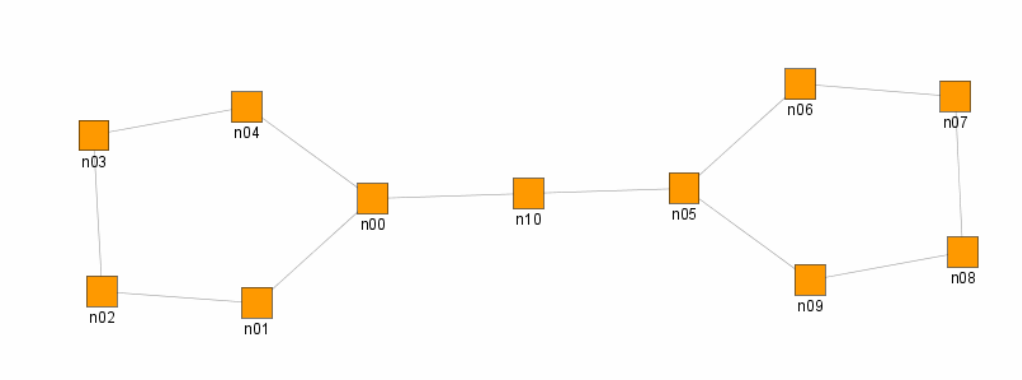


Figure 82 : un graphe simple

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
 <graph edgedefault="undirected">
  <node id="n00"/>
  <node id="n01"/>
  <node id="n02"/>
  <node id="n03"/>
  <node id="n04"/>
  <node id="n05"/>
  <node id="n06"/>
  <node id="n07"/>
  <node id="n08"/>
  <node id="n09"/>
  <node id="n10"/>
  <edge source="n00" target="n01"/>
  <edge source="n01" target="n02"/>
  <edge source="n02" target="n03"/>
  <edge source="n03" target="n04"/>
  <edge source="n04" target="n00"/>
  <edge source="n05" target="n06"/>
  <edge source="n06" target="n07"/>
  <edge source="n07" target="n08"/>
  <edge source="n08" target="n09"/>
  <edge source="n09" target="n05"/>
  <edge source="n00" target="n10"/>
  <edge source="n05" target="n10"/>
 </graph>
</graphml>
```

Figure 83 : fichier GraphML pour un graphe simple

Le premier élément signifie que ce fichier est un document XML 1.0 encodé en UTF-8. Cette déclaration est commune à tous fichiers XML. Le second élément est la racine du document GraphML, elle contient l'intégralité de la définition du graphe exemple. Elle signifie que le document est un document GraphML et spécifie le namespace par défaut du document. La notion de namespace est encore une fois commune à tous documents XML.

Les éléments suivants décrivent le graphe. Un graphe est caractérisé par ses sommets et arêtes, en GraphML il est représenté par un élément **graph** qui contient des éléments **node** et **edge**. L'élément **graph** possède un attribut **edgedefault** qui spécifie l'orientation par défaut des arêtes. Chaque élément **node** définit un sommet du graphe, il doit être identifié de manière unique par un attribut **id**. Chaque arête est définie par un élément **edge** dont les attributs **source** et **target** spécifient respectivement l'identifiant des sommets de départ et d'arrivée.

Ces quelques éléments et leurs attributs permettent de définir la structure de n'importe quel graphe. Cependant des informations supplémentaires sont souvent associées à un graphe et ses éléments. Deux mécanismes d'extensions permettent de stocker ces informations en dans un document GraphML.

4.3.2 Attributs XML

Les attributs XML sont les attributs « normaux » en XML, ils peuvent être associés à n'importe quel élément. Dans l'exemple ci-dessus **id** est un attribut XML de l'élément **node**. En GraphML il est possible d'ajouter des attributs XML aux éléments **graph**, **node** et **edge**. Par exemple dans un réseau social les sommets peuvent être étiquetés par le nom d'une personne, et les arêtes par un type de relation.

```
<node id="n00" label="Charles"/>
<node id="n01" label="Brian"/>
<node id="n02" label="Anna"/>
<edge source="n00" target="n01" relation="amicale"/>
<edge source="n00" target="n02" relation="professionnelle"/>
<edge source="n01" target="n02" relation="amoureuse"/>
```

Dans un graphe valué, les arêtes sont pondérées, on peut définir un attribut XML **weight** qui définit le poids de chaque arête.

```
<edge source="n00" target="n01" weight="0.75"/>
```

Cette solution est tout à fait acceptable, mais elle ne permet pas de préciser le type de l'attribut, ici un nombre réel. De plus, si un grand nombre d'arêtes partagent la même valeur pour un attribut, il est obligatoire de la spécifier pour chaque arête, ce qui peut devenir particulièrement redondant et rendre très volumineux la description d'un graphe. C'est pourquoi il existe un second type d'attributs, spécifiques aux documents GraphML.

4.3.3 Attributs GraphML

Les attributs GraphML sont en fait des éléments XML, et non plus des attributs XML, qui permettent de définir des données typées et potentiellement complexes. Un attribut GraphML doit être

déclaré. L'élément **key** permet de déclarer un attribut GraphML. Cet élément doit être placé dans l'élément **graphml**, avant l'élément **graph**.

```
<key id="w0" for="edge" attr.name="weight" attr.type="float"/>
```

Le fragment GraphML ci-dessus définit un attribut nommé *weight* de type *float*, qui peut s'appliquer exclusivement aux arêtes. Pour définir le poids d'une arête on ajoute un élément **data** dans l'élément **edge** correspondant. Ici, par exemple une arête de poids 0.75. Comme pour les sommets l'attribut **id** de l'élément **key** est obligatoire et permet de l'identifier facilement par la suite dans les éléments **data** le référençant.

```
<edge source="n00" target="n01">
   <data key="w0">0.75</data>
</edge>
```

On peut de plus définir une valeur par défaut pour un attribut GraphML.

```
<key id="w" for="edge" attr.name="weight" attr.type="float">
   <default>1.0</default>
</key>
```

Dans ce cas, les arêtes qui ne définissent par leur poids auront le poids par défaut, ici 1.

On remarque ici que les valeurs des attributs sont de vrais éléments XML, il est donc tout à fait possible de créer un attribut de type complexe. Dans le cas d'un réseau social, les sommets sont des personnes ayant une identité, un âge et bien d'autres caractéristiques. On peut définir une personne en XML.

```
<person>
   <firstname>Douglas</firstName>
   <lastname>Smith</lastname>
   <age>36</age>
   ...
</person>
```

Cet élément définissant une personne peut être utilisé comme valeur d'attribut GraphML.

4.3.4 Discussion autour des attributs XML et GraphML

Les attributs GraphML sont plus puissants que les attributs XML, mais sont moins pratiques à utiliser dans le cas de types primitifs, c'est pourquoi j'ai choisi d'autoriser dans Grapho la déclaration d'attributs typés GraphML utilisé comme attributs XML.

```
<key id="w" for="edge" attr.name="weight" attr.type="float">
   <default>1.0</default>
</key>
```

L'attribut GraphML déclaré ci-dessus peut être utilisé comme un attribut GraphML,

```
<edge source="n00" target="n01">
   <data key="w0">0.75</data>
</edge>
```

ou comme un attribut XML.

```
<edge source="n00" target="n01" weight="0.75"/>
```

Cela ajoute de la flexibilité au langage sans en briser la cohérence.

4.3.5 Les types d'attributs reconnus

Les types d'attributs les plus courants sont reconnus lors du chargement d'un fichier GraphML. Le Tableau 7 présente les types reconnus par le parser GraphMLSAXReader, ainsi que les mots-clés à utiliser dans le fichier et les classes qui les implémentent une fois chargés en mémoire.

Type	Notation GraphML	Classe Java
Texte	string	`java.lang.String`
Date	date	`java.util.Date`
Booléen	boolean	`java.lang.Boolean`
Entier	int	`java.lang.Integer`
Entier long	long	`java.lang.Long`
Réel	float	`java.lang.Float`
Réel double précision	double	`java.lang.Double`

Tableau 7 : types reconnus par le parser GraphML

Lorsqu'un graphe est chargé en mémoire, il n'y plus de restriction, tout type d'objets peut être stocké comme attribut d'une entité. Cependant lors de l'écriture du graphe sur fichier, tous les types autres que ceux définis ici seront considérés inconnus et les objets correspondant seront écrits comme des chaînes de caractères.

Une solution à ce problème serait d'utiliser les fonctions de sérialisation native de Java. La sérialisation permet de sauvegarder les objets Java, pourvu qu'ils implémentent l'interface Serializable. Cette solution pose problème car le format d'un objet sérialisé n'est pas compatible avec XML. La sérialisation ne produit pas des données lisibles comme le XML, le format utilisé est plus proche d'un codage binaire. De plus, Sun parle depuis longtemps de proposer un nouveau type de sérialisation basé sur XML, cela constituerait la solution idéale.

4.3.6 Entrées/Sorties en GraphML

J'ai adopté le format GraphML pour le stockage des graphes. J'ai donc implémenté deux classes, GraphMLSAXReader pour la lecture et GraphMLDOMWriter pour l'écriture.

Il y a deux API pour la lecture des documents XML en Java. DOM, pour Document Object Model et SAX, pour Simple API for XML. DOM permet de charger un document XML en mémoire de manière générique, qu'importe s'il s'agit de GraphML ou d'un autre langage XML. En mémoire un document XML est une arborescence dans laquelle il est possible de naviguer pour accéder aux divers éléments. C'est très simple à utiliser, mais en pratique cela fonctionne très mal sur les gros documents. La construction de l'arborescence mémoire est alors très coûteuse en temps et en mémoire. SAX suit une toute autre logique, il fournit un parser événementiel générique qui permet d'identifier un à un, durant la lecture, les différents éléments du document. L'implémentation avec SAX est un petit plus difficile mais les performances sont au rendez-vous, pas de problème de mémoire car SAX stocke peu d'information, et pas de problème de temps, le processus est de complexité linéaire par rapport à la taille du document.

Ces contraintes m'ont donc conduit à opter pour une implémentation SAX pour la lecture. L'écriture est quant à elle réalisée avec DOM avec des performances satisfaisantes.

4.4 Conception

4.4.1 Un modèle de graphe visuel

4.4.1.1 Conception du modèle

La création en mémoire d'un graphe n'implique que trois classes. Si le graphe est abstrait, c'est-à-dire sans propriétés graphiques, les trois classes **`GraphImpl`**, **`NodeImpl`** et **`EdgeImpl`**, suffisent. Si, par contre, le graphe est destiné à être visualisé, les classes **`ZGraphImpl`**, **`ZNodeImpl`** et **`ZEdgeImpl`**, doivent être utilisées. La Figure 84 montre le schéma d'héritage ente les différentes interfaces et classes constituant le modèle. Il y a deux axes d'héritage, du modèle de graphe abstrait vers le modèle de graphe graphique, et de la spécification vers l'implémentation.

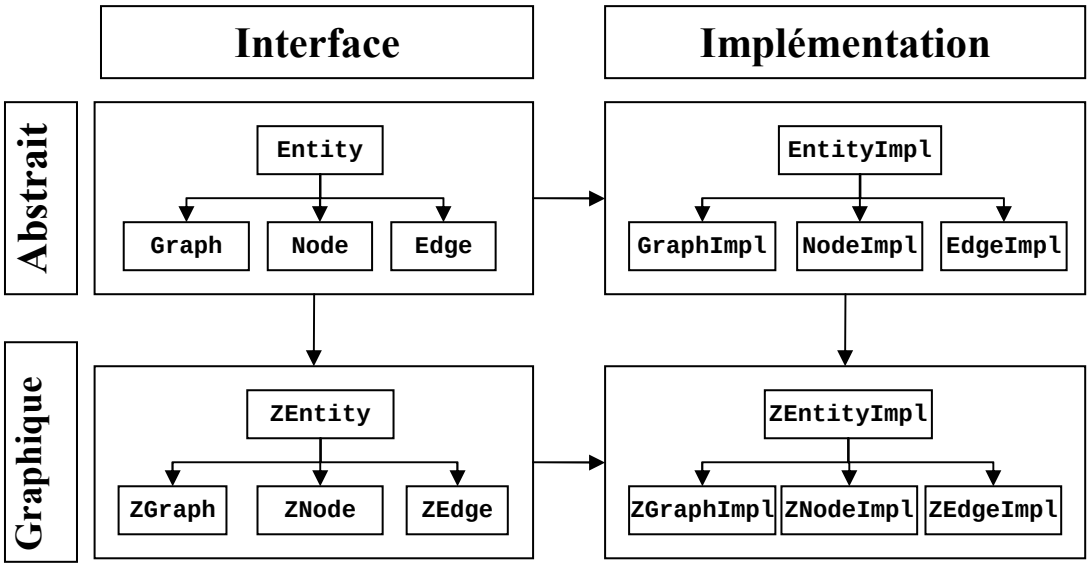


Figure 84 : schéma d'héritage des classes

4.4.1.2 Attributs de données

L'interface **`Entity`** définit un objet qui peut posséder des attributs ou propriétés. Chacun de ces attributs est un couple (nom, valeur). Le graphe est responsable du contrôle des types des attributs de ses sommets et arêtes. L'interface **`Graph`** hérite de l'interface **`EntityManager`**. La gestion des types d'attributs est un élément important, les possibilités de codage dépendent largement du type des attributs. Lorsque l'on ajoute un nouvel attribut de donnée à une ou plusieurs entités, il est important de préciser son type. Les attributs de données dont le type est connu peuvent être utilisés plus efficacement comme source d'un codage. De plus certains éléments de l'interface sont sensibles aux types de données. La fonction de tri des colonnes sur la table ne fonctionne que si le type de la colonne est connu.

4.4.1.3 Structure

Les interfaces **`Graph`**, **`Node`** et **`Edge`** définissent respectivement les fonctionnalités d'un graphe, et des sommets et arêtes. L'interface **`Graph`** permet d'accéder aux ensembles de sommets et d'arêtes et de les modifier. L'interface **`Node`** permet l'accès et la modification de l'ensemble d'arêtes propre à un sommet. L'interface **`Edge`** définit une arête comme un couple de sommets. Ces interfaces héritent de **`Entity`**, ce qui permet de stocker des attributs de données pour un graphe et chacun de ses éléments.

Par défaut le modèle du graphe est implémenté selon le codage par listes d'adjacences. Les sommets et arêtes sont stockés de manière redondante. Le graphe stocke la liste de ses sommets et la liste de ses arêtes, de plus, chaque sommet contient localement la liste de ses arêtes incidentes. Cette structure permet de représenter relativement efficacement les graphes peu denses.

Plusieurs codages de graphes ont été proposés pour la représentation des graphes. Auber [Auber03] a étudié les différences de quatre codages, en comparant leur complexité mémoire et algorithmique. Les codages étudiés sont :

- codage par tableau de tableaux d'index noté $\boldsymbol{C_{tti}}$. Les arêtes sortantes d'un sommet sont stockées dans un tableau, et ces tableaux sont eux-mêmes stockés dans un tableau.
- codage par listes doublement chaînées noté $\boldsymbol{C_{ldc}}$. C'est le même principe que pour le codage C_{tti}, on remplace les tableaux par des listes doublement chaînées.
- codage redondant noté $\boldsymbol{C_{lr}}$. Les sommets et arêtes sont stockés dans des listes doublement chaînées. Les voisinages entrant et sortant de chaque sommet sont, eux aussi, stockés dans des listes doublement chaînées.
- codage par matrice noté $\boldsymbol{C_{mat}}$. On stocke la matrice d'adjacence.

Le Tableau 8 présente l'ensemble des mesures de complexité pour ces différents codages.

	$\boldsymbol{C_{tti}}$	$\boldsymbol{C_{ldc}}$	$\boldsymbol{C_{lr}}$	$\boldsymbol{C_{mat}}$
Occupation mémoire	$1+2\cdot\lvert V\rvert+\lvert E\rvert$	$1+4\cdot\lvert V\rvert+3\cdot\lvert E\rvert$	$2+5\cdot\lvert V\rvert+11\cdot\lvert E\rvert$	$\lvert V\rvert^2$
Parcours de V	$\Theta(\lvert V\rvert)$	$\Theta(\lvert V\rvert)$	$\Theta(\lvert V\rvert)$	$\Theta(\lvert V\rvert)$
Parcours de E	$\Theta(\lvert V\rvert+\lvert E\rvert)$	$\Theta(\lvert V\rvert+\lvert E\rvert)$	$\Theta(\lvert E\rvert)$	$\Theta(\lvert V\rvert)^2$
Parcours voisinage sortant	$\Theta(\deg_+(v))$	$\Theta(\deg_+(v))$	$\Theta(\deg_+(v))$	$\Theta(\lvert V\rvert)$

Parcours voisinage entrant	$\Theta(\|V\|+\|E\|)$	$\Theta(\|V\|+\|E\|)$	$\Theta(\deg_-(v))$	$\Theta(\|V\|)$
Sommets d'une arête	-	-	$\Theta(1)$	-
Suppression d'un sommet	$\Theta(\|V\|)$	$\Theta(1)$	$\Theta(\deg(v))$	$\Theta(\|V\|)^2$
Suppression d'une arête	$\Theta(\deg_+(v))$	$\Theta(1)$	$\Theta(\deg(v)+\deg(u))$	$\Theta(1)$
Ajout d'un sommet	$\Theta(\|V\|)$	$\Theta(1)$	$\Theta(1)$	$\Theta(\|V\|)^2$
Ajout d'une arête	$\Theta(\deg_+(v))$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Existence d'une arête	$\Theta(\|V\|+\|E\|)$	$\Theta(\|V\|+\|E\|)$	$\Theta(\|E\|)$	$\Theta(1)$

Tableau 8 : Complexité des opérations sur différents codages

Le codage par matrice est le plus coûteux en mémoire et en temps pour la plupart des opérations. Ce codage peut être intéressant pour des graphes très denses, lorsque $|E| \simeq |V|^2$. Ce n'est toutefois pas le cas des graphes que l'on manipule en visualisation d'informations, qui sont généralement peu denses. Le codage redondant est un peu plus coûteux en mémoire que les codages par tableaux et listes chaînées, mais il est aussi le plus efficace pour la plupart des opérations, excepté pour la suppression d'arêtes.

La Figure 85 présente un schéma du codage redondant implémenté dans l'API Grapho. Les ensembles sont implémentés par des listes doublements chaînées, ce qui assure de bonnes performances pour les opérations de suppressions et d'ajouts de sommets et d'arêtes.

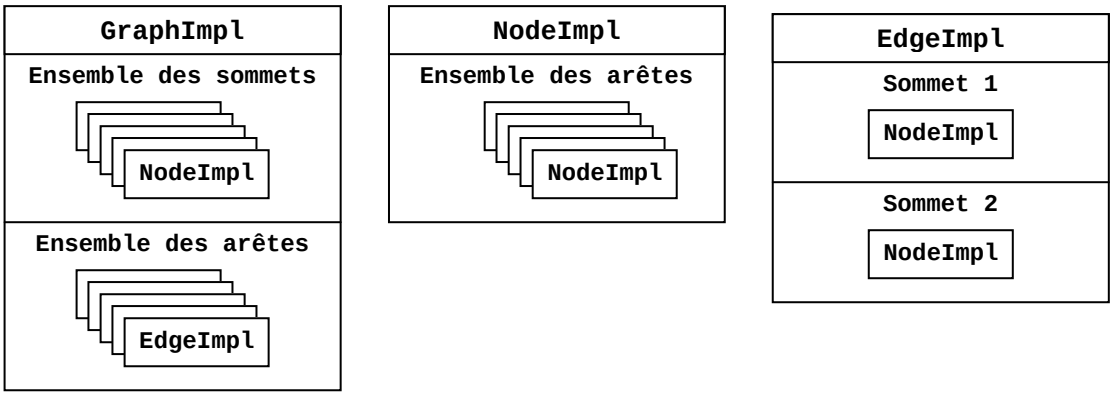


Figure 85 : Implémentation par codage redondant

4.4.1.4 Attributs graphiques

L'interface **`ZEntity`** regroupe l'ensemble des attributs graphiques communs aux graphes, sommets et arêtes.

Size spécifie la taille d'une entité. Pour un sommet la taille correspond à sa surface, pour une arête c'est son épaisseur. La taille d'un graphe détermine l'espace dans lequel ses sommets peuvent être placés.

L'attribut *Paint* spécifie l'apparence d'une entité. Cet attribut sert principalement à définir la couleur des objets, mais l'interface **`java.awt.Paint`** utilisée ici correspond à une méthode de remplissage plus générale. Elle permet par exemple d'appliquer un dégradé de couleurs ou une texture.

L'attribut *Transparency* contrôle la transparence d'un objet. La transparence peut être utilisée pour réduire la visibilité de certains objets. Attention toutefois car l'utilisation de la transparence modifie l'apparence des couleurs.

Les graphes, sommets et arêtes peuvent être étiquetés. L'attribut *Label* est une chaîne de caractères qui est affichée à proximité de son entité.

L'attribut *Visible* est un booléen qui détermine si l'entité doit être affichée. L'attribut *Selected* est aussi un booléen, il détermine si une entité est sélectionnée. Enfin l'attribut *Highlight* est un entier qui permet de définir plusieurs états de mise en relief pour une entité.

L'interface **`ZNode`** hérite de **`ZEntity`**, elle l'enrichit en incluant les caractéristiques graphiques propres aux sommets.

Un sommet possède une position dans le plan qui est stockée dans l'attribut *Location*. L'attribut *Shape* détermine la forme du sommet. *Pinned* est un booléen qui précise si un sommet peut être déplacé ou non. Ce dernier attribut est utilisé dans les différents algorithmes de placement.

4.4.2 Le module de Présentation

La couche Présentation est responsable de l'affichage du diagramme *nœud-lien* mais aussi des fonctions élémentaires d'interactions, comme la gestion des évènements souris primitifs comme le picking. Cette partie de l'API est implémentée avec Piccolo [Bederson03]. Piccolo est une API du HCIL de l'université du Maryland qui permet de créer des visualisations interactives, cette API est décrite dans la section suivante.

4.4.2.1 La librairie Piccolo

Piccolo est une API qui permet de créer des interfaces graphiques interactives dites zoomables. Piccolo succède à Jazz, Pad++ [Bederson96] et Pad dans la famille des interfaces zoomables. Dans ces interfaces, le zoom est un paradigme d'interaction essentiel, et est donc pleinement intégré dans l'API.

Piccolo permet de créer des interfaces dynamiques et interactives par le biais de graphes de scène. On peut différencier trois types d'objets dans un graphe de scène Piccolo. La caméra qui détermine le point de vue de l'utilisateur sur la scène. Le layer qui est l'espace observé par la caméra. Et enfin les éléments de la scène, comme du texte, des formes géométriques et des images.

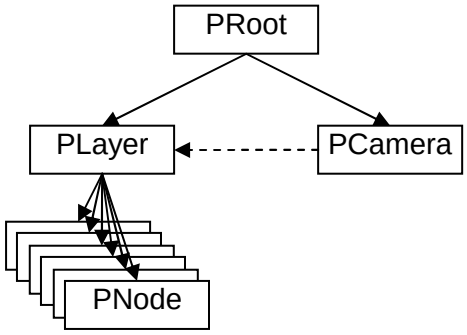


Figure 86 : Graphe de scène Piccolo élémentaire

La Figure 86 présente un graphe de scène Piccolo élémentaire. Le graphe de scène Piccolo est principalement un arbre d'inclusion. La racine du graphe de scène est toujours une instance de la classe **`PRoot`**. L'instance de **`PLayer`** est le layer principal et doit contenir tous les éléments visuels et interactifs de la scène. Ces éléments doivent dériver de la classe **`PNode`** qui regroupe les fonctions communes de tous les objets du graphe de scène, y compris les objets **`PRoot`**, **`PCamera`** et **`PLayer`**. Le graphe de scène de la Figure 86 spécifie que la scène est constituée d'un layer qui contient plusieurs éléments visuels et interactifs. La caméra permet de paramétrer le point de vue de l'observateur, c'est-à-dire la partie visible du layer, ainsi que le coefficient de zoom.

`PNode` est la classe de base de tous les objets Piccolo. Une instance de **`PNode`** n'a cependant pas de représentation visuelle. Les objets de cette classe sont les briques élémentaires de la structure hiérarchique d'inclusion. Chaque objet dérivant de **`PNode`** est un nœud dans cette hiérarchie, il a donc un père et des fils.

Cette hiérarchie contrôle l'héritage des transformations. A chaque nœud peut-être assignée une transformation affine, cette transformation est par définition une composition de translations, rotations, homothéties et changements d'échelle. La transformation affine d'un nœud est appliquée à lui et toute sa descendance.

Enfin, la hiérachie intervient dans les processus d'affichage et de gestion du picking. Ces deux processus impliquent de calculer efficacement la boîte englobante de chaque élément à afficher. La boîte englobante d'un élément visuel est le plus petit rectangle dont les côtés sont respectivement parallèles à l'horizontale et la verticale qui englobe cet élément. L'utilisation des boîtes englobantes permet de déterminer plus facilement les parties de l'écran à redessiner lorsque la scène a changé. Le picking est un processus qui intervient lorsque l'utilisateur manipule la souris. Il permet d'identifier les nœuds pointés par le curseur. En utilisant l'arborescence et les boîtes englobantes, on peut améliorer la complexité du picking de $\Theta(N)$ à $\Theta(\log N)$, N étant le nombre d'éléments.

La classe **`PNode`** regroupe les fonctionnalités décrites ci-dessus. Ces fonctionnalités sont donc communes à tous les objets dérivant de **`PNode`**. Cependant, la classe **`PNode`** ne permet pas directement d'afficher du contenu graphique. Trois types d'éléments graphiques ont été implémentés pour le texte, les images et les figures vectorielles. Ces trois types d'éléments visuels interactifs dérivent de la classe **`PNode`** et sont :

- **`PText`**, pour l'affichage de labels,
- **`PPath`**, pour l'affichage de courbes vectorielles,
- **`PImage`** pour l'affichage des bitmaps.

Ces classes permettent de manipuler les briques visuelles primitives. On peut ensuite créer des objets plus complexes en les composant dans un arbre d'inclusion. Par exemple, la Figure 87 montre comment construire une image étiquetée et encadrée avec trois nœuds primitifs. Ici le cadre est le père du label et de l'image, cela lui permet de se redimensionner si nécessaire pour s'adapter à l'image et au texte.

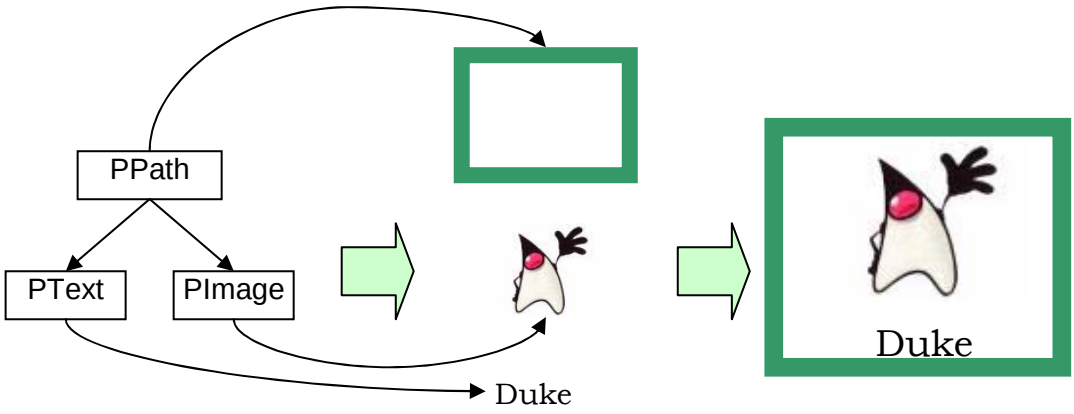


Figure 87 : Création d'éléments complexes par composition

L'héritage offre cependant plus de liberté, lorsque l'on souhaite créer des objets plus complexes et leur ajouter des comportements dynamiques et interactifs. C'est le choix que nous avons fait pour l'implémentation de couche Présentation.

4.4.2.2 Implémentation

La couche Présentation est constituée des trois classes **`DefaultGraphView`**, **`DefaultNodeView`** et **`DefaultEdgeView`** qui implémentent respectivement les vues d'un graphe, d'un sommet et d'une arête. Ces héritent toutes trois de classes Piccolo.

La classe **`DefaultGraphView`** est donc une spécialisation de la classe **`PNode`**, composée de deux calques différents. Le premier calque est utilisé pour les arêtes et le second pour les sommets. Ces deux calques permettent de toujours afficher les sommets par dessus les arêtes.

La classe **`DefaultNodeView`** hérite aussi de **`PNode`**. Elle permet de composer dans un même objet une forme géométrique colorée instance de la classe **`PPath`**, et une étiquette textuelle instance de la classe **`PText`**. L'héritage nous a permis d'automatiser les calculs de repositionnement de l'étiquette par rapport à la forme lors d'un changement de taille par exemple.

La classe **`DefaultEdgeView`** hérite de **`PCurve`**. La classe **`PCurve`**, que nous avons créée pour l'occasion, permet de représenter des segments de courbes. Nous y avons ajouté la possibilité de dessiner des extrémités fléchées.

Ces classes implémentent toutes trois l'interface **`EntityView`**. Cette interface très simple spécifie le comportement générique et décrit le cycle de vie d'un objet de la couche Présentation. Un tel objet doit pouvoir être initialisé, mis à jour et détruit. Il a de plus connaissance de l'objet du Modèle dont il est la représentation.

```
public interface EntityView
{
  public ZEntity getZEntity();
  public void initialize(GraphController controller);
  public void update(GraphController controller, int propertiesFlag);
  public void garbageCollect(GraphController controller);
}
```

La méthode `update` est appelée par le contrôleur lorsque l'objet du modèle a été modifié. Le paramètre `propertiesFlag` indique l'ensemble des modifications subies par l'objet du modèle.

Tous les objets de Présentation sont sensibles aux évènements utilisateurs. Piccolo gère la gestion des évènements primitifs et les redirige vers le composant en charge de l'affichage. Ce composant, instance de la classe **`PCanvas`**, fait le lien entre Swing et Piccolo. Nous avons spécialisé ce composant afin d'offrir des évènements utilisateurs de plus haut niveau. En effet, lorsque l'on développe avec Grapho, les évènements utilisateurs à traiter doivent faire référence aux objets Grapho et non aux objets Piccolo. Nous avons donc créé l'interface **`ControlListener`** qui permet de facilement créer des gestionnaires d'évènements spécifiques.

4.4.3 Synchronisation du Modèle et de la Présentation

La synchronisation entre les deux couches Modèle et Présentation est un point clef de la conception d'un outil de visualisation de graphes. La couche de Contrôle se charge de cette synchronisation.

Nous avons vu auparavant le fonctionnement du modèle. Toute modification du modèle peut être notifiée par envoi de messages. Ce processus de notification est implémenté selon le motif de conception **`Listener`**. Le modèle du graphe peut notifié un ensemble d'objets pourvu qu'ils implémente l'interface **`GraphListener`**. Les objets qui implémentent cette interface peuvent s'abonner auprès d'une instance du modèle afin de recevoir les messages décrivant les modifications qui lui sont appliquées.

La couche Contrôle se résume dans Grapho à une seule classe, **`GraphController`** qui implémente l'interface **`GraphListener`**. Le contrôleur est une instance de cette classe, il est notifié de toutes modifications du modèle, et assure les mises à jour de la couche Présentation. Les messages en provenance du modèle arrivent par le processus Modèle, ils sont placés dans un buffer afin d'être traités par le processus Swing. La couche Contrôle assure dont principalement les échanges entre les deux processus différents. Je rappelle que l'utilisation de deux processus différents pour les calculs sur le Modèle et pour la mise à jour de la Présentation assure de meilleures performances et permet surtout d'éviter les problèmes de blocage de l'affichage lors d'opérations complexes sur le Modèle.

La technique de buffering permet aussi de factoriser plusieurs petites mises à jour en une mise à jour multiple de la Présentation, afin d'éviter de trop fréquentes modifications de la Présentation qui dégrade les performances de l'affichage et des éléments interactifs. Une technique de buffering similaire est aussi utilisée pour le rafraîchissement de la table des entités. Ce composant Swing, présenté en section 4.5.2, affiche les attributs des entités d'un graphe. Sa mise à jour lors de l'ajout ou de la suppression d'entité est particulièrement coûteuse. Lors d'un filtrage massif, il est

particulièrement important de factoriser l'ensemble des suppressions afin de mettre à jour la table lorsque tous les objets ont été supprimés. Cette factorisation nous a permis de gagner un facteur 100 sur le temps d'exécution des filtrages.

La technique de buffering des messages fonctionne avec un Timer. Lorsqu'un message est reçu, la modification est stockée dans une queue et le Timer est réinitialisé. Les mises à jour de la vue, correspondants aux modifications stockées, ne seront réalisées que lorsque aucun message n'arrivera pendant le temps de vie du Timer. Après plusieurs essais le temps de vie du Timer a été fixé à 100 ms.

Les messages entre le Contrôleur et la Présentation s'exécutent dans le processus Swing. Ce processus est responsable de l'exécution des messages de la librairie graphique de Java. Les messages entre le Contrôleur et le Modèle s'exécutent dans un processus dédié. Cela permet de décharger le processus Swing et de s'assurer ainsi que des tâches complexes peuvent être exécutées sur le Modèle sans figer l'affichage ou nuire aux temps de réponses de l'interface graphique, cf. Figure 88.

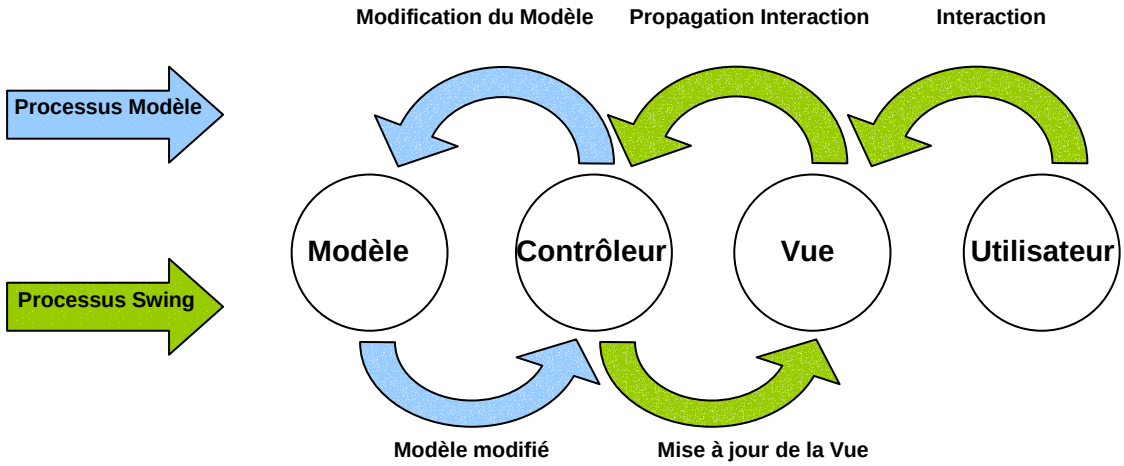


Figure 88 : communication entre les différentes couches de l'API

4.5 Grapher : prototype d'exploration de graphes

4.5.1 Description

Grapher est l'application de visualisation de graphes conçue avec l'API Grapho. Grapher permet d'utiliser toutes les fonctionnalités de l'API pour l'exploration et l'analyse de graphes.

Le panneau d'affichage principal est constitué des deux vues de graphe, la table et le diagramme *nœud-lien*. La table permet une lecture linéaire des différents éléments du graphe et la visualisation de leurs différents attributs. Le diagramme *nœud-lien* présente la topologie du graphe et permet une analyse visuelle de la structure du graphe.

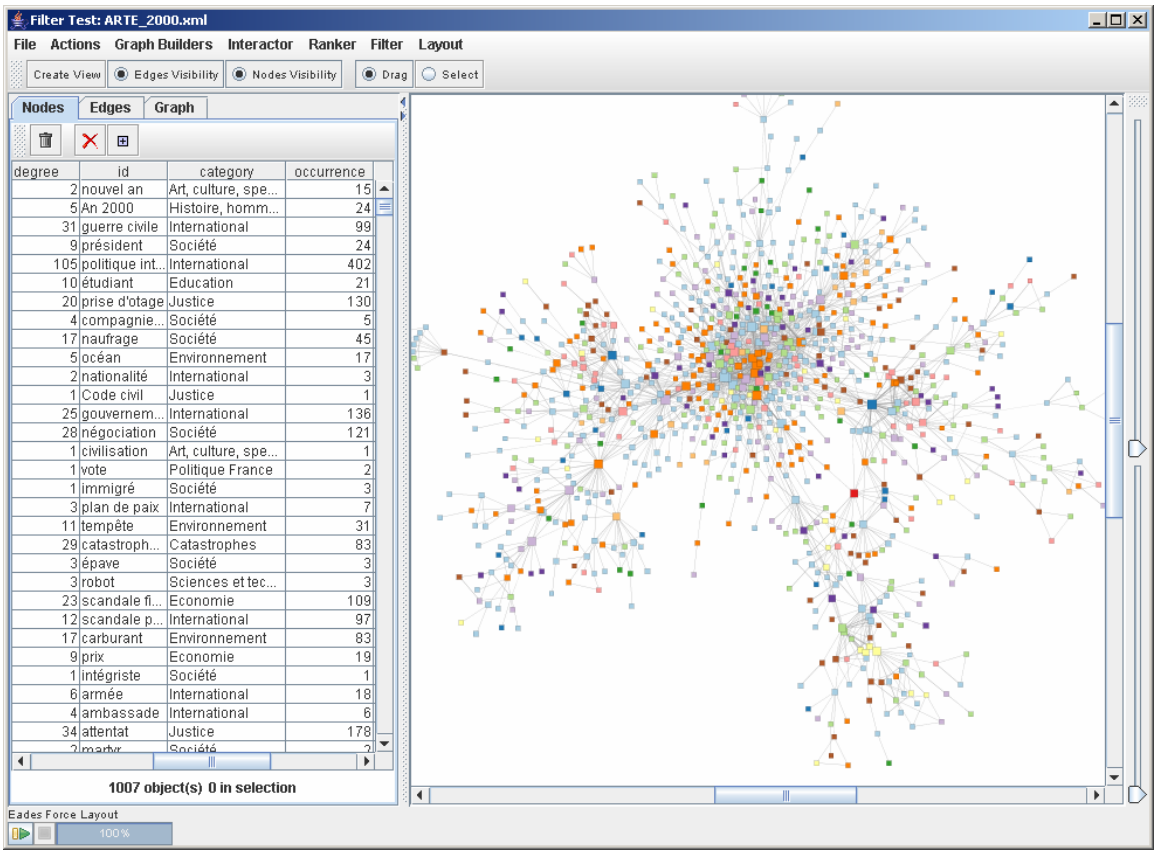


Figure 89 : La fenêtre principale de visualisation de Grapher

La barre de menu donne l'accès aux différentes fonctionnalités de Grapho comme les entrées/sorties, les paramètres de codages graphiques, le calcul de métriques, l'application de filtres et la sélection d'un algorithme de placement.

4.5.2 Tables des données interactives

Les tables ou tableaux de données offrent une lecture linéaire de l'information. Cette lecture est tout à fait complémentaire à la lecture structurelle ou topologique qu'offre la visualisation d'un graphe placé.

La visualisation d'un graphe permet de rechercher des éléments par voisinages et par régions. Par contre il est très difficile de repérer de manière précise un objet selon la valeur d'un ou plusieurs de ces attributs. Si, par exemple, on recherche un sommet dont on connaît le label, le graphe placé n'est pas la meilleure solution. Il faudrait alors, dans le pire des cas, lire tous les labels pour être sûr d'identifier le sommet en question. La vue d'un graphe placé n'offre pas un ordre de lecture pertinent pour la recherche de certaines caractéristiques des sommets et arêtes. Il est bien sûr possible d'utiliser un codage pour rendre visible, par la couleur par exemple, certains attributs, mais cela ne permet pas d'évaluer de manière exacte leur valeur, l'information résultant d'un codage est plus floue. Le but du codage n'est pas d'avoir une lecture précise des valeurs d'attributs, mais une lecture structurelle globale rendue rapide par son adéquation avec nos capacités préattentives. C'est pourquoi l'utilisation d'une table de données s'avère quasiment obligatoire en complément du diagramme *nœud-lien*.

Chaque type d'entités peut être visualisé dans une table. Le prototype implémenté avec l'API Grapho propose donc trois tables, une pour le graphe, une pour les sommets et la dernière pour les arêtes. Chaque entité est représentée par une ligne dans la table correspondante. A chaque attribut des entités correspond une colonne de la table, cf. Figure 90.

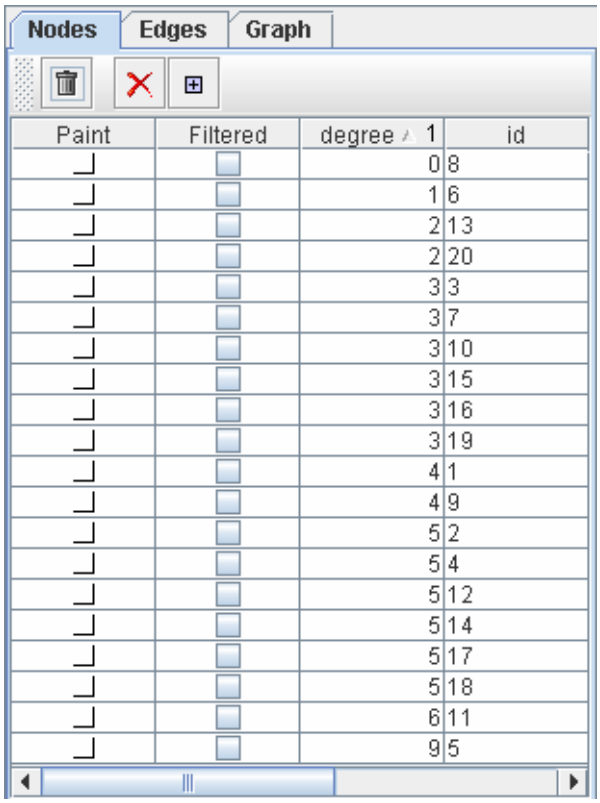


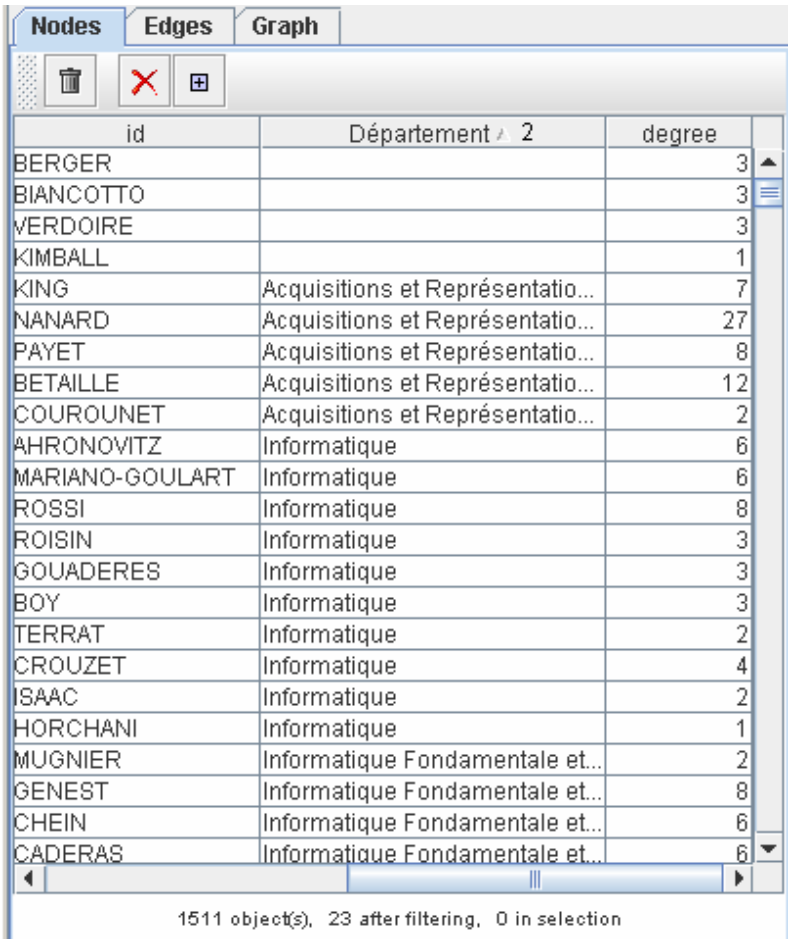
Figure 90 : exemple de table de sommets

Tris

Les lignes d'une table peuvent être ordonnées selon un ou plusieurs attributs. La table est donc ordonnable par rapport aux valeurs d'une ou plusieurs de ses colonnes.

Tout d'abord on parle de tri simple lorsque la table est ordonnée selon un attribut. Le comportement d'un tri dépend du type de l'attribut choisi. Si l'attribut sélectionné est de type chaîne de caractères, le tri est basé sur l'ordre alphanumérique, ce comportement est le comportement par défaut. Si le type d'un attribut n'est pas connu, l'ordre alphanumérique est utilisé, et on considère alors que les valeurs de l'attribut sont des chaînes de caractères. Les valeurs sont des objets Java, et possèdent donc la méthode toString() qui donne une représentation textuelle de l'objet, cette chaîne de caractère est alors utilisée pour le tri. Lorsque le type de l'attribut, c'est-à-dire la classe Java, est connu et qu'il hérite de l'interface Comparable, alors les valeurs sont triés selon l'ordre conventionnel, de façon croissante ou décroissante pour des nombres ou des dates par exemples. Afin d'obtenir des tris pertinents il est nécessaire de spécifier le type des attributs. Cela peut-être fait dans le fichier GraphML du graphe en suivant la syntaxe requise, ou de manière programmatique lorsqu'on crée un nouvel attribut par un processus, comme le calcul de métriques par exemple. Une fonctionnalité permet toutefois de deviner le type des attributs dont le type n'aurait pas été correctement déclaré. Pour trier une table selon les valeurs d'un attribut il suffit de double cliquer sur la colonne correspondante, un icône indique alors le sens du tri, croissant ou décroissant. La Figure 91 illustre un

tri simple sur un petit graphe de six auteurs. Les auteurs sont ordonnés dans la table selon l'attribut qualitatif Département.



id	Département 2	degree
BERGER		3
BIANCOTTO		3
VERDOIRE		3
KIMBALL		1
KING	Acquisitions et Représentatio...	7
NANARD	Acquisitions et Représentatio...	27
PAYET	Acquisitions et Représentatio...	8
BETAILLE	Acquisitions et Représentatio...	12
COUROUNET	Acquisitions et Représentatio...	2
AHRONOVITZ	Informatique	6
MARIANO-GOULART	Informatique	6
ROSSI	Informatique	8
ROISIN	Informatique	3
GOUADERES	Informatique	3
BOY	Informatique	3
TERRAT	Informatique	2
CROUZET	Informatique	4
ISAAC	Informatique	2
HORCHANI	Informatique	1
MUGNIER	Informatique Fondamentale et...	2
GENEST	Informatique Fondamentale et...	8
CHEIN	Informatique Fondamentale et...	6
CADERAS	Informatique Fondamentale et...	6

Figure 91 : Tri simple selon le Département

Si l'on souhaite qu'en plus les auteurs d'un même Département soient ordonnés par degré, il faut alors appliquer un tri multiple par Département puis par degré, cf. Figure 92.

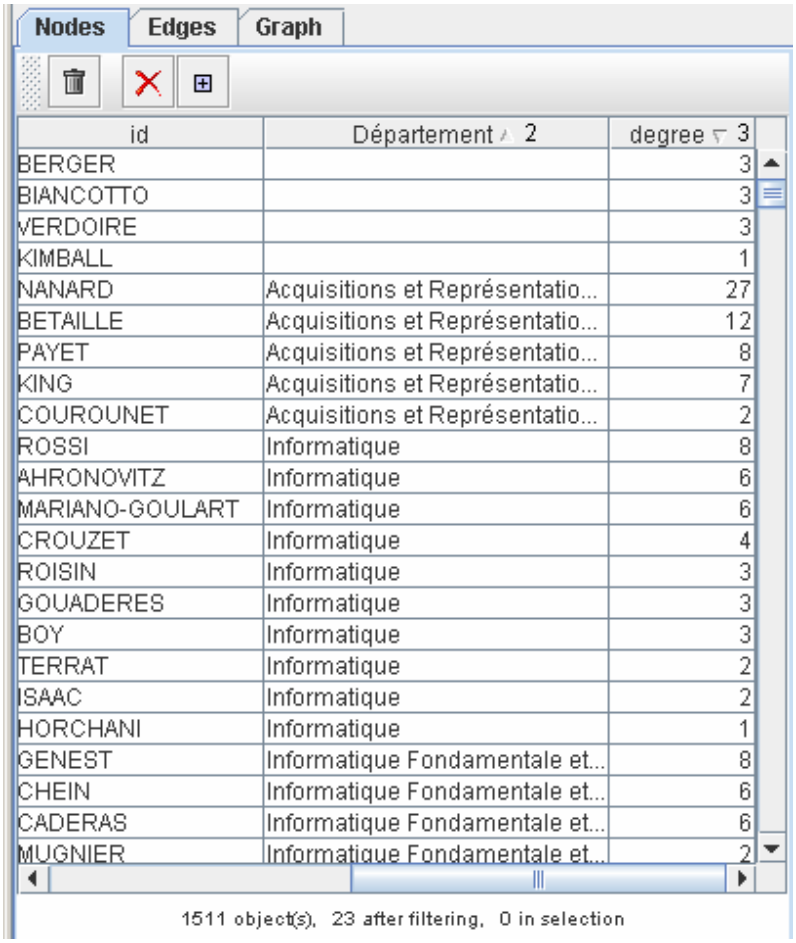


Figure 92 : Tri multiple selon le Département puis le degré

Un point sur la complexité de cette fonctionnalité de tri. Elle consiste simplement à réordonner une liste d'entités selon un critère. Java offre des facilités pour trier les structures de listes. L'implémentation de l'algorithme de tri assure une complexité en $\Theta(N.\log N)$ dans le pire des cas, pour une liste de N éléments.

4.5.3 Techniques de Filtrage

4.5.3.1 Type de filtrage

Plusieurs niveaux de filtrage coexistent dans Grapho et sont donc accessibles dans Grapher. Le filtrage visuel permet de définir l'ensemble des entités visibles dans le diagramme *nœud-lien*. Le filtrage visuel ne modifie pas la structure du graphe, les sommets ou arêtes invisibles sont tout de même présents dans le graphe et seront donc pris en compte par les algorithmes de placements ou de calcul de métriques. Le filtrage réversible permet de supprimer temporairement des éléments de la structure du graphe. Les éléments filtrés temporairement apparaissent tout de même dans les tables, mais une marque permet de les repérer. Les tables des arêtes et des sommets possèdent une colonne indiquant par un booléen si une entité est filtrée. Les entités peuvent être réintroduites dans le graphe à tout moment. Une entité filtrée temporairement reste en mémoire. Seul le filtrage définitif libère les entités filtrées de la mémoire, cette opération est quant à elle irréversible.

Il faut préciser que lorsque des sommets sont filtrés de manière structurelle, nous avons fait le choix de filtrer aussi les arêtes incidentes à ces sommets. Une arête ne peut être réintroduite dans le graphe que si ses deux sommets ne sont pas filtrés.

4.5.3.2 Performances du filtrage

Les différents types de filtrage ne sont pas équivalents en complexité. Le filtrage visuel consiste seulement à modifier la visibilité des entités, qui est décrite par un booléen. La complexité de cette opération est linéaire par rapport au nombre d'entités dont la visibilité est modifiée, et la mise à jour du modèle est quasiment instantanée, même pour de grands graphes. Le filtrage visuel s'exécute donc toujours dans des temps compatibles avec l'interactivité.

Le filtrage définitif consiste à mettre à jour la structure du graphe. Les éléments filtrés sont donc effectivement supprimés du modèle. Rappelons-nous que le modèle du graphe est implémenté selon le codage des listes redondantes. Lorsqu'une arête est ajoutée ou supprimée il est nécessaire de mettre à jour la liste des arêtes du graphe ainsi que les listes des arêtes des deux sommets incidents. Quand on ajoute ou supprime un sommet, il faut seulement mettre à jour la liste des sommets du graphe. La suppression d'un sommet entraîne la suppression de toutes ses arêtes incidentes.

Le filtrage temporaire fonctionne à peu de chose près comme le filtrage définitif. Dans le modèle du graphe de Grapho, toutes les listes d'entités sont dédoublées. A chaque liste d'entités correspond une liste des mêmes entités mais après filtrage. Les objets effectivement présents dans la structure sont donc présents dans les deux listes. Lorsqu'une entité est filtrée temporairement, elle est marquée comme filtrée par un booléen et retirée de la ou les listes des entités non filtrées. Pour une arête cela implique donc de mettre à jour la liste des arêtes non filtrées et les listes d'arêtes non filtrées des deux sommets incidents.

Les différents types de filtrage sont tous de complexité linéaire, le temps d'exécution d'un filtrage reste proportionnel au nombre d'éléments à filtrer. La section 4.5.7 présente plus en détail les performances de l'API.

4.5.3.3 Filtrage et Interaction

La seule manière de filtrer des entités avec Grapher est de sélectionner les entités, dans la table ou le diagramme *nœud-lien*, puis de spécifier le type de filtrage. On peut parler de filtrage manuel, en ce sens, que l'utilisateur doit sélectionner les objets à supprimer ou à ajouter. La table et ses fonctions de tri permettent de facilement réaliser une sélection relative aux attributs. Le diagramme *nœud-lien* est quant à lui particulièrement efficace pour sélectionner des régions du graphes.

Ce choix d'une sélection manuelle avant filtrage impose à l'utilisateur de désigner explicitement les entités qui seront filtrées. C'est pourquoi les différents algorithmes de filtrage de Grapho ne filtrent pas directement le graphe, ils créent en fait de nouveaux attributs qui devront être utilisés comme des marqueurs pour sélectionner les éléments à supprimer. Nous développons actuellement un module de création de requêtes complexes pour faciliter la sélection. La sélection manuelle d'entités peut en effet devenir fastidieuse lorsque l'on manipule de grands graphes. Ce module est décrit plus en détail en section 4.5.3.7.

4.5.3.4 Filtrage autour d'un sommet focus : vue locale

Les méthodes de placement implémentées dans Grapho offrent principalement des vues globales de graphes. Deux sommets voisins peuvent être très éloignés dans un placement global.

Lorsque l'on s'intéresse plus particulièrement au voisinage d'un sommet, il est possible d'activer la fonction d'exploration par proximité. L'utilisateur peut alors choisir un sommet focus par simple double-clic. Le graphe est automatiquement filtré pour ne laisser que les K sommets les plus proches du focus dans un rayon R. Nous utilisons la distance géodésique comme mesure de distance. Par défaut le rayon maximum R est de 3, et le nombre de sommets à afficher K est égal à 100.

Tant que le nombre de sommets reste peu élevé, nous utilisons une méthode d'animation fine pour la mise à jour des positions des sommets, cf. 4.5.6.2. Cette technique permet de suivre facilement les sommets durant le placement.

Cette fonction de filtrage automatique offre une bonne vue locale du voisinage d'un sommet, et permet une exploration par proximité du graphe.

4.5.3.5 Filtrage en graphes arborés

Deux algorithmes de filtrage produisent des graphes arborés. Ils sont tous les deux basés sur le même principe, la construction d'un arbre couvrant, et le calcul pour chaque arête de sa longueur dans cet arbre. Cette longueur d'arête est une mesure théorique qui correspond à la distance géodésique dans l'arbre couvrant qui sépare les deux sommets incidents d'une arête. Si l'arête appartient à l'arbre couvrant sa longueur est donc de 1. Par contre si elle n'en fait pas partie, sa longueur est supérieure à 1. Ce marquage des arêtes est particulièrement intéressant pour le filtrage. En filtrant les arêtes selon les longueurs décroissantes on simplifie le graphe en le faisant tendre vers son arbre couvrant.

Il y a deux possibilités pour obtenir un arbre couvrant, l'algorithme de Kruskal qui donne un arbre couvrant minimal (ou maximal) et l'algorithme de Boutin qui diffère par le choix des arêtes de l'arbre couvrant.

Arbre couvrant minimal

Un arbre couvrant est dit minimal (resp. maximal) lorsque la somme des poids des arêtes appartenant à l'arbre est minimal (resp. maximal). Pour obtenir un arbre couvrant minimal ou maximal on utilise l'algorithme de Kruskal, qui peut être décrit ainsi :

```
Ajouter toutes les arêtes dans la liste L.
Trier L selon le poids des arêtes, de façon croissante pour un arbre minimal et
décroissante pour un arbre maximal.
Tant que L n'est pas vide
{
      Retirer la première arête de L.
      Si cette arête ne crée pas de cycle, l'ajouter au graphe.
}
```

L'opération de tri des arêtes est la phase la plus complexe de l'algorithme en $\Theta\left(|E|.\log|E|\right)$. Au cours de l'algorithme on utilise une structure de données qui décrit les différentes composantes connexes et qui permet de tester si une arête crée un cycle en $\Theta\left(\log|V|\right)$. Le coût total du maintien de cette structure et des tests de présence de cycle s'exprime en $\Theta\left(|V|.\log|V|\right)$.

Arbre couvrant de Boutin

Boutin [Boutin06] propose une technique de construction d'arbre couvrant qui préserve l'aspect sans échelle et l'indice de clustering d'un graphe. Une des particularités des graphes sans échelle est que la distribution des degrés des sommets suit une loi de puissance. La plupart des sommets ont un faible degré, et quelques-uns ont un très fort degré. Les graphes issus de données réelles présentent très souvent cette propriété, qui provient du fait que les sommets les plus connectés ont plus de chance d'acquérir de nouvelles connexions au cours du temps. On peut penser, par exemple, à l'autorité d'une page sur le Web. La visibilité d'une page Web dépend de cette autorité, les moteurs de recherche présentent les pages de grande autorité dans les premiers résultats. Cette grande visibilité a pour résultat de favoriser la création de nouveaux liens vers cette page, et donc d'augmenter son autorité. C'est la politique « du riche devient plus riche ».

La technique de Boutin tient compte de cette propriété dans la sélection des arêtes de l'arbre couvrant.

```
Ajouter tous les sommets dans la liste L.
Trier L selon le degré (ou une autre mesure d'autorité) des sommets, de façon
décroissante.
Tant que L n'est pas vide
{
	Retirer v le premier sommet de L.
	Ajouter les voisins de v dans la liste Lv.
	Ordonner Lv selon la mesure d'autorité des sommets, de façon décroissante.
	Tant que Lv n'est pas vide
	{
		Retirer s le premier sommet de Lv.
		Si l'arête entre v et s ne crée pas de cycle, l'ajouter au graphe.
	}
}
```

Ces deux techniques ont l'avantage de simplifier le graphe sans pour autant casser les clusters, ou dénaturer leur nature sans échelle. L'exemple suivant le confirme. Considérons le graphe des descripteurs des sujets du journal télévisé d'Arte pour l'année 2000, cf. 4.6.1 pour plus de détails sur ce graphe. Il présente les caractéristiques des graphes sans-échelle à comportement petit-monde.

Après le calcul de l'arbre couvrant qui maximise le poids des arêtes (ici l'occurrence), le graphe est filtré pour différentes valeurs de longueur dans l'arbre. Les distributions des degrés suivent toujours une loi normale et la distribution des indices de clustering varie très peu. La Figure 93 présente l'histogramme de l'indice de clustering des sommets du graphe non filtré. Sur la Figure 94, 600 arêtes dont la longueur est supérieure à 10 ont été supprimées. L'indice de clustering moyen diminue peu et l'histogramme des indices garde le même profil. Les figures Figure 95, Figure 96 et Figure 97 montrent l'évolution de l'indice de clustering lorsque l'on filtre le graphe de manière plus aggressive. Dans le cas le plus extrême, près de 2/3 des arêtes ont été filtrées, et le graphe présente encore un indice de clustering du même ordre que le graphe original, le profil de la distribution reste quant à lui comparable.

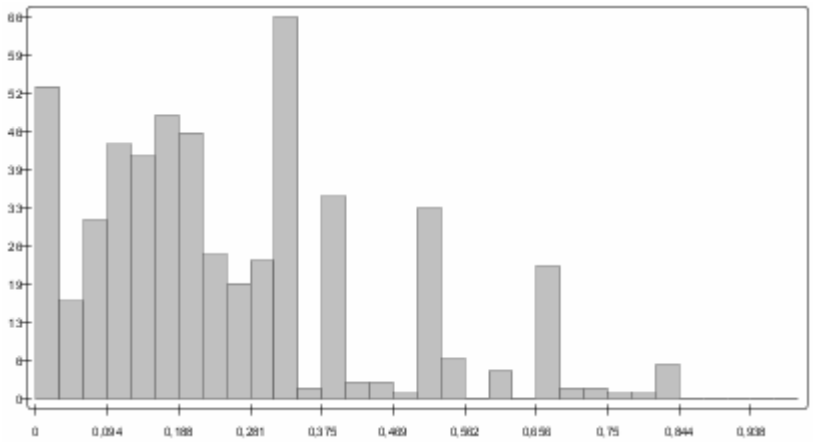


Figure 93 : graphe JT Arte non filtré : longueur max = 21
1007 sommets, 3423 arêtes et IC = 0.383

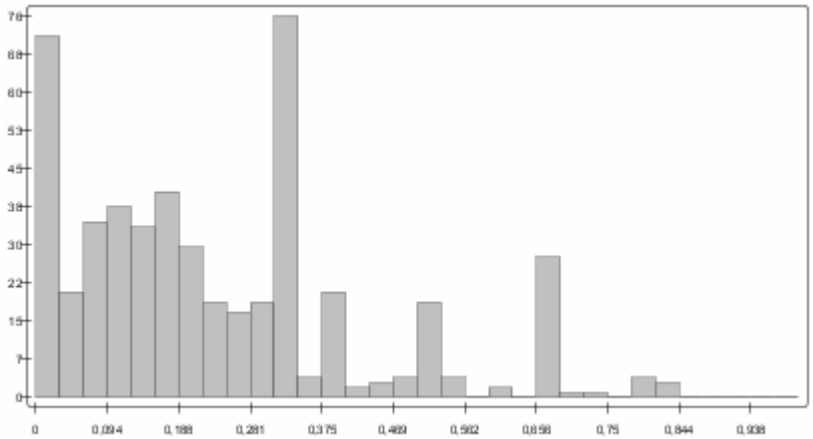


Figure 94 : graphe JT Arte filtré : longueur max = 10
1007 sommets, 2812 arêtes et IC = 0.35

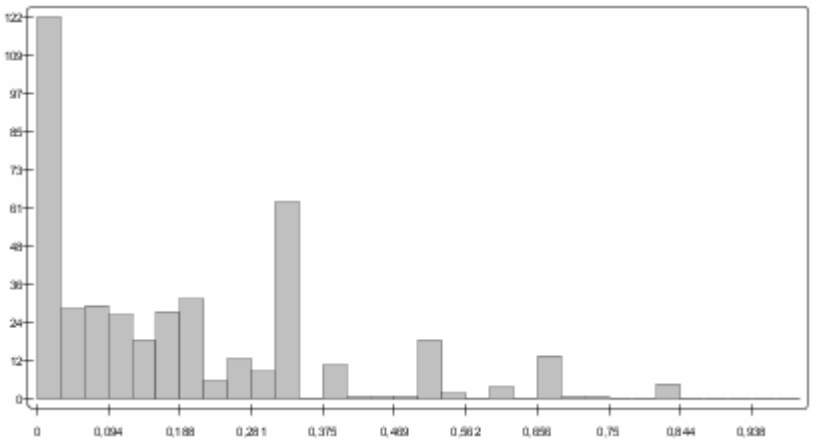


Figure 95 : graphe JT Arte filtré : longueur max = 7
1007 sommets, 2129 arêtes et IC = 0.289

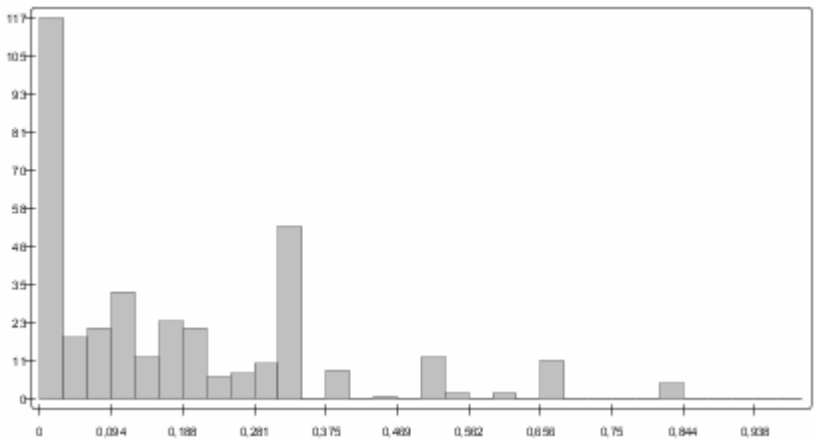


Figure 96 : graphe JT Arte filtré : longueur max = 5
1007 sommets, 1703 arêtes et IC = 0.255

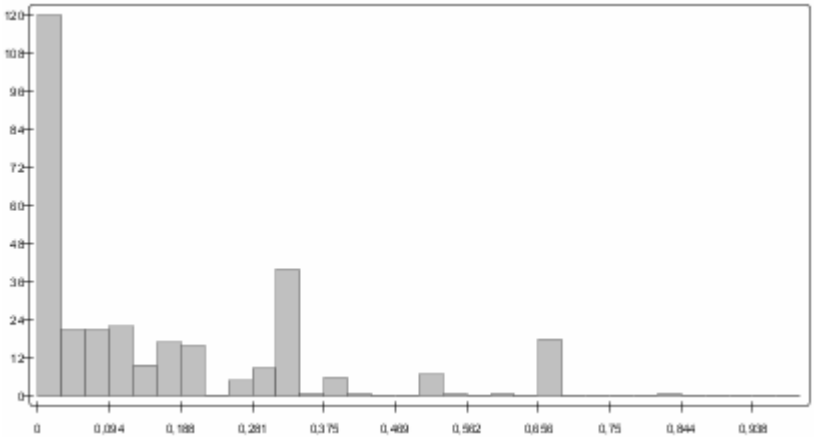


Figure 97 : graphe JT Arte filtré : longueur max = 3
1007 sommets, 1374 arêtes et IC = 0.23

L'évolution de la distribution consiste principalement en une diminution homogène de la population de chaque classe. Seule la population des sommets ayant un indice proche de 0 augmente, cette classe contient les sommets dont la plupart des arêtes ont été filtrées.

Au fur à mesure que des arêtes sont supprimées l'aspect du graphe placé évolue. Au départ très compact et semblable à un cluster géant, il est à chaque étape plus éclaté, tendant vers l'arbre couvrant, cf. Figure 98, Figure 99 et Figure 100.

Figure 98 : graphe JT Arte non filtré

Figure 99 : graphe JT Arte filtré : longueur max = 5

Figure 100 : graphe JT Arte filtré : longueur max = 3

Le filtrage par arbre couvrant minimal/maximal peut être appliqué en utilisant différentes métriques pour les arêtes. Lorsque le graphe est « naturellement » valué, par une mesure de poids exogènes comme l'occurrence dans l'exemple ci-dessus, on peut le filtrer à partir de l'arbre couvrant qui maximise ce poids. S'il n'est pas valué il est possible d'utiliser la *betweenness centrality* ou la edge force comme métrique à maximiser dans l'arbre couvrant. Une dernière solution consiste à placer le graphe avec un des modèles de force de Noack. Dans ce cas la longueur des arêtes est une métrique qui donne de bons résultats. Dans ce cas nous construisons l'arbre couvrant qui minimise la longueur des arêtes.

4.5.3.6 Identification des clusters

Nous proposons une méthode d'identification des clusters basée sur l'utilisation des modèles LinLog. Nous avons vu que ces modèles produisent des placements qui séparent les différents clusters d'un graphe. Ce placement peut donc être utilisé comme entrée d'un algorithme de clustering géométrique. Cette fonctionnalité a été implémentée très simplement dans Grapher. L'algorithme procède de la manière suivante :

```
Ajouter toutes les sommets dans la liste L.
Tant que L n'est pas vide
{
  Retirer le sommet v de L.
  Construire un arbre couvrant T à partir de v tel que :
       Les arêtes de T ont une longueur inférieure au seuil l.
  Marquer les sommets de T comme déjà visités.
  Retirer les sommets de T de L.
}
```

Cet algorithme consiste à filtrer les arêtes dont la longueur est supérieure à un seuil fixe, puis à identifier les composantes connexes. Chaque composante correspond à un cluster étiqueté de manière unique. Cet algorithme est de complexité linéaire.

4.5.3.7 Sélection d'entités par requêtes complexes

La sélection manuelle de sommets ou arêtes du graphe dans les tables d'entités peut devenir pénible lorsque le nombre d'entités est très grand ou lorsque les critères de sélection sont complexes.

Le parcours de grandes tables peut en effet prendre beaucoup de temps lorsqu'elles contiennent un grand nombre d'éléments. Les déplacements de la barre de déroulement perdent beaucoup de précision et le déplacement d'un seul pixel de cette barre peut correspondre à un mouvement de plusieurs pages dans la table. C'est un problème inhérent à toutes les tables de données, et de nombreuses idées ont été développées pour y remédier. Je citerais plus particulièrement TableLens [Rao94] qui permet de représenter synthétiquement un très grand ensemble de donnée dans une seule page. La table permet d'utiliser une représentation graphique particulièrement compacte pour les valeurs numériques et qualitatives. La Figure 101 illustre son utilisation sur des données automobiles.

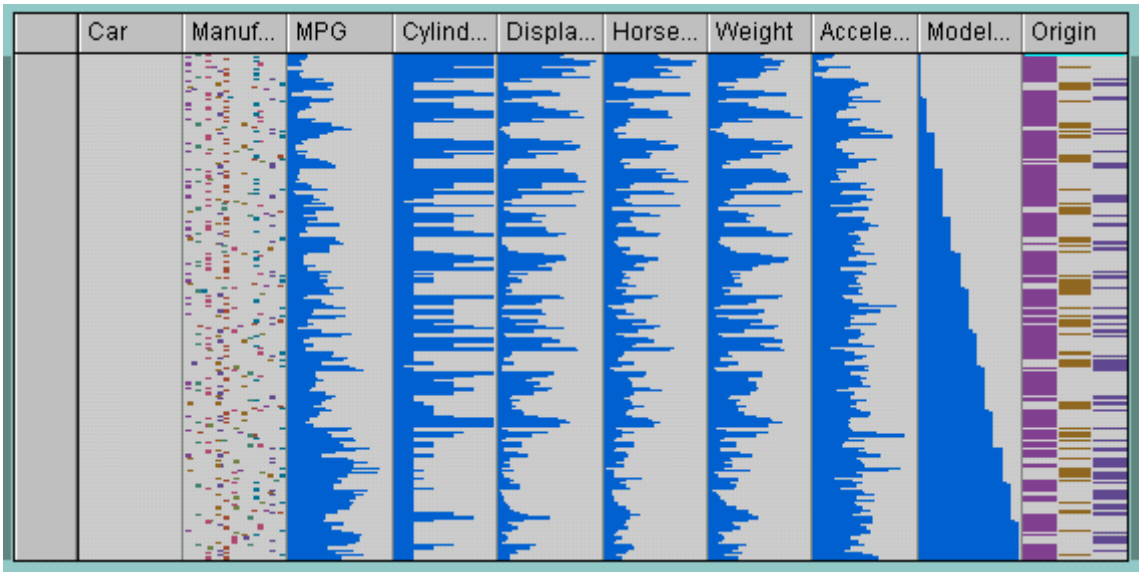
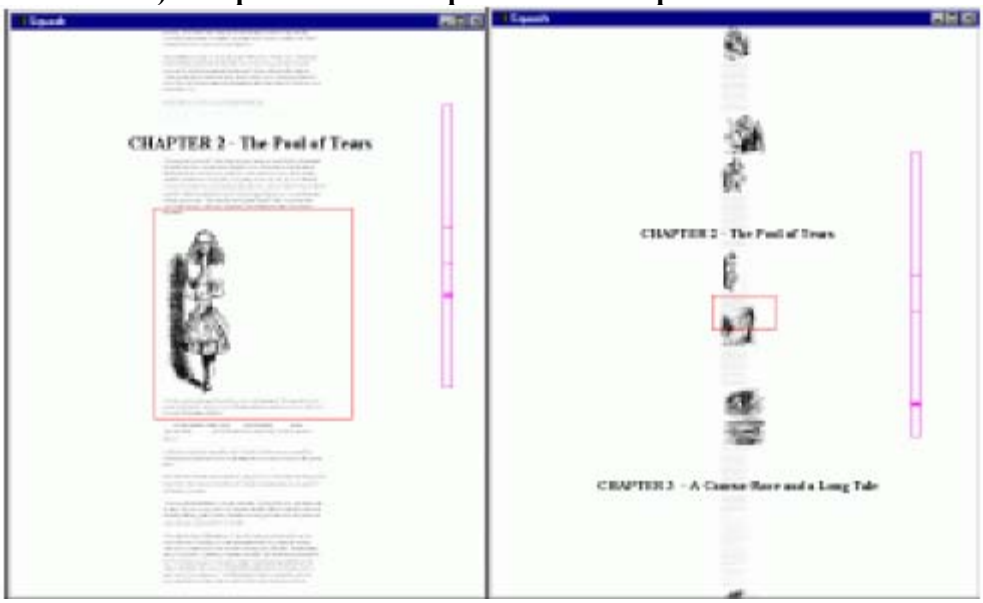


Figure 101 : TableLens

Une seconde technique [Igarashi00] paraît intéressante pour naviguer dans de très grandes tables. Conçue pour la navigation dans des documents composés d'un grand nombre de pages, l'application d'un zoom arrière lors du déroulement des pages permet d'obtenir un défilement en vitesse constante de l'affichage, et ce quelque soit la vitesse à laquelle est manipulé la barre de déroulement. Cette technique permet d'éviter la perte de contexte lors de défilements rapides. Cette technique semble toutefois être d'une utilité limitée dans le cas de données textuelles, le texte offrant généralement peu de repères visuels. La Figure 102 illustre le principe appelé *Speed-dependent Automatic Zooming* ou *SDAZ*, sur un document riche en illustrations. On voit sur cet exemple que les nombreuses illustrations offrent autant de repères lors d'un défilement rapide. Sans ces illustrations, il devient difficile de se repérer réellement dans une masse de texte affichée dans une petite échelle.

a) Comportement classique durant un déplacement lent



b) Zoom arrière durant un déplacement rapide
Figure 102 : Speed-dependent Automatic Zooming

Ces deux exemples rappellent qu'il est possible de naviguer dans une grande masse de donnée linéaire ou tabulaire, mais les solutions ne sont toutefois pas toujours faciles à mettre en oeuvre lorsque l'on utilise des composants graphiques existants comme dans notre cas la JTable Swing.

Par ailleurs, les fonctionnalités de tris multiples ne permettent pas toujours d'offrir un ordre de lecture compatible avec des critères de sélection complexes. C'est pourquoi nous avons choisi de développer un module de sélection par requêtes ou filtres complexes. Ce module n'a pas encore été développé mais nous avons déjà effectué plusieurs choix de conception que je développe ici.

Spécification des filtres

Un filtre ou une requête est un objet qui accepte et rejette des entités. L'interface **`Filter`** en spécifie le comportement élémentaire.

```
public Interface Filter
{
      public boolean acceptsEntity(Entity entity) ;
}
```

Les critères de filtrage portent généralement sur les valeurs d'un attribut des entités. La classe **`AttributeFilter`** implémente l'interface **`Filter`** en ajoutant la possibilité de spécifier un attribut et ses intervalles de valeurs autorisées.

Pour composer des filtres dont le critère dépend de plusieurs attributs, nous avons créé la classe **`ChainedFilter`** qui permet d'enchaîner successivement plusieurs filtres. Les filtres contenus dans une chaîne de filtres sont appliqués les uns après les autres, c'est une composition en série.

Les classes **`NoFilter`**, **`AndFilter`** et **`OrFilter`** offrent quant à elles la richesse de formulation du langage booléen. *And* et *Or* sont habituellement des opérateurs à deux arguments, pour des raisons pratiques nous les définissons comme des opérateurs n-aires. Cela permet entre autre de limiter le nombre de filtres booléens à créer, cf. Figure 104. On peut noter que le filtre booléen **`AndFilter`** applique ses filtres fils en parallèle.

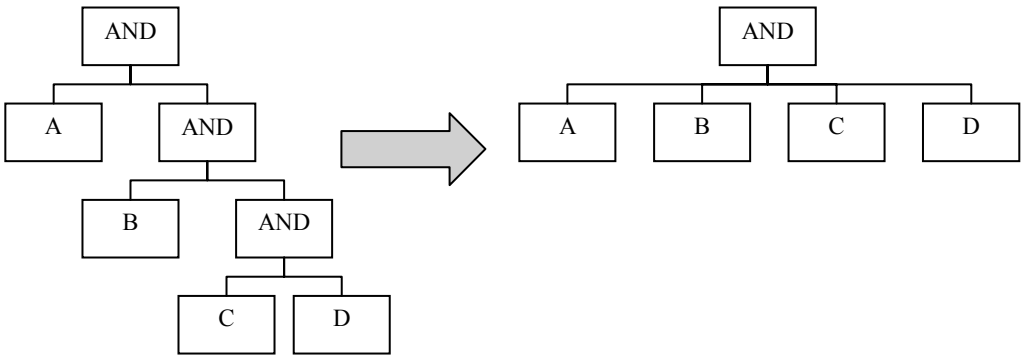


Figure 103 : opérateurs binaires et n-aires

Ces classes sont d'ors et déjà inclues dans Grapho. Nous devons à présent développer les composants graphiques qui permettent de les instancier en objets interactifs.

Pour les filtres de seuillage d'un attribut, nous avons choisi un composant qui affiche l'histogramme des valeurs de cet attribut sur lequel il est possible de spécifier à la souris les intervalles de valeurs à seuiller, cf. Figure 104. L'histogramme donne une bonne idée du nombre d'éléments rejetés par le filtre.

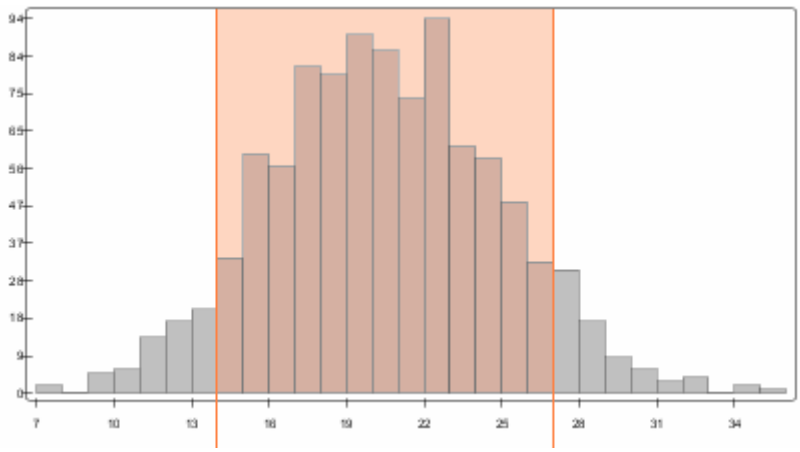


Figure 104 : Filtre sur les sommets dont le degré est inférieur à 14 ou supérieur à 27

Pour représenter les filtres complexes (chaînés et booléens) nous nous sommes inspirés des interfaces de gestions d'effets des logiciels de musiques. Les différents effets s'enchaînent, chaque effet prenant en entrée la sortie de l'effet précédent. C'est un comportement tout à fait similaire à nos filtres chaînés. Par exemple, le logiciel Tracktion [Tracktion] propose de composer une ligne d'effets par manipulation directe. Chaque effet est symbolisé par un petit composant graphique. Ces composants s'emboîtent les uns après les autres pour composer des effets complexes. Chaque composant permet de contrôler le paramètre principal d'un effet. Il peut aussi prendre le focus utilisateur afin d'accéder à tous les paramètres de l'effet. La Figure 105 illustre ces différentes fonctionnalités.

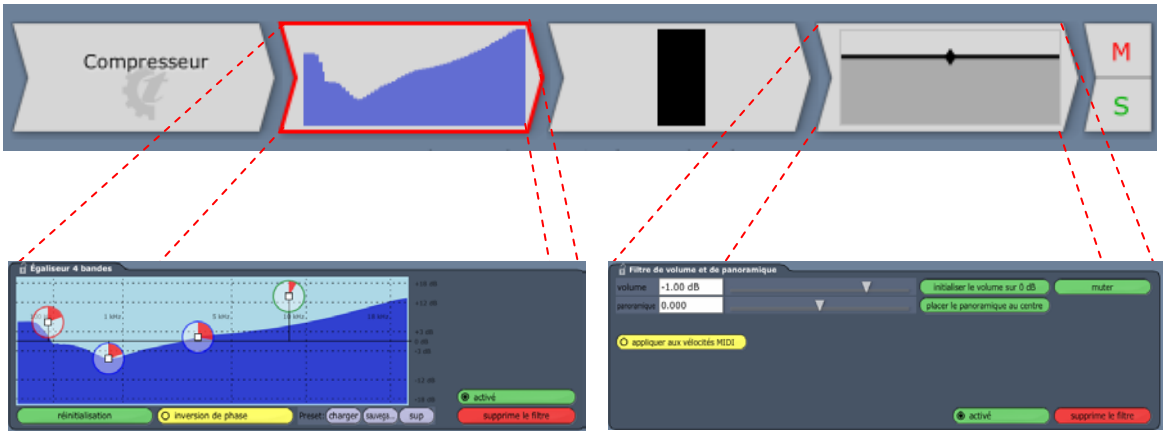


Figure 105 : Filtres chaînés du logiciel de musique Tracktion

Les expressions booléennes se composent de manières arborescentes, nous pouvons donc utiliser les algorithmes de placement d'arbres pour calculer la position des différents filtres. La Figure 106 montre ce à quoi pourrait ressembler l'interface de filtrage.

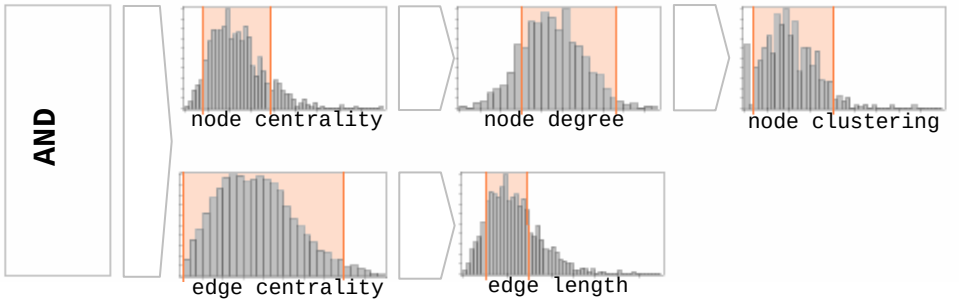


Figure 106 : Prototype d'interface pour les requêtes complexes

4.5.4 Calculs de métriques

4.5.4.1 Mesures globales

Cette fonction permet de calculer les différentes mesures caractéristiques d'un graphe :

- nombre de sommets,
- nombre d'arêtes,
- degré moyen,
- ratio d'arêtes,
- densité,
- diamètre et diamètre de la plus grande composante connexe.

Ces mesures sont calculées en temps constant excepté le diamètre qui est calculé en $\Theta\left(|V|^2\right)$.

4.5.4.2 Mesures de clustering

Nous avons implémenté deux mesures de clustering, l'indice de clustering des sommets [Watts99] et la mesure de force des arêtes [Auber03a]. La force d'une arête est un équivalent à l'indice de clustering pour un sommet, les calculs sont de la même nature. Les arêtes de force faible peuvent être filtrées afin d'isoler les clusters d'un graphe. Ces deux mesures sont normées sur [0,1] et peuvent être utilisées pour un codage par couleur ou par taille.

4.5.4.3 Mesures de centralités

Nous avons implémenté le calcul de la mesure de betweenness centrality [Freeman77]. Nous avons adapté le meilleur algorithme actuel [Brandes01] avec une complexité temps en $\Theta\left(|V|\cdot|E|\right)$. De plus, la mesure a été étendue aux arêtes.

4.5.4.4 Mesures géométriques

Les mesures géométriques se basent sur la structure et la position des sommets. La fonction de mesure des longueurs d'arêtes calcule simplement la longueur de chaque arête. Nous avons utilisé cette fonction pour nos études du comportement des différents modèles de force pour le placement de graphes. Le clustering géométrique est une fonction plus utile. Il se base sur la longueur des arêtes pour identifier des clusters dans un graphe. Le processus est simple, si deux sommets sont liés par une arête dont la longueur est inférieure à un seuil donné, alors ils appartiennent au même cluster. La méthode est ici appliquée plusieurs fois avec des seuils croissants, ce qui permet d'obtenir une

hiérarchie de clusters. L'algorithme a une complexité temps linéaire. Chaque entité se voit assigner une étiquette pour chaque seuil, qui nomme le cluster auquel elle appartient. Cette étiquette peut ensuite être codée par une couleur.

4.5.5 Codage graphique

4.5.5.1 Couleur

Nous avons intégré deux méthodes de codage couleur. La première méthode est basée sur l'interpolation de couleur dans les principaux modèles de couleurs, RGB, HSB, CIEXYZ, CIELAB et CIELUV. L'utilisateur est donc inviter à sélectionner les deux couleurs extrêmes de son échelle de couleur qui coderont les valeurs minimales et maximales, et un modèle de couleur. Les couleurs intermédiaires sont interpolées linéairement dans le modèle choisi. L'appariement couleur / valeur sera appliqué linéairement ou par quantiles, cf. section 3.4.4.

La seconde option est d'utiliser les différentes échelles de couleurs proposées par Brewer. Ces échelles sont regroupées en trois catégories, ordonnée, divergente et qualitative. Les deux méthodes d'appariement linéaire et par quantiles sont sélectionnables.

4.5.5.2 Taille

Notre implémentation du codage de valeurs par la taille correspond exactement aux méthodes définies en section 3.4.4. L'utilisateur choisit donc une taille minimale et une taille maximale qui coderont respectivement les valeurs minimale et maximale. L'appariement pourra être linéaire ou par quantiles.

4.5.6 Algorithmes de placement

4.5.6.1 Placement aléatoire

Les sommets du graphe sont positionnés de manière aléatoire dans l'affichage. Ce placement peut être utilisé comme phase d'initialisation pour un placement par modèle de force.

4.5.6.2 Placement par modèles de force

L'implémentation des modèles de force provient de Prefuse [Heer05]. Cette implémentation utilise des 4-tree afin d'optimiser le calcul des forces de répulsions. Prefuse est bien conçu, et permet notamment de facilement créer de nouveaux modèles de forces. Les modèles implémentés sont :

- forces dites naturelles
- une version simplifiée du Kamada-Kawai prenant en compte la distance théorique entre sommets
- Fruchterman-Reingold
- Eades
- Node LinLog
- Edge LinLog

Les placements à base de modèles de forces peuvent être animés durant le processus de calcul. Deux formes d'animation sont supportées :

- animations des positions avec transitions calculées par interpolation
- rafraîchissements des positions à intervalles réguliers

Par défaut, l'API utilise le rafraîchissement à intervalles réguliers pour des raisons de performances. En effet, sur de grands graphes le temps pris pour un rafraîchissement des positions est trop grand pour pouvoir obtenir des animations assez fluides. Cependant, il est possible d'utiliser une animation plus fine, dans ce cas nous procédons par interpolation entre la position actuelle et la nouvelle position. Attention cette méthode ne fonctionne que sur des graphes de très petite taille.

Les modèles de forces standard convergent généralement assez vite en partant d'un positionnement aléatoire, par contre les placements Node et Edge LinLog convergent très lentement. C'est pourquoi il est souvent intéressant pour ces deux modèles de partir des positions calculées par un modèle standard.

4.5.6.3 Placement radial de l'arbre couvrant

Cette méthode consiste à construire l'arbre couvrant d'après un sommet sélectionné par l'utilisateur, puis à le placer dans un espace en coordonnées polaires. L'arbre couvrant est issu d'un parcours en largeur d'abord. Le choix de l'arbre couvrant et la méthode de placement assurent que la distance d'un sommet au sommet focus indique sa distance géodésique au focus. Nous nous sommes basés sur l'implémentation Prefuse [Heer05] de cet algorithme. Cette implémentation permet d'animer la transition entre deux changements de focus. Les transitions animées assurent la continuité de la représentation, ce qui facilite la mémorisation du contexte lors des changements de focus [Yee01].

4.5.7 Discussion autour des performances

Cette section présente les résultats des performances des deux principaux modules de Grapho, le modèle et l'affichage. Nous avons procédé à trois types de mesures :

- temps de mises à jour du modèle seul
- temps de mises à jour de la présentation
- temps d'un cycle d'affichage

4.5.7.1 Environnement d'évaluation des performances

Je prendrai comme machine de référence un PC avec CPU à 2Gh et mémoire de 1Go, équipée d'une carte vidéo standard. Au niveau logiciel, le système d'exploitation est Windows XP, et j'utilise la version 1.5 la machine virtuelle Java.

4.5.7.2 Le Modèle

Nous présentons ici les performances du Modèle en tant que module autonome. Les différentes mesures sont donc effectuées sans activer le module de Présentation. Nous mesurons les temps d'exécution de différentes opérations et l'occupation mémoire du Modèle.

Nous avons choisi de mesurer les temps d'exécution des opérations de modifications du Modèle les plus représentatives :

- création d'un graphe,
- modifications d'un attribut des entités,
- filtrage et suppression de la moitié des sommets et de leurs arêtes adjacentes,
- filtrage et suppression de la moitié des arêtes.

Le graphique de la Figure 107 montre que l'ensemble des ces opérations sont exécutées dans des temps linéaires au nombre d'entités du graphe. On constate que l'opération de création est significativement la plus coûteuse. Un zoom sur les autres opérations, en Figure 108, montre que le filtrage est plus rapide que la suppression d'entités. Cette dernière opération est de complexité comparable à la modification d'attributs.

Les performances du modèle sont tout à fait correctes pour une application Java. Le filtrage est particulièrement rapide et cela même avec des graphes de très grandes tailles. Le processus de création est nettement plus lent que les autres opérations, mais cela peut s'expliquer par les temps de création des objets Java. De plus, la création est une opération peu critique dans une application réelle, car elle n'a lieu vraisemblablement qu'une fois, à l'initialisation. Les opérations de filtrage, et de modifications des attributs sont nettement plus critiques, et on peut constater qu'elles s'exécutent toujours dans des temps très raisonnables au vu du nombre d'objets.

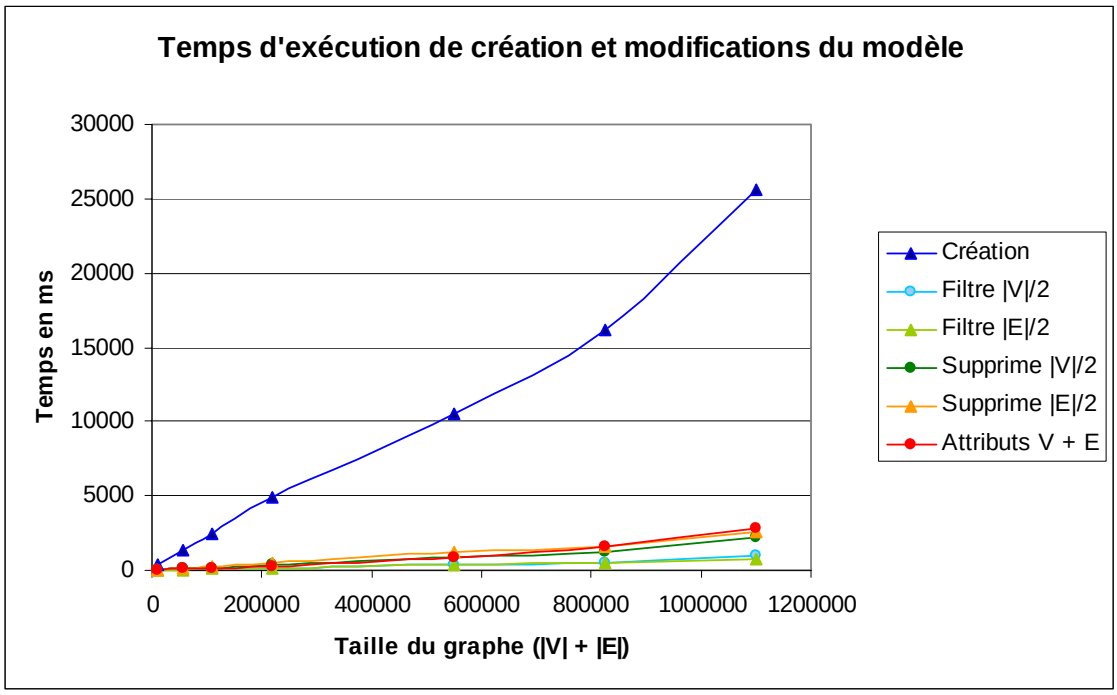


Figure 107 : Temps d'exécution de création et de modifications du modèle

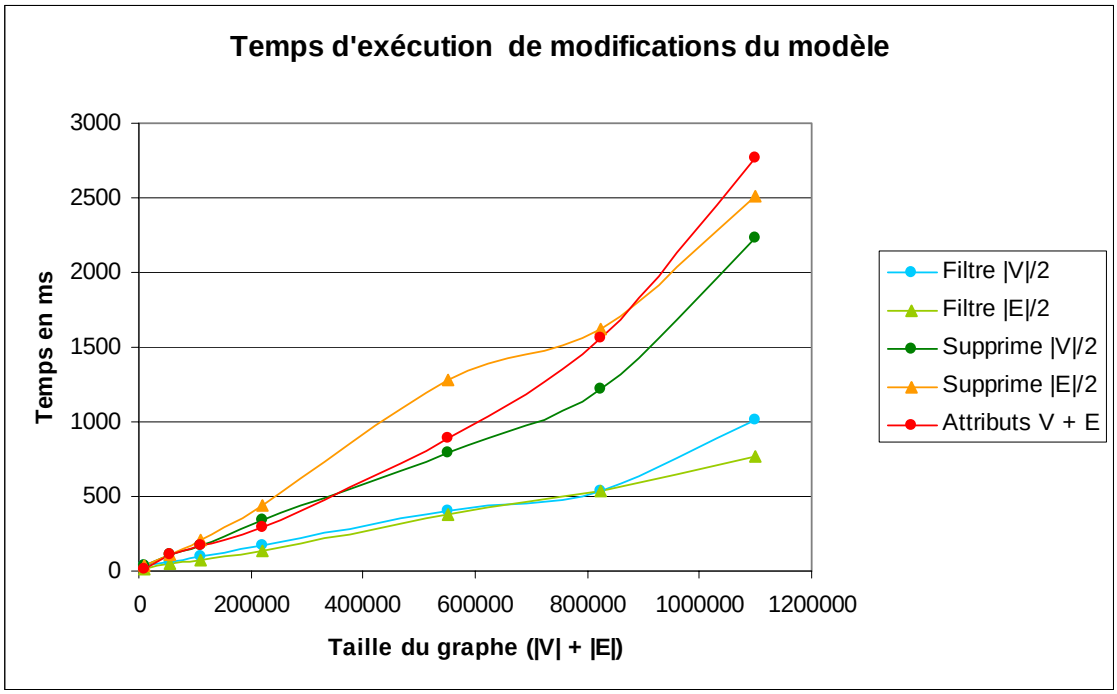


Figure 108 : Temps d'exécution de modifications du modèle

Au niveau occupation mémoire, le modèle est relativement gourmand, cf. Figure 109. Chaque entité du graphe occupe en moyenne 500 octets. Cela implique qu'il est difficile de dépasser les un à deux millions d'entités avec une machine actuelle. Toutefois, ce chiffre dépasse largement la taille des graphes que nous avons eus à traiter jusqu'à présent.

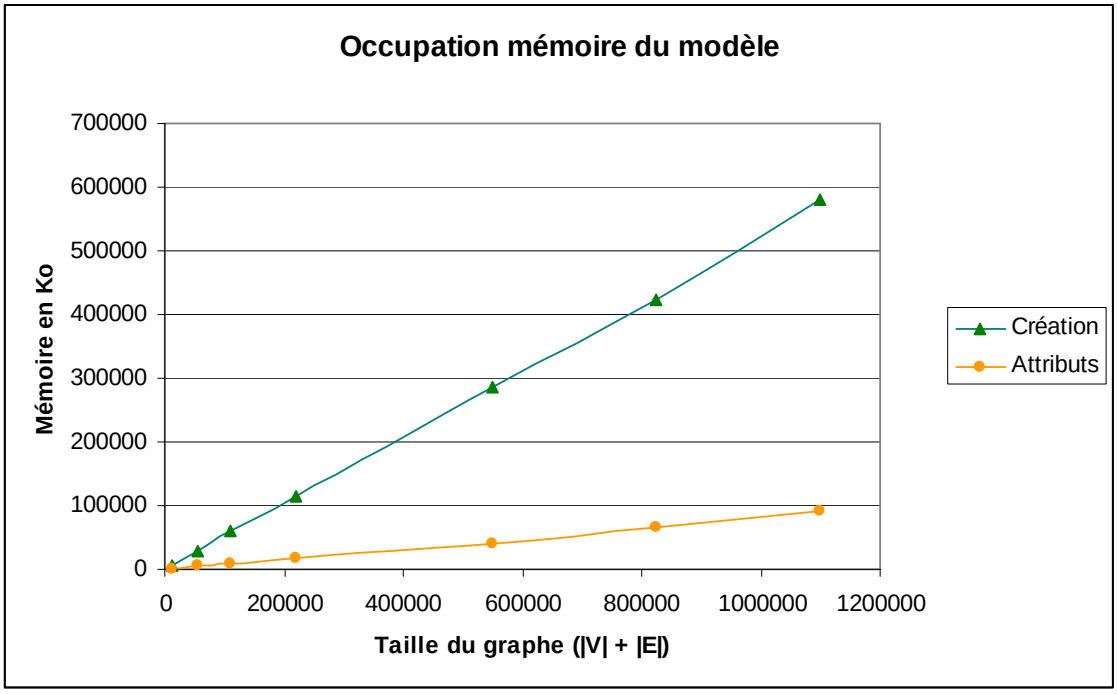


Figure 109 : Occupation mémoire du modèle

Le Modèle de Grapho offre des performances tout à fait respectables, aussi bien en temps d'exécution qu'en occupation mémoire. Les performances du filtrage temporaires sont quant à elles

très satisfaisantes. Le Modèle est bien adapté à des graphes de plusieurs centaines de milliers d'entités, qui se décomposent le plus souvent dans les graphes que nous avons pu observer en quelques dizaines de milliers de sommets et quelques centaines de milliers d'arêtes.

4.5.7.3 Présentation

Lorsque l'interface de visualisation de graphe est activée, la Présentation est chargée en mémoire. Elle est mise à jour à chaque modification du Modèle. Nous mesurons ici ses performances, en mesurant à nouveau, les temps d'exécution des opérations les plus fréquentes et l'occupation mémoire.

Sur le premier graphique, cf. Figure 107, sont représentées les courbes des temps des mises à jour de la Présentation suite à des opérations de création et de suppression. Ces opérations impliquent de modifier le graphe de scène Piccolo. La modification de l'arborescence Piccolo implique un surcoût important.

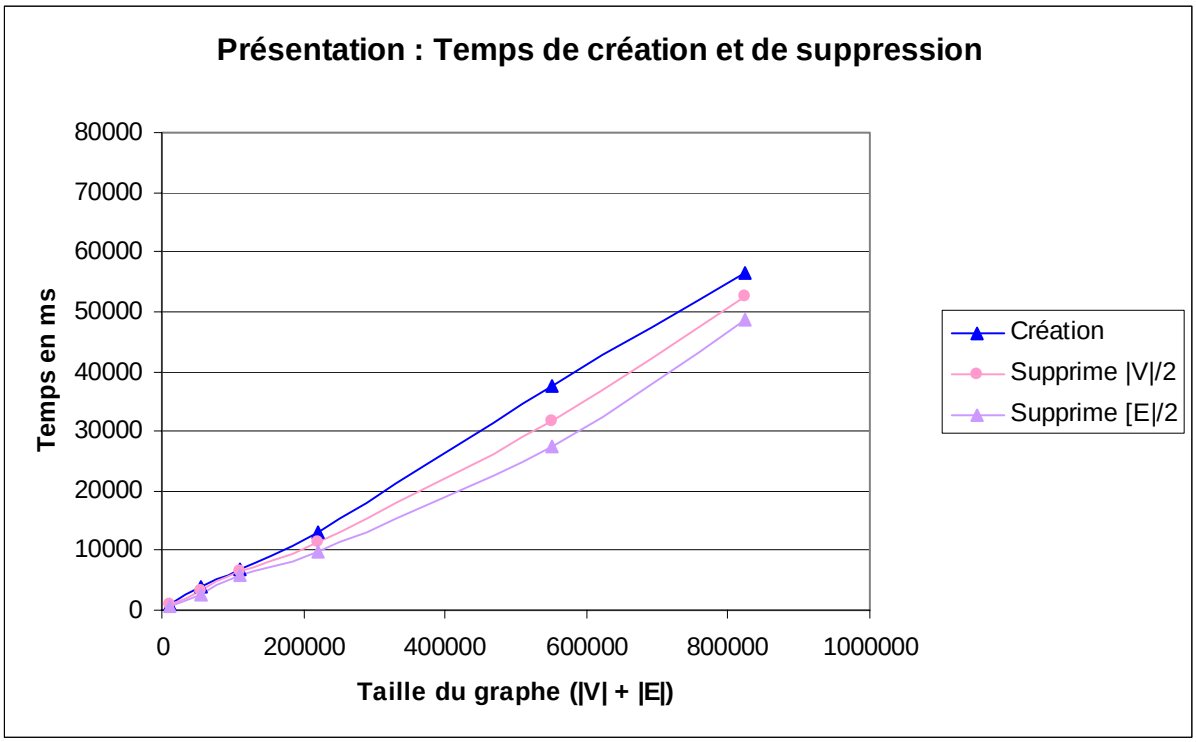


Figure 110 : Temps de mise à jour de la Présentation

Le filtrage et le changement de visibilité des entités sont de complexités comparables, cf. Figure 108. Pas de surprise car les objets visuels filtrés sont simplement masqués. Les temps pour repositionner un sommet ou une arête sont comparables. Cependant, les arêtes sont beaucoup plus nombreuses que les sommets dans cette expérience mais aussi dans les graphes issus de données réelles. Le repositionnement des arêtes est donc toujours beaucoup plus coûteux que pour les sommets.

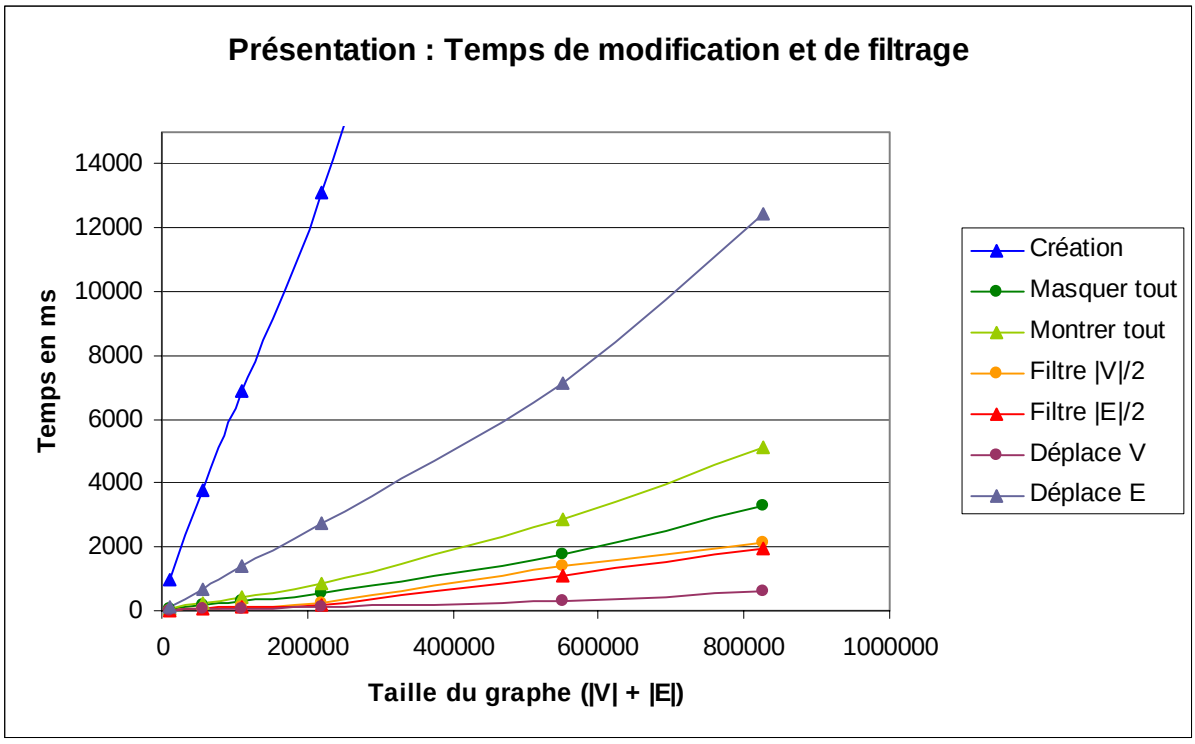


Figure 111 : Temps de mise à jour de la Présentation (zoom)

La création et la suppression d'entités, et le changement de positions des arêtes posent réellement problème dans le module de Présentation. Ces trois opérations sont en effet particulièrement coûteuses et pourraient être un obstacle à l'utilisation de Grapho pour visualiser des graphes de plusieurs centaines de milliers d'entités. Ce n'est pas pourtant pas le cas. Les processus de création et de suppressions sont de loin les plus coûteux. La création n'a cependant lieu qu'une fois, durant l'initialisation de la Présentation. Et la suppression peut être avantageusement remplacée par un filtrage. Reste le repositionnement des arêtes. Une arête est repositionnée automatiquement lorsqu'un de ses sommets a été déplacé. Lors de l'exécution d'un algorithme de placement, les positions des sommets sont modifiées à intervalles réguliers. Dans ce cas le repositionnement des arêtes ralentit effectivement énormément tout le processus de placement. Le repositionnement des sommets reste par contre très rapide et généralement négligeable. Nous offrons donc la possibilité à l'utilisateur de désactiver l'affichage et le repositionnement des arêtes durant les calculs de placement.

Les temps cumulés de mises à jour du Modèle et de la Présentation restent faibles pour les opérations de filtrage et de déplacement, les plus utiles lors de l'exploration. Jusqu'à 200 000 entités ces opérations s'exécutent en moins d'une seconde. Le seuil des 10 secondes est dépassé à partir de 800 000 entités.

Présentation et Modèle occupent approximativement le même espace mémoire. Lors de la visualisation d'un graphe avec Grapho, il faut donc compter près de 1 Ko de mémoire occupée par entité du graphe. Sur la machine de référence, nous sommes donc limités à des graphes de moins de 1 000 000 d'entités.

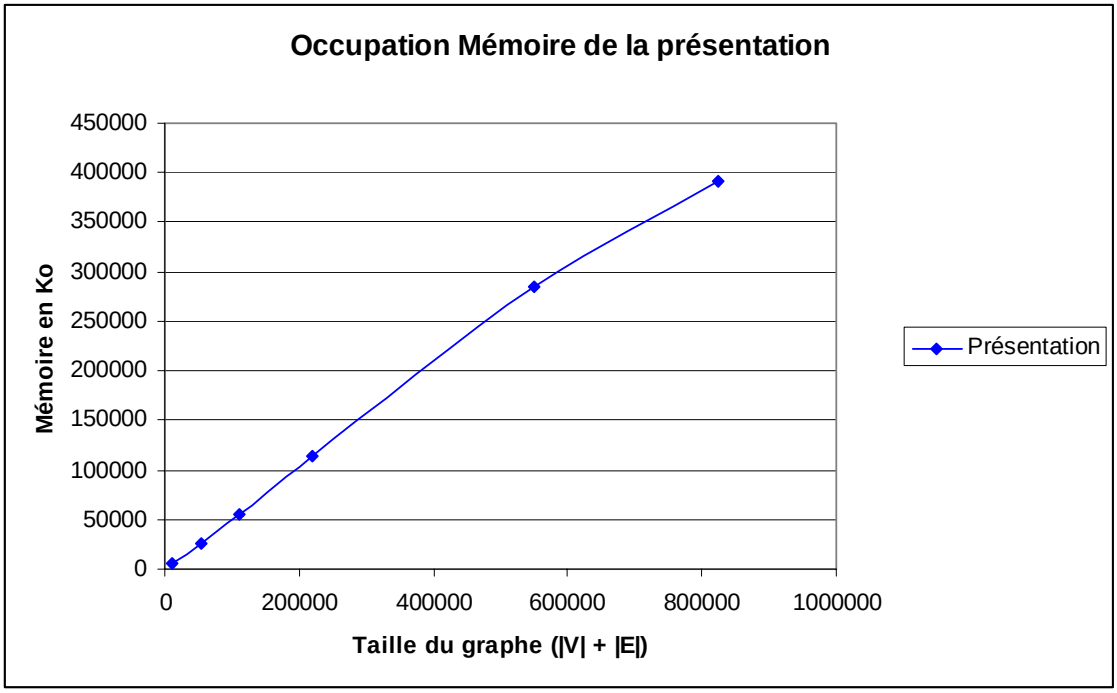


Figure 112 : Occupation mémoire du module de Présentation

4.5.7.4 L'Affichage

Le temps d'affichage est une caractéristique critique. La manipulation de la Présentation lors d'une exploration implique des changements très fréquents du point d'observation de l'utilisateur. Chaque changement de point de vue, ainsi que chaque modification du graphe nécessite au moins un réaffichage. De plus, nous avons recours de manière intensive à l'animation, pour les changements de points de vue et les changements de positions des sommets. La fluidité de l'animation dépend entièrement des performances de l'affichage.

Les graphiques suivants montrent l'évolution du temps pris pour redessiner les sommets et arêtes d'un diagramme *nœud-lien*. Ainsi, sur la Figure 113, nous constatons que dessiner un sommet est en moyenne quatre à cinq fois plus coûteux que de dessiner une arête.

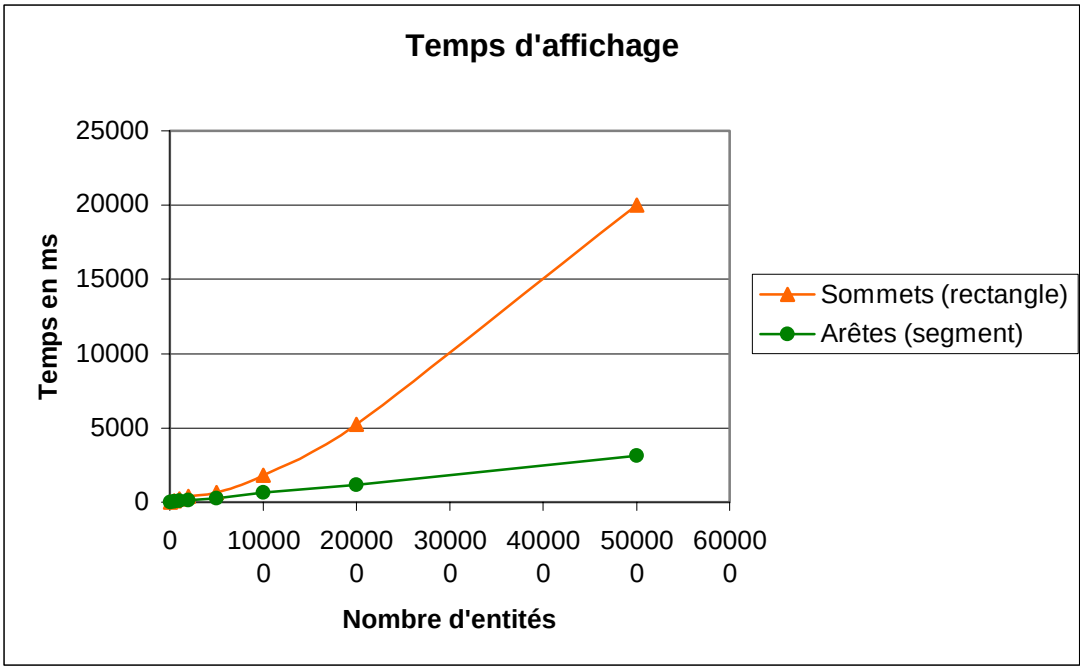


Figure 113 : Affichage des sommets et arêtes

Ces résultats correspondent aux performances du processus d'affichage des primitives graphiques. Le rendu d'un rectangle ou de toute autre forme dont l'intérieur est coloré est toujours plus coûteux que le rendu d'un segment de droite.

Afin d'obtenir des animations parfaitement fluides, la vitesse d'affichage doit être proche des 25 images par secondes, ce qui correspond à un temps d'affichage d'approximativement 40 ms. Toutefois, pour certains types d'animations nous pouvons tolérer des vitesses d'affichage beaucoup plus faibles. Les changements de points de vue de type pan ou zoom sont programmés sur une durée de deux secondes, ils tolèrent des vitesses d'affichages de l'ordre de 4 à 5 images, ce qui fait au total une dizaine d'images pour toute l'animation. Ce seuil de 4 images par seconde correspond à une vitesse de 250 ms.

Les figures Figure 114 et Figure 115 présentent à nouveaux les temps d'affichage des sommets et arêtes. Les temps de référence pour une bonne fluidité des animations et une interactivité confortable y sont ajoutés. En 40 ms il est possible de dessiner seulement quelques milliers de sommets ou une dizaine de milliers d'arêtes. En 250 ms, on peut dépasser la dizaine de milliers de sommets ou près de 50 000 arêtes.

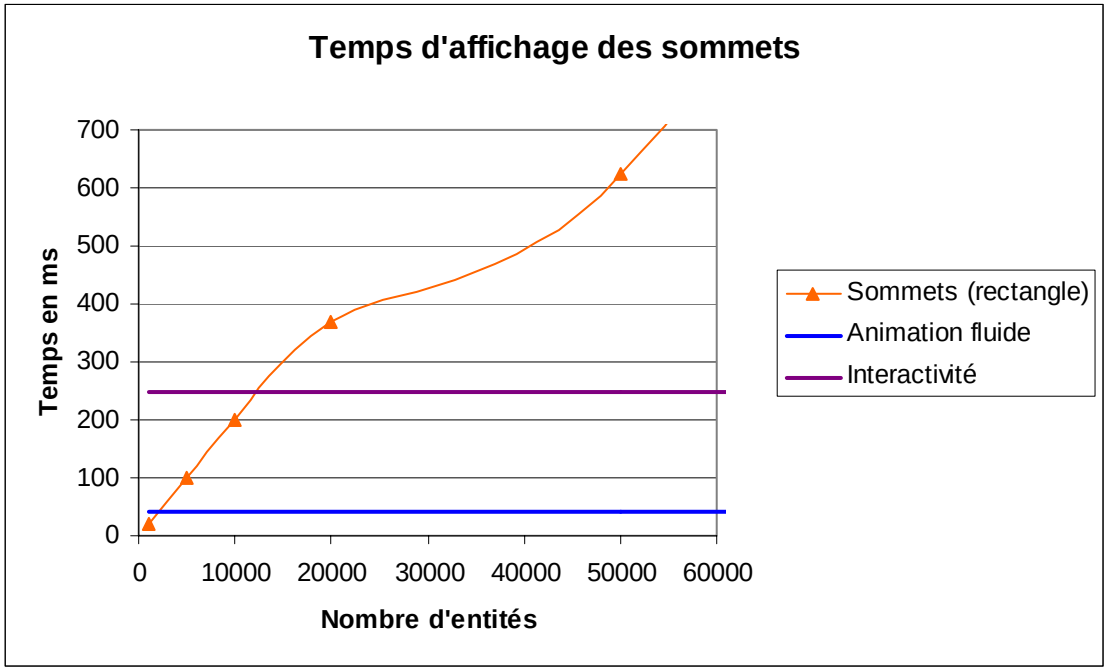


Figure 114 : Affichage des sommets

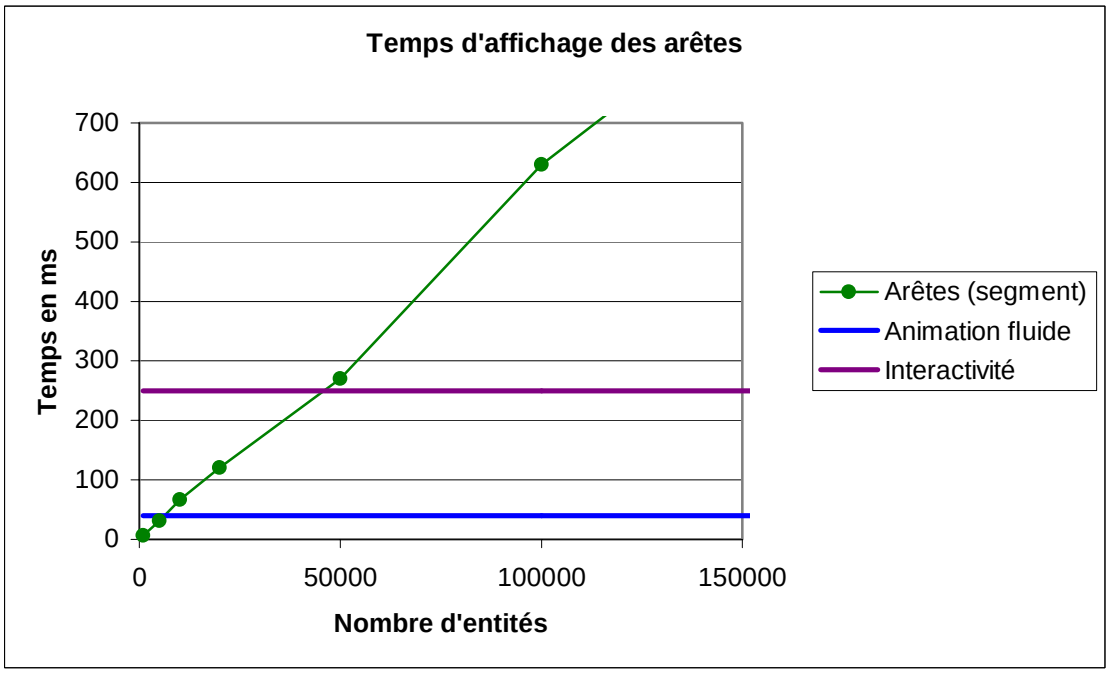


Figure 115 : Affichage des arêtes

Les performances de l'affichage contraignent donc énormément les possibilités de l'application en imposant des limites bien plus drastiques que pouvaient le faire les performances du Modèle ou de la Présentation. Les différents graphes sur lesquels nous avons mené des expérimentations, cf. 4.6, sont composés de moins de 5 000 sommets mais peuvent impliquer plusieurs dizaines de milliers d'arêtes. Le temps d'affichage des sommets est dans ce cas tout à fait raisonnable, par contre pendant des tâches d'exploration intensives ou de repositionnement du graphe le temps d'affichage des arêtes nous oblige à les masquer.

Il est possible d'améliorer considérablement les performances de l'affichage en désactivant certaines caractéristiques des objets graphiques. La méthode de tracé des contours, par exemple, contrôle l'épaisseur et le type des contours. En Java cette fonctionnalité altère considérablement les performances du rendu de toutes les primitives graphiques. La désactivation de cette fonctionnalité peut donc améliorer les performances du dessin des sommets et arêtes. Toutefois, on perd alors la possibilité de codage d'un attribut des arêtes par la taille qui est implémenté directement en utilisant les objets Stroke.

Ces performances d'affichage assez médiocres proviennent de l'implémentation du rendu Java, qui n'utilise que très anecdotiquement les possibilités d'accélération matérielle offertes par les cartes graphiques. Cependant, les choses devraient changer très prochainement avec la version 1.6 de l'API Java. Le processus de rendu doit être en partie remanié afin de pouvoir utiliser efficacement la librairie OpenGL compatible avec la grande majorité des cartes graphiques 3D. Les vitesses d'affichages pour les primitives de type segment de droite et rectangle seront respectivement multipliées par un facteur 5 et 10, comme on le constate sur le graphique de la Figure 116 émis par le responsable du module d'affichage de l'API Java.

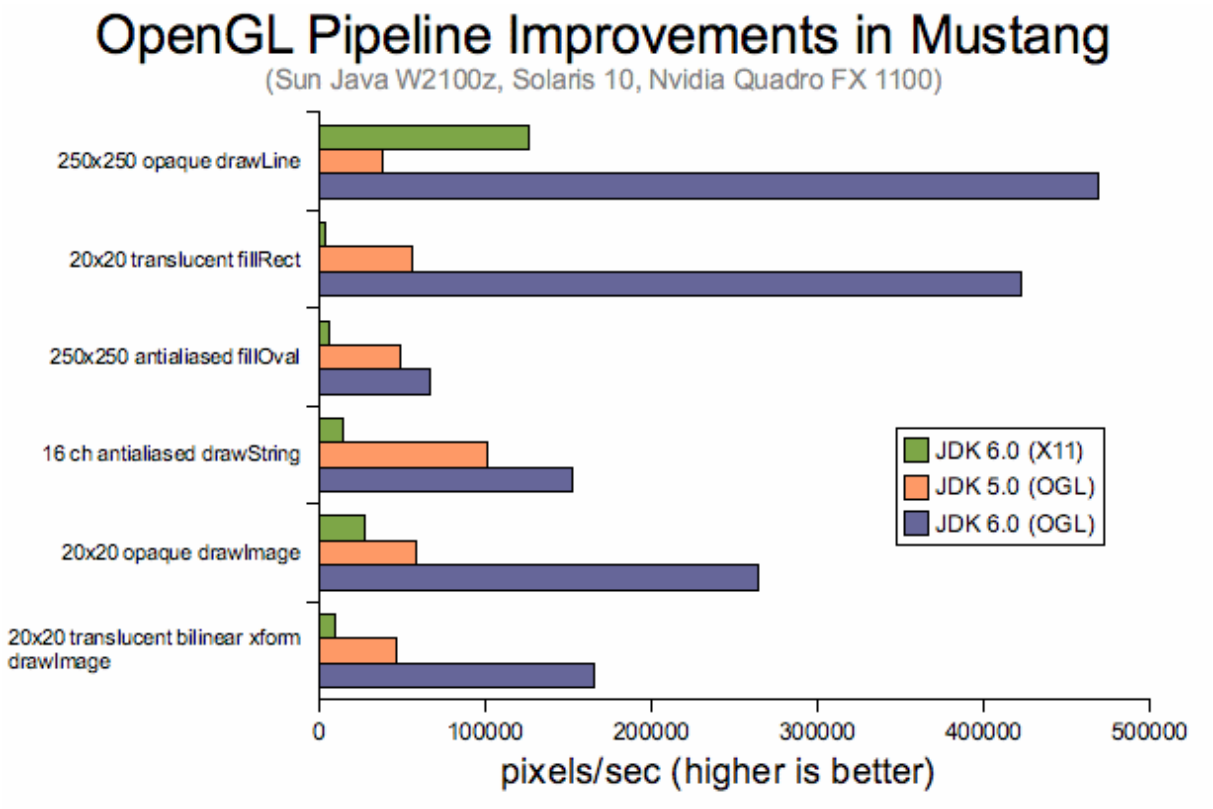


Figure 116 : Amélioration des performances de rendu avec OpenGL

Ces futures améliorations devraient donc nous permettre de visualiser confortablement des graphes de taille bien plus conséquente.

4.6 Les applications

4.6.1 Baromètre thématique de l'information : Cartographie des JT (Ina)

4.6.1.1 Motivation

Disposant de la collection complète des journaux télévisés des chaînes publiques et des chaînes privées jusqu'en 1993, l'Ina se trouve dans une position privilégiée pour établir une cartographie de l'information télévisuelle. En effet, chaque sujet de journal télévisé fait l'objet d'une indexation documentaire composée d'un résume synthétique, d'une description par mots clés ou grille sémantique, et d'une description générique. La description par mots clés constitue pour plusieurs raisons une excellente base pour la réalisation d'une cartographie de l'information. En effet, la fonction essentielle du langage documentaire, qui permet la désambiguïsation du langage naturel, assure que deux sujets sémantiquement distincts ne seront pas corrélés par la polysémie d'un terme. D'autre part, l'utilisation de grilles sémantiques et de termes choisis pour la description d'évènements type génère des groupes redondants de co-descripteurs, le plus souvent distincts les uns des autres. Dans une modélisation sous forme de graphe, la co-description génère des agrégats de type « petits monde ». De plus, les descripteurs ont un degré de généricité variable et peuvent donc avoir un nombre de co-descripteurs très différents. Par exemple, un sujet sur le naufrage de l'Erika en 2000 sera décrit par : *catastrophe, pollution, naufrage, Bretagne, pétrole, plages*. *Catastrophe* est un terme très générique qui est associé à plus d'une trentaine d'évènements et à plus de 130 descripteurs, alors que *plage* est relié uniquement à 6 autres descripteurs. De par ces caractéristiques, il semble intéressant de modéliser la carte de l'information par un graphe dont les entités représentent les termes descripteurs, et les arêtes la relation « décrit un même sujet de journal télévisé». En effet, ces données présentent des caractéristiques structurelles de graphe « sans échelle », combinées avec des parties « petit monde ».

4.6.1.2 Construction des graphes des JT

Les données nous ont été fournies par le dépôt légal sous formes de fichiers XML. Chaque journal télévisé est associé à un ficher XML qui le décrit. Dans cette description le journal télévisé est représenté comme une liste de sujets. Un sujet est caractérisé par un titre, un résumé, des mots-clés descripteurs et les journalistes et intervenants qui y participent. L'ensemble des mots-clés d'un sujet permet comme nous l'avons déjà dit d'en caractériser synthétiquement et sans ambiguïté le contenu.

La première phase pour la construction d'un graphe de JT est de sélectionner un ensemble de JTs ayant un sens. Nous avons choisi de représenter l'information par chaîne sur l'année 2000. Nous avons donc obtenu les données relatives aux cinq chaînes suivantes : France2, France3, Canal+, Arte et M6 afin de comparer leur contenu informationnel.

Nous allons construire un graphe pour chacun de ces ensembles. Je rappelle que dans ces graphes, chaque mot-clé est représenté par un sommet, et qu'il existe une arête entre deux sommets si les deux mots-clés correspondants sont contenus dans un même sujet. Le processus de construction est simple, chaque sujet contient une liste de mots-clés, qui correspondra alors à une *clique* de sommets dans le graphe. On souhaite, de plus, connaître combien de fois est apparu un mot-clé, et combien de fois ont été utilisés conjointement deux mots-clés, dans l'ensemble des sujets. Cette information est calculée durant le processus de construction et stockée dans les sommets et arêtes comme un attribut de type entier nommé *occurrence*.

Ce processus de construction induit une propriété particulière sur les graphes que nous obtenons, à savoir la présence de nombreuses cliques de tailles variables. Cette propriété implique que l'indice de clustering d'un graphe construit de cette manière sera élevé. On peut préciser que cette technique génère des graphes non orientés simples, sans boucles ni arêtes parallèles. Toutefois une arête d'occurrence *k* peut être interprétée comme k arêtes parallèles, dans ce cas le graphe n'est plus simple. Nous avons fait le choix de cette interprétation, et les différents algorithmes de placements que nous utilisons prennent en compte cet élément. Pour les différentes mesures, nous considérons les deux manières de compter les arêtes, avec ou sans l'occurrence. Les mesures avec prise en compte de l'occurrence sont notées X* dans le tableau ci-dessous.

	N	M	M*	|degré|	|degré*|	densité	densité*	Diam	C	C*
France2	1569	7686	14091	9.8	18.0	0.0062	0.0115	7	0.449	1.984
Aléatoire	1569	7688	-	9.8	-	0.0062	-	6	0.005	-
France3	1384	5065	8366	7.3	12.1	0.0053	0.0087	9	0.428	1.341
Aléatoire	1384	5072	-	7.3	-	0.0053	-	7	0.005	-
Canal+	750	2244	3573	6.0	9.5	0.0080	0.0127	8	0.504	1.083
Aléatoire	750	2247	-	6.0	-	0.0080	-	7	0.005	-
Arte	1007	3423	5913	6.8	11.7	0.0068	0.0117	8	0.383	1.176
Aléatoire	1007	3429	-	6.8	-	0.0068	-	7	0.006	-
M6	1406	4577	7539	6.5	10.7	0.0046	0.0076	9	0.406	1.231
Aléatoire	1406	4582	-	6.5	-	0.0046	-	8	0.004	-

Tableau 9 : profil des cinq graphes de JT

Le Tableau 9 présente les cinq graphes de JT pour l'année 2000. On peut comparer les mesures de chacun de ces graphes avec celles de son graphe aléatoire « équivalent ». On constate notamment que le diamètre d'un graphe de JT est très légèrement supérieur à celui de son homologue aléatoire et surtout que son indice de clustering est de 60 à 100 fois plus grand. Les distributions des degrés des graphes de JT suivent une loi de puissance (cf. Figure 117), ces différentes caractéristiques sont celles des graphes sans échelle. Les graphes obtenus partagent tous ces caractéristiques.

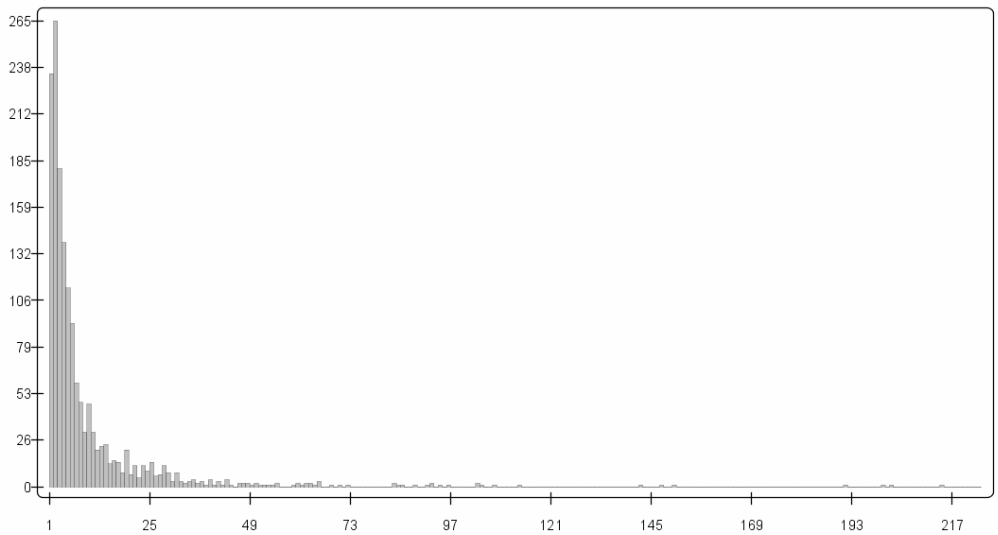


Figure 117 : Distribution des degrés du graphe des JT de France2

4.6.1.3 Analyse visuelle

Ainsi, pour obtenir la Figure 118, nous avons sélectionné le corpus des journaux télévisés de la chaîne Arte pour l'année 2000. Nous avons généré le graphe des descripteurs sur l'ensemble du corpus, qui apparaît le plus souvent comme un agrégat dense de points et d'arêtes, inexploitable visuellement. Tout d'abord nous avons éliminé les descripteurs d'occurrence faible qui représentent des sujets d'actualité peu traités, puis les arêtes d'occurrence faible afin de ne garder que les liaisons les plus significatives des amas. Nous avons permis l'étalement du graphe et l'émergence de regroupements de termes signifiants en éliminant temporairement certaines arêtes isolées reliant deux groupes denses. Dans certains cas, le filtrage de quelques nœuds de degré très élevé peut s'avérer extrêmement efficace pour améliorer la carte. En effet, ces nœuds perturbent la mise à plat du graphe en rassemblant, au centre du graphe, des groupes potentiellement distincts. Dans le cas où les entités représentent des termes, il s'agit le plus souvent de termes « outils » multi facette comme *manifestation*. Les liens et les descripteurs filtrés peuvent être réintégrés à la visualisation dès qu'un placement satisfaisant est réalisé.

Chaque attribut peut être associé interactivement à un attribut graphique afin de mettre en évidence visuellement les groupes ou l'importance d'un terme ou d'une liaison. Ainsi sur la Figure 118, un dégradé de couleur (vert-rouge) est associé à la « durée d'usage cumulée » du nœud sur l'année, marquant en rouge les termes correspondant aux durées de traitement les plus élevées sur l'année. La taille est associée à l'occurrence du terme, introduisant ainsi une échelle de visibilité des entités en fonction de leur fréquence d'utilisation. Cet effet, cumulé à l'association d'un dégradé de couleur à l'occurrence des arêtes, amplifie la perception des groupes de termes significatifs.

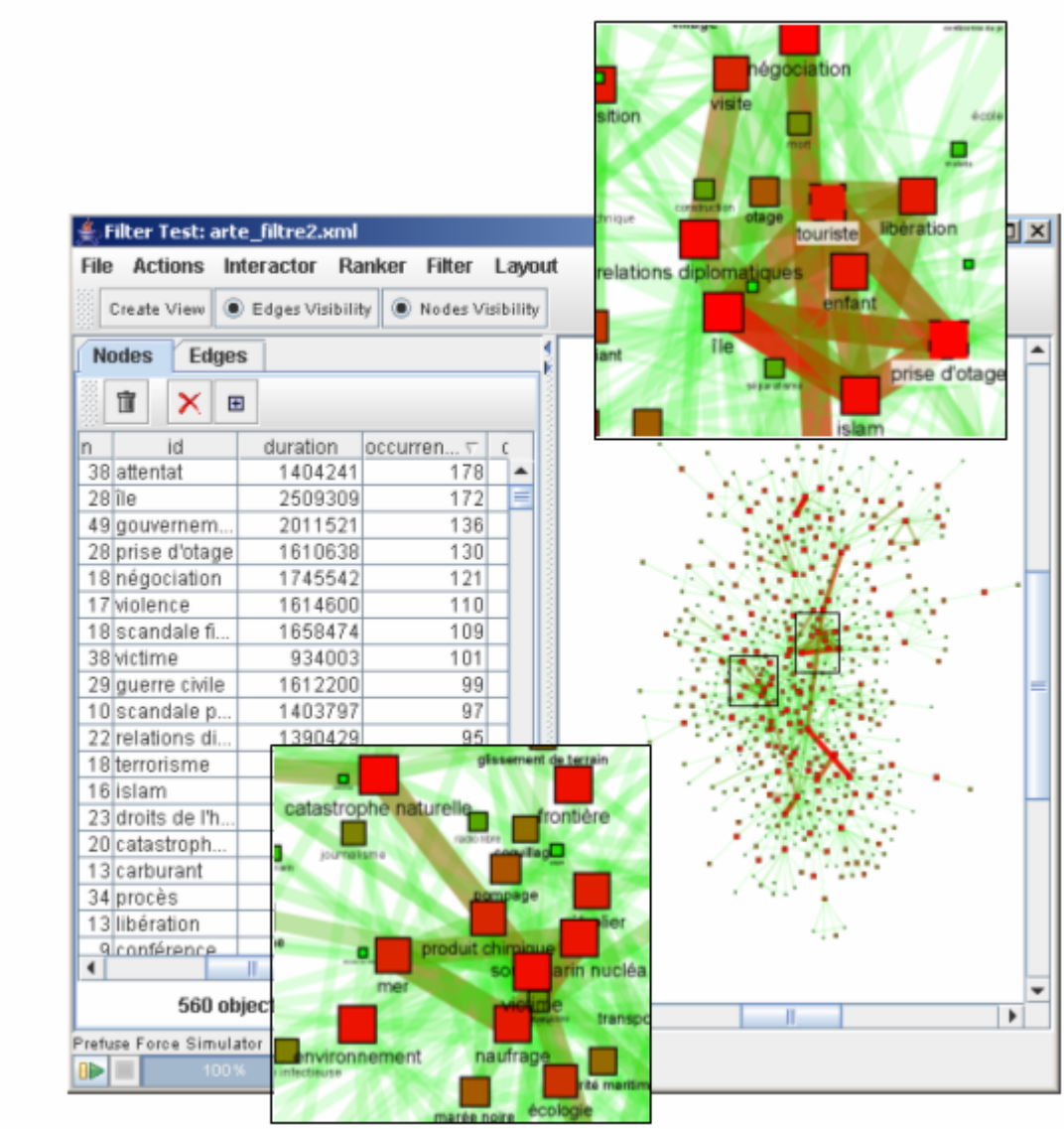


Figure 118 : JT d'ARTE sur l'année 2000 avec zoom sur les deux agrégats les plus importants

L'utilisateur dispose de deux types de vues : une vue globale et une vue locale. La vue globale, cf. Figure 118, permet d'analyser les groupes qui émergent des données. La vue locale, cf. Figure 119, représente l'environnent d'un nœud sélectionné et peut être paramétrée en terme de nombre et de caractéristiques de nœuds visibles. Le passage entre deux vues locales est interactif par sélection du nœud cible.

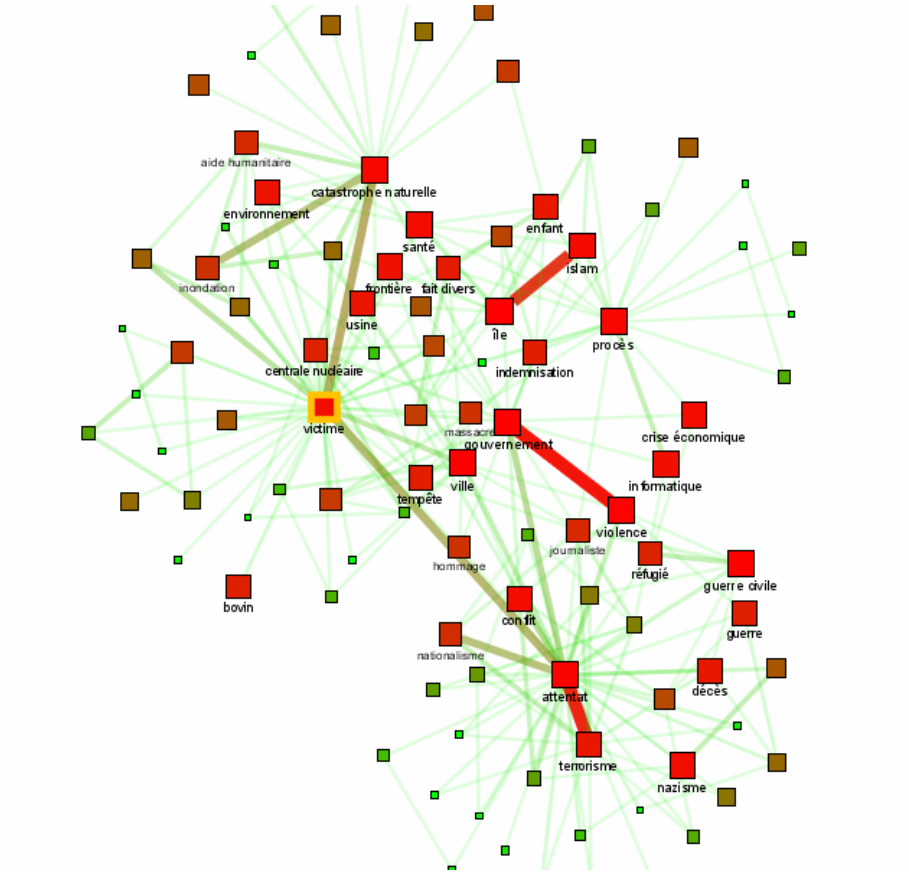


Figure 119 : Graphe local du corpus des JT d'ARTE sur l'année 2000 autour du terme « victime ».

Une remarque pour commencer une analyse très succincte des résultats : les descripteurs les plus fréquemment utilisés pour l'année 2000 sur ARTE sont attentat, île, gouvernement, prise d'otage et violence. La figure fait apparaître instantanément plusieurs types d'information : les descripteurs importants en terme de durée globale des sujets qu'ils indexent, en terme d'occurrence, les associations de termes, et les groupes de descripteurs plus reliés entre eux qu'avec le reste du graphe. Les descripteurs fréquemment utilisés correspondent soit à un évènement unique mais très relayé par les journaux télévisés, comme la tempête ou la prise d'otages aux Philippines pour l'année 2000, soit à des évènements types récurrents, comme par exemple le descripteur *faits divers*, ou encore à des grand thèmes d'information comme l'illustre le terme *santé*. Le nombre de relations qu'un descripteur entretient avec le reste du graphe informe sur le traitement informationnel dont il fait l'objet. De manière générale, l'analyse des relations entre descripteurs fait apparaître deux types d'agrégats : des petits groupes de termes et de liens présentant une forte occurrence (agrégats rouges sur la figure) et des regroupements plus vastes caractérisés par une connectivité relativement dense mais d'importance informationnelle plus faible (agrégats verts). D'un point de vue global, *naufrage* se trouve intégré dans un agrégat rouge composé de *sous-marin*, et *mer*, lui-même inclu dans un agrégat vert de descripteurs liés à l'environnement et au milieu marin. Très rapidement, l'analyse des agrégats rouges fait émerger les groupes suivants : « *naufrage/océan/pollution/sous-marin* », « *île/prise d'otage/islam* »[1], « *scandales politiques/financiers/corruption/témoignage*». Deux arêtes correspondent à des cooccurrences très fortes donc à des sujets d'actualité très souvent abordés *: « carburant/politique*

[1] Touristes et journalistes pris en otage aux philippines, juillet/septembre 2000.

économique », « *gouvernement/violence »*[2] . La vue locale permet d'étudier dans le détail le contexte d'un terme. La connectivité élevée du descripteur spécifique v*ictime*, par exemple*,* donne un aperçu des facettes principales de son utilisation. Les deux agrégats principaux s'articulent autour de *catastrophes naturelles (inondation*, *aide humanitaire, environnement santé*) et *attentats (guerre civil, terrorisme, conflit…).* On retrouve l'évènement très médiatisé des journalistes pris en otage aux Philippines dans le lien rouge *« île-islam* » ainsi que le lien *« gouvernement-violence ».* Les aspects sociétaux *procès indemnisation* sont aussi présents. Ces premiers résultats ne constituent que l'ébauche de la démarche d'analyse. En effet, ces cartes ne pourront être interprétées qu'en regard des évènements ou groupes d'évènements auxquels elles se rapportent. Cet aspect très intéressant ne constitue pas le sujet de cette section, aussi nous arrêterons-nous à ce niveau quant à l'analyse de cette figure. En résumé, ces cartographies de l'information permettent d'obtenir très rapidement des vues globales et locales de sujets d'actualités traités en intégrant des données quantitatives et qualitatives. Elles font émerger automatiquement les thèmes les plus traités de l'information et leur contexte. Elles peuvent être utilisées comme point d'entrée pour naviguer, effectuer des analyses sur les média, faire des comparaisons entre chaînes ou périodes de temps, assurer l'exhaustivité du traitement d'un sujet. L'Ina a développé, sous le nom de *baromètre thématique,* une méthode de classification automatique de tous les sujets de JT en 14 catégories afin de proposer une ébauche d'analyse globale du traitement de l'actualité sur les chaînes TV. Cet outil constitue une approche complémentaire pour l'analyse des médias télévisuels.

4.6.2 Le dépôt légal du Web

Depuis sa création, il y a plus de 400 ans, le Dépôt Légal s'est adapté aux nouveaux médias à mesure que leur intérêt patrimonial devenait évident. Le Web représente une convergence de ces médias, et son extrême volubilité, tant d'un point de vue technologique qu'éditorial, pose aujourd'hui la question de sa conservation. Jugeant que la valeur patrimoniale de ce nouveau média ne pouvait être ignorée, le législateur posera prochainement, dans la loi sur l'économie numérique, le cadre juridique du Dépôt Légal des sites Web français.

Considérant le Web comme une extension aux médias actuels, la charge de ce Dépôt Légal sera partagée entre L'Ina et la BNF, dans la continuité des collections des deux institutions. L'Ina se concentrera ainsi essentiellement sur les sites relevant des industries culturelles et des médias radios et télévisions. Dans cette optique, l'Ina mène des études et des expérimentations sur les techniques, procédés et méthodes permettant la constitution et la gestion d'une archive du Web.

La constitution d'une archive d'une partie du Web posent plusieurs types de problèmes. Des problèmes de définition du domaine, ou comment déterminer la frontière du domaine des sites tombant dans l'escarcelle du Dépôt Légal. La mise en oeuvre pratique d'un Dépôt Légal nécessite en premier lieu l'établissement d'une cellule de veille documentaire, chargée tout d'abord de la délimitation, puis du suivi du domaine.

Des problèmes techniques liés à l'évolution permanente des technologies utilisées sur le Web, et à la multiplicité des formats des contenus multimédia. Afin d'assurer la viabilité de ces contenus sur le long terme, il est nécessaire de mettre en place une politique de migration ou d'émulation vers des

[2] L'étude des sujets (ARTE année 2000) comportant l'association *violence-gouvernement* montre que cette association est presque exclusivement utilisée pour décrire le conflit israélo-palestinien, et constitue donc un bon indice pour mesurer son importance médiatique.

formats normés (les deux principes sont assez proches pour une archive numérique, l'émulation pouvant se concevoir comme une migration "à la demande" lors de la consultation, le tout étant de conserver les algorithmes de conversion), tandis que la collecte et l'indexation exige un suivi des évolutions technologiques pour pouvoir appréhender ces contenus.

Le Web est construit – au moins en partie – sur une logique d'interaction : l'utilisateur sollicite lui-même l'affichage d'une page en cliquant sur un hyperlien ou en remplissant les champs d'un formulaire. La collecte doit reproduire cette interaction pour parcourir et archiver les pages d'un site, à l'aide de logiciels appelés "robots de collecte" (ou *crawlers*). Le principe en est simple: le robot rapatrie une page d'un site, l'archive, répertorie l'ensemble des hyperliens qu'elle contient, puis rapatrie chacune des pages pointées par ces hyperliens, et procède de la même manière pour chacune d'entre elles et ce de manière itérative, jusqu'à ce que l'ensemble des liens aient été suivis (les hyperliens pointant vers des pages déjà rapatriées sont ignorés, de même que ceux pointant vers des pages extérieures au site). La mise en œuvre d'un tel système n'est cependant pas sans poser certains problèmes :

- Le processus étant itératif, sa durée et sa complexité dépendent du nombre total de pages du site.
- Le robot est limité dans ses modes d'interactions : il peut imiter le clic d'un utilisateur sur un hyperlien, mais pas la saisie des champs d'un formulaire.
- Le robot a besoin de connaître de manière formelle les frontières du site à explorer pour déterminer quels hyperliens pointent vers des pages extérieures au site.
- Le robot doit être capable de "comprendre" les différents formats techniques des contenus pour pouvoir trouver les hyperliens qui s'y trouvent.
- Certains sites peuvent générer une infinité d'hyperliens pointant vers des pages identiques, pouvant tromper le robot.

4.6.2.1 Construction d'un Graphe du Web

Le processus de captation des pages Web est actuellement dans sa phase de test. Chaque captation ou *crawl* génère un réseau de pages Web. Ce réseau peut être post-traités afin de construire le graphe des pages d'un site ou le graphes des sites.

Nous avons choisi de présenter ici le graphe des sites issu d'un *crawl* en largeur d'abord initié de la page d'accueil du site de l'Ina (www.Ina.fr). L'appartenance des sites au domaine du Dépôt Légal ne peut pas être évaluée à l'heure actuelle. Le crawl a donc lieu sans aucun contrôle de la « qualité » des sites explorés. Le contenu des pages est tout de même analysé afin d'évaluer automatiquement l'appartenance des sites explorés au domaine du Dépôt Légal. Ce processus est pour l'heure basé sur l'identification de mots-clés spécifiques au domaine.

4.6.2.2 Analyse visuelle

Une première constatation s'impose. Avec 2000 sites et plus de 200 000 liens les graphes obtenus sont très denses, et donc particulièrement difficile à visualiser comme on peut le constater sur la Figure 120. Le graphe est présenté sans aucun filtrage, après application de l'algorithme de placement de *Fruchterman-Reingold*. La taille et la couleur sont utilisées pour coder la valeur d'*authority* des sites. Je rappelle que l'*authority* d'un sommet dans un graphe est une mesure d'importance liée à la connectivité, vous trouverez une définition plus précise en section 2.2.2.1.2. Cette première tentative montre que le graphe obtenu est dense compact, et que son noyau est constitué des sites les plus importants.

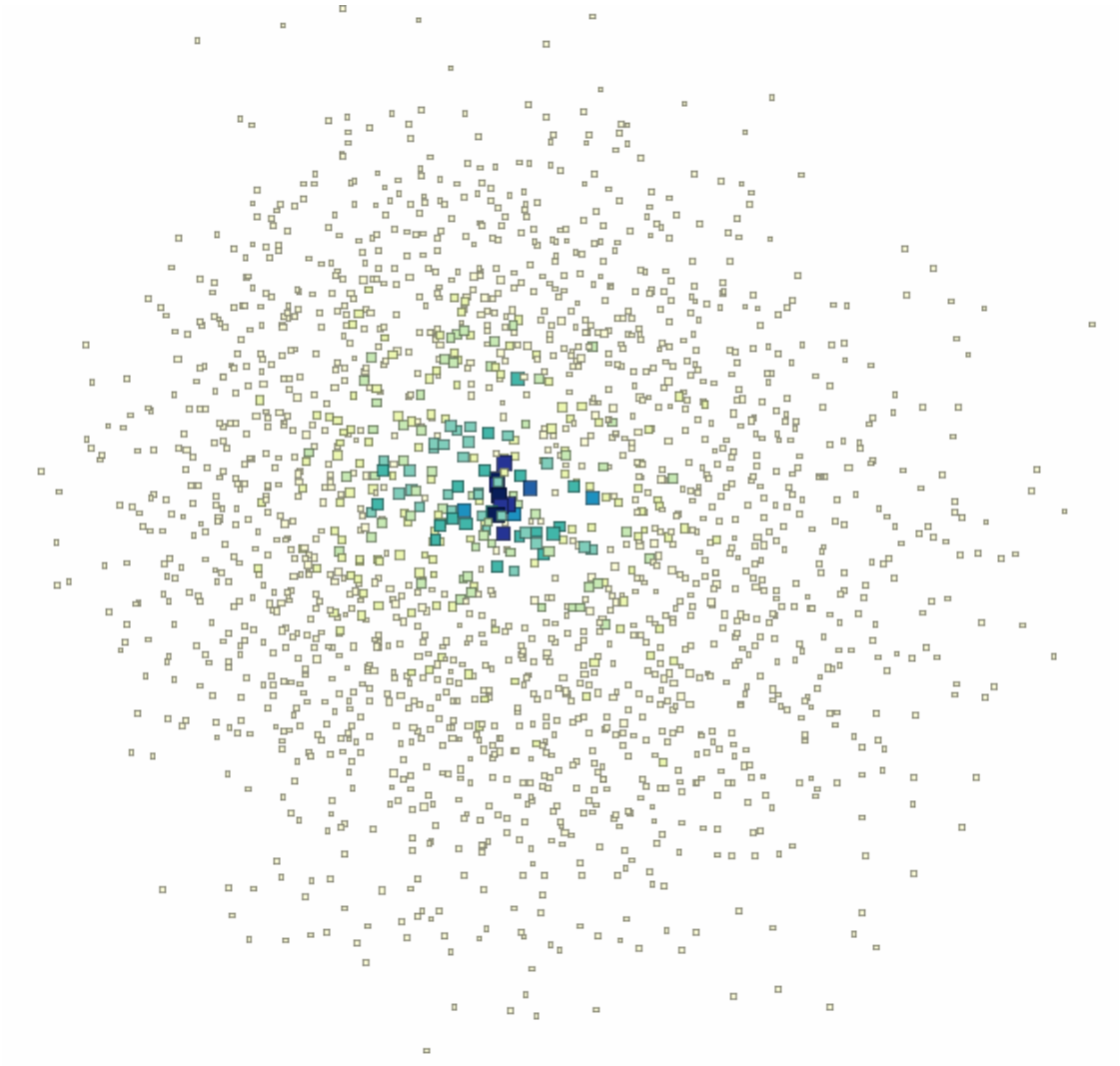


Figure 120 : graphe non filtré des sites Web placé par Fruchtermn-Reingold

Dans un second temps nous utilisons l'algorithme de placement *Edge LinLog* afin d'identifier éventuellement des clusters et faire émerger une structure. La Figure 121 est un zoom sur le noyau du graphe. Une grande partie des sites du noyau sont les sites de grandes institutions publiques. Sur la partie droite du noyau se trouvent principalement les sites des différents ministères à l'exception des ministères de l'éducation et de la culture. Ces deux sites se trouvent regroupés dans la partie gauche avec les sites des grandes écoles, universités et institutions culturelles. Le site de l'Ina, identifié par un bord orange, est lui aussi placé dans ce regroupement.

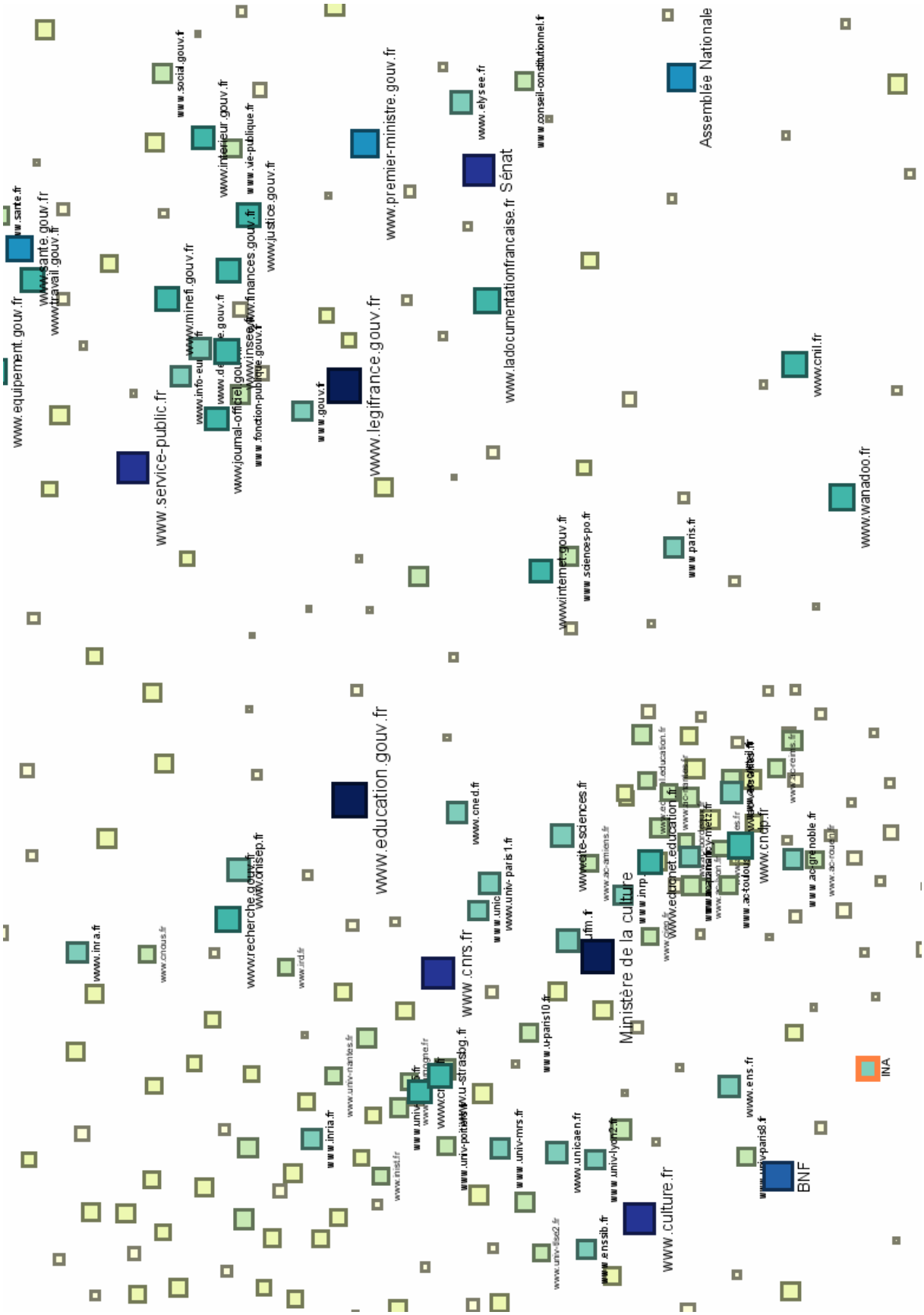
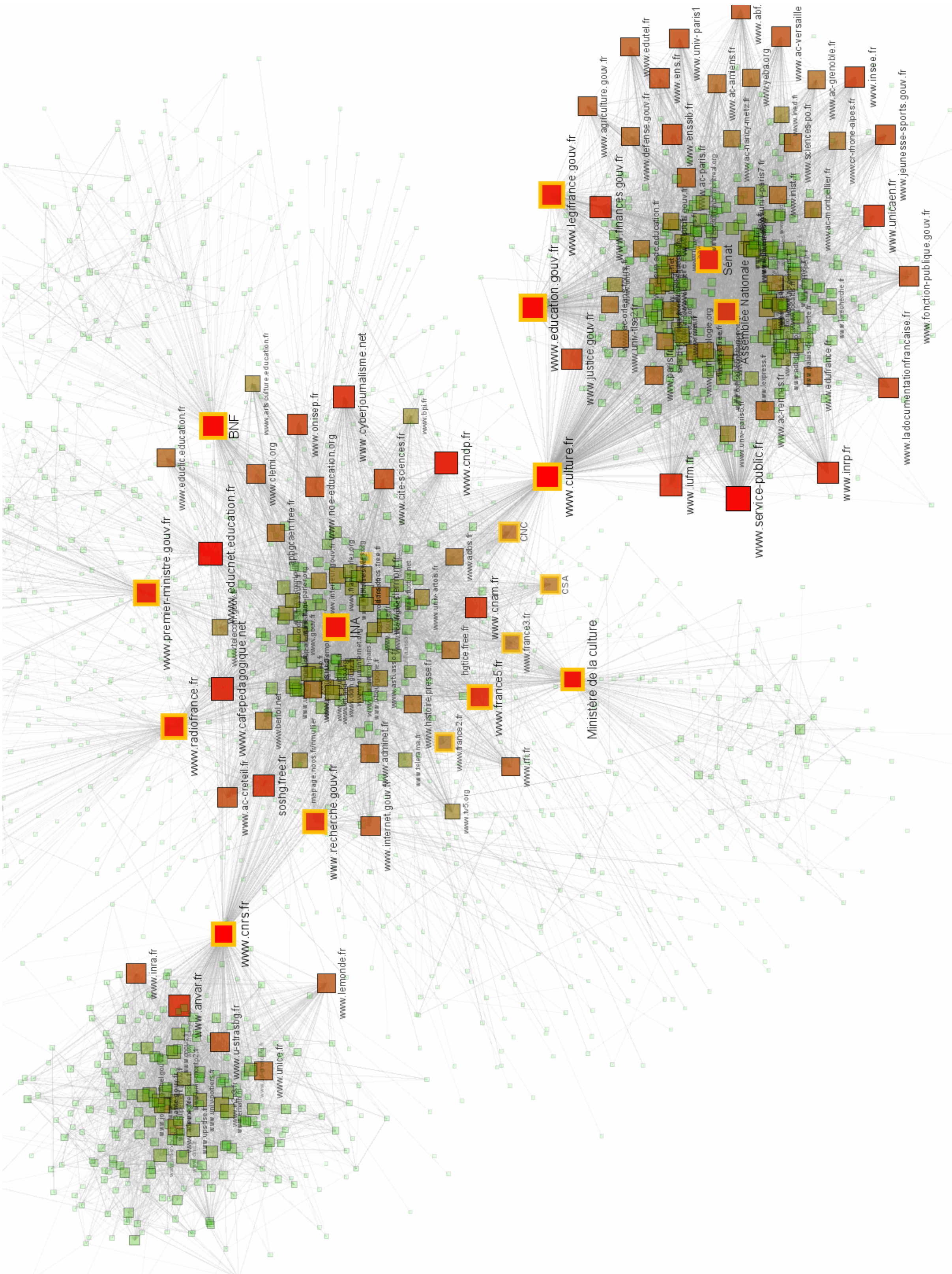


Figure 121 : zoom sur le noyau du graphe des sites Web placé par Edge LinLog



www.legifrance.gouv.fr
www.education.gouv.fr
www.culture.fr
www.service-public.fr
Sénat
Assemblée Nationale
www.finances.gouv.fr
www.justice.gouv.fr
www.agriculture.gouv.fr
www.defense.gouv.fr
www.edutel.fr
www.ens.fr
www.univ-paris1
www.ac-amiens.fr
www.abf.
www.ac-versaille
www.ac-grenoble.fr
www.insee.fr
www.jeunesse-sports.gouv.fr
www.unicaen.fr
www.fonction-publique.gouv.fr
www.ladocumentationfrancaise.fr
www.inrp.fr
www.iufm.fr
BNF
www.educlic.education.fr
www.premier-ministre.gouv.fr
www.educnet.education.fr
www.cafepedagogique.net
www.radiofrance.fr
www.clemi.org
www.onisep.fr
www.cyberjournalisme.net
www.cndp.fr
www.cite-sciences.fr
www.noe-education.org
INA
CNC
CSA
www.cnam.fr
www.france3.fr
www.france5.fr
Ministère de la culture
www.rfi.fr
www.recherche.gouv.fr
www.internet.gouv.fr
soshg.free.fr
www.ac-creteil.fr
www.cnrs.fr
www.inra.fr
www.anvar.fr
www.u-strasbg.fr
www.unice.fr
www.lemonde.fr

Figure 122 :

Durant le *crawl*, la page d'accueil de chaque site est analysée. Le contenu textuel est parcouru afin de mesurer l'appartenance au domaine du Dépôt Légal en vérifiant la présence de mots-clés. Une note de « sémantique » est donc calculée et attribuée à chaque site.

Cette mesure est utilisée pour effectuer un filtrage. La Figure 122 présente ce graphe filtré et placé par l'algorithme de Fruchterman-Reingold. Le placement a été finalisé à la main, afin de rendre lisibles les sites les plus représentatifs du domaine. La couleur, la taille et la transparence codent la note de sémantique. Le placement révèle trois grands clusters. Le cluster de gauche est composé des sites d'université scientifiques et de laboratoires de recherche. Il est connecté au cluster central par le site du CNRS. Ce cluster central est organisé autour du site de l'Ina et essentiellement constitué des sites des chaînes télévisées, radio et institutions culturelles. Le cluster de droite regroupe les sites des ministères et de plusieurs universités en sciences sociales, économiques et politiques.

4.6.3 Les auteurs et publications de LIRMM

L'étude de données bibliographiques est source de nombreuses données de type graphe. Les articles scientifiques en se citant les uns les autres constituent un réseau complexe de documents. L'analyse de la structure d'un tel réseau permet d'identifier les publications les plus importantes de par le rôle qu'elles occupent au sein du réseau. Le degré entrant du sommet d'une publication constitue par exemple une bonne mesure de son importance. De la même manière on peut aussi considérer que ce sont les auteurs qui se citent. Le réseau qu'ils forment permet d'identifier les auteurs majeurs et l'influence qu'ils ont sur une communauté.

Un article scientifique est aussi souvent le résultat du travail de plusieurs personnes qui collaborent ponctuellement ou régulièrement ensemble. L'analyse du réseau des collaborations entre chercheurs permet d'identifier ces groupes de personnes.

4.6.3.1 Source de données

Le LIRMM maintient à jour la liste des publications réalisées en son sein. Nous avons obtenu un export au format texte de ces publications. Après nettoyage le fichier est composé de blocs séparés par un saut de ligne. Chaque bloc correspond à une publication différente du laboratoire. Un de ces blocs est présenté ci-dessous.

```
Titre    Improving Defect Detection in Static-Voltage Testing
Auteurs  RENOVELL M. AZAIS F.    BERTRAND Y.
Type de document    Art. dans Revue
Référence     IEEE Design and Test of Computers
Date de parution    2002
Année de parution   2002
Nbre/N° de page     pp. 83-89
Niveau d'autorisation     0
Prix    0
Numéro  9559
Date d'indexation   19/03/2003
Volume  17
N° de revue / série 6
Département   Microélectronique
```

Les descriptions des publications ne sont pas complètement homogènes, cependant les champs les plus utiles sont systématiquement présents. Ces champs sont en gras dans le bloc ci-dessus. Une

publication sera caractérisée par son titre, ses auteurs, l'année ou la date de parution et le département qui l'a émise.

La liste des publications permet de construire plusieurs graphes. Le plus simple est de considérer que chaque publication est connectée à ses auteurs et chaque auteur à ses publications. On obtient ainsi le graphe bipartite des publications et auteurs. Ce graphe a la très bonne propriété de coder sans redondance et sans perte d'information la liste des publications. Chaque publication ou auteur est représenté exactement par un sommet. Par contre, ce graphe n'est pas facile à interpréter. De manière générale, il est plus simple de raisonner sur un graphe dont les sommets sont du même type. Par exemple, l'évaluation de l'importance de la collaboration entre deux chercheurs est beaucoup plus simple à réaliser dans un graphe constitué uniquement d'auteurs. Ce graphe bipartite est en fait équivalent à l'hypergraphe des auteurs où chaque publication est une hyperarête qui connecte ses auteurs. Une hyperarête étant une arête qui peut connecter plus de deux sommets. Il est plus facile de se ramener à un graphe plus simple. Il reste deux possibilités, les graphes des auteurs et des publications. Ces deux graphes sont duaux. Ils sont tout deux le résultat d'une simplification du graphe bipartite des publications et auteurs. Le graphe des auteurs est une expansion de l'hypergraphe des auteurs, où chaque hyperarête connectant un groupe de sommets est développée tel que le groupe de sommets devient une clique. Une n-arête devient donc $n(n-1)/2$ arêtes. Comme une publication correspond à une hyperarête, elle correspond dans le graphe des auteurs à plusieurs arêtes. Le graphe des publications peut être obtenu de façon tout à fait similaire.

Nous nous sommes intéressés plus particulièrement au graphe des auteurs, ce réseau est le bon support pour tenter d'identifier les groupes de chercheurs qui travaillent souvent ensemble. On peut remarquer que le titre et les auteurs d'une publication sont les seules informations indispensables pour construire le graphe des collaborations des auteurs. La date et le département sont cependant des informations supplémentaires intéressantes, nous les intégrons donc comme attributs des sommets et arêtes dans le graphe des auteurs. Comme les auteurs sont susceptibles d'avoir publié dans différents départements, nous les rattachons au département dans lequel ils ont le plus publié.

4.6.3.2 Analyse du graphe des auteurs du LIRMM

Le graphe des auteurs est un graphe non orienté. La relation de collaboration est mutuelle et sans direction. De plus, pour une paire de sommets donnés, il y autant d'arêtes qui les connectent qu'il y a eu de collaborations entre les deux auteurs correspondants. Le graphe des auteurs n'est donc pas un graphe simple. Ce facteur est à prendre en compte afin de bien interpréter les valeurs des différentes métriques qui ne sont généralement pas bornées dans un graphe non simple.

Le premier réflexe est de tenter de placer le graphe tel quel avec le modèle de force de Fruchterman-Reingold. Le nombre de publications par auteurs est un bon indice d'importance. Nous codons cette information par la taille des sommets. Le département est une information typiquement qualitative, de plus il n'y a que cinq départements différents. Cet attribut est particulièrement adapté à un codage par la couleur.

Le fait que ce graphe contient de nombreuses arêtes parallèles a une incidence sur le placement. La présence de q arêtes entre deux sommets induit le même comportement que la présence d'une arête dont le coefficient de raideur serait q fois plus grand. Plus il y a d'arêtes entre deux sommets plus le ressort virtuel entre eux est raide, et donc plus la distance qui les sépare sera proche de la longueur d'arête par défaut.

Le graphe contient 1500 auteurs et 8000 liens de collaboration, dont 4500 liens parallèles. Il est composé d'une composante connexe géante et d'une myriade de petites composantes. Les différentes composantes sont facilement identifiables. La Figure 123 montre la composante géante qui comporte 1000 auteurs et concentre 7600 liens de collaborations, son diamètre est de 13. Nous nous concentrons sur cette composante géante, les autres composantes ne contenant au plus qu'une petite dizaine de sommets chacune, elles sont particulièrement simple à analyser.

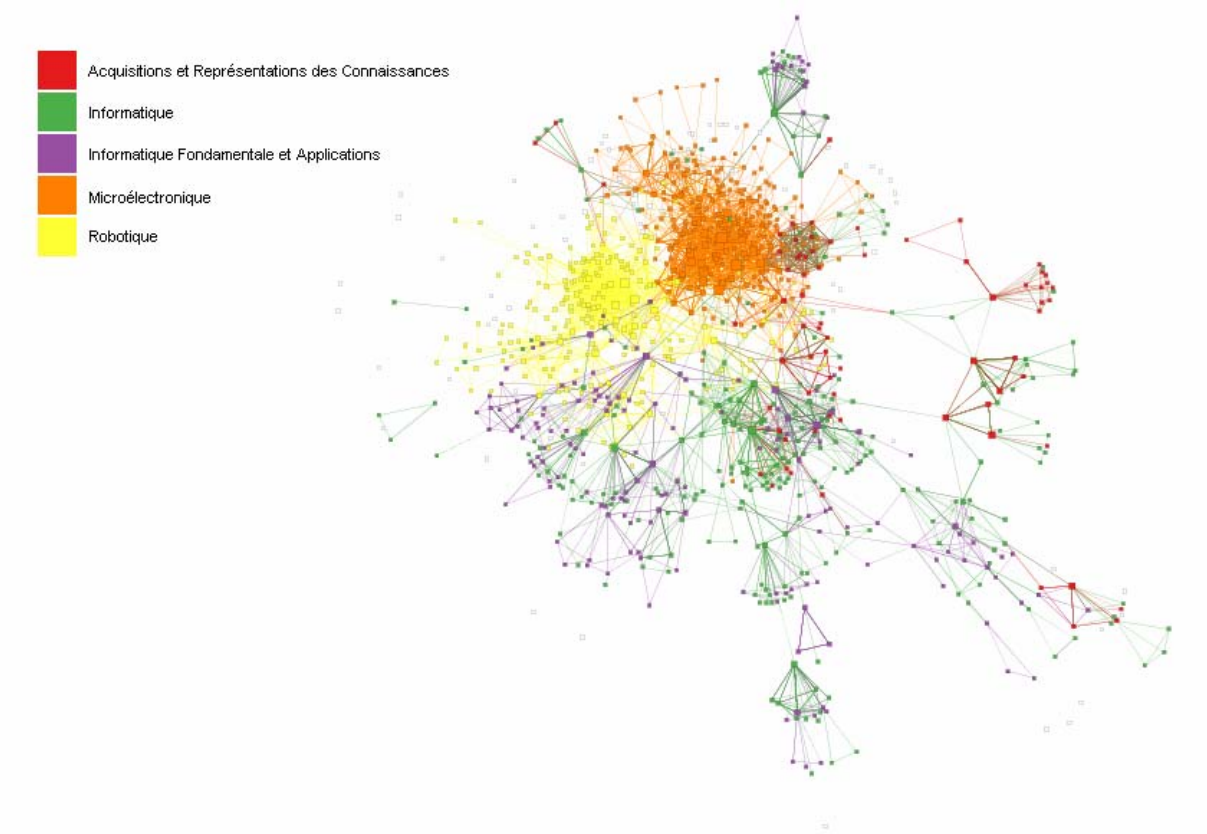


Figure 123 : composante géante du graphe des auteurs

Cette visualisation permet de rapidement constater que la composante géante étudiée possède deux noyaux relativement denses autour des deux départements de microélectronique et de robotique. Les auteurs des trois autres départements sont plus dispersés. Il n'y pas de frontières apparentes entre les départements d'informatique et de d'informatique fondamentale. Au sein des trois départements d'informatique, la formation de collaborations semble peu guidée par l'appartenance au même département. On peut tout de même observer qu'un groupe d'auteurs du département d'acquisition de connaissance semble s'être constitué près du cœur microélectronique.

En masquant les arêtes, et en naviguant par pan et zoom on peut aisément identifier les auteurs qui ont le plus publié, cf. Figure 124. Attention toutefois quant à l'interprétation du nombre de publications. Les publications des départements de Robotique et de Microélectronique sont généralement le fruit du travail de nombreux chercheurs. Le nombre moyen d'auteurs pour une publication est plus élevé dans ces départements, et le nombre moyen de publications par auteur est donc lui aussi plus élevé.

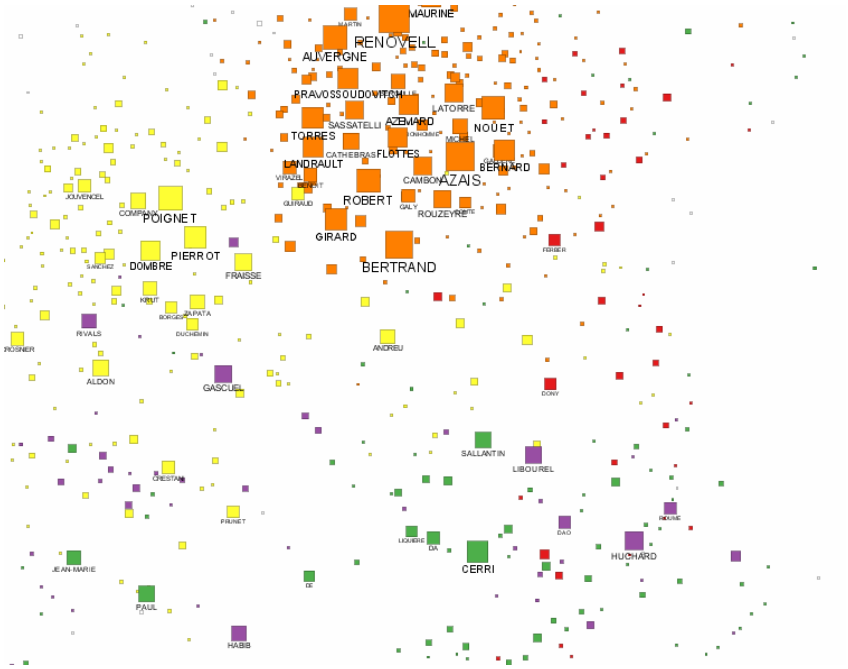


Figure 124 : le masquage des arêtes facilite la lecture de la taille des sommets

Dans un second temps, nous souhaitons identifier plus précisément les groupes d'auteurs qui travaillent fréquemment ensemble. Le placement Edge LinLog donne un placement tout à fait pertinent pour ce type de tâches. Nous utilisons le placement précédent comme initialisation du modèle LinLog. De nombreux clusters de tailles différentes peuvent être identifiés, cf. Figure 125. En sélectionnant les arêtes internes à un cluster, il est possible de retrouver les articles pour lesquels les auteurs ont collaboré, et ainsi de définir une étiquette pour le cluster.

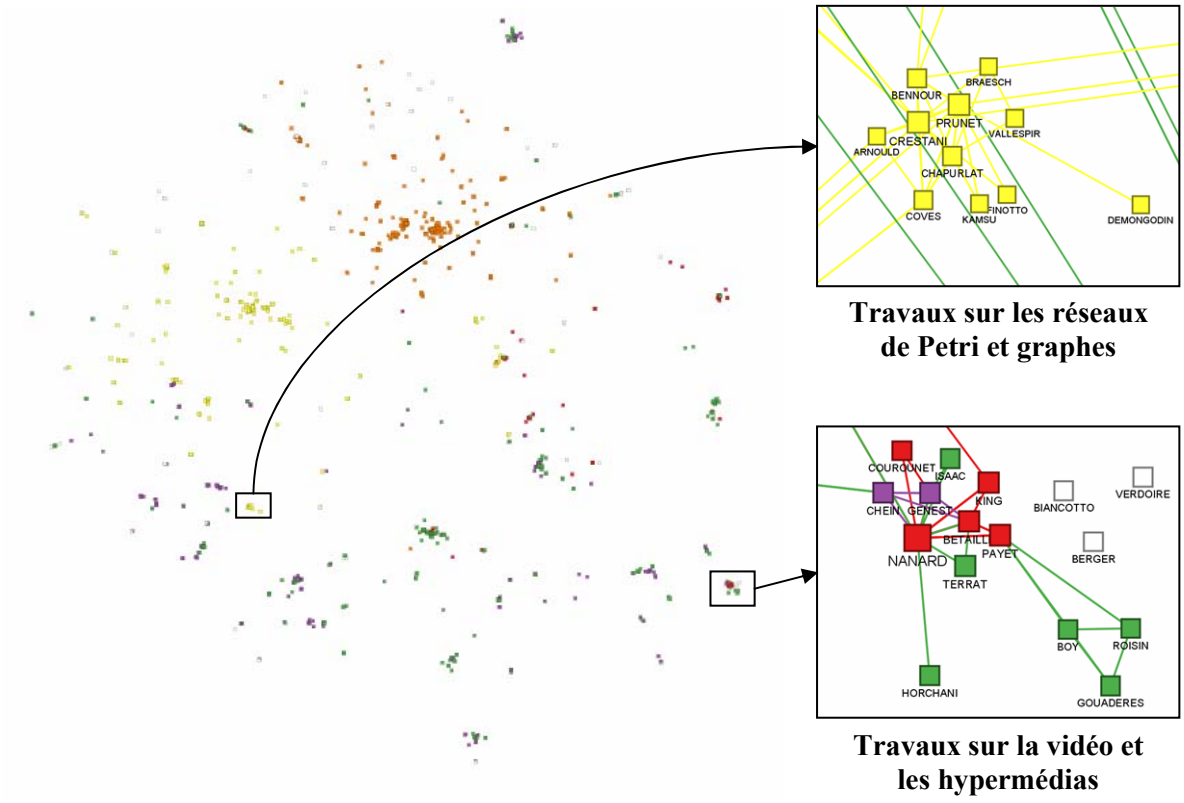


Figure 125 : identification de groupes d'auteurs fortement corrélés

Enfin, nous avons calculé la centralité (betweenness centrality) des sommets et arêtes du graphe. Cette mesure rend compte du nombre de plus courts chemins passant par chaque sommet et arête. Dans ce réseau très clusterisé, comme celui-ci, cette mesure permet d'identifier les auteurs et les articles les plus centraux au sein d'un cluster, mais aussi les entités qui assurent la jonction entre plusieurs clusters. Nous utilisons une variation de la taille pour coder la centralité des sommets et arêtes. Ce codage renforce la visibilité des sommets et arêtes ayant une grande valeur de centralité, cf. Figure 126.

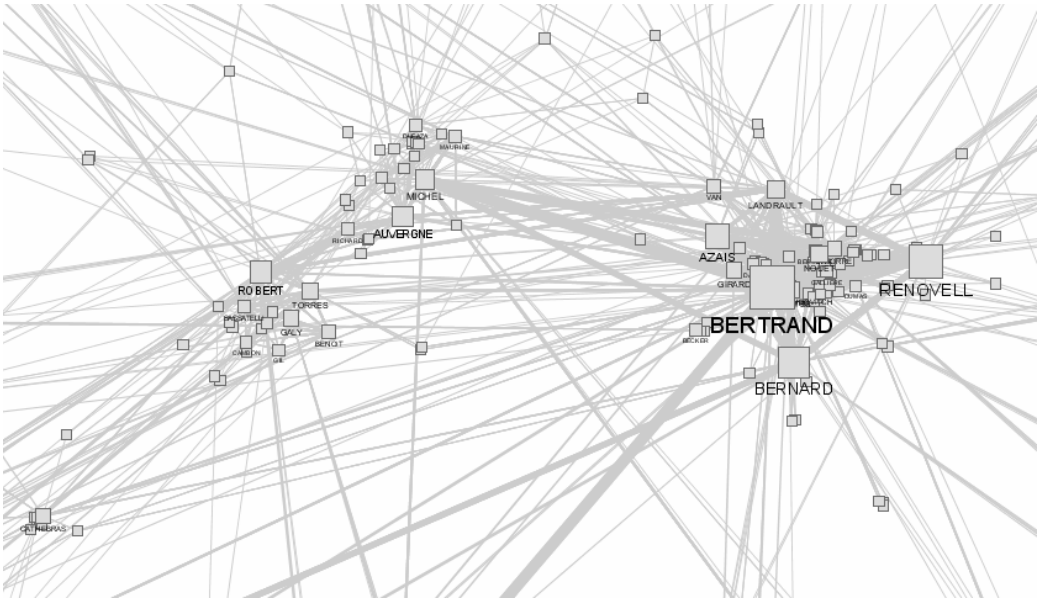


Figure 126 : mise en relief des auteurs et articles les plus centraux

4.6.4 Les visualisations du thésaurus et de la thématisation

Les bibliothèques doivent apporter l'aide nécessaire à leurs utilisateurs au cours de la consultation. Les structures documentaires sont complexes mais représentent un point d'accès essentiel pour comprendre l'organisation et le contenu des bibliothèques. A l'heure où une grande part des consultations commence à s'effectuer en ligne, un enjeu de taille est d'assurer la *continuité sémantique* entre les processus d'indexation et de recherche. L'élargissement de la consultation en ligne rend difficile l'intervention directe de professionnels de la documentation. Aussi, la création d'interfaces conviviales de consultation de ces structures paraît-elle constituer un nouvel élément incontournable du service d'assistance aux utilisateurs.

Une des spécificités des bibliothèques réside dans leur cœur de métier documentaire, à savoir la gestion des contenus en fonction de critères multiples soigneusement étudiés. Ces critères décrivent les documents au niveau sémantique mais aussi en terme de formats, de droits d'accès, de disponibilités, d'appartenance à des collections, éventuellement d'aspects marketing et commerciaux ou de matériels…. En effet, le rôle de la documentation est de positionner le nouveau document dans l'ensemble des ressources par l'attribution de données descriptives. Cet ensemble d'information structure les ressources et permet de définir le contexte de l'offre proposée. Dans le cas de ressources audiovisuelles, le travail documentaire constitue une part importante du travail d'archivage :

contrairement aux ressources textuelles, les ressources audiovisuelles ne s'autodécrivent pas encore[3]. Ainsi, chaque document est décrit, au niveau d'archivage recommandé, par une notice documentaire textuelle créée par des documentalistes et intégrée dans une base de données pour en permettre l'accès.

La garantie « qualité » d'une bibliothèque en terme d'accès sémantique réside en partie dans le paradigme suivant : toute œuvre inaccessible est une œuvre perdue. Or, sans prendre en compte les compétences des utilisateurs, deux phénomènes peuvent rendre une œuvre documentée inaccessible. Lorsqu'un document représentant une réponse pertinente à une requête se trouve mêlé à un trop grand nombre de documents résultats de plus faible pertinence, l'utilisateur ne peut plus analyser les résultats et atteindre sa cible : il est confronté au *bruit documentaire*. A l'opposé, si aucun document résultat ne répond à une requête correctement formulée et dans les domaines couverts par la bibliothèque, l'utilisateur est confronté au *silence documentaire*. Un « bon » système documentaire doit minimiser le bruit en garantissant que tous les documents résultats d'une requête sont pertinents, et le silence en assurant que la grande majorité (dans l'idéal la totalité) des documents pertinents est présente dans l'ensemble résultat. C'est la structuration sémantique et catégorielle des contenus qui constitue la réponse « métier » des bibliothèques à cette problématique.

De manière générale, le langage naturel n'est pas interprétable de manière univoque. Aussi, les bibliothèques ont-elles élaboré des langages documentaires afin de produire des descriptions formalisées et normalisées de leurs ressources. Les quelques exemples suivants illustrent l'ambiguïté d'interprétation du langage naturel et la nécessité de la structuration documentaire. Le terme « PARIS », posé comme requête en texte libre sur tous les champs de notices donne pour résultat un ensemble de … quelques centaines de milliers de documents! En effet, les notices peuvent contenir « PARIS » comme lieu de production, de tournage, d'événement, comme sujet architectural, social... Dans le monde télévisuel, il est relativement fréquent qu'un intervenant possède des rôles multiples tel M.Drucker : présentateur, producteur d'émissions, invité et même quelques fois interprète... La polysémie peut aussi générer du bruit documentaire : virus est actuellement employé dans le domaine médical et informatique. Dans ces trois cas, une désambiguïsation rapide peut s'effectuer grâce à la structuration documentaire, soit en posant la requête « PARIS » en précisant le champ d'indexation correspondant à la requête désirée, soit en spécifiant le type de rôle de la personne, soit en considérant le terme « père » dans la structure hiérarchique que constitue le thésaurus.

Les connaissances[4] qui servent à décrire le contenu des ressources de l'Ina se présentent sous forme de listes spécifiques et de thésaurus. Les listes peuvent être de différents types, comme la liste des métiers de l'audiovisuel, des matériaux supports, des canaux de diffusion…Un thésaurus est un ensemble de termes organisés en arborescence et contenant des liens transversaux de type « voir aussi », des informations de synonymie ou de précision.. Le thésaurus des noms communs comporte plus de 10000 termes. Un « bon » terme de thésaurus est un terme qui ne génère pas de bruit documentaire. Il doit donc décrire précisément son objet, être non polysémique et de sens constant dans le temps. Par exemple, « organisation » est un terme très générique, inutilisable en tant que

[3] De grands champs de recherche (transcription automatique, indexation d'images…) ont pour but d'analyser des flux sonores ou audiovisuels afin d'en produire une description automatique. Si ces techniques ne présentent pas une qualité suffisante pour être exploitées au niveau de qualité exigé pour la documentation, elles peuvent déjà constituer un point d'accès intéressant si aucune indexation n'est disponible.

[4] Si les approches ontologiques apparaissent prometteuses, elles sont pour le moment difficiles à mettre en œuvre, tout particulièrement pour des bibliothèques généralistes comme l'Ina. En effet, la modélisation des concepts et de leurs relations, qui concerne potentiellement l'intégralité du monde paraît extrêmement complexe à réaliser…

requête s'il est employé seul. Un « bon » terme de thésaurus ne doit pas non plus générer de silence. Un terme trop spécifique aura tendance à créer du *silence documentaire*. Par exemple, le terme « galéace » ne sera pas un bon terme pour indexer un document sur « les bateaux à voile et à rame chargés d'artillerie ». Le thésaurus peut être vu comme un filtre de normalisation de vocabulaire descripteur. D'une part, ces termes descripteurs doivent se faire le reflet de l'évolution des contenus eux-mêmes. Ainsi, l'évolution du langage qui exprime le cours des évènements mondiaux ou nationaux, des pratiques sociales, culturelles ou technologiques, constitue-t-elle aussi un élément à prendre en compte. La difficulté de cette tâche est de différencier les évolutions langagières qui constituent des « effets » de mode de celles qui accompagnent les changements profonds. D'autre part, ces termes descripteurs doivent se faire le reflet de l'évolution des usages. L'étude des requêtes des utilisateurs permet d'établir l'intérêt de l'introduction de nouveaux types de descripteurs. Par exemple, depuis quelques années, toute scène présentant une anomalie de situation ou de tournage sera indexée de façon à être facilement repérée pour intégrer un bêtisier. La gestion d'un thésaurus est donc une opération complexe qui suppose une surveillance des termes déjà acceptés, pour les préciser ou éventuellement les supprimer, et des nouveaux termes qui sont soumis, étudiés sur une période donnée, puis rejetés ou intégrés. L'utilisation de ces données lors de l'indexation maintient la cohérence descriptive de l'ensemble des contenus.

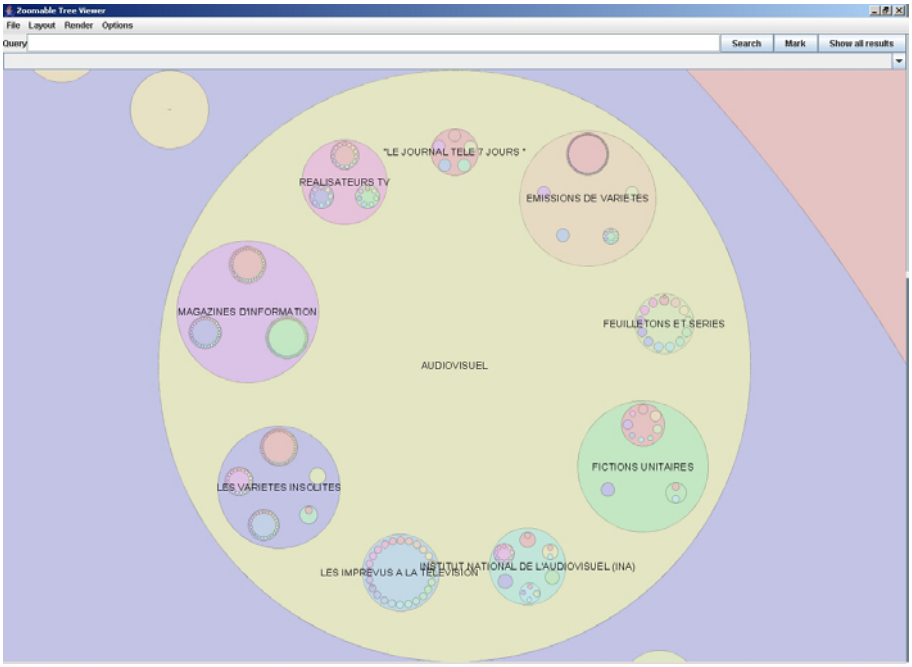


Figure 127 : Visualisation hiérarchique du thésaurus.

L'importance fonctionnelle du thésaurus nous a conduit à proposer un outil de visualisation interactive présentant deux modes de représentation de l'arborescence. Un premier mode de visualisation (cf. Figure 127) est adapté à une vision de l'arbre dans sa largeur et s'inspire de PAD et PAD++, travaux de K. Perlin [Perlin93]. L'arborescence apparaît comme un ensemble de cercles imbriqués. Chaque cercle représente un nœud de l'arbre et contient à sa périphérie ses nœuds enfants. La navigation s'effectue par une animation continue en cliquant avec la souris sur les cercles apparents ou en cherchant des termes dans le champ de requête. L'imbrication imparfaite de cercles dans un cercle rend l'identification des enfants et descendants d'un nœud bien plus facile et permet l'insertion de descripteurs textuels.

Un deuxième mode est plus adapté à une vision en profondeur de l'arborescence. Il est basé sur la représentation diagramme nœud-lien de type « space tree » [Plaisant02] qui permet de développer la

hiérarchie à la demande en sélectionnant les branches à développer. Les deux représentations sont associées à un moteur de recherche textuelle : l'utilisateur pose une requête, les nœuds cibles sont sélectionnés, marqués et les branches les atteignant sont développées alors que les autres sont masquées.

Ce prototype a été proposé aux documentalistes pour parcourir le thésaurus et/ou vérifier la présence d'un mot, discuté puis mis à leur disposition. Il leur est apparu qu'il pourrait constituer un bon outil pour contrôler l'usage des termes. Un attribut correspondant au nombre de documents indexés par chaque terme a été rajouté aux données et a été associé à un attribut graphique de couleur. Cette représentation permet une analyse très rapide des mots qui peuvent être source de bruit ou de silence. Elle conserve le contexte des termes ce qui permet de visualiser instantanément le voisinage du terme afin d'analyser éventuellement la cause du silence ou du bruit. Par exemple, un mot trop proche sémantiquement dans le voisinage peut influer sur la sous-utilisation d'un terme. Le prototype dispose de filtres permettant d'afficher les nœuds en fonction de paramètres comme leur fréquence d'usage, leur première ou dernière date d'utilisation. Il offre donc des fonctionnalités d'analyse et de contrôle global de l'usage des termes du thésaurus. Il est utilisé pour analyser les pratiques d'indexation et pour effectuer la mise à jour du thésaurus : le prototype de départ a été appréhendé, intégré et détourné par les experts pour un usage particulier. De la même façon, on pourrait appliquer ce prototype à l'analyse des requêtes utilisateurs : l'occurrence des termes constituant les requêtes permettrait de projeter sur le thésaurus une répartition des thématiques demandées par les usagers.

Ce type d'interface est aussi particulièrement adapté pour servir d'accès à des collections structurées en arborescence alors qu'elles présentent une structure de graphe. En effet, il est plus aisé de concevoir des classifications sous forme d'arborescence car l'ajout d'un élément de l'arborescence n'implique pas une mise à jour de la totalité des relations. Cependant, il arrive toujours un moment où on atteint les limites de la classification arborescente et où deux classes distinctes deviennent pertinentes pour référencer un même document, même si la logique de l'auteur en privilégie une. Depuis quelques années, l'Ina propose une offre thématisée ou éditorialisée sur ses fonds numérisés. Cette offre regroupe quelques 200 corpus thématiques organisés en arborescence. La possibilité d'effectuer une recherche textuelle sur les descriptions de corpus et de documents permet d'afficher toutes les réponses à une requête dans le contexte de l'arborescence. L'utilisateur peut alors naviguer dans le voisinage de ses résultats. La Figure 128 montre la partie de l'arborescence correspondant à la requête « Sartre ». Un document est relatif à la présence de Sartre lors d'une grève des usines Renault. L'utilisateur dispose du voisinage de ce document dans l'arborescence, voisinage qui lui permet tout d'abord de s'informer sur le contenu du document. Le sujet principal de ce document repose sur les grèves Renault et non sur Sartre, donc sa présence à l'image sera ponctuelle. Ce voisinage lui permet aussi d'appréhender le contexte général de la requête : Renault, le pouvoir syndical et la position de Sartre dans ce conflit social. Cette interface permet à l'utilisateur de comprendre la logique de la classification et d'en tirer partie. Cette interface d'interrogation et de visualisation sera appliquée au parcourt de l'offre thématisée du service internet Inamedia dans le courant de l'année.

Figure 128 : Visualisation sélective d'arborescence de corpus thématiques.

Conclusion

Durant cette thèse nous avons travaillé principalement sur deux représentations cartographiques de l'information, les diagrammes de Venn-Euler pour la visualisation d'ensembles et de leurs intersections, et les diagrammes nœud-lien pour la visualisation de réseaux.

Nous avons conçu un prototype de formulation de requêtes booléennes et de visualisation des ensembles résultats de recherche utilisant les diagrammes de Venn. Ce prototype a fait l'objet d'un retour positif de la part des documentalistes de l'Ina. Toutefois, les diagrammes de Venn présentent un défaut de scalabilité et de flexibilité. C'est pourquoi nous nous sommes intéressés à la construction dynamique de diagrammes d'Euler. Nous avons donc créé et implémenté un nouvel algorithme qui permet de construire des diagrammes d'Euler représentant jusqu'à huit ensembles. Nous n'avons pas encore pu intégrer ces diagrammes dans notre prototype de formulation, il n'a donc pas été possible de comparer ces deux types de diagrammes afin de vérifier plusieurs hypothèses. La minimisation du nombre de régions dans un diagramme d'Euler facilite-t-elle vraiment leur lecture ? Les diagrammes de Venn ont l'avantage d'être des représentations statiques, nous utilisons toujours les cinq mêmes diagrammes de Venn alors qu'il existe une infinité de diagrammes d'Euler. On peut donc supposer que si les diagrammes de Venn peuvent faire l'objet d'un apprentissage, il n'en sera jamais de même pour les diagrammes d'Euler. Ces hypothèses devraient être vérifiées par une évaluation auprès d'utilisateurs.

Par ailleurs, Nous projetons de faire évoluer notre algorithme de création de diagrammes d'Euler afin d'être capables de contrôler la surface des régions. La surface d'une région dans un diagramme de Venn-Euler est l'incarnation de la variable graphique de taille, elle est donc naturellement perçue comme représentant la taille de l'ensemble qui lui est associé. Le contrôle des surfaces permettraient aux utilisateurs de déterminer plus efficacement la taille des différents sous-ensembles. Ce contrôle implique de retravailler le placement du graphe dual et l'algorithme de dessin.

Notre deuxième réalisation concerne la visualisation de graphes avec les diagrammes *nœud-lien*. Nous avons conçu l'API de visualisation Grapho et Grapher, son prototype d'application. Cette API intègre la plupart des calculs de métriques, des techniques placement et de codage graphique décrites dans ce document. Nous avons utilisé le prototype Grapher afin d'analyser différents graphes issus de données réelles. Les fonctionnalités de placement, de filtrage et de codage graphique nous ont permis de déterminer de nombreuses propriétés sur la structure de ces réseaux, d'en déterminer les entités d'intérêt.

Nous avons vu que les algorithmes de placement basés sur les modèles de forces donnent des résultats de bonne qualité. Il apparaît de plus, que les modèles classiques et de clustering visuel sont relativement complémentaires. Les modèles de Fruchterman-Reingold, Eades et Kamada-Kawai permettent de visualiser la structure des graphes de faible densité, et les modèles LinLog de Noack produisent des placements qui révèlent les sous-structures ou clusters des graphes denses. Dans ce dernier cas, il est cependant difficile de visualiser la manière dont s'articulent ces différents groupes. Un réseau dense reste malgré tout composé localement de régions peu denses, et les sommets les composants sont généralement placé de manière inadéquate. Ce comportement nous empêche d'identifier certaines caractéristiques comme l'articulation des clusters entre eux. Considérons par exemple, le cas simple de deux clusters très denses connectés par un seul chemin. Les sommets sur ce chemin ont un coefficient de clustering très faible. Le chemin peut être considéré comme un graphe de

densité négligeable que les modèles LinLog sont tout à fait incapables de traiter. Or la présence d'un chemin unique entre deux clusters est une propriété particulièrement intéressante que nous souhaiterions être capables d'identifier. Nous aimerions donc proposer une méthode mixte de placement, dans laquelle les modèles LinLog seraient utilisés pour identifier des clusters. La connaissance à priori des clusters pourraient ensuite être utilisée pour produire un placement multi-échelle basé sur un modèle de force classique. Le modèle classique aura alors pour tâche de placer les clusters de même niveaux les uns par rapport aux autres. Nous pensons que ce type de méthode pourrait permettre de repérer des caractéristiques des graphes denses considérées comme du bruit par les modèles LinLog.

Nous souhaitons continuer nos expérimentations de visualisation de graphes dans le futur. Le Dépôt-Légal du Web devrait bientôt être officialisé et nous restons donc en étroite collaboration avec son équipe sur les questions d'accès et de visualisation. Le Département des Projets et Méthodes, responsable des développements informatique internes à l'Ina s'est aussi montré très intéressé par cette approche. Notamment pour la visualisation de réseaux de documents ou de termes descripteurs contextuels aux recherches d'un utilisateur. Enfin, l'Ina participe au projet européen Infomedia, notamment sur la conception d'une application de navigation dans des contenus multimédias utilisant des liens de similarité, entre textes, images et sons, voir l'annexe B pour plus de détails.

Annexe A : Taxonomie des tâches

Quelles sont les tâches qu'un utilisateur doit pouvoir effectuer sur un graphe. La structure de graphe peut être employée pour de nombreuses applications différentes, comme la supervision d'un réseau de communication ou de transport, l'analyse de réseaux sociaux, l'étude de l'évolution du Web,…

Nous proposons une taxonomie généraliste des tâches réalisables sur un graphe. Pour réaliser cette taxonomie nous nous sommes basés sur FADIVA [Ioannidis95] qui est une plateforme de tests pour les applications de visualisation d'information. FADIVA fournit une liste de tâches abstraites supportées par les systèmes de visualisation. Nous avons associé à chacune de ces dix tâches abstraites un certain nombre de tâches spécifiques à l'analyse des graphes. Cette liste de tâches concrètes ne se veut pas exhaustive, mais elle nous semble constituer un bon point de départ.

1. **Trouver un petit nombre d'éléments correspondant à une requête précise**
   - des catégories d'entités (attribut qualitatif)
   - des classes d'entités par rapport à un attribut numérique
   - des outliers par rapport à un attribut numérique
2. **Trouver un petit nombre d'éléments par simple exploration**
   - identifier les arêtes entrantes/sortantes à un sommet
   - identifier les sommets voisins entrants/sortants d'un sommet
   - identifier les arêtes dans le k-voisinage d'un sommet
   - identifier les sommets dans le k-voisinage d'un sommet
3. **Identifier un motif :**
   a. **Identifier une tendance**
   b. **Identifier un cluster**
      - identifier des structures ressemblant à
        - des arbres
        - des cliques (agrégats)
        - des treillis
4. **Identifier des anomalies**
   - identifier différentes composantes connexes
5. **Identifier un petit nombre d'éléments en « suivant un chemin dans les données »**
   - identifier les plus courts chemins entre 2 sommets
   - identifier les plus courts chemins entre 2 groupes de sommets
6. **Comparer les résultats de plusieurs requêtes**
   - comparer deux graphes ou sous-graphes différents
   - comparer deux placements du même graphe
   - analyser l'évolution d'un graphe
7. **Obtenir une vue d'ensemble d'une grande masse de données**
   - calculer un placement global du graphe.
8. **Interpréter/discriminer/expliquer des motifs**
   - identifier le couplage entre 2 groupes de sommets
     - goulets d'étranglement
     - chemins multiples

- identifier les chemins de diffusions
    - la propagation
    - l'importance du flux pour chaque sommet et arêtes

9. Ajouter/supprimer des éléments

- ajouts d'entités
- création d'une méta-structure comme une hiérarchie de clusters par exemple
- filtrer : simplification de la structure
    - supprimer des entités selon des critères sur leurs attributs endogènes
        - ex : filtrage arborescent
        - ex : filtrage d'arêtes inter/intra agrégats
    - supprimer des entités selon des critères sur leurs attributs exogènes
        - ex : supprimer les sommets/arêtes de poids faible

10. Modifier des éléments

- modifications d'entités
- calcul d'attributs

Certaines tâches concrètes peuvent être rattachées à plusieurs tâches abstraites de FADIVA. Par exemple, pour obtenir une vue d'ensemble d'un graphe complexe, le placement est vraiment essentiel, mais l'application d'un codage couleur/taille peut aussi jouer un rôle déterminant. Le filtrage est naturellement placé dans la tâche abstraite suppression (tâche 9) de FADIVA, mais c'est aussi une manière de sélectionner un ensemble d'entités correspondant à une requête (tâche 1).

De plus pour certaines fonctions spécifiques aux graphes, comme la propagation d'un flux dans un réseau ou l'analyse de l'évolution d'un réseau dans le temps, l'intégration dans une plateforme généraliste comme FADIVA n'est pas facile.

Annexe B : Analyse de liens de similarité entre documents audiovisuels

Ces impressions d'écran sont issues des travaux de Hervé Goëau, doctorant à l'Ina. Elles ont été réalisées avec Grapher. Ces travaux étudient l'intérêt des graphes pour présenter des vues synthétiques de programmes audiovisuels, ici un journal télévisé.

Les graphes considérés ont pour sommets des images issues du flux vidéo. Ces images sont sélectionnées par détection de pics d'activité. Les arêtes peuvent correspondre à différents types de relations :

- relation temporelle, une arête orientée (ou arc) connecte deux images qui se suivent dans le flux,
- relation de similarité, l'arête porte la mesure d'une distance entre les deux images qu'elle relie. Dans ce cas il y a une arête entre chaque paire de sommets.

Les distances utilisées sont des mesures de similarité entre images.

En Figure 129, le graphe est constitué à l'origine des arêtes temporelles et des arêtes représentant la distance moyenne entre images dans l'espace RGB. Les arêtes dont la distance image dépassait un certain seuil ont été supprimées, le graphe a ensuite été placé par le modèle de Fruchterman-Reingold. Cette visualisation permet de dégager une structure grossière du journal télévisé.

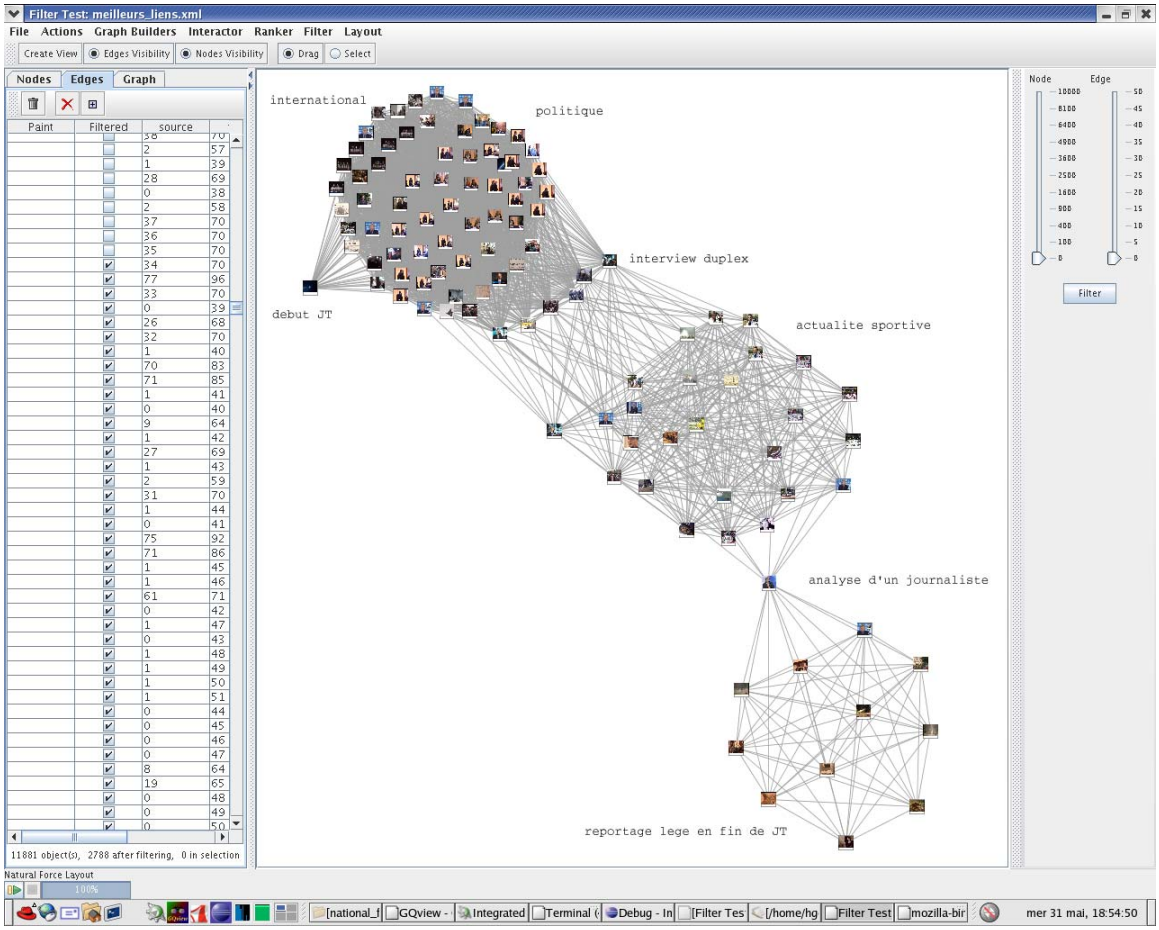


Figure 129 : analyse de la structure d'un journal télévisé

Sur la Figure 130, un procédé plus raffiné a été utilisé. Une contrainte temporelle plus forte a été imposée. A l'origine seule les arêtes temporelles ont été conservées, ce qui donne un graphe qui présente une structure de liste. Ensuite, les liens de similarité image les plus proche de l'image du présentateur sont réinsérés. Cela a pour conséquence de connecter tous les sommets représentant le présentateur, et de faire apparaître plusieurs boucles. Il en résulte une visualisation de la structure narrative du journal télévisé. Nous pouvons voir la colonne vertébrale comme étant le fil conducteur de la retransmission, induite par le présentateur. Chaque boucle représente un reportage. L'intérêt d'une telle visualisation est d'appréhender rapidement le contenu du journal. Nous repérons immédiatement 10 reportages dont une boucle plus longue indiquant l'évènement du jour. La dominante bleue au centre droit et l'alternance de petites boucles nous indique qu'il s'agit d'une interview plateau, l'intervention d'un analyste en l'occurrence.

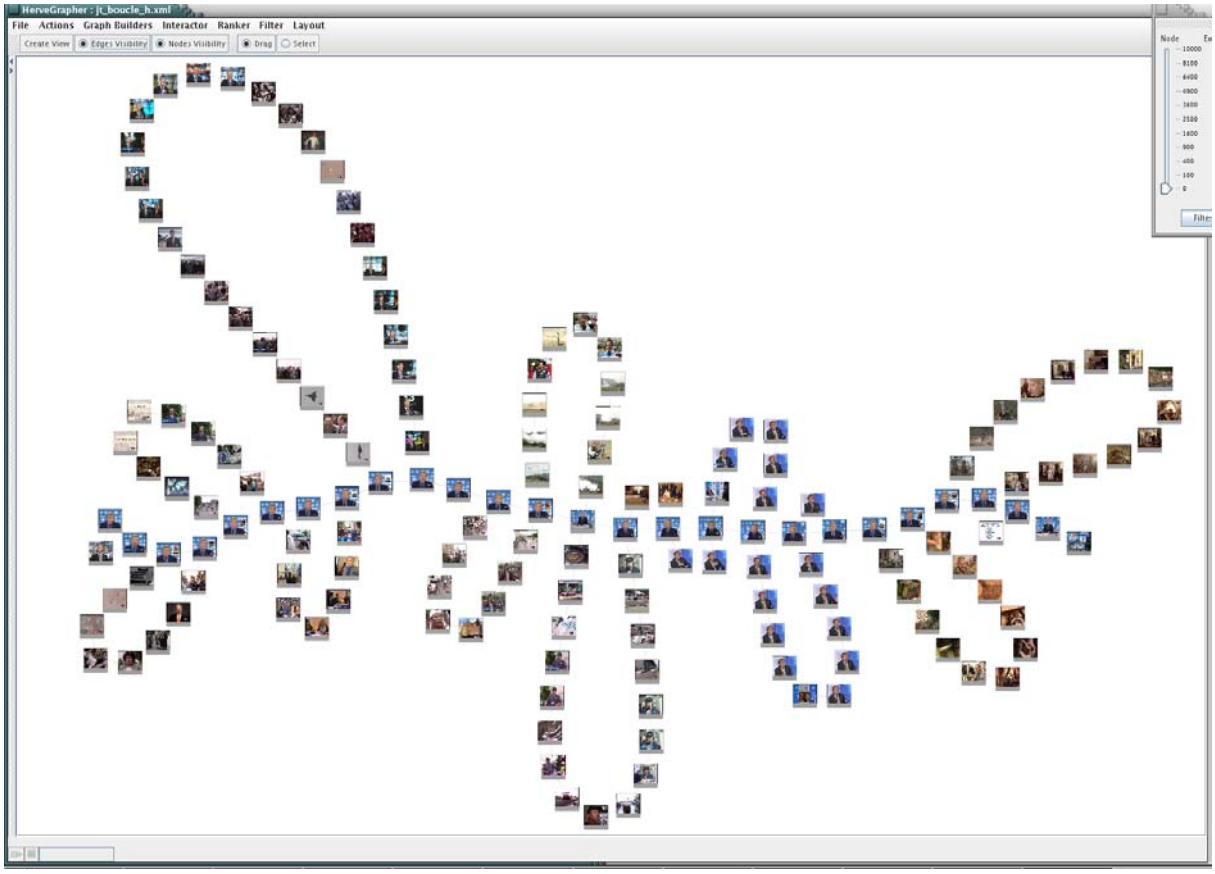


Figure 130 : représentation temporelle d'un journal télévisé

Références

[**Anick90**] P.G. Anick, J.D. Brennan, R.A. Flynn, D.R. Hanssen, B. Alvey and J. Robbins. "*A direct manipulation interface for Boolean information retrieval via natural language query*". Proceedings of the ACM SIGIR Conference, 135-150, 1990.

[**Anthonisse71**] J.M. Anthonisse. "*The Rush in a Directed Graph*". Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam, Oct. 1971.

[**Adar06**] E. Adar. "GUESS*: A Language and Interface for Graph Exploration*". To appear at CHI 2006,

[**Albert02**] R. Albert and A. Barabasi. "*Statistical mechanics of complex networks*". Review Modern Physics, 74, 47, 2002.

[**Auber03a**] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon. "*Multiscale visualization of small world networks*". In Proc. of IEEE Symposium on Information Visualization, pages 75--81, 2003.

[**Auber03b**] D. Auber. "*Tulip : A huge graph visualisation framework*". In P. Mutzel and M. Jnger, editors, Graph Drawing Softwares, Mathematics and Visualization, pages 105--126. Springer-Verlag, 2003.

[**Balzer05**] M. Balzer, O. Deussen and C. Lewerentz. "*Voronoi treemaps for the visualzation of software metrics*". in Proceedings of the 2005 ACM symposium on Software visualization, pp. 165-172. ACM Press, 2005.

[**Baesa99**] R. Baeza-Yates and B. Ribeiro-Neto. "*Modern Information Retrieval*", ACM Press, New York, NY, 1999.

[**Barabasi00**] A. Barabasi, R. Albert and H. Jeong. "Scale-*free characteristics of random networks: The topology of the world wide web*". Physica A, 281:69-77, 2000.

[**Barnes86**] J. Barnes and P. Hut, "*A Hierarchical O(N Log N) Force Calculation Algorithm*". Nature 324(4), 1986.

[**Batagelj98**] V. Batagelj and A. Mrvar. "*Pajek - Program for Large Network Analysis*". Connections 21:47-57 (1998).

[**Battista94**] G. Battista, P. Eades, R. Tamassia, and I.G. Tollis. "*Algorithms for drawing graphs: An annotated bibliography*". Computational Geometry: Theory and Applications, 4 (5). 235-282, 1994.

[**Beauchamp65**] M. A. Beauchamp. "*An improved index of centrality*". Behavioral Science, 10:161-- 163, 1965.

[**Bederson96**] B. Bederson, J. Hollan, K. Perlin, J. Meyer, D. Bacon and G. Furnas. "*Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics*". Journal of Visual Languages and Computing, 7, pp. 3-31, 1996.

[**Bederson03**] B. Bederson, J. Grosjean and J. Meyer. "*Toolkit Design for Interactive Structured Graphics*". Tech Report HCIL2003 -01, Computer Science Department, University of Maryland, College Park, MD.

[**Benoy05**] F. Benoy and P. Rodgers. "*Evaluating the Comprehension of Euler Diagrams*". Proceedings of Euler Diagrams Workshop 2005.

[**Bergman95**] L. D. Bergman, B. E. Rogowitz and L. A. Treinish. "*A Rule-based Tool for Assisting Colormap Selection*". IBM TJ Watson Research. 1995.

[**Bollobas85**] B. Bollobas. "*Random graphs*". Academic Press, London, England, 1985.

[**Bonacich95**] P. Bonacich. "*Factoring and weighting approaches to status scores and clique identification*". Journal of Mathematical Sociology, 2. 113-120.

[**Boutin04**] F. Boutin and M. Hascoët. "Cluster *Validity Indices for Graph Partitioning*". Proceedings of IV 2004, IEEE, 2004.

[**Boutin05**] F. Boutin, J Thièvre and M. Hascoët. "*Multilevel Compound Tree – Construction, Visualization and Interaction*". Interact'05, Roma, 2005.

[**Boutin06**] F. Boutin, J. Thièvre and M. Hascoët. "*Focus-based filtering + clustering technique for power-law networks with small world phenomenon*". Visualization and Data Analysis 2006, San Jose, 2006.

[**Bowles71**] L. Bowles. "*Logic Diagrams for up to n classes*". The Mathematical Gazette 55, pp.370-373, 1971.

[**Boyle83**] J. Boyle, K. Bury and R. Evey. "*Two studies evaluating learning and use of QBE and SQL*". Proceedings of the Human Factors Society 27th Annual Meeting, 663-667. Santa Monica, CA: Human Factors Society. 1983.

[**Brandenburg96**] F. J. Brandenburg, M. Himsolt, and C. Rohrer. "*An experimental comparison of force-directed and randomized graph drawing algorithms*". In F. J. Brandenburg, editor, Graph Drawing (Proc. GD '95), volume 1027 of Lecture Notes in Computer Science, pp. 76-87. Springer-Verlag, 1996.

[**Brandes99**] U. Brandes, P. Kenis, J. Raab, V. Schneider, and D. Wagner. "*Explorations into the visualization of policy networks*". Journal of Theoretical Politics, 11(1):75-106, 1999.

[**Brandes01**] U. Brandes. "*A faster algorithm for betweenness centrality*". Journal of Mathematical Sociology, 25(2):163--177, 2001.

[**Brandes04**] U. Brandes and D. Wagner. "*Visone - Analysis and Visualization of Social Networks*". In M. Jünger and P. Mutzel (Eds.): Graph Drawing Software, pp. 321-340. © Springer-Verlag, 2004.

[**Brin98**] S. Brin and L. Page. "*The anatomy of a large-scale hypertextual web search engine*". In Proceedings of WWW7, 1998.

[**Brewer98**] C.A. Brewer. "*Color Use Guidelines for Mapping and Visualization*". In A.M.MacEachren and D.R. Fraser Taylor (eds.) Visualization in Modern Cartography. Elsivier Science Ltd, 1994, pp.123-147. G. & N. Andrienko, GMD SET.KI, 1998 Last updated 30th of November, 1998

[**Bruls00**] M. Bruls, K. Huizing, and J.J. vanWijk. "*Squarifed treemaps*". In Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization, pp. 33-42. IEEE Computer Society, 2000.

[**Carroll00**] J. Carroll. "*Drawing Venn triangles*". Technical Report HPL-2000-73, HP Labs (2000).

[**Carroll05**] J. Carroll. "*Which n-Venn diagrams can be drawn with convex k-gons?*". Personal communication, May 2005.
[**Chow04**] S. Chow and F. Ruskey. "Towards *a General Solution to Drawing Area Proportional Euler Diagrams*". Proceedings of Euler Diagrams 04. ENTCS. Springer Verlag, 2004.
[**Davidson89**] R. Davidson and D. Harel. "*Drawing graphs nicely using simulated annealing*", Technical Report CS89-13, Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot, Israel, 1989. A revised version, dated April, 1991, has been submitted to Communicationsof the ACM.
[**Eades84**] P. Eades. "*A Heuristic for Graph Drawing*", Congressus Numerantium 42, pp.149-160, 1984.
[**Eades92**], P. Eades. "*Drawing Free Trees*". Bulletin of the Institute of Combinatorics and its Applications, pp.10-36, 1992.
[**Edwards89**] A. W. F. Edwards. "*Venn diagrams for many sets*". New Scientist 7, pp. 51-56, 1989.
[**Erdos59**] P. Erdos and A. Renyi. "*On random graphs*". Publ. Math. Debrecen, 6:290--297, 1959.
[**Euler61**] L. Euler. "*Lettres a Une Princesse d'Allemagne*". Volume 2. Letters No. 102--108. 1761.
[**Faust95**] K. Faust and W. Stanley. "*Social Network Analysis : Methods and Applications*". Cambridge University Press, 1995.
[**Fekete04**] J.D. Fekete. "*The InfoVis Toolkit*". Proceedings of the 10th IEEE Symposium on Information Visualization (InfoVis'04), Austin, TX, Oct 2004. IEEE Press. pp. 167-174
[**Fisher88**] J. Fisher, Chris et al. "*Diagrams Venn ans How*". Mathematics Magazine 61, pp.36-40, 1988.
[**Flower02**] J. Flower and John Howse. "*Generating Euler Diagrams*". Proceedings of the Second International Conference on Diagrammatic Representation and Inference, pp. 61 – 75, 2002.
[**Freeman77**] L.C. Freeman. "*A Set of Measures of Centrality Based on Betweenness*". Sociometry. 40(6):35--41. 1977.
[**Fruchterman91**] T. M. J. Fruchterman and E. M. Reingold. "*Graph drawing by force-directed placement*". Software: Practice and Experience, 21(11), pp.1129-1164, 1991.
[**Furnas86**] G. W. Furnas. "*Generalized fisheye views*", in CHI '86, pp.16-23, 1986.
[**Ghoniem04**] M. Ghoniem, J.D. Fekete and P. Castagliola. "*A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations*". In Proceedings of the IEEE Symposium on information Visualization (infovis'04) - Volume 00 (October 10 - 12, 2004). INFOVIS. IEEE Computer Society, Washington, DC, 17-24, 2004.
[**Gibson77**] J.J. Gibson. "*On the analysis of change in the optic array*". Scandinavian Journal of Psychology 18(3): 161-163, 1977.
[**Glassner03**] A. Glassner. "*Venn and Now*". IEEE Computer Graphics and Applications, Volume 23 (no. 4), pp.82-95, 2003.
[**Gosset04**] N. Gosset and B. Chen. "*Intuitive Color Mixing and Compositing for Visualization*". In Proceedings of Infovis 2004.
[**Grokker04**] Grokker, http://www.groxis.com, 2004.
[**Grunbaum75**] B. Grünbaum. "*Venn diagrams and Independent Families of Sets*". Mathematics Magazine 48, pp.12-23, 1975. [Grünbaum awarded the MAA Lester R. Ford prize for this paper in 1976 (see AMM, Aug-Sept. 1976, pg. 587)]
[**Grunbaum99**] B. Grünbaum. "*The Search for Symmetric Venn Diagrams*". Geombinatorics 1, pp.104-109, 1999.
[**Harary94**] F. Harary. "*Graph Theory*". Reading, MA: Addison-Wesley, pp. 113-115, 1994.
[**Healey96**] C. Healey. (1996). "*Choosing effective colours for data visualization*". IEEE Visualization. p. 263.
[**Hearst99**] M. Hearst. "*User Interfaces and Visualization*", chapter 10 in [Baeza99], 1999.
[**Heer05**] J. Heer, S. K. Card and J. A. Landay. "*Prefuse : a toolkit for interactive information viusalization*". Proceedings of CHI 2005.
[**Herman00**] I. Herman, G. Melancon and M.S. Marshall. "*Graph visualization and navigation in information visualisation: a survey*". IEEE Transactions on Visualization and Computer Graphics, 6(1):24--43, 2000.
[**Hopcroft74**] J. Hopcroft and R.E. Tarjan. "*Efficient planarity testing*". J. ACM 21 (1974), 549--568.
[**Igarashi00**] T. Igarashi and K. Hinckley. "*Speed-dependent automatic zooming for browsing large documents*". In Proc. UIST 2000, 139-148. 200.
[**Ioannidis95**] Y. Ioannidis. "*The FADIVA Benchmark - Version 0**", 1995.
[**Johnson91**] B. Johnson and B. Shneiderman. "*Tree maps: A space-filling approach to the visualization of hierarchical information structures*". In Proceedings of the 2nd International IEEE Visualization Conference, pp. 284-291. IEEE Computer Society, 1991.
[**Jones98**] S. Jones. "Graphical *query specification and dynamic result previews for a digital library*". Proceedings: ACM Symposium on User Interface Software and Technology, pp. 143-151, 1998.
[**Kamada89**] T. Kamada and S. Kawai. "*An Algorithm for Drawing General Undirected Graphs*". Information Processing Letters 31, pp.7-15, 1989.
[**Katz53**] L. Katz. "*A new status index derived from sociometric analysis*". Psychmetrika, 18(1):39--43, 1953.
[**Kim04**] D.H. Kim, J.D. Noh and H. Jeong. "*Scale-free trees: the skeletons of complex networks*". Physical Review, 2004. E 70(046126).
[**Kleinberg99**] J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. "*The Web as a Graph: Measurements, Models, and Methods*". In 5th International Conference on Computing and Combinatorics, Tokyo, July 26-28, 1999.
[**Kleinberg99**] J. M. Kleinberg. "*Authoritative sources in a hyperlinked environment*". JACM, 46(5):604--632, 1999.
[**Koffka35**] K. Koffka. "*Principles of Gestalt Psychology*". Harcourt-Brace, New York, 1935.
[**Kruskal56**] J. Kruskal. "*On the shortest spanning subtree of a graph and the traveling salesman problem*". Proc. American Mathematical Society, 7:48--50, 1956.

[**Kumar00a**] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tompkins and E. Upfal. "*The web as a graph*". In Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pages 1--10. ACM Press, 2000.
[**Kumar00b**] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. "*Stochastic models for the web graph*". In Proc. 41st IEEE Symp. on Foundations of Computer Science, 2000.
[**Kuratowski30**] K. Kuratowski. "*Sur le problème des courbes gauches en topologie*". Fundamenta Mathematicae 15, pp.271-283, 1930.
[**Lamping94**] J. Lamping and R. Rao. "*Laying out and visualizing large trees using a hyperbolic space*". In Proceedings of UIST'94, pp.13-14, 1994.
[**Larkin87**] J. H. Larkin and H. A. Simon. "*Why a diagram is (sometimes) worth ten thousand words*". Cognitive Science, Issue 11, PP. 65-99, 1987.
[**Luhn57**] H.P. Luhn. "A *statistical approach to mechanized encoding and searching of literary information*". IBM Journal of Research and Development, 1957.
[**Marshall01**] M. S. Marshall, I. Herman and G. Melancon. "*An Object-oriented Design for Graph Visualization*". Software Practice and Experience, 2001, Vol. 31, pp 739-765.
[**Melancon98**], G. Melançon and I. Herman. "*Circular Drawings of Rooted Trees*". Reports of the Centre for Mathematics and Computer Sciences, Report number INS--9817, 1998.
[**Michard82**] A. Michard. "*Graphical Presentation of Boolean Expressions in a Database Query Language: Design Notes and an Ergonomic Evaluation*". Behaviour and Information Technology, 1, 3 (1982), 279-288.
[**Milgram67**] S. Milgram. "*The small world problem*". Psychology Today, 22, 61-67. 1967
[**Miller56**] G. A. Miller. "*The magical number seven, plus or minus two: some limits on our capacity for processing information*". Psychological Review, vol. 63, pp. 81-97. 1956.
[**More59**] T. More. "*On the construction of Venn Diagrams*". The Journal of Symbolic Logic 24, pp.303-304, 1959.
[**Newman02**] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. "*Random graph models of social networks*". Proceedings of the National Academy of Sciences, USA, 99(Suppl. 1):2566--2572, 2002.
[**Noack03**] A. Noack. "*An energy model for visual graph clustering*". In G. Liotta, editor, Proceedings of the 11th International Symposium on Graph Drawing (GD 2003), LNCS 2912, pp. 425-436, Springer-Verlag, Berlin, 2004.
[**Noack05**] A. Noack. "*Energy-Based Clustering of Graphs with Nonuniform Degrees*". Accepted for publication in: Proceedings of the 13th International Symposium on Graph Drawing (GD 2005, Limerick, Ireland, Sep. 12-14), © Springer-Verlag, 2005.
[**Perlin93**] K . Perlin and D. Fox. "*Pad - An Alternative Approach to the Computer Interface*". In SIGGRAPH 93 Conference Proceedings, pp. 57-64, 1993.
[**Plaisant02**] C. Plaisant, J. Grosjean and B. Bederson. "*SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation*". In Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02), pp.57, 2002.
[**Polythress71**] Polythress, Vern ans Hugo Sun. "*A Method to Construct Convex Connected Venn Diagrams for Any Finite Number of Sets*". The Pentagon 31, pp.80-83,1971.
[**Purchase97**] H.C. Purchase, R.F. Cohen, and M.I. James. "*An experimental study of the basis for graph drawing algorithms*". Journal of Experimental Algorithmics , 2:4, 1997.
[**Rao94**] R. Rao, and S.K. Card. "*The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus and Context Visualization for Tabular Information*". In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Celebrating Interdependence (CHI'94), ACM Press, pp.318—22. 1994.
[**Reingold81**] E.M. Reingold and J.S. Tilford. "*Tidier drawings of trees*". IEEE Transactions on Software Engineering, 7(2):223—228, 1981.
[**Robertson86**] P. K. Robertson and J. F. O'Callaghan. "*The Generation of Color Sequences for Univariate and Bivariate Mapping*". IEEE Computer Graphics and Applications, 6(2):24—32. 1986.
[**Robertson91**] G. G. Robertson, J. D. Mackinlay, and S. K. Card. "*Cone trees: Animated 3D visualizations of hierarchical information*". In Proceedings of the Conference on Human Factors in Computing Systems (CHI'91), pp.189-194, 1991.
[**Rogowitz96**] B. Rogowitz and L. Treinish. "*How NOT to lie with visualization*". Computers in Physics, 10(3):268—274. 1996.
[**Ruskey01**] F. Ruskey and M. Weston. "A *survey of Venn diagrams*". The Electronic Journal of Combinatorics, 1997. Dynamic survey, Article DS5. http://www.combinatorics.org/Surveys/ds5/VennEJC.html. Revised 2001, 2005.
[**Sadibussi96**] G. Sabidussi. "*The centrality index of a graph*". Psychometirka 31, 581-606, 1996.
[**Sarkar93**] M. Sarkar and M. Brown. "*Graphical fisheye views*". Communications of the ACM, 37(12), pp.73-84, 1993.
[**Saulnier05**] A. Saulnier. "*La perception du mouvement dans les systèmes de visualisation d'information*". Proceedings of IHM 2005.
[**Shneiderman98**] B. Shneiderman, D. Byrd, and W. B. Croft. "*Sorting out Searching: A User-Interface Framework for Text Searches*". Communications of the ACM. 41. 4. pp.95-98, 1998.
[**Shneiderman01**] B. Shneiderman and M. Wattenberg. "*Ordered treemap layouts*". In Proceedings of the IEEE Symposium on Information Visualization, pp. 73-78. IEEE Computer Society, 2001.
[**Spoerri99**] A. Spoerri. "*InfoCrystal: A Visual Tool for Information Retrieval*". Readings in Information Visualization: Using Vision to Think. Eds. S. Card, J. Mackinlay and B. Shneiderman. (San Francisco, California: Morgan Kaufmann), pp. 140 – 147, 1999. (Originally published in 1993).

[**Spoerri04**] A. Spoerri. "MetaCrystal*: Visualizing the Degree of Overlap Between Search Engines*". WWW 2004. Proceedings of the ACM International World Wide Web Conference. New York, NY, May 17 – 22. 2004.
[**Thievre03**] J. Thièvre et M. Viaud. "*Diagramme de Venn et Recherche d'Information*". IHM 2003, Poitiers, 2003.
[**Thievre04**] J. Thièvre, M. Viaud and A. Verroust-Blondet. "*Using Euler Diagrams in Traditional Libary Environments*". Euler Diagrams 04, Brighton, 2004.
[**Thievre05**] J. Thièvre, A. Verroust-Blondet and M. Viaud. "*Drawing Diagrams from Labelled Graphs*". Euler Diagrams 05, Paris, 2005.
[**Tracktion**] *Tracktion v2*, www.mackie.com/products/tracktion/.
[**Valente98**] T. W. Valente and R. K. Foreman. "*Integration and radiality: Measuring the extent of an individual's connectedness and reachability in a network*". Social Networks, 20(1):89--105, 1998.
[**VanHam03**] F. van Ham. "*Using Multi-Level Call Matrices in Large Software Projects*". Proc. IEEE Symp. Information Visualization 2003.
[**Venn80**] J. Venn. "*On the diagrammatic and mechanical representation of propositions and reasonings*". The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 9 (1880) 118.
[**Verroust04**]A. Verroust. and M-L. Viaud. "*Ensuring the drawability of Extended Euler Diagrams for up to 8 sets*". (reduced version) Diagrams'2004. International Conference on the Theory and Application of Diagrams, Cambridge, Mars 2004.
[**Walker90**] J. Q. Walker II. "*A node-positioning algorithm for general trees*". Software, Practice and Experience, 20(7), pp.685-705, 1990.
[**Ware95**] C. Ware, and W. Knight. "*Using visual texture for information display*". In ACM Trans. Graph. 14, 1, pp. 3--20. 1995.
[**Ware04**] C. Ware. "*Information Visualization: Perception for Design*". Morgan Kaufmann Publishers, 2nd Edition, 2004.
[**Wattenberg98**] M. Wattenberg. "*Map of the Market*". http://www.smartmoney.com/marketmap, 1998.
[**Wattenberg99**] M. Wattenberg. "*Visualizing the Stock Market*". In Proceedings of Extended Abstracts of Human Factors inComputing Systems (CHI 99) ACM Press, pp. 188-189, 1999.
[**Watts99**] D.J. Watts. "*Small Worlds*". Princeton University Press, 1999.
[**Yee01**] K.P. Yee, D. Fisher, R. Dhamija, M. Hearst. "*Animated exploration of dynamic graphs with radial layout*". In Proceeding of IEEE Symposium on Information Visualization 2001.
[**Young93**] D. Young and B. Shneiderman. "A *Graphical Filter Flow Representation of Boolean Queries: A Prototype Implementation and Evaluation*". In Journal of the American Society for Information Scienc', Vol. 44(6), pp. 327—339, 1993.