La visualisation d'information pour les données massives : une approche par l'abstraction de données

Joris Sansen

THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

par **Joris Sansen**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : INFORMATIQUE

La visualisation d'information pour les données massives: Une approche par l'abstraction de données

---

Date de soutenance : 04 Juillet 2017

Devant la commission d'examen composée de :

C. Hurter	Professeur des universités, ENAC	Rapporteur
P. Poncelet	Professeur des universités, LIRMM	Rapporteur
G. Venturini	Professeur des universités, EPU - Polytech'Tours	Président du jury
G. Blin	Professeur des universités, LaBRI	Examinateur
D. Auber	Maître de conférence, LaBRI	Directeur de thèse
R. Bourqui	Maître de conférence, LaBRI	Encadrant

– 2017 –

Résumé L'évolution et la démocratisation des technologies ont engendré une véritable explosion de l'information et notre capacité à générer des données et le besoin de les analyser n'a jamais été aussi important. Pourtant, les problématiques soulevées par l'accumulation de données (stockage, temps de traitement, hétérogénéité, vitesse de captation/génération, *etc.*) sont d'autant plus fortes que les données sont massives, complexes et variées. La représentation de l'information, de part sa capacité à synthétiser et à condenser des données, se constitue naturellement comme une approche pour les analyser mais ne résout pas pour autant ces problèmes. En effet, les techniques classiques de visualisation sont rarement adaptées pour gérer et traiter cette masse d'informations. De plus, les problèmes que soulèvent le stockage et le temps de traitement se répercutent sur le système d'analyse avec par exemple, la distanciation de plus en plus forte entre la donnée et l'utilisateur : le lieu où elle sera stockée et traitée et l'interface utilisateur servant à l'analyse. Dans cette thèse nous nous intéressons à ces problématiques et plus particulièrement à l'adaptation des techniques de visualisation d'informations pour les données massives. Pour cela, nous nous intéressons tout d'abord à l'information de relation entre éléments, comment est-elle véhiculée et comment améliorer cette transmission dans le contexte de données hiérarchisées. Ensuite, nous nous intéressons à des données multi-variées, dont la complexité à un impact sur les calculs possibles. Enfin, nous présentons les approches mises en œuvre pour rendre nos méthodes compatibles avec les données massives.

Title Information visualization for big data: a data abstraction approach

Abstract The evolution and spread of technologies have led to a real explosion of information and our capacity to generate data and our need to analyze them have never been this strong. Still, the problems raised by such accumulation (storage, computation delays, diversity, speed of gathering/generation, *etc.*) is as strong as the data are big, complex and varied. Information visualization, by its ability to summarize and abridge data was naturally established as appropriate approach. However, it does not solve the problem raised by Big Data. Actually, classical visualization techniques are rarely designed to handle such mass of information. Moreover, the problems raised by data storage and computation time have repercussions on the analysis system. For example, the increasing distance between the data and the analyst : the place where the data is stored and the place where the user will perform the analyses are rarely close. In this thesis, we focused on these issues and more particularly on adapting the information visualization techniques for Big Data. First of all focus on relational data : how does the existence of a relation between entity is transmitted and how to improve this transmission for hierarchical data. Then, we focus on multi-variate data and how to handle their complexity for the required computations. Finally, we present the methods we designed to make

our techniques compatible with Big Data.

Keywords information visualization, data exploration, big data, relational data, multivariate data, hierarchical data, directed weighted graphs

Mots-clés Visualisation d'information, exploration, données massives,données relationnelles, données multivariées,données hiérarchiques,graphes orientés pondérés

Laboratoire d'accueil Laboratoire Bordelais de Recherche en Informatique
Unité Mixte de Recherche CNRS (UMR 5800)
351, cours de la Libération F-33405 Talence cedex

Remerciements

Je tiens tout d'abord à remercier La Direction générale des Entreprises (DGE) et BPIFrance qui, dans le cadre du Projet Investissement d'Avenir *SpeedData* ont financé mes travaux de thèse. Je remercie également l'Université de Bordeaux et le LaBRI pour m'avoir permis de réaliser ces travaux dans les meilleures conditions possibles. Je remercie bien évidemment les rapporteurs, les professeurs Hurter et Poncelet, pour avoir accepté de relire ce manuscrit ainsi que pour leurs remarques, nécessaires à son amélioration. Je remercie également les membres de mon jury de thèse, Messieurs Blin, Auber et Bourqui pour avoir accepté d'assister à ma soutenance.

Je tiens également à remercier l'équipe *MABioVis* et tout particulièrement l'ensemble des membres du thème *EVADoMe* pour le soutien et les conseils insufflés pendant ces quelques années ainsi que pour toute la bonne humeur et tous les bons moments partagés. Je ne remercierais jamais assez Romain et David qui m'ont donné la chance d'accomplir cette thèse avec eux, ont dirigé et guidé mes premiers pas dans la Recherche et dans l'enseignement.

Je remercie mes parents, Karell, Conrad et Évie qui m'ont toujours soutenu, poussé et encouragé dans mon parcours personnel et académique. Je pense également à ma belle-famille qui a su trouver les mots juste lorsque c'était nécessaire et m'a supporté le reste du temps. Une pensée en particulier pour mes collègues et amis Charlotte, Phi-Vu et Marie qui m'ont apporté plus que je ne saurais l'exprimer.

Enfin, je tiens à remercier ma compagne Émilie sans qui rien n'aurait été possible. Je ne saurais jamais assez la remercier pour le partage et le soutien qu'elle m'apporte à chaque instant.

à Bordeaux, le 04 Juillet 2017

Table des matières

Table des figures

Liste des tableaux

Données massives et visualisation d'information : Introduction

1 Contexte

La notion d'accumulation d'information et de données massives, malgré la popularité croissante dont elle fait l'objet depuis ces dernières années, est beaucoup moins récente que ce qu'on pourrait penser. Déjà dans les années 40 apparaissaient des mentions de la croissance en volume de l'information. En 1944, Fremont Rider (Bibliothèque de l'université de Wesleyan) estimait que la taille des bibliothèques universitaires américaines avait doublé tous les 16 ans et prévoyait qu'en 2040, la bibliothèque de l'université de Yale nécessiterait 6 000 miles d'étagères[113]. Une autre mention de cette *explosion de l'information* apparaît dans le *Lawton Constitution* en 1964 d'après l'*Oxford English Dictionnary* (voir figure 1). On retrouve ainsi différentes apparitions des problèmes soulevés par l'augmentation de l'information et de son stockage, que ce soit dans des articles scientifiques, dans des journaux de presse publique ou dans des publications de réflexion [106]. Ces différentes apparitions montrent de façon sous-jacente l'évolution progressive de notre capacité à générer et stocker des données et ses implications dans les processus de traitements et d'analyses. Le terme *Big data*, traduit ici en données massives, n'est apparu qu'en 1997 [30] avec la création et l'utilisation des *super-ordinateurs* pour décrire le problème

MIRROR OF THE MIND

By JOSEPH WHITNEY

Are people better informed than they used to be?

Answer: Yes, thanks to the information explosion. Nuclear scientist Wernher von Braun said all knowledge accumulated up to 1750 was doubled by 1900; doubled again by 1950; redoubled in the 1950-1960 decade, and will double again by 1967. Sociologists see two major possibilities of coping with this torrent of knowledge. One lies in selectivity, with each individual learning only what is vital to him. The other is automation, providing more leisure for study.

Should medicine cabinets be locked up?

Answer: It might be simpler to keep them well above the reach of children. Since 86 per cent of all accidental poisoning cases involve children under six years of age, the bathroom medicine cabinet is not often involved. A recent A.M.A. News item reported that more poisonings take place in the kitchen than in any other room. Substances more frequently swallowed include aspirin, bleach, detergents, insecticides, and vitamin and iron preparations.

Is swimming natural to mankind?

Answer: Probably not; tales of infants who swim instinctively have never been proven. Most swimming teachers say that young children learn more easily than older ones. Humans do not have an instinctive fear of water, but once fear is acquired, learning to swim becomes slower and more difficult. Persons over age 16 or so often have great difficulty learning to swim, partly due to acquired fears, or they tighten up and cannot relax in water.

FIGURE 1: L'une des première mention de l'explosion de l'information dans le *Lawton Constitution* le 30 novembre 1964.

soulevé par la masse de données que ces machines génèrent lors des calculs. Cette augmentation de plus en plus importante de la génération et du stockage d'information s'est confirmée avec les années (voir figure 2). Ainsi, Eric Schmidt, directeur général de Google, mentionnait durant la conférence *Techonomy* de 2010 : "Depuis l'aube de l'humanité et jusqu'à l'année 2003, l'Homme a créé 5 Exabytes d'informations ; cette même quantité d'informations est désormais créée tous les deux jours." [1]

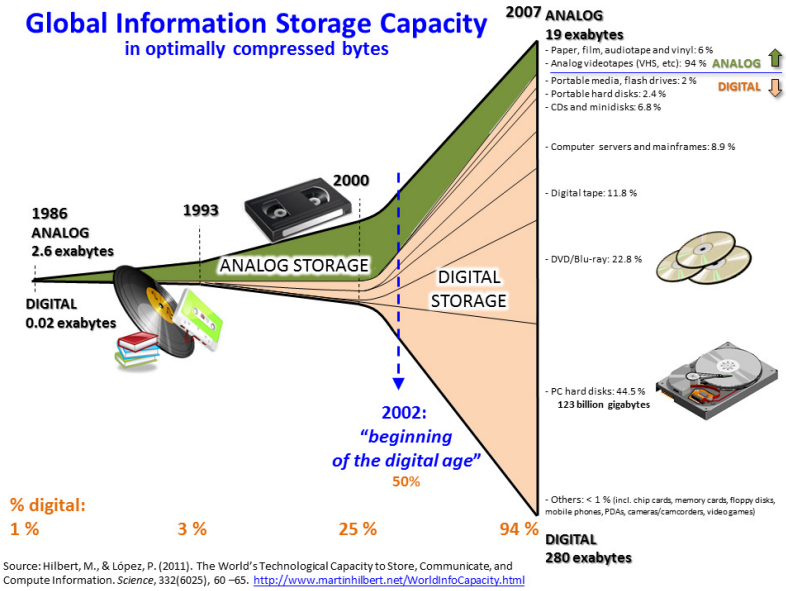


FIGURE 2: Évolution du stockage d'information et sa répartition par support entre 1986 et 2007.

Une telle masse d'information pose cependant de nombreux problèmes. Tout d'abord, elle doit pouvoir être stockée, mais surtout traitée. Or, les systèmes de gestions de bases de données classiques sont de moins en moins aptes à répondre à ces problématiques [73]. De même, il est rapidement devenu évident que de telles masses de données ne pouvaient non seulement pas être traitées sur un simple ordinateur, mais pas non plus sur des infrastructures matérielles (machine locale ou serveur unique) et logicielles classiquement utilisées. Les temps de traitement et de parcours deviennent beaucoup trop importants au fur et à mesure que la masse d'information augmente. Ces limites définissent ce qu'on appelle les données massives : un ensemble de données difficile à traiter avec les outils classiques. Cette définition n'impose pas de distinction réelle entre des données non massives et des données massives. En effet, cette limite est variable en fonction de l'infrastructure disponible, de l'évolution du matériel et des algorithmes de traitement, *etc.* .

1. "There were 5 Exabytes of information created between the dawn of civilization through 2003, but that much information is now created every 2 days."

1.1 Problème du stockage et de l'accès

Généralement, lorsqu'on aborde les problématiques de traitement de données massives, on se confronte rapidement à ce qui est communément appelé le problème des *3V* : Volume, Variété et Vélocité. Le *Volume*, de façon relativement simple, concerne la masse de données qu'il faut gérer. La *Variété* des données est souvent la cause de ce problème de volume : les données ont différentes sources induisant une grande hétérogénéité dans l'information récoltée. Enfin, la *Vélocité* définit la vitesse de génération ou d'acquisition des données, les évolutions technologiques permettant de générer de plus en plus de données à des fréquences de plus en plus rapides. Il est fréquent d'associer un quatrième V aux trois précédents : La véracité. En effet, l'hétérogénéité des sources de données induit généralement une certaine imprécision voire une baisse de la qualité des données qu'il faut prendre en compte lors du traitement et de l'analyse.

Afin de répondre à ces problématiques, deux approches sont généralement envisagées : l'utilisation de *super-ordinateurs* ou l'utilisation d'une infrastructure *cloud*. Historiquement, les premiers tiennent une grande place, surtout en Physiques, ou les besoins en capacité de calculs et de stockage pour les simulations (météorologie, physiques élémentaires ou simulation de dynamique des fluides par exemple) sont très consommateurs de ressources. Cependant, l'intérêt pour l'analyse de données massives issues d'internet (site de vente, parcours de site web, suivi d'utilisateurs, *etc.*) et l'éveil à l'intérêt économique que peuvent représenter de telles analyses associées à l'amélioration des matériels et techniques ont fait émerger la seconde. En effet, bien que généralement considérée comme moins performante que les super-ordinateurs [117], une infrastructure *cloud* comprenant des unités de stockage et de calculs distribués est une solution performante et bien plus facile d'accès pour un coût plus abordable.

1.2 Rôle de la visualisation d'information

L'intérêt de la visualisation d'information en tant que vecteur de transmission d'information et en tant qu'outil d'analyse et d'exploration de données n'est plus à démontrer. En effet, les origines de la représentation visuelle pour véhiculer une information remonte aux sumériens [14] et est même rentrée dans les adages : qui n'a jamais entendu le proverbe " *Une image vaut mille mots* " ? Cela reste vrai pour les données massives ou une telle quantité d'information est difficile à analyser en l'état. La visualisation n'est évidemment pas la seule approche envisageable pour étudier des données, les statistiques tiennent également un rôle prépondérant dans ce domaine et les deux approches sont généralement combinées. La visualisation, cependant, tient une place prépondérante dans le système cognitif humain et représente un réel outil d'interprétation et de

compréhension pour l'Homme [138]. Les problèmes soulevés par les données massives, introduit précédemment, se répercutent cependant sur les systèmes et techniques de visualisation. En effet, ceux-ci ne sont que rarement adaptés pour la représentation ou l'exploration de jeux de données tellement grands qu'on ne peut pas envisager de les stocker sur une simple machine, ni même les traiter avec des approches traditionnelles. La transition entre les deux méthodes, pour données massives ou non, doit se faire au sein même du *Pipeline* de visualisation décrit par Santos et Brodlie [122] (voir figure 3). Dans ce processus, la première étape consiste en l'analyse des données, résultant en la création d'une abstraction des données. Une fois les données abstraites, elles sont filtrées afin de ne conserver que les données d'intérêt. Ensuite vient la création des données géométriques, c'est-à-dire la transformation des données en une structure que l'on pourra utiliser pour générer la représentation. Finalement se fait la création de l'image qui sera *rendue* pour être observée et analysée par l'utilisateur.

La distance entre le client de visualisation et le lieu de stockage et de traitement des données (le serveur) implique le remaniement des étapes de visualisation, accentuant ainsi la séparation en deux composantes : d'un coté le stockage et le traitement des données, et de l'autre la représentation et l'interaction utilisateur (voir figure 4). Une telle distanciation a pourtant un coût, celui du transfert des données d'une composante à l'autre, impliquant des délais lors d'interactions ou de chargements dont on ne peut s'affranchir. L'objectif devient ainsi de réduire au maximum ces délais et cela ne peut se faire que par la mise en place d'infrastructures logicielles et de processus de traitement des données adaptés.

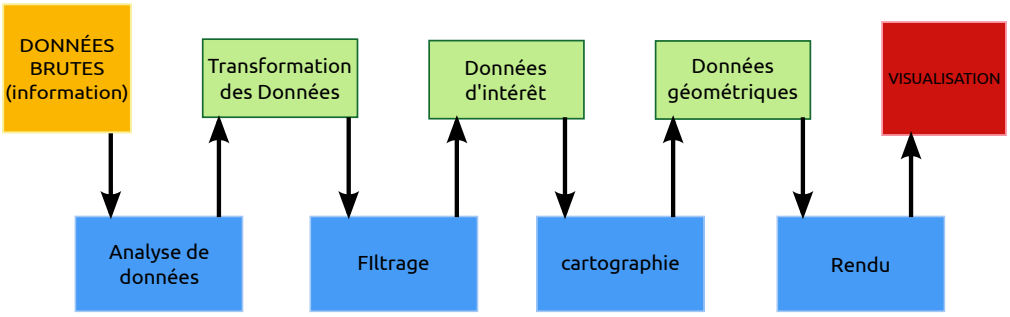


Figure 3: Adaptation du Processus de visualisation de données décrit par Santos et Brodlie [122].

2 Organisation du manuscrit

Tout d'abord nous fournirons la terminologie et les définitions essentielles utilisées dans cette thèse. Certaines notions étant peu utilisées dans ce manuscrit,

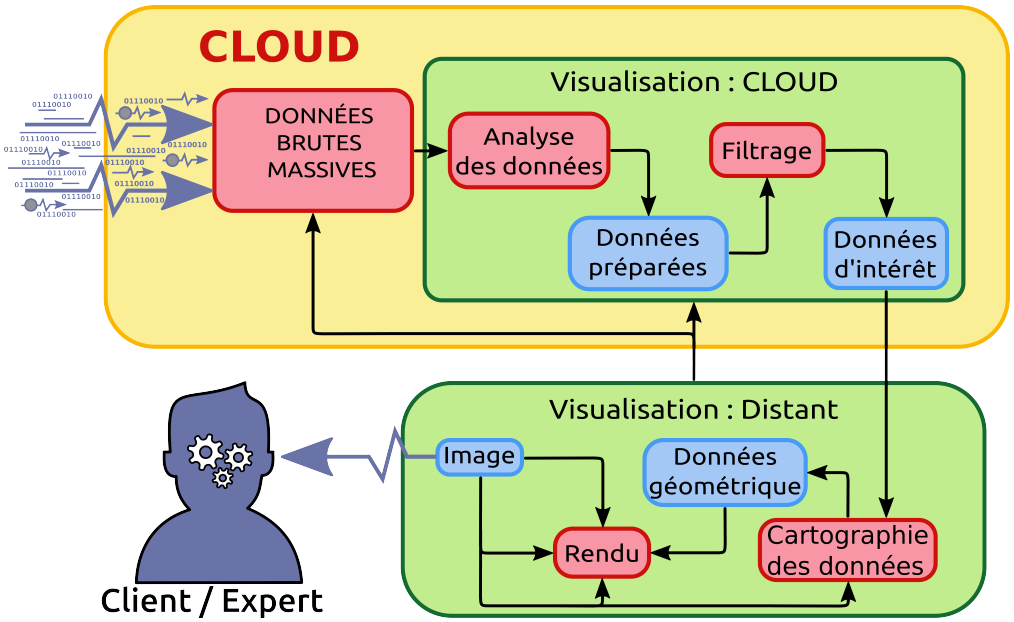


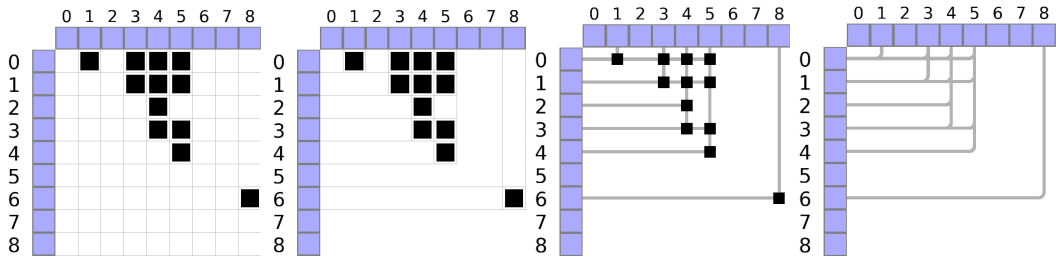
FIGURE 4: Processus de visualisation de grandes masses de données

nous ne les définirons qu'au moment de leur utilisation dans les chapitres concernés.

Le chapitre I est un état de l'art des différents domaines auxquels sont rattachés les travaux décrits dans ce manuscrit. Nous présenterons ainsi les travaux majeurs concernant la représentation de données relationnelles puis multi-variées, l'exploration multi-échelle et la représentation de données massives. Afin que chaque chapitre puisse être lu indépendamment, les introductions présenteront également l'état de l'art des différents domaines concernés par le chapitre.

Le chapitre II porte sur l'encodage visuel des arêtes, et plus particulièrement sur les effets que peut avoir l'encodage visuel utilisé pour représenter les arêtes sur la visualisation de données relationnelles. Il existe plusieurs techniques de représentation de données relationnelles décrites dans la littérature[62], mais deux sont particulièrement connues et utilisées : le diagramme nœuds-liens et la matrice d'adjacence. Quelques travaux se sont déjà intéressés à comparer ces deux techniques, mais tous souffrent du même biais : le positionnement et la représentation des éléments. En effet, ces travaux n'évaluent pas les différences entre ces deux représentations uniquement, ils évaluent en même temps l'algorithme de positionnement ou d'ordonnancement des éléments utilisés durant l'évaluation. Pour pallier cela, nous avons mis en place quatre encodages visuels des arêtes permettant une transformation progressive depuis un encodage matriciel vers un encodage lien des connexions entre éléments sans devoir modifier le positionnement ou l'ordre des nœuds (voir figure 5). Nous présentons deux évaluations utilisateurs visant à déterminer l'influence de l'encodage visuel

des relations pour des tâches simples et une tâche plus complexe. Les résultats obtenus avec ces deux évaluations utilisateurs ne permettent pas de tirer de conclusions franches sur la supériorité statistiques d'un encodage visuel par rapport à un autre. En revanche, même s'il ne semble pas y avoir de différences statistiquement notables, les retours utilisateurs indiquent qu'une simplification raisonnable de l'encodage visuel des arêtes permet d'augmenter le confort de lecture des matrices d'adjacences.



(a) encodage 1 **(b) encodage** 2 **(c) encodage** 3 **(d) encodage** 4

FIGURE 5: Transformation progressive de l'encodage visuel des arêtes depuis un encodage matriciel jusqu'à un encodage nœuds-lien.

Dans le chapitre III nous nous appuyons sur les conclusions du chapitre précédent pour la création d'une technique de représentation de données relationnelles pondérées et hiérarchisées appelée *Adjasankey* (voir figure 6). L'état de l'art ne nous a pas permis d'identifier de technique de représentation de données relationnelles permettant de clairement mettre en avant à la fois les relations entre éléments et leurs importances tout en permettant l'exploration interactive multi-échelle des données. Pour répondre à ces critères, nous nous sommes basés sur la représentation matricielle, permettant de séparer les éléments sources et destinations des relations, ainsi que sur l'un des encodages visuels mis en place dans le chapitre II adapté à ces besoins. Nous nous appuyons également sur l'abstraction de données et le partitionnement hiérarchique des données pour en permettre l'exploration interactive multi-échelle. Bien que cette nouvelle technique de représentation ait été pensée et créée pour la représentation de données massives, elle reste utilisable avec des données de taille classique. Nous présentons l'efficacité de notre représentation avec un cas d'étude utilisant des données réelles : des parcours utilisateurs sur un site internet.

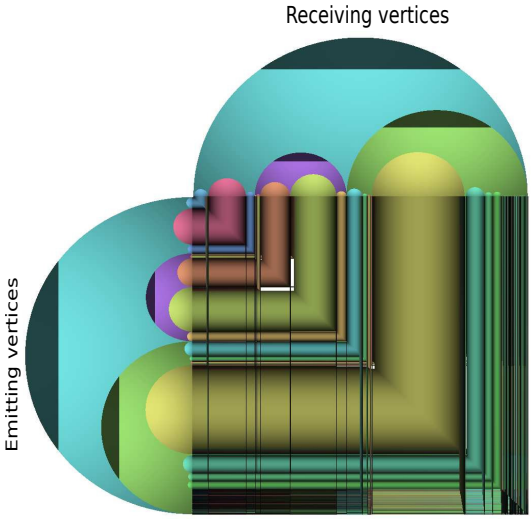


FIGURE 6: Adjasankey permet la représentation de données relationnelles en mettant clairement en avant l'importance des relations tout en permettant l'exploration multi-échelle des données.

Le chapitre IV porte sur la représentation de données multi-variées massives. Pour cela, nous nous appuyons sur la technique de coordonnées parallèles présentée par Inselberg [71, 72]. On peut d'ores et déjà trouver de nombreux travaux dans la littérature présentant différentes versions et amélioration des coordonnées parallèles. Cependant, peu ont été présentées dans un objectif d'utilisation avec des données massives. L'objectif de ce chapitre porte donc sur la description d'une technique de visualisation de données abstraites avec des coordonnées parallèles permettant l'interaction sans délais majeurs. Pour cela, nous nous appuyons sur une architecture en deux composantes, le client de visualisation et le serveur de stockage et calculs. Ce chapitre décrit le client de visualisation ainsi que le processus d'abstraction des données que nous avons mis en place pour permettre non seulement la représentation mais l'interaction sur des données massives (voir figure 7).

Le chapitre V présente finalement les approches utilisées pour la représentation de données massives avec les outils de visualisation décrits dans les chapitres III et IV. Pour cela nous nous appuyons sur l'infrastructure HADOOP dédiée aux calculs et stockage de données massives. Nous présentons ainsi pour Adjasankey une technique basée sur le pré-calcul et le stockage des requêtes utilisateurs envisageable grâce à sa structure de données très compacte. Nous voyons que pour les coordonnées parallèles, cette solution est également possible mais avec un coût computationnel et un coût en stockage non négligeable. Nous présentons également une autre approche, basée sur le calcul à la demande, fournissant des performances moindre que l'approche stockage mais avec des coûts de stockage et de pré-calculs réduits.

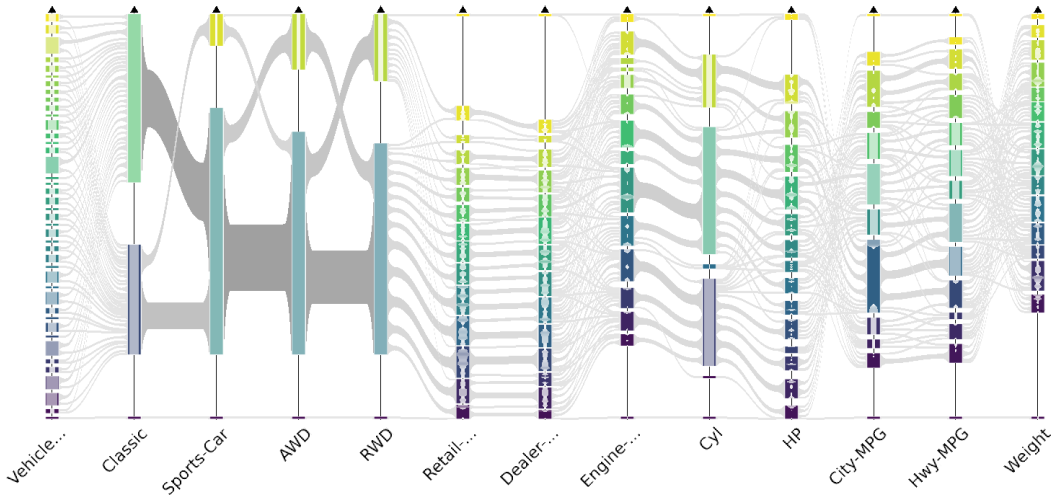


Figure 7: Coordonnées parallèles pour la représentation de données massives par abstraction de données.

Enfin, nous donnons en dernière partie les conclusions de nos travaux de recherche ainsi que les perspectives de recherche qu'ils offrent.

3 Publications

Certains des travaux que nous présentons ici ont été publiés. Une phase préliminaire de l'évaluation utilisateur de l'encodage visuel des arêtes a été présentée à l'*International conference on Information Visualisation* et a été publiée dans les actes de la conférence [120]. Durant cette même conférence, nous avons également présenté nos travaux sur *Adjasankey* et l'utilisation d'une approche stockage pour la visualisation de données massives, eux aussi publiés dans les actes [121]. Ces derniers travaux ont par ailleurs été récompensés par le prix du meilleur article de la conférence (*Best paper Award*). Enfin, les travaux concernant les coordonnées parallèles ont été soumis au journal *MDPI informatics*.

Définitions et Terminologie

Ensembles

Définition 1 (Décomposition d'ensemble)
Soient E un ensemble et $P = \{E_i\}_{1 \leq i \leq p}$ un ensemble d'ensembles E_i tels que $\forall i \in [1..p]$, $E_i \subseteq E$. P est une décomposition de E si et seulement si

$$\bigcup_{i=1}^{p} E_i = E$$

Définition 2 (Partition d'ensemble)
Soient E un ensemble et $P = \{E_i\}_{1 \leq i \leq p}$ une décomposition de E. P est une partition de E si et seulement si $\forall i \in [1..p]$, $j \in [1..p]$, $i \neq j$, on a $E_i \cap E_j = \emptyset$.

Définition 3 (Paire d'éléments)
Soit E un ensemble. On appelle paire d'éléments de E tout ensemble non-ordonné de deux éléments $\{u, v\}$ tel que $\{u, v\} \subseteq E$.

Définition 4 (Couple d'éléments)
Soient E un ensemble et u et v deux éléments de E. On appelle couple des éléments u et v, la donnée de u et v dans un ordre déterminé, noté (u, v).

Graphes

Définition 5 (Graphe non-orienté)
Soient V et E deux ensembles, tels que $E \subseteq \{\{u, v\} | u \in V, v \in V\}$. On appelle graphe non orienté, noté $G = (V, E)$, la structure $< V, E >$. Les éléments de V (resp. de E) sont appelés des sommets (resp. des arêtes). Soit $e = \{u, v\}$ une arête, les sommets u et v sont appelés les extrémités de l'arête e.

Définition 6 (Graphe orienté)
Soient V et A deux ensembles, tels que $A \subseteq \{(u, v) | u \in V, v \in V\}$. On appelle graphe orienté, noté $G = (V, A)$, la structure $< V, A >$. Les éléments de V (resp. de A) sont appelés des sommets (resp. des arcs). Soient $a = (u, v)$ un arc, on appelle le sommet u (resp. v) la source (resp. la destination) de l'arc a.

Remarque : On appelle *boucle* une arête $\{u, u\}$ qui relie un sommet à lui-même.

Remarque : Les définitions suivantes données pour un graphe $G = (V, E)$ concernent tout types de graphes, qu'ils soient orientés ou non.

Définition 7 (Ordre d'un graphe)
Soit $G = (V, E)$ un graphe. On appelle ordre du graphe $Ordre_G$, le nombre de sommets $|V|$ dans le graphe G.

Définition 8 (Taille d'un graphe)
Soit $G = (V, E)$ un graphe. On appelle taille de graphe $Taille_G$, le nombre d'arêtes $|E|$ dans le graphe G.

Définition 9 (Sous-graphe)
Soient $G = (V, E)$ un graphe. On appelle $G' = (V', E')$ un sous-graphe de G si et seulement si les trois conditions suivantes sont vérifiées :

— G' est un graphe,

— $V' \subseteq V$ et

— $E' \subseteq E$.

Définition 10 (Sous-graphe induit)
Soit $G = (V, E)$ un graphe. On appelle $G' = (V', E')$ sous-graphe induit de G par V' si et seulement si les conditions suivantes sont vérifiées :

— G' est un sous-graphe de G et

— $E' = \{\{u, v\} | \{u, v\} \in E, u \in V' \text{ et } v \in V'\}$.

Le sous-graphe G' de G induit par V' est noté $G[V']$.

Remarque : Pour tout ensemble $V' \subseteq V$, on peut construire le sous-graphe induit G' de G par V'.

Définition 11 (Voisinage d'un sommet)
Soient $G = (V, E)$ un graphe et $u \in V$. On appelle voisinage de u dans G, noté $N_G(u)$ l'ensemble $\{v | \{u, v\} \in E\}$.

Remarque : Il est clair que $N_G(u) \subseteq V$.

Définition 12 (Adjacence d'un sommet)
Soient $G = (V, E)$ un graphe et $u \in V$. On appelle adjacence de u dans G, noté $adj_G(u)$ l'ensemble $\{\{u, v\}|\{u, v\} \in E\}$.

Définition 13 (Adjacence entrante et sortante d'un sommet)
Soient $G = (V, A)$ un graphe orienté et $u \in V$. On appelle adjacence entrante (resp. sortante) de u dans G, noté $adj^-_G(u)$ (resp. $adj^+_G(u)$), l'ensemble $\{(v, u)|(v, u) \in A\}$ (resp. $\{(u, v)|(u, v) \in A\}$).

Définition 14 (Degré d'un sommet)
Soit $G = (V, E)$ un graphe et $u \in V$. On appelle degré de u dans G noté $deg_G(u)$ la quantité $|adj_G(u)|$.

Définition 15 (Degré entrant et sortant d'un sommet)
Soient $G = (V, A)$ un graphe orienté et $u \in V$. On définit les degrés entrant et sortant de u dans G, respectivement notés $deg^-_G(u)$ et $deg^+_G(u)$ comme suit :

$$deg^-_G(u) = |adj^-_G(u)| \text{ et } deg^+_G(u) = |adj^+_G(u)|$$

Remarque : Le degré d'un sommet u dans un graphe orienté $G = (V, A)$ est $deg_G(u) = deg^-_G(u) + deg^+_G(u)$.

Définition 16 (chemin non-orienté)
Soit $G = (V, E)$ un graphe non-orienté. On appelle chemin non-orienté (ou chemin) dans G, une séquence $(v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k)$ avec :

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

Remarque : Un chemin ne passe pas deux fois par le même sommet, le même arc ou la même arête.

Définition 17 (pondération)
Soient $G = (V, E)$ un graphe et K un ensemble. On appelle pondération des sommets (resp. des arcs et arêtes) du graphe toute application $f : V \to K$ (resp. $f : E \to K$). On dit alors que G est sommet-pondéré (resp. arête-pondéré) et on le note $G = (V, E, f)$.

Remarque : Si G est un graphe orienté, on parle alors de graphe orienté arc-pondéré.

Définition 18 (longueur pondérée d'un chemin)
Soient $G = (V, E, w)$ un graphe arc-arête-pondéré avec $u \in V$, $v \in V$,la fonction de pondération w et $p = (u = v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k = v)$ un chemin de u à v composé de k sommets (et $k - 1$ arêtes). La longueur pondérée du chemin est $\sum_{i=1} k - 1w(e_i)$.

Remarque : La longueur (non pondérée) du chemin p est le nombre d'arcs et arêtes qu'il contient.

Remarque : On définit de manière analogue les longueurs (pondérées ou non) des chemins orientés.

Définition 19 (Distance dans un graphe)
Soient $G = (V, E)$ un graphe, $u \in V$ et $v \in V$. On appelle distance dans G entre u et v, notée $dist_G(u, v)$, la longueur du plus court chemin dans G de u à v.

Définition 20 (Distance pondérée dans un graphe)
Soient $G = (V, E, w)$ un graphe arête-pondéré, $u \in V$ et $v \in V$ et w la fonction de pondération. Soit $P = p_1, p_2, \ldots, p_n$ l'ensemble de tous les chemins dans G de u à v. On appelle distance pondérée dans G entre u et v, notée $dist_{G,w}(u, v)$, la plus petite longueur pondérée des chemins de P.

Définition 21 (Graphe connexe)
Soit $G = (V, E)$ un graphe non-orienté. Le graphe G est connexe s'il existe un chemin (non-orienté) entre toute paire de sommets dans G.

Définition 22 (Graphe couvrant)
Soient $G = (V, E)$ et $H = (V_H, E_H)$ deux graphe. H est un graphe couvrant de G si et seulement si $V_H = V$, $E_H \subseteq E$.

Définition 23 (Graphe complet)
Soit $G = (V, E)$ un graphe. On dit que le graphe G est un graphe complet si et seulement si $\forall u \in V$ et $\forall v \in V$, $\{u, v\} \in E$.

Remarque : Un graphe complet à n sommets, noté K_n, possède $\frac{n(n-1)}{2}$ arêtes.

Définition 24 (Densité d'un graphe)
Soit $G = (V, E)$ un graphe. On appelle densité du graphe G la mesure de comparaison de la taille $|E(G)|$ avec celle d'un graphe complet de même ordre. Cette mesure est déterminée comme suit :

$$dens_G = \frac{2 \cdot |E|}{|V|(|V| - 1)}$$

Définition 25 (Arbre)
On dit que T est un arbre si et seulement si T est un graphe acyclique et connexe.

Remarque : De manière à éviter de possibles ambiguïtés, les sommets d'un arbre seront appelés *nœuds* si T est clairement défini comme étant un arbre. Les nœuds $u \in V$ tels que $d_T(v) = 1$ sont appelés feuilles de l'arbre. L'ensemble des feuilles est noté $\mathcal{F}_T$. Si un nœud u de T n'est pas une feuille on dit que u est interne à T.

Définition 26 (Arbre enraciné)
Soit $G = (V, A)$ un graphe orienté. On dit que G est un arbre enraciné si et seulement si :

— G est un graphe acyclique dirigé,

— $|A| = |V| - 1$

— $\exists! r \in V, deg_G^-(r) = 0$ et

— $\forall v \in V \backslash r, deg_G^-(v) = 1$.

Remarque : Le sommet r est appelé racine de l'arbre.

Définition 27 (Sous-Arbre)
Soit $T = (V, E)$ un arbre, tout sous-graphe connexe de T est appelé un sous-arbre de T.

Remarque : Pour un graphe enraciné T, le sous-arbre enraciné en $u \in V$ est noté T_u et correspond au sous-graphe induit formé par u et l'ensemble des nœuds $v \in V$ accessibles à partir de u.

Définition 28 (Profondeur d'un sommet)
Soient $G = (V, A)$ un arbre enraciné de racine r et $u \in V$. On définit la profondeur (aussi appelée hauteur) de u dans G, notée $prof_G(u)$, comme suit :

$$prof_G(u) = dist_G(r, u)$$

Définition 29 (Niveau)
Soit $T = (V, E)$ un arbre enraciné en r, le i-ème niveau de G noté $N_i(T)$ est l'ensemble de feuilles dans le sous-arbre induit par l'ensemble $\{u \in V, h_G(u) \leq i\}$.

Remarque : On appelle *hauteur* de l'arbre le nombre de niveau existant dans l'arbre, soit le nombre d'arcs sur un chemin de longueur maximale.

Définition 30 (Graphe biparti)
Soit $G = (V, E)$ un graphe. On dit que G est un graphe biparti si il existe V_1 et V_2 tels que $V_1 \subset V$ et $V_2 \subset V$, $V_1 \cap V_2 = \emptyset$, $V_1 \cup V_2 = V$ et V_1 et V_2 sont des stables.

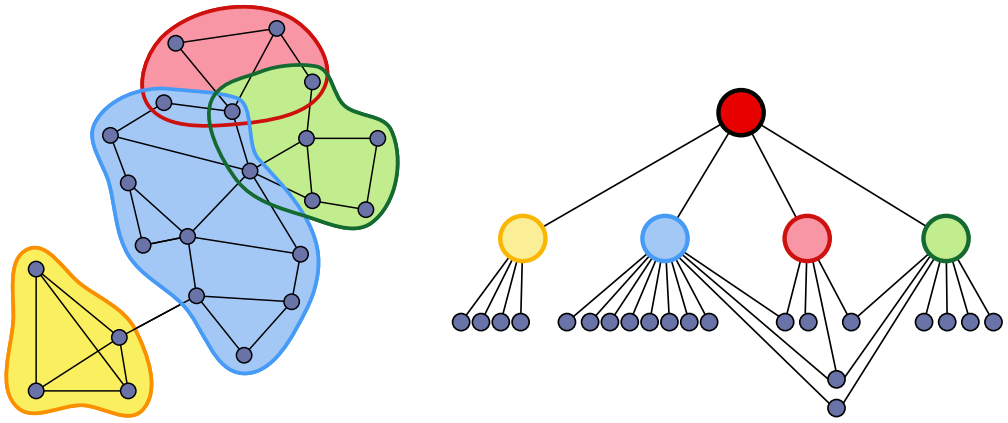
Définition 31 (Graphe décomposé)
Soient $G = (V, E)$ un graphe et $H = (V_H, A_H)$ un graphe orienté acyclique tel que $\mathcal{F}(H) = V$. Le graphe décomposé (G, H) est défini comme suit : tout sommet $u \in V_H \setminus \mathcal{F}(H)$ représente un groupe C_u de sommets de G tel que $C_u = \{v \in V | deg_H^+(v) = 0$ et il existe un chemin de u à v dans $H\}$ (voir figure 8).

Remarque : On appelle niveau i de (G, H), l'ensemble des sommets u de H tels que :

$$prof_H(u) = i.$$

Définition 32 (Graphe partitionné)
Soit $G' = (G, H)$ un graphe décomposé, G' est un graphe partitionné si H est un arbre enraciné.

Remarque : Le graphe H est appelé arbre de partition.



(a) graphe H **(b) DAG de décomposition**

FIGURE 8: Décomposition de graphe

Définition 33 (Arête inter-groupes / intra-groupe)
Soient $G = (V, E)$ un graphe et $C = \{C_1, C_2, \ldots, C_k\}$ une décomposition des sommets de G. On dit que $e = \{u, v\} \in E$ est une arête inter-groupes (resp. intra-groupe) s'il existe C_i et C_j tels que $i \neq j, u \in C_i$ et $v \in C_j$ (resp. s'il existe C_i tel que $u, v \in C_i$).

Remarque : On note $E(C_i, C_j)$ l'ensemble des arêtes reliant un sommet de C_i et un sommet de C_j.

Définition 34 (Graphe quotient)
Soient (G, H) un graphe décomposé, $\{u_1, \ldots, u_p\}$ le niveau i de (G, H) et $C = \{C_1, C_2, \ldots, C_p\}$ l'ensemble des groupes correspondant dans G.
Le graphe quotient $Q_i = \{V_Q, E_Q\}$ du niveau i de (G, H) est défini comme suit :

- $V_Q = \{u_1, u_2, \ldots, u_p\}$,
- $e = \{u_j, u_k\} \in E_Q$, si et seulement si $j \neq k$ et $\exists u \in C_j, v \in C_k$ tels que $\{u, v\} \in E$

Remarque : Un sommet (resp. arête et arc) de Q_G est appelé *méta-nœud* (resp. *méta-arête* et *méta-arc*).

Définition 35 (Graphe quotient pondéré)
Soient $Q = (V_Q, E_Q, w)$ un graphe arête-pondéré. On dit que Q est un graphe quotient pondéré du niveau i d'un graphe arête-pondéré décomposé (G, H) si le graphe $Q' = (V_Q, E_Q)$ est le graphe quotient du niveau i de (G, H).

Remarque : Pour pondérer une méta-arête ou un méta-arc, il existe plusieurs méthodes, notamment la valeur minimale, maximale, médiane ou encore moyenne des arêtes ou arcs qu'il ou elle représente.

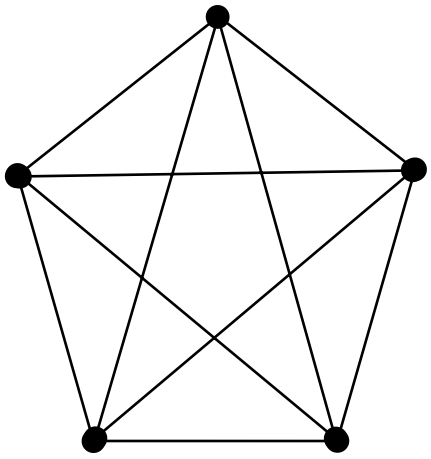
Représentation de graphe

Les notions données dans cette section sont définies pour les graphes non orientées mais peuvent être trivialement adaptées aux graphes orientés.

Définition 36 (Dessin de graphe)
Un dessin de graphe est une représentation graphique d'un graphe G dans un espace à 2 ou 3 dimensions.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

(a) Matrice d'adjacence



(b) Diagramme nœuds-liens

Figure 9: Exemple de représentation de graphe utilisant : en figure 9a, une matrice d'adjacence et en figure 9b un diagramme nœuds-liens.

Définition 37 (Matrice d'adjacence)
Soit $G = (V, E)$ un graphe, $u \in V$, $v \in V$ où $n = |V|$. La matrice d'adjacence A de G est une représentation graphique d'un graphe sous la forme d'une matrice booléenne $(n \times n)$ avec pour chaque couple (u, v) :

$$A_{uv} = \begin{cases} 1 \ si \ (u,v) \in E \\ 0 \ sinon \end{cases}$$

Remarque : La matrice d'adjacence A d'un graphe non-orienté est symétrique puisque $A_{u,v} = A_{v,u}$.

Remarque : Dans ce manuscrit, la matrice d'adjacence d'un graphe orienté sera représentée avec les sommets sources en abscisse et les sommets destinations en ordonnée.

Définition 38 (Matrice d'adjacence pondérée)
Soit $G = (V, E, p)$ un graphe arête-pondéré, $u \in V$, $v \in V$ où $n = |V|$ et w la fonction de pondération . La matrice d'adjacence pondéré A de G est une matrice $(n \times n)$ avec pour chaque couple (u, v) :

$$A_{uv} = \begin{cases} p(u,v) \ si \ (u,v) \in E \\ \emptyset \ sinon \end{cases}$$

Définition 39 (Diagramme Noeuds-liens)
On appelle diagramme noeud-lien la représentation visuelle d'un graphe G pour lequel les sommets sont représentés par des formes (points par exemple) et les arêtes par des lignes ou polylignes reliant deux sommets.

Remarque : Dans le cas d'un graphe orienté, des flèches peuvent être ajoutées à l'extrémité des arcs pour montrer l'orientation

Définition 40 (Dessin en lignes brisées)
Soit $G = (V, E)$ un graphe. On appelle *dessin en ligne brisée de* G, un dessin de G dans lequel chaque arête de G est représentée par une ligne brisée (appelée poly-ligne), *i.e.* une séquence de segments contigus.

Définition 41 (Représentation hybride)
On appelle méthode hybride une technique de représentation d'un graphe qui combine tout ou partie de plusieurs techniques de représentation pré-existante.

Définition 42 (Dessin orthogonal)
Soit $G = (V, E)$ un graphe. On appelle *dessin orthogonal de* G, un dessin en lignes brisées de G dont les angles formés par deux segments adjacents (de deux arêtes) ou par deux segments contigus d'une ligne brisée (d'une même arête) sont orthogonaux ou sont plats.

Définition 43 (Point de contrôle)
Dans un dessin en ligne brisée, on appelle *point de contrôle* d'une arête, chaque "brisure" de la ligne brisée représentant l'arête, *i.e.* l'extrémité commune de chaque paire de segments contigus.

Autres définitions

Définition 44 (Complexité d'un algorithme)
La complexité d'un algorithme est la quantité d'espace mémoire utilisée (complexité en espace) et/ou le nombre d'opérations élémentaires réalisées par l'algorithme (complexité en temps) en fonction de la taille des données en entrée de l'algorithme.

Définition 45 (Problème d'arrangement linéaire minimum)
Le problème d'arrangement linéaire minimum (PALM) est définit dans [105] par : Étant donné un graphe $G = (V, E)$ avec $|V| = n$, un *positionnement* est une fonction définit par $\varphi : V \to 1 \dots n$. Le problème d'arrangement linéaire minimum (PALM) est un problème d'optimisation combinatoire définit par : Trouver pour un graphe $G = (V, E)$ un positionnement φ qui minimise l'arrangement linéaire

$$LA(G, \varphi) = \sum_{uv \in E} |\varphi(u) - \varphi(v)|$$

.

Chapitre I

État de l'art

Dans ce chapitre nous allons présenter les quatre domaines majeurs que nous retrouverons tout au long de ce manuscrit : la visualisation de données relationnelles, la visualisation de données multi-dimensionnelles, la visualisation de données abstraites et la visualisation de données massives. En section 1 nous présenterons des exemples de techniques et outils permettant la représentation de données relationnelles, caractérisées par des éléments et des relations entre ces éléments et généralement modélisées par des graphes. La section 2 introduira ensuite les techniques de visualisation et d'interaction pour données multi-dimensionnelles, notamment avec les différentes approches que l'on peut retrouver dans la littérature : orientées points et orientées lignes. Nous verrons tout au long de ce manuscrit que l'un des problèmes majeurs rencontrés en visualisation de données consiste à représenter efficacement plus d'informations que ne le permet un écran d'ordinateur ou la perception humaine. Nous présenterons ainsi en section 3 les techniques d'exploration multi-échelles qui tentent de répondre à ce problème. Finalement, nous introduirons en section 4 le problème des données massives et ses implications sur la visualisation de données.

1 Visualisation de données relationnelles : Généralités

Lorsqu'on considère la visualisation d'éléments et de relations entre éléments, l'un des modèles les plus utilisés est le modèle graphe qui représente les éléments par des sommets et les relations par des arêtes reliant ces sommets. Les graphes sont ainsi utilisés dans de nombreux domaines, et en particulier ceux en lien avec l'analyse de réseaux. La sociologie avec les réseaux sociaux, la biologie avec les réseaux métaboliques, les télécoms avec les réseaux de communications, en sont quelques exemples.

1.1 Visualisation de graphe : cas général

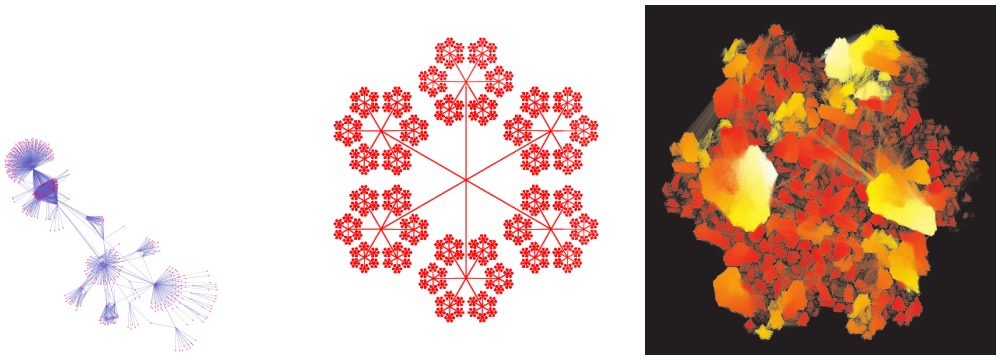
Lorsqu'on considère la représentation de graphes, deux représentations sont généralement utilisées : les *Diagrammes Nœuds-Liens* (que nous appellerons DNL(s)), ou les *Diagrammes Orientés Matrices* (DOM(s)). Nous présenterons ici différents travaux s'appuyant sur ces techniques, mais également d'autres techniques. Parmi ces autres techniques, certaines sont dites hybrides et visent à combiner deux techniques différentes pour bénéficier de leurs avantages. Finalement nous présenterons les résultats de travaux visant à déterminer dans la mesure du possible quelle technique de représentation est la plus adaptée en fonction de tâches courantes en analyse de graphes.

Diagrammes nœuds-liens

Dans un diagramme nœuds-liens, une entité est représentée par une forme géométrique que l'on appelle *glyphe*. Les relations entre les entités sont quant à elles représentées par des lignes droites ou brisées, appelées *liens*. On parle ainsi de ligne polygonale, généralement simplifiée en polyligne.

L'amélioration d'un DNL se passe essentiellement par l'amélioration des techniques de dessin automatiques et notamment l'optimisation de critères esthétiques afin de rendre les informations de relations plus faciles à lire, comprendre et utiliser. Dans leurs travaux, Purchase [107], Purchase *et al.* [110], Ware *et al.* [139] mènent ainsi plusieurs études expérimentales pour tenter de déterminer quels sont les critères esthétiques les plus importants. Différents critères sont ainsi évalués : la symétrie de la représentation, la minimisation du nombre de croisements d'arêtes et du nombre de brisures, l'utilisation d'une grille pour le positionnement des nœuds et le routage orthogonal des arêtes, la minimisation du nombre d'angles aigus dans les brisures de polylignes, *etc.* . Ces études ont ainsi montré que la réduction du nombre de croisements d'arêtes et du nombre de segments des polylignes tient un rôle majeur dans la lisibilité des DNLs. Différents travaux se sont ainsi attachés à l'optimisation de ces différents critères et cela avec différentes approches. L'amélioration de ces critères pose cependant le problème de la complexité algorithmique sous-jacente. En effet, la réduction du nombre de croisements d'arêtes par exemple a été prouvée comme étant un problème NP-complet par Garey et Johnson [49]. Afin de simplifier le traitement de ce problème, des heuristiques ont été décrites dans la littérature, avec des approches différentes (voir figure I.1). Les algorithmes par modèles de contraintes tentent de trouver un positionnement des nœuds permettant de minimiser une ou plusieurs fonctions [31, 32]. Le positionnement par modèle de forces [44, 53, 88] en est une version où la fonction à minimiser représente l'énergie d'un modèle physique. Les décompositions topologiques de graphes [9, 127] sont quant à elles basées sur la détection de sous-structures dans le graphe tandis que les algorithmes de remplissage de l'espace [97] tentent de répartir les éléments du graphe afin d'occuper au maximum l'espace libre dans

la représentation.



(a) (b) (c)

Figure I.1: Exemples de représentation de graphes par DNL en utilisant différents algorithmes de positionnement : (a) positionnement par algorithme de modèle de forces, ici *FM3*[53], (b) positionnement par l'algorithme de décomposition topologique *Topolayout*[9] et (c) positionnement avec remplissage de l'espace par l'algorithme *Gospermap*[97].

Diagrammes Orientés Matrices

Un Diagramme orienté matrice correspond à un tableau dans lequel un élément est représenté par une ligne et une colonne du tableau. Dans ce diagramme, l'existence d'une relation entre deux éléments est représentée par un glyphe positionné à l'intersection des lignes et colonnes des deux entités considérées. Afin de faciliter l'identification des éléments et des structures du graphe (des *cliques* par exemples), l'ordre des nœuds utilisés est le même en abscisse et en ordonnée. De cette manière, les *cases* présentes sur la diagonale de la matrice représentent les *boucles*. Les matrices étant carrées et les éléments qui composent le graphe étant représentés deux fois, les relations entre deux éléments du graphe peuvent aussi être dupliquées. Dans le cas où les auteurs ne veulent les représenter qu'une fois, ils peuvent alors choisir de placer les arêtes dans l'une ou l'autre des deux demi-matrices. Cette technique est par ailleurs utilisée pour représenter les graphes orientés. Dans ce cas, l'attribution d'un axe aux sources des arcs et de l'autre à leurs destinations permet d'identifier l'orientation des relations. Une première description des matrices comme outils d'analyse visuelle des données est donnée par Bertin [20]. Il y décrit ainsi les matrices comme outils permettant de donner à voir la structure de l'information. Ces travaux ont récemment été modernisés avec *Bertifier* [103], une version numérique des matrices de Bertin (voir figure I.2).
D'autres travaux se sont attachés à l'optimisation de la représentation visuelle des DOM et la plupart considère l'amélioration de la représentation dans l'ob-

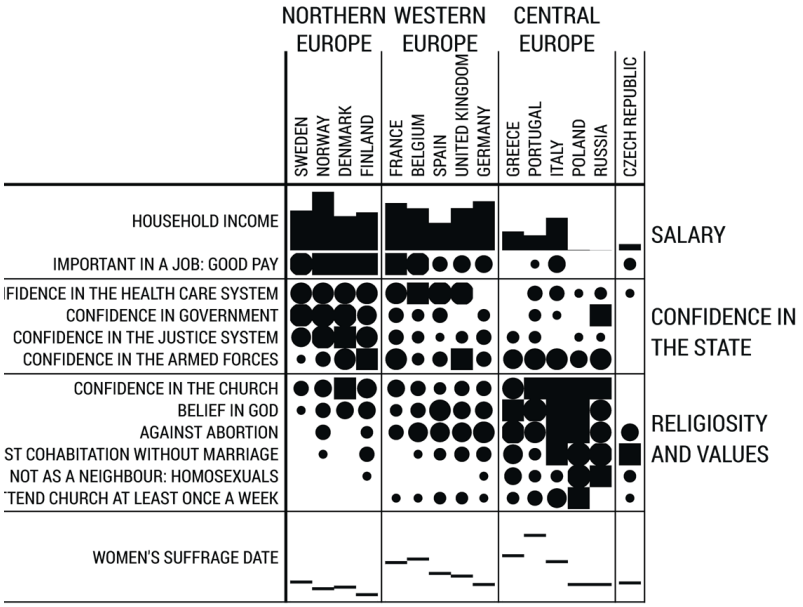


Figure I.2: Une version numérique des matrices de Bertin, bertifier [103], permet de manipuler les données afin d'explorer les données et de mettre en évidence des schémas et ou relations.

jectif de faciliter une interaction ou une analyse spécifique. Nous en présenterons quelques exemples en sections 1.2 et 1.3. Même si l'amélioration des DOM peut se faire de façon esthétique la plupart des travaux se sont essentiellement intéressés à l'optimisation de l'ordre des éléments pour faciliter la détection de certains motifs (voir figure I.3). Cependant, ici aussi se pose le problème de complexité algorithmique. En effet, l'optimisation de l'ordre des nœuds correspond au problème d'arrangement linéaire minimum, qui consiste à trouver un alignement des nœuds du graphe minimisant la distance entre les éléments connectés (voir définition 45) et qui est NP-difficile [105]. Par conséquent, la communauté s'attache, là aussi, à la création de méthodes heuristiques d'ordonnancement des nœuds. Là encore plusieurs approches sont décrites dans la littérature. La détection de minimum locaux par exemple [82, 115] vise à retourner un ordre correspondant à un optimal local mais qui peut ne pas être le positionnement optimal global. D'autres utilisent des approches par décomposition spectrale [80] ou la combinaison des deux approches [105]. D'autres encore utilisent des techniques d'ordonnancement partiel inspirées des algorithmes de partitionnement [23, 130].

Dans leurs travaux, Mueller *et al.* [98] ont effectué une comparaison de différents algorithmes d'ordonnancement afin de déterminer leurs effets sur la lisibilité des DOM pour des graphes de tailles allant jusqu'à 1000 nœuds. Ils comparent ainsi trois classes d'algorithmes d'ordonnancement (simples, dédiés aux matrices creuses et par décomposition spectrale) pour huit catégories de

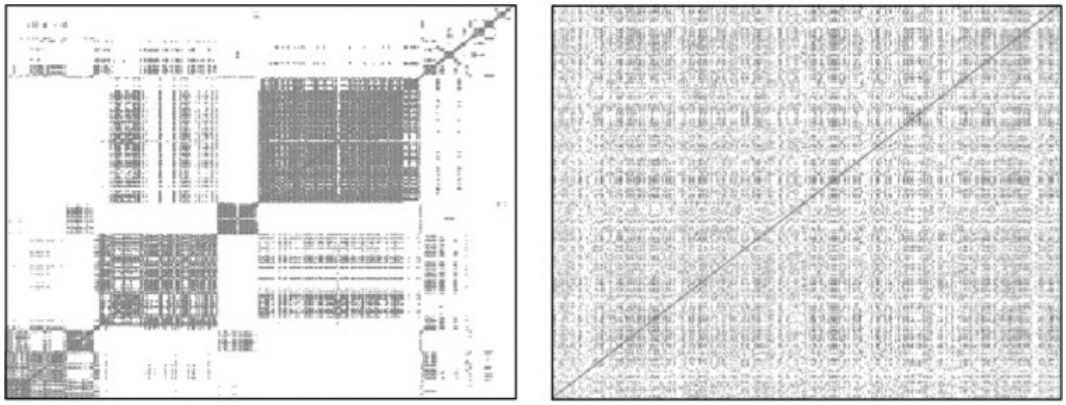
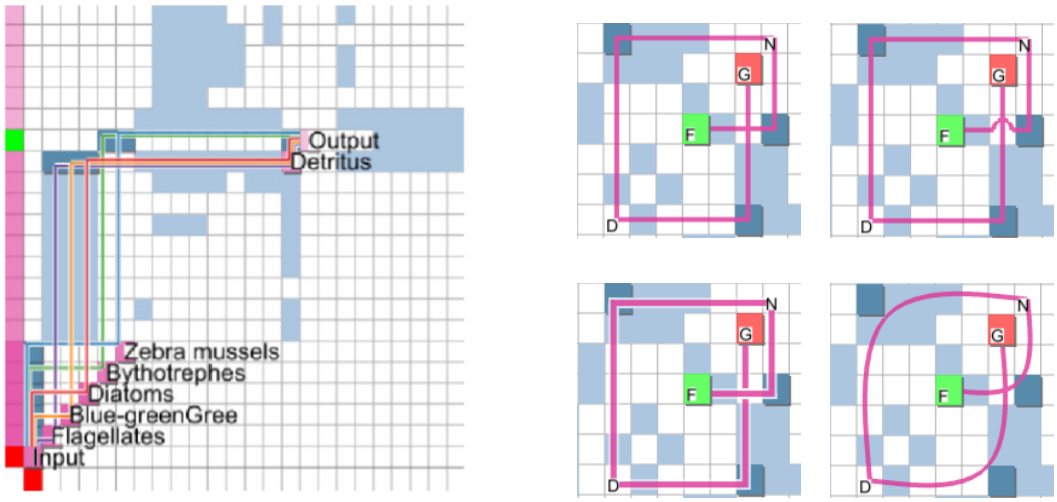


Figure I.3: L'ordre des nœuds peut avoir une influence sur la lisibilité des informations que représente la matrice. A droite, une matrice avec un ordre aléatoire, à gauche, des données ordonnées afin de mettre en avant la structuration des données. Images extraite de [98].

graphes (cinq synthétiques et trois réels). Ils déduisent de leurs travaux que l'un des principaux aspects esthétiques pouvant améliorer les DOM est de rapprocher autant que possible dans la représentation les entités proches en terme de distance dans le graphe. Cette conclusion rejoint les critères esthétiques des DNL qui visent à rapprocher dans le plan les éléments connectés.

Techniques hybrides

Les deux approches DNL et DOM sont très différentes et pourtant représentent la même information, chacune avec des avantages et inconvénients. Certains se sont donc demandés s'il était possible de créer une nouvelle méthode de représentation, dite *hybride*, qui combinerait les avantages de ces deux techniques. La technique décrite par Shen et Ma [125] présente l'utilisation d'un DOM pour représenter un graphe et utilisant les polylignes d'un DNL afin de montrer les chemins connectant deux entités spécifiques (voir figure I.4a). Pour cela, la technique se base sur l'utilisation de la diagonale de la matrice pour représenter les nœuds. Des polylignes orthogonales sont ensuite tracées entre les représentations de nœuds présentes sur la diagonale. La brisure de la polyligne est ainsi placée sur la case correspondant à l'arête. Avec cette technique, les polylignes peuvent se croiser et il peut être difficile de distinguer un croisement d'arêtes d'une brisure de polyligne. La technique s'appuie donc tout d'abord sur l'utilisation de couleurs distinctes pour chaque arête afin de faciliter la distinction. D'autres techniques sont cependant présentées afin d'accentuer la distinction, soit en ajoutant un *pont* à l'une des polylignes, soit en rompant une des polylignes dans la zone du croisement ou encore en modifiant la courbure de la polyligne (voir figure I.4b).

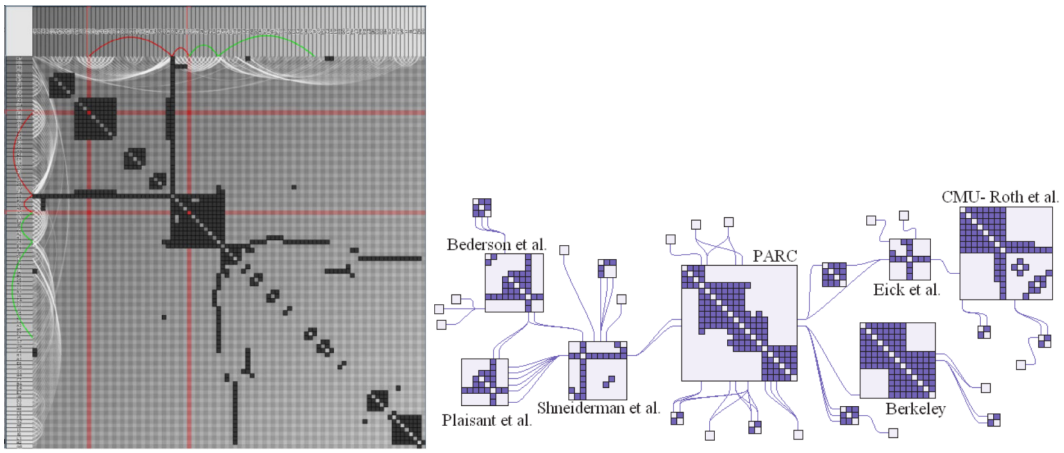


(a) (b)

FIGURE I.4: Technique introduite dans [125] : un chemin dans le graphe est représenté par des polylignes reliant les nœuds de la diagonale. Les quatre figures de droites montrent les différentes représentation d'un croisement d'arêtes : sans distinction visuelle, avec un *pont*, avec une coupure de la polyligne ou avec une modification de la courbure.

La technique présentée dans [60], appelée *matlink*, consiste en la représentation d'un graphe par une matrice d'adjacence dont les arêtes sont également représentées par des lignes courbes apposées sur la matrice et reliant les nœuds des axes (voir figure I.5a). Cette technique peut par ailleurs s'apparenter à une hybridation entre une matrice d'adjacence et un diagramme en arcs, qui est un DNL dont les nœuds sont alignés et reliés par des lignes courbes représentant les arêtes. Dans *matlink*, les arcs sont représentées en transparence, permettant de mettre en évidence le degré de chaque nœud par des arcs plus ou moins visibles. L'observation des arcs ainsi représentés permet d'identifier rapidement les nœuds fortement connectés.

Dans [61], les auteurs présentent une nouvelle approche, appelée *NodeTrix*, pour mettre en avant les communautés dans un réseau et en faciliter l'exploration. Cette technique utilise une représentation DNL d'un graphe permettant la transformation en DOM de communautés sélectionnées par l'utilisateur (voir figure I.5b). Cette transformation permet de bénéficier à la fois du positionnement des nœuds dans le plan d'un DNL associé à la facilité d'analyse d'un graphe par une représentation DOM. Cette technique facilite l'analyse des communautés présentes dans le graphe mais aussi les éléments qui connectent ces communautés entre elles. Watson *et al.* [140] présente également une nouvelle technique hybride entre DOM et DNL appelée *Quilts* afin de représenter



(a) matlink **(b) nodetrix**

FIGURE I.5: A gauche, *matlink* La technique présentée dans [60] représentant un graphe en utilisant un DOM sur lequel est apposé des arcs afin de mettre en avant le degré et le voisinage d'un nœud. A droite, *Nodetrix*, une hybridation entre DNL et DOM afin de faciliter l'exploration de communautés dans le graphe.

des graphes orientés acycliques (DAG). Cette technique utilise la structure en couche du DAG pour créer une matrice d'adjacence pour chaque niveau du DAG. Ces sous-matrices sont ensuite positionnées côte-à-côte en diagonal *coin-à-coin* (voir figure I.6).

Les arêtes sont ensuite représentées de la même manière que pour un DOM classique ; elles sont représentées par un glyphe noir à l'intersection des lignes et colonnes des nœuds correspondant à ses extrémités. Lorsqu'une arête relie deux nœuds de deux alignements (couches) non successifs, le glyphe représentant cette arête est positionné sur la ligne ou colonne du nœud source mais en dehors de la *matrice* et coloré de la même couleur que le nœud distant.

Autres techniques

Même si les DNL et les DOM sont les techniques les plus souvent utilisés, d'autres techniques ont été présentées dans la littérature. On peut ainsi citer l'exemple de l'arc diagramme tel qu'introduit par Wattenberg [141], qui est très ressemblant à la représentation DNL. Dans un arc diagramme, les nœuds sont alignés horizontalement et les arêtes sont représentées par des arcs de cercles reliant les entités connectées. L'objectif consiste ici en l'utilisation des arcs pour représenter une séquence spécifique de nœuds. La différence entre ces deux techniques se fait sur le positionnement des nœuds. Tandis que l'optimisation du positionnement des nœuds dans un DNL tient un rôle prépondérant afin d'améliorer la qualité de la représentation, avec un arc diagramme, l'analyse est

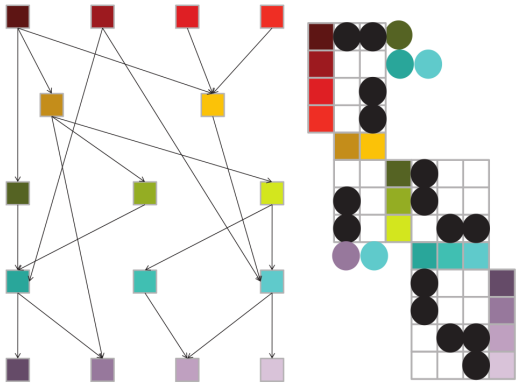
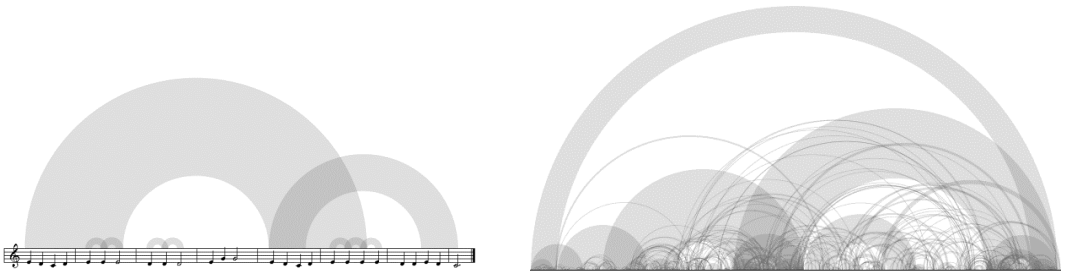


Figure I.6: Technique *quilt* présentée dans [140]. A gauche, représentation DNL d'un DAG, chaque alignement horizontal de nœuds correspond à une couche du DAG. à droite, la représentation *quilt* correspondante.

inversée, c'est la *qualité* de la représentation qui nous informe sur la séquence. Par exemple, la figure I.7a montre que les éléments du graphe, ici des ensembles de notes, sont reliées par un arc si cet ensemble se retrouve à un autre endroit du graphe. On peut ainsi effectuer l'analyse d'un morceau de musique (séquence répétée, etc) en observant son arc diagramme (voir figure I.7b).

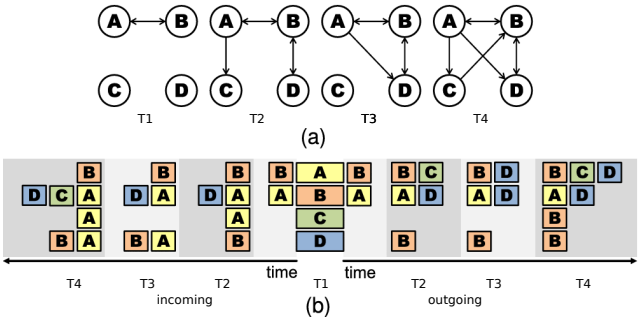


(a) (b)

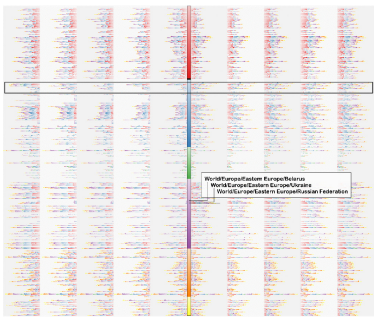
Figure I.7: (a) Dans un arc diagramme, un arc relie deux sous-séquences identiques. L'analyse d'une séquence avec un arc diagramme permet d'identifier les motifs répétés ainsi que la portée de cette répétition. (b) Exemple de résultats obtenus sur un morceau de musique complet, ici *Lettre à Élise*.

On peut également citer l'exemple de Hlawatsch *et al.* [64] qui se sont intéressés à l'utilisation des listes d'adjacences pour la visualisation de graphes orientés et pondérés dynamiques. Leur technique positionne les nœuds sur une colonne centrale et place en ligne horizontale de part et d'autre de cette colonne les voisins à distance 1 de chaque nœud. La position à gauche ou à droite du nœud est déterminée par l'orientation des arcs (voir figure I.8a). Cette technique

met également à profit la taille des nœuds pour indiquer les informations de pondération ainsi qu'un système de blocs visuels qui se décalent dynamiquement depuis la colonne centrale vers l'extérieur pour indiquer l'évolution au cours du temps du graphe considéré. De par la représentation compacte des éléments, cette technique permet la visualisation de grands graphes dynamiques. L'observation de la représentation dans sa globalité permet d'effectuer une analyse par la détection visuelle de motifs globaux tels que les changements de couleurs, leurs positions, *etc.* (voir figure I.8b). On retrouve ce type d'analyse sur des représentations orientées pixels ou de type *heatmap*.



(a) construction de la représentation en liste d'adjacence



(b) exemple de liste d'adjacence visuelle

Figure I.8: Exemple de visualisation de graphes orientés pondérés : Liste d'adjacence visuelle pour les graphes dynamiques décrits dans [64].

Comparaisons des techniques de représentations

Les deux techniques les plus utilisées, DOM et DNL sont des représentations visuellement très différentes et pour autant, elles transmettent exactement la même information. Certains auteurs se sont donc focalisés non pas sur l'amélioration de l'une ou l'autre de ces techniques, mais plutôt à leurs comparaisons en fonction de différentes tâches. Ghoniem *et al.* [51] a ainsi comparé les DNL et les DOM en utilisant, pour les premiers un algorithme de positionnement des nœuds par modèle de forces et pour le second un ordonnancement alphabétique des entités. Ils définissent pour leur protocole d'évaluation sept tâches :

— estimation de l'ordre du graphe (nombre de sommets),

— estimation de la taille du graphe (nombre d'arêtes),

— identification du nœud le plus connecté (*i.e.* ayant le plus grand nombre de voisin à distance 1),

— recherche d'un nœud à l'aide de son étiquette,

— recherche d'une arête reliant deux nœuds,

— recherche d'un voisin commun à deux nœuds, et
— recherche d'un chemin entre deux nœuds.

Ces tâches sont évaluées à l'aide de graphes de trois différentes tailles : 20, 50 ou 100 nœuds ; et trois différentes densités (voir définition 24) : 0.2, 0.4 ou 0.6. Les graphes générés sont non orientés aléatoires et les nœuds sont étiquetés par des règles alphabétique ou alphanumérique suivant la taille du graphe. Cette évaluation est faite à partir de l'analyse des résultats de 36 personnes connaissant les DNL mais non habituées aux DOM, pour les 126 questions (9 graphes $\times$ 2 diagrammes $\times$ 7 tâches). Leurs résultats montrent ainsi que pour 6 tâches sur 7, la lisibilité des DNL diminuait lorsque la taille et la densité des graphes augmentaient. Seuls les résultats pour la tâche *recherche un chemin entre deux nœuds* se sont vus meilleurs lorsqu'on utilisait un DNL au lieu d'un DOM. Suite à l'analyse de leurs résultats, les auteurs recommandent ainsi l'utilisation des DNL dans le cas de petits graphes, car permettant d'obtenir des représentations plus lisibles et plus simples à interpréter que les représentations matricielles. Au contraire, dans le cas de grands graphes, les performances obtenues avec les DNL déclinent rapidement pour devenir moins bonnes que celles obtenues avec les DOM. Les auteurs émettent cependant une réserve sur les résultats obtenus, notamment concernant l'algorithme de positionnement des nœuds pour les DNLs. Des travaux de Purchase [108] montrent que l'algorithme de positionnement a peu d'impact sur la lisibilité de ces diagrammes mais essentiellement pour les petits graphes.

Keller *et al.* [81] ont confirmé ces résultats en comparant les DNL et les DOM avec des paramètres expérimentaux proches sur des graphes orientés. Ils effectuent deux expériences, l'une visant à identifier les paramètres influant la lisibilité des DOM, l'autre visant à comparer la lisibilité des deux représentations. Ils évaluent ainsi trois paramètres dans la première évaluation : l'ordre, la densité et le positionnement des étiquettes. Pour cela, le protocole inclue l'évaluation de trois groupes de 12 graphes d'ordre 10, 20 et 40, chaque groupe divisé en deux pour représenter des densités de 10 et 20%. 21 personnes ont participé à cette évaluation et ont répondu à deux questions : compter le nombre d'arêtes sortantes et compter le nombre d'arêtes entrantes pour un nœud spécifique. Les résultats obtenus permettent de tirer peu de conclusions ; selon les auteurs, la variabilité des temps de réponse entre participants ayant une influence plus importante que celle créée par les paramètres évalués. La principale conclusion mentionnée pointe l'importance de l'étiquetage des entités pour l'identification des lignes et colonnes et en particulier pour l'analyse de grandes matrices.

La seconde évaluation, portant sur la comparaison des deux diagrammes, est basée sur cinq tâches : 1. trouver et sélectionner un nœud en particulier, 2. trouver et sélectionner une arête reliant deux nœuds en particulier, 3. estimer le nombre de connexions entrantes d'un nœud, 4. estimer le nombre de connexions sortantes d'un nœud, 5. estimer le nombre de voisins communs

entre deux nœuds, et 6. estimer la taille du plus court chemin reliant deux nœuds.Le protocole est basé sur l'utilisation de deux jeux de données réels : un de taille moyenne (22 nœuds) avec une faible densité (env. 6 %) et un de taille plus importante (50 nœuds) avec une densité plus forte (env. 27%). Un troisième jeu de donné de petite taille est utilisé pour la démonstration et les premiers essais des participants. Les nœuds des DNL sont positionnés à l'aide du programme *neato* décrit par Gansner et North [47] qui utilise un algorithme de dessin par modèle de force. Les 16 participants ont ainsi dû répondre à 2 × 24 questions différentes (6 tâches × 2 graphes × 2 représentations), l'ordre d'apparition des questions (combinaisons de tâche-graphe-représentation) étant défini aléatoirement.

Les résultats obtenus sont globalement similaires à ceux de l'évaluation de Ghoniem *et al.* [51]. En effet, là aussi les résultats montrent de meilleurs performances en utilisant les DNL pour les petits graphes tandis que les DOM se révèlent meilleurs pour des graphes de plus grandes tailles. Seuls les résultats pour la tâche de recherche de chemin étaient meilleurs en utilisant un DNL qu'en utilisant un DOM avec des grands graphes. Les auteurs concluent leurs analyses en recommandant de choisir le diagramme à utiliser en fonction de la tâche et du type de graphe (ordre et densité) et émettent l'hypothèse qu'une vue multiple en utilisant les deux diagrammes conjointement permettrait de choisir l'un ou l'autre directement en fonction de la tâche à effectuer.

Bae et Watson [15] ont de leurs cotés comparé la représentation *Quilt* sous ses différentes formes ainsi qu'avec les représentations nœuds-liens et matricielle pour une tâche de recherche de chemin. Les résultats montrent que les *Quilts* sont plus adaptés que les deux autres représentations pour la tâche évaluée sur des graphes orientés acycliques. Ils confirment aussi partiellement les résultats des deux évaluations précédentes indiquant que dans leurs résultats les DOM apparaissaient moins efficaces que les DNL pour la tâche de recherche de chemin.

1.2 Visualisation de graphes pondérés

Il est possible d'adapter la plupart de ces représentations afin d'ajouter une information de pondération des éléments et des relations (arêtes) du graphe, nous ne présenterons ici que quelques exemples.

La plupart des techniques de représentation de la pondération s'appuient sur la modification de la longueur, épaisseur ou couleur des arêtes en fonction de leurs poids.

Dans leur article, Collberg *et al.* [29] utilisent une représentation par DNL et appliquent une correspondance inverse entre la longueur d'une arête et son poids. Les nœuds sont ensuite positionnés par un algorithme de force. Le poids associés aux éléments et relations intervient ainsi dans l'algorithme en associant le poids des éléments à leurs inerties. Selassie *et al.* [124] mettent en avant les relations entre éléments d'un graphe par la mise en évidence des

regroupements d'arêtes en fonction de leurs orientations (les arêtes ayant des directions similaires sont agrégées ensemble) et dont la largeur varie en fonction de l'importance du flux. (voir figure I.9)

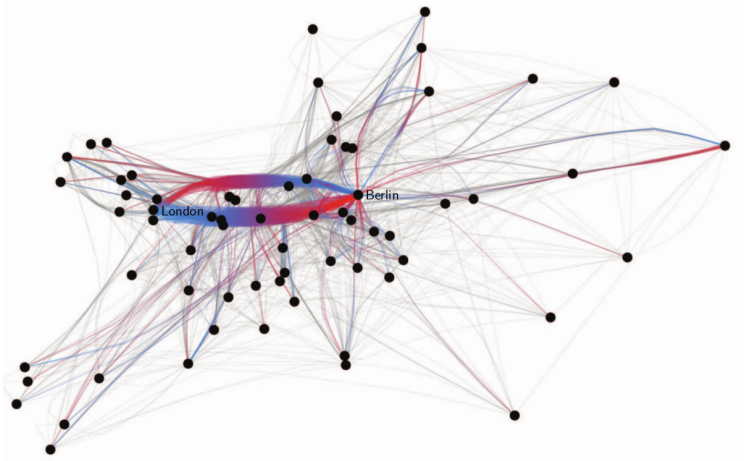
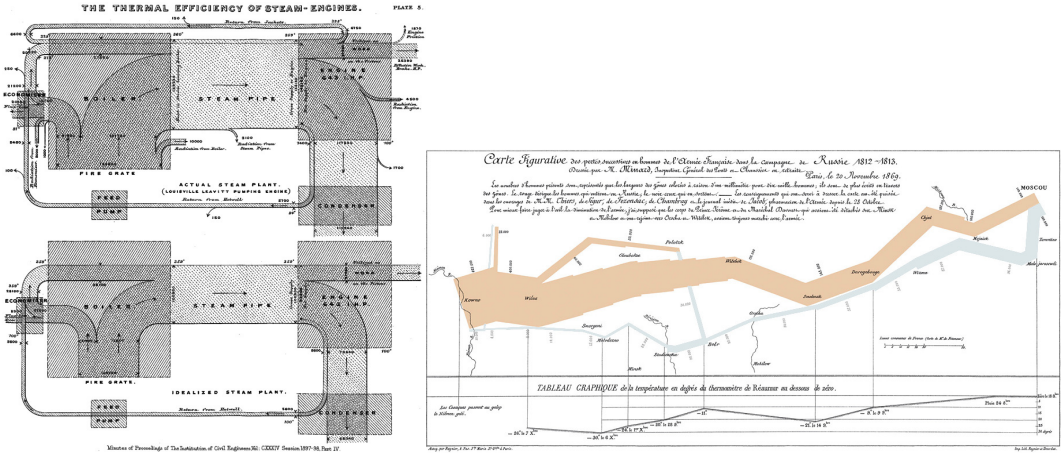


Figure I.9: Exemple de visualisation de graphe orienté avec un diagramme nœuds-liens : illustrations extraite de [124]

La représentation de la pondération dans les DOM fait appel à des mécanismes similaires : la plupart des solutions trouvées dans la littérature font appel soit à une modification de la couleur ou de la transparence des cases des arêtes, soit à une modification de leurs tailles en fonction de leur importance : plus une arête aura un poids important, plus elle sera dans les valeurs hautes de l'échelle de couleurs (ou de transparence) ou plus la case correspondante sera grande. Les techniques décrites dans [20, 59] utilisent notamment cette coloration. Lorsque toute la matrice est colorée, on peut considérer le résultat de cette coloration comme une carte de chaleur permettant d'identifier les différentes valeurs dans la matrice. Pour de plus amples informations, nous recommandons la lecture de [144] qui présente les différentes évolutions de cette technique.

D'autres techniques que les DOM et les DNL ont cependant été créées pour représenter la notion de pondération entre éléments d'un graphe, notamment les méthodes qui s'inspirent des diagrammes de flux. Une des techniques les plus connues est celle du diagramme de Sankey [119] initialement utilisé par Minard [96] (voir figure I.10). Cette technique peut être assimilée à un cas particulier du DNL dédié à la représentation de flux. En effet, elle représente les éléments du graphe par une forme géométrique et les connexions entre éléments par des lignes dont la largeur indique leur importance. Cela permet de mettre en avant les tendances majeures tout en représentant les quantités associées et de faciliter l'identification des contributions majeures dans l'ensemble du flux. Plusieurs publications dans la littérature utilisent ce type de diagrammes [5, 114, 136, 146] apportant de nouveaux outils d'interactions ou de nouveaux

cas d'utilisations. Cependant la représentation en elle-même a relativement peu évolué, preuve de son efficacité.



(a) (b)

Figure I.10: Diagramme de Sankey et carte de Minard permettant la représentation de données pondérées : (a) diagramme de Sankey montrant l'efficacité énergétique d'un moteur à vapeur et (b) Carte de Minard montrant l'évolution du nombre de soldats dans l'armée de Napoléon lors de la campagne de Russie de 1812.

1.3 Représentation de données partitionnées

Il est possible de partitionner les données en fonction de différents algorithmes afin de regrouper les éléments en fonction de caractéristiques communes ou de valeurs proches. Associés à une représentation adaptée, ces partitionnements peuvent mettre en avant la structure des données pour en faciliter l'analyse. Notre travail se concentrera essentiellement sur la représentation de graphes partitionnés (voir définition 32) et non sur l'amélioration de techniques existantes ou la mise en place de nouvelles techniques de partitionnement. La représentation des données partitionnées peut ensuite être effectuée par différentes techniques. Nous présenterons quelques unes de ces techniques, et notamment celles basées sur le dessin d'arbre pour représenter l'arbre de partitionnement ainsi que les techniques dédiées.

Représentation d'arbre

Les représentations d'arbre font partie des techniques permettant de représenter les données partitionnées. De nombreuses techniques permettent de représenter un arbre, on les classe généralement en deux catégories : les techniques *space-filling* et les non *space-filling*. Les techniques *space-filling* sont

les techniques qui maximisent l'utilisation de l'espace disponible (les *treemaps* par exemple).

On retrouve dans les techniques non *space-filling* les DNL où les partitions et les éléments du graphe sont représentés par des nœuds et les relations entre nœuds et celles d'appartenance aux partitions sont représentées par des liens. Il existe également d'autres techniques qui ne sont pas DNLs ; par exemple les arborescences indentées positionnent les éléments verticalement, la distance à la racine étant représenté par une indentation [1].

Le principal inconvénient de ces techniques est que toutes ne permettent pas de représenter les arêtes entre nœuds du graphe en complément de l'arbre de partitionnement. Nous nous attacherons à présenter dans les sections suivantes les techniques permettant de représenter l'arbre de partitionnement ainsi que les relations entre éléments du graphe.

Représentation non *space-filling*

Les représentations non *space-filling* utilisent généralement des DNL et les communautés du dessin de graphes et de la visualisation d'information ont présenté différents algorithmes de dessin permettant la représentation de données partitionnées. L'objectif de ces algorithmes consiste à présenter à la fois les nœuds et arêtes du graphe mais également les différents niveaux de partitionnement des nœuds. Le fonctionnement général de ces algorithmes consiste à rapprocher dans le plan les éléments appartenant à une même partition et à les rassembler au sein d'une structure (des formes rectangulaires ou circulaires par exemple). De nombreuses techniques permettant d'améliorer le positionnement pour éviter les croisements d'arêtes ou améliorer la mise en évidence des relations hiérarchiques ont été présentées dans la littérature [18, 67, 128].

Techniques *space-filling*

On appelle technique *space-filling* les techniques qui tentent d'utiliser l'espace disponible dans sa totalité. La technique la plus représentative est celle des *treemaps*. Cette technique initialement présentée dans [126] se base sur des divisions récursives d'un rectangle pour chacune des partitions de l'arbre. Ainsi, pour toute partition de l'arbre correspond un rectangle dans la représentation. Ce rectangle est divisé en autant de rectangles que la partition contient de sous-partitions. Différentes améliorations ont été apportées à cette technique, notamment pour améliorer la détection de rectangles imbriqués [75, 134], les proportions des subdivisions [24] ou utiliser d'autres formes que le rectangle[17](voir figure I.11).

1. les arborescences indentées sont généralement utilisée pour les aperçus d'arborescence sur les systèmes de parcours de fichiers et dossiers

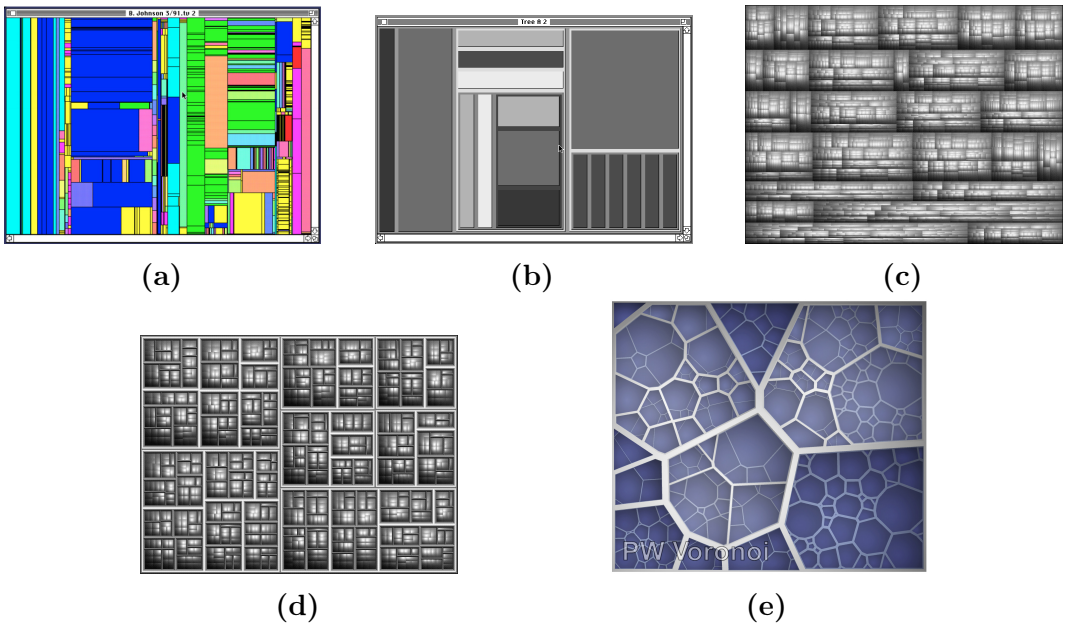
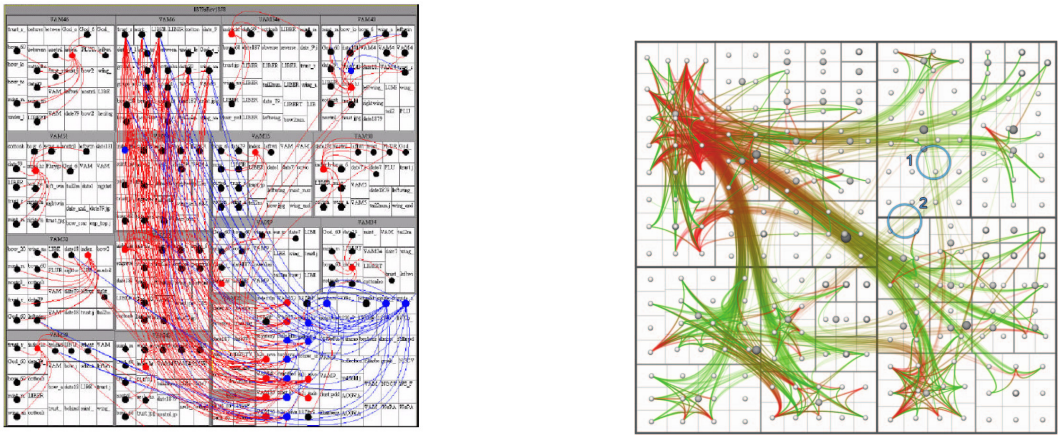


FIGURE I.11: Différentes techniques de treemaps : (a) treemaps telle qu'introduite par Shneiderman [126], utilisant une imbrication des éléments de la même lignée dans l'arbre de partitions présenté par [75] (b) et [134] (c) une fonction de division de l'espace proche du carré (d) ou par cellules de Voronoï (e).

Initialement, cette technique a été introduite pour la représentation d'arbres mais on trouve dans la littérature des exemples présentant également les liens entre éléments appartenant à ces partitions. Ainsi dans [39], la technique décrite utilise une apposition d'arcs par dessus la *treemap* afin de représenter les relations entre éléments. Le principal inconvénient de cette technique est l'encombrement visuel induit par la superposition des liens(voir figure I.12a). Dans [67], l'auteur décrit une technique de faisceautage d'arêtes permettant d'améliorer la représentation en réduisant cet encombrement visuel (voir figure I.12b).

Autres techniques

D'autres techniques ont également été présentées dans la littérature, certaines utilisent des approches hybrides, d'autres non. Neumann *et al.* [100] présentent *arcTree*, une technique hybride combinant deux représentations : une variante des treemaps pour laquelle les entités sont toutes positionnées sur la même ligne et un arc diagramme afin de relier les nœuds du graphe (voir figure I.13a). On peut également mentionner la technique introduite dans [54] qui utilisent un DOM dont le partitionnement est représenté sur les bords de la matrice (voir figure I.13b). Ainsi les nœuds sont ordonnés afin de rapprocher



(a) (b)

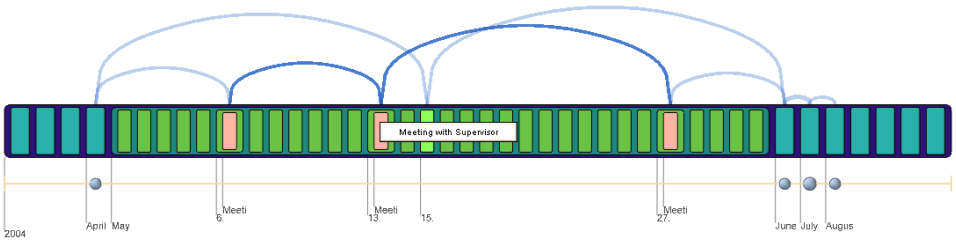
FIGURE I.12: Visualisation de liens apposés par dessus la représentation Treemap. (a) technique présentée dans [39] utilisant la courbure pour indiquer l'orientation des relations et (b) technique de faisceautage introduite dans [67] pour réduire l'encombrement visuel et utilisant la coloration pour indiquer l'orientation des relations.

les éléments appartenant à une même partition.

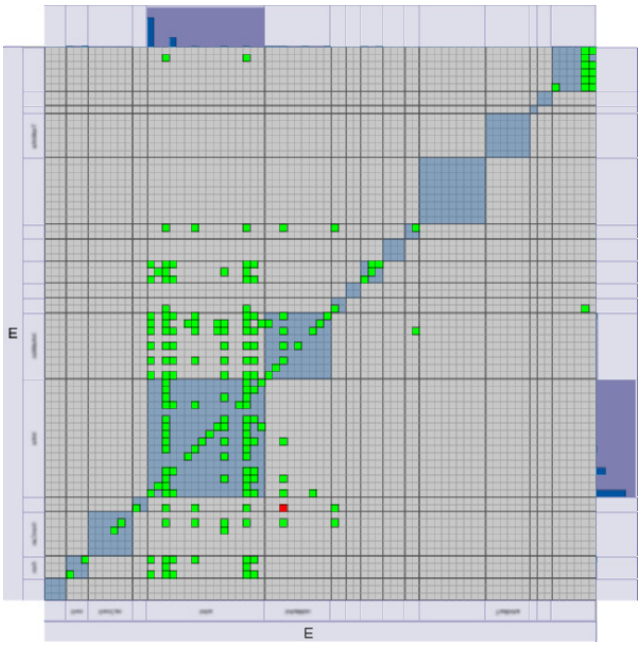
Représentation abstraite

Sans même considérer la visualisation de données massives, le nombre d'éléments à représenter peut rapidement devenir trop important pour en permettre la représentation. En effet, soit le nombre d'éléments peut être trop important pour en permettre le rendu par la machine, soit la représentation devient trop encombrée pour qu'une analyse puisse être effectuée par un utilisateur. Ainsi, différentes techniques ont été mises au point pour réduire la quantité de données à représenter. Nous n'aborderons ici que les techniques basées sur le partitionnement de données, mais il en existe d'autres, basées par exemple sur l'échantillonnage.

Si on considère un graphe partitionné (voir figure I.14a), il est possible de définir une coupe maximale de l'arbre de partitionnement. On appelle *coupe maximale* c un ensemble de nœuds de l'arbre tel que pour tout nœud n appartenant à c, c ne contient pas d'ancêtre de n. La figure I.14b présente un exemple de coupe de l'arbre, représentée par une ligne pointillée rouge. On distingue deux types d'abstractions de données partitionnées : le cas simple et le cas multi-échelle. Le cas simple correspond à effectuer une coupe dans un seul niveau de l'arbre des partitions et à représenter le graphe abstrait, appelé graphe quotient (voir définition 34), correspondant à cette coupe (voir figure I.14c). Le cas complexe consiste en la représentation d'une coupe à différents niveaux de l'arbre, il faut donc permettre l'identification du niveau



(a)

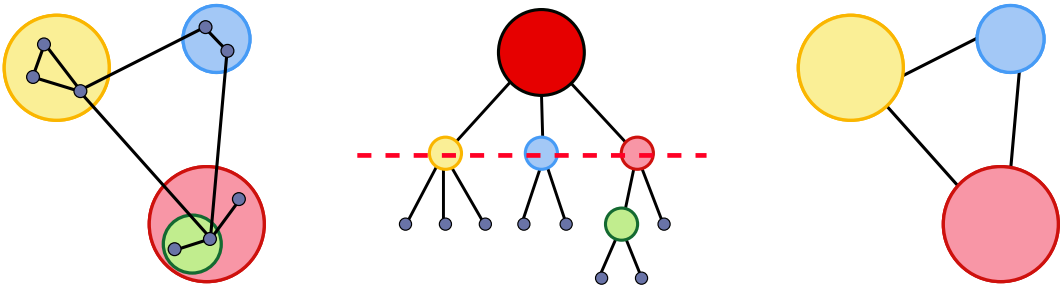


(b)

Figure I.13: Autres techniques de visualisation d'arbres avec relations entre éléments. (a) La technique Arc-tree se base sur une treemap horizontale avec des liens représentant les relations entre éléments. (b) une matrice avec le partitionnement hiérarchique des éléments représenté sur les cotés de la matrice.

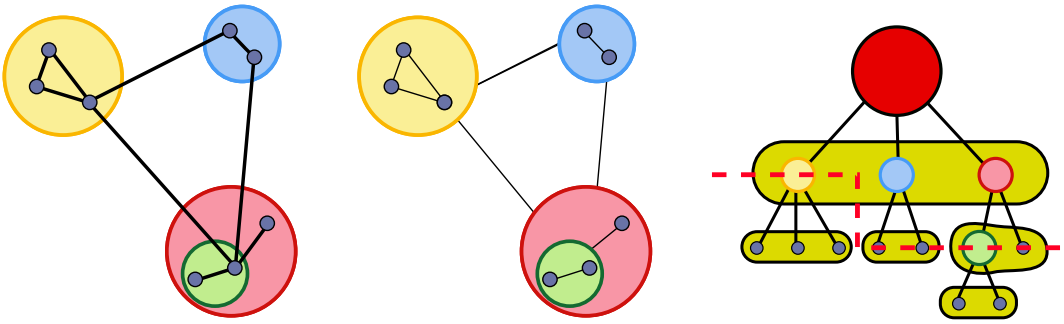
de l'arbre observé.

L'abstraction multi-échelle des données consiste à effectuer une coupe de l'arbre à différents niveaux de l'arbre. Par exemple on peut voir sur la figure I.15 que l'arbre de partitionnement est divisé en différents ensembles (encadré en vert sur la figure I.15c) correspondant aux descendants appartenant à une partition. La figure I.15b correspond à l'abstraction multi-échelle. On retrouve cette technique dans différents articles de la littérature. La première utilisation de cette technique a été présentée dans [33]. La technique décrite représente un graphe partitionné dans un espace tri-dimensionnel où chaque niveau de l'arbre est représenté par un niveau de profondeur dans l'espace 3D. Ainsi il est



(a) (b) (c)

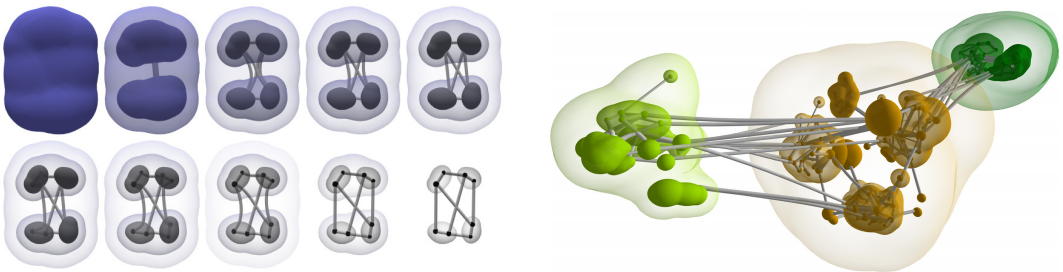
FIGURE I.14: Visualisation abstraite d'un graphe partitionné : (a) le graphe d'origine partitionné, (b) une coupe maximale de l'arbre de partitionnement (ligne pointillée rouge), (c) graphe quotient résultant de la coupe de l'arbre.



(a) (b) (c)

FIGURE I.15: Visualisation abstraite multi-échelle d'un graphe partitionné : (a) graphe partitionné, (b) visualisation abstraite multi-échelle résultante et (c) le cas complexe avec une coupe de l'arbre de partitionnement.

possible d'analyser le partitionnement du graphe et les relations entre éléments au sein d'un niveau de la hiérarchie. Cependant, cette représentation souffre des problèmes d'encombrements induits par la représentation de tous les niveau de l'arbre de partitionnement ainsi que des problèmes d'occlusions induit par la représentation tri-dimensionnelle. Une autre approche présentée dans [16] représente les nœuds et partitions par des volumes imbriqués (voir figure I.16b). La modification de la transparence de la surface de ces volumes permet de modifier de façon interactive le niveau de l'arbre de partitions visualisables. D'autres approches [46, 123, 131] utilisent cette technique pour réduire le nombre d'éléments à afficher en représentant le graphe quotient correspondant à une coupe de l'arbre.



(a) (b)

Figure I.16: Représentation tri-dimensionnelle de données partitionnées par objets 3D imbriqués. La modification de la transparence de la surface permet de visualiser différents niveaux de l'arbre de partitionnement.

2 Visualisation de données multi-variées

Les données multi-variées aussi appelées multi-dimensionnelles peuvent se résumer à des données pour lesquelles à un élément, correspond plusieurs attributs. Une façon de se les représenter consiste à visualiser un tableau dont les lignes correspondent aux éléments et les colonnes aux attributs ; chaque case correspond à la valeur de l'élément considéré pour un attribut particulier. On retrouve ce type de données dans de nombreux domaines, ingénierie, santé ou encore en sciences humaines, *etc.* . L'obtention d'une représentation efficace de ce type de données est un réel défi, de par la nature même des données et la diversité de l'information à représenter. Pour cela, différentes approches sont décrites dans la littérature, approches que nous classerons en deux catégories : les techniques inspirées du nuage de points et les techniques inspirées des coordonnées parallèles. Pour une présentation plus complète, il existent plusieurs travaux s'intéressant tout particulièrement à présenter l'ensemble des techniques envisageables [25, 52, 65].

2.1 Matrice de nuage de points et techniques similaires

Si l'on considère la visualisation de données à deux attributs numériques, l'une des représentations les plus fréquentes est celle des nuages de points (*scatterplot* en anglais). Cette représentation permet de projeter deux attributs sur les axes x-y des coordonnées cartésiennes (où trois attributs avec les axes x-y-z en représentation 3D) et permet ainsi d'analyser l'évolution des paramètres l'un par rapport à l'autre et de mettre en avant différents motifs dans les données tels que les corrélations, tendances, anomalies ou cas particuliers, *etc.* . Une généralisation de ce diagramme pour des données multi-variées consiste en la création d'une matrice des comparaisons deux à deux des variables du jeu de données. Pour cela, pour des données contenant k variables, V_1, V_2, ... V_k,

on crée une matrice de k lignes et k colonnes. Pour chaque ligne i et colonne j, on place à l'intersection des lignes et colonnes le diagramme en nuage de points correspondant aux variables V_i et V_j. On appelle le diagramme résultant une *Matrice de nuage de points* (MNP). Ce diagramme permet de comparer deux-à-deux chacun des attributs des données sous la forme d'un ensemble de nuages de points placés les uns à coté des autres sur une grille (voir figure I.17). De la même manière que dans les DOM, l'ordre des éléments (ici les variables) est le même sur l'abscisse et sur l'ordonnée. Il en résulte également que la diagonale du MNP correspond à un nuage de points qui compare une variable à elle-même (V_i par rapport à V_i). De même, comme les éléments suivent le même ordre sur l'abscisse et l'ordonnée, cela implique un dédoublement des données représentées qui peut être résolu par la représentation d'une demi-matrice. Un outil d'interaction souvent associé à la MNP permet la sélection d'un ensemble d'éléments dans un des nuages de points provoquant la mise en surbrillance des éléments correspondant à cette sélection dans les autres nuages de points de la matrice (*linking* et *brushing*).

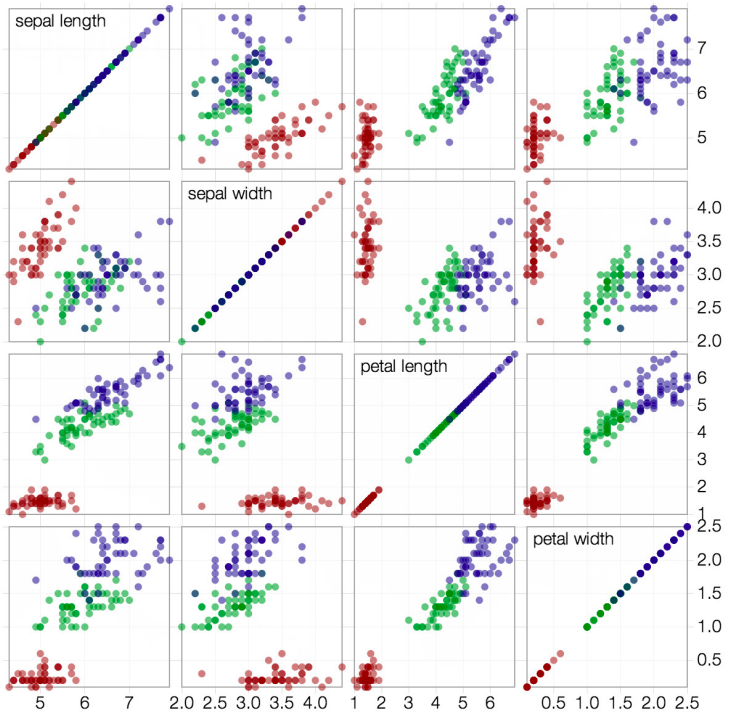


Figure I.17: Exemple de représentation de données multi-variées par une Matrice de Nuage de points. Ici les données représentent des caractéristiques de fleurs (longueurs des pétales, sépales, etc) et regroupent au total 4 attributs tous comparés deux-à-deux à l'aide de nuage de points.

Une des publications les plus récentes visant à l'optimisation de cette

technique est celle des travaux de Elmqvist *et al.* [37] (voir figure I.18). Les auteurs y présentent une nouvelle technique de visualisation de nuage de points associée à une représentation de MNP interactive. Il présente ainsi un système de visualisation en deux parties : le nuage de points et la MNP. Dans la MNP, l'utilisateur peut se déplacer de façon interactive d'un nuage de points à un autre. Ces déplacements induisent une transformation animée du nuage de points vers son état suivant montrant de nouvelles variables. Ainsi, en s'appuyant sur la position des variables dans la MNP et en effectuant une rotation dans l'espace 3D dans le sens adéquat, la technique transforme le nuage de points par une animation similaire à la rotation d'un cube. Cette rotation implique que sur les deux variables représentées avant l'animation, une sera remplacée dans la représentation et l'autre sera maintenue. Cette technique permet notamment de maintenir le contexte des données représentées et notamment de maintenir le lien avec la variable qui est conservée.

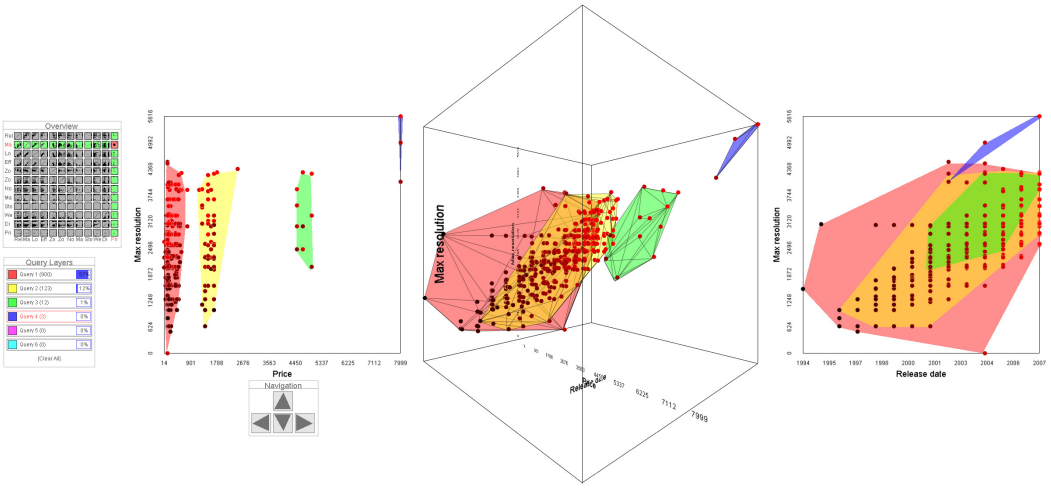
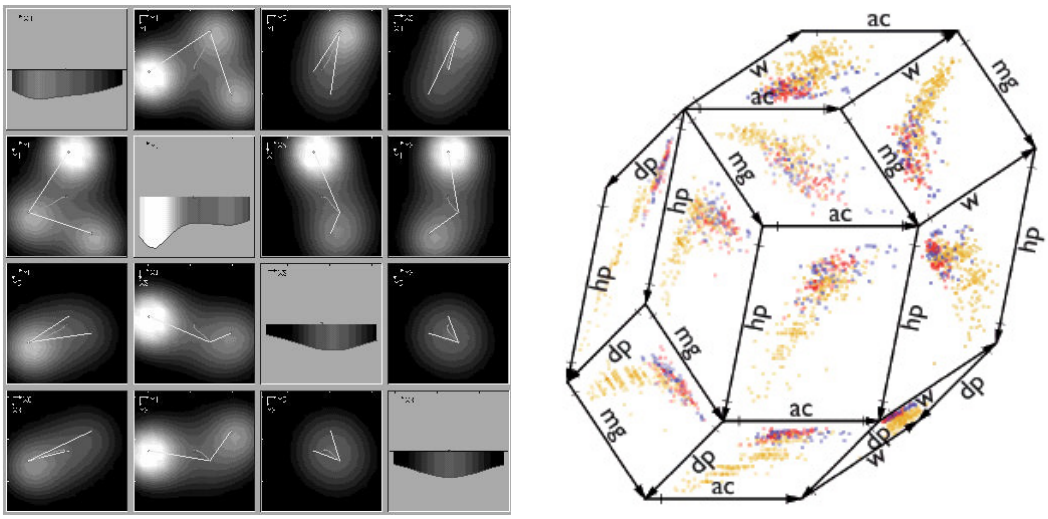


FIGURE I.18: Technique de visualisation de Nuage de Points interactive associée à la représentation d'une MNP. L'animation du nuage de points par une rotation de la représentation permet de maintenir le contexte des données visualisées.

D'autres travaux de la littérature présentent différentes adaptations des nuages de points pour des données multi-variées. Dans [133, 145], la technique présentée est très similaire mais est utilisée pour représenter des données scalaires et est appelée *HyperSlices* (voir figure I.19a). Une autre technique permet l'attribution d'une dimension de l'espace à chaque variable. Ainsi si deux variables sont utilisées, la représentation forme un carré, avec trois variables elle forme un cube. Cette technique, appelée *Hyperboxe* et introduite dans [7], ne se limite cependant pas à la représentation de trois variables mais permet la projection de N variables dans un espace à N-*dimensions* (voir figure I.19b).



(a) (b)

Figure I.19: Adaptations de scatterplot : (a) HyperSlice présentée par Van Wijk et Liere [133] pour données scalaires et (b) hyperbox présentée par Alpern et Carter [7] pour n-dimensions (image extraite de [28]).

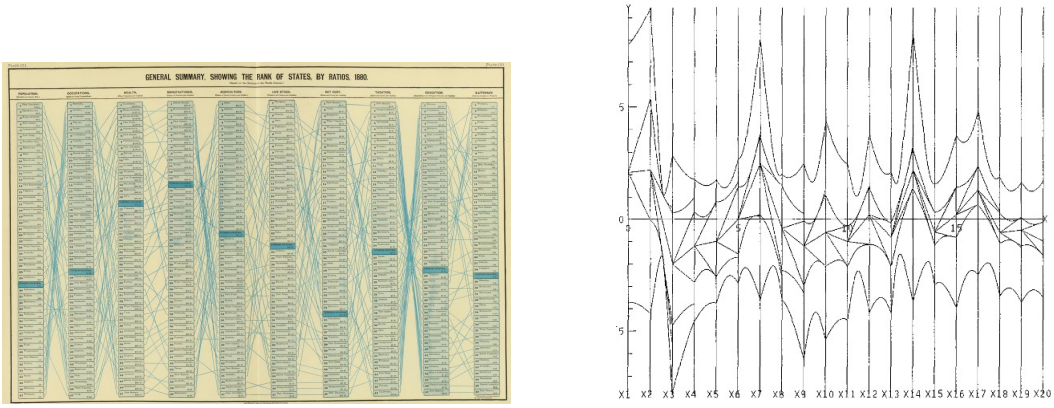
2.2 Coordonnées Parallèles

Une autre technique souvent utilisée pour la représentation de données multi-dimensionnelles est celle des coordonnées parallèles. Cette technique représente les attributs par des axes verticaux parallèles (des dimensions) et les éléments des données par des polylignes, chaque polyligne croisant les axes à la valeur de l'élément pour l'attribut concerné.

Techniques classiques

Depuis sa première utilisation dans [45] (voir figure I.20a), et sa présentation dans un contexte scientifique plus récent dans [71, 72] (voir figure I.20b), les coordonnées parallèles ont été la base de nombreux travaux visant à les optimiser pour différentes tâches comme le montre l'étude de Heinrich et Weiskopf [57]. On retrouve différents travaux utilisant les coordonnées parallèles et proposant diverses améliorations, que ce soit sur la technique de représentation, qui peut être orientée géométrie ou densité, ou sur les techniques et outils d'interaction. Cependant, la capacité de la technique à représenter l'intégralité des attributs, induit certaines limites lorsqu'on considère la visualisation de jeux de données de taille importante. Parmi ces limites, on trouve notamment le problème d'encombrement visuel induit par les croisements de polylignes.

Une première approche pour prendre en compte ce problème d'encombrement consiste à utiliser la transparence ou des cartes de chaleur pour mettre en



(a) **(b)**

Figure I.20: (a) Première présentation des coordonnées parallèles par [45] dans le cadre d'un classement des états nord-américains. (b) Utilisation des coordonnées parallèles par [72].

évidence les zones d'importance [12, 143], on parle ainsi d'approches orientées densités (voir figures I.21a et I.21b). Dans [74] la technique présentée s'appuie sur des textures et des fonctions de transfert entre la densité d'éléments et l'opacité afin de faciliter la détection de motifs dans les données tels que les changements de *comportements* (passage d'une valeur haute à une valeur basse par exemple) ou l'identification de regroupements (*clusters* en anglais)(voir figures I.21c et I.21d). Zhou *et al.* [152] proposent de leurs cotés une technique qui agrège les polylignes en utilisant un modèle de forces pour réduire les interférences visuelles (voir figure I.22). Pour cela, la technique modifie la courbure des lignes en suivant un modèle de *gravitation* : les arêtes s'attirent les unes les autres pour minimiser la distance entre segments parallèles et minimiser les surfaces de croisement.

Ces techniques, bien qu'utiles restent cependant limitées par le nombre de valeurs distinctes que peut générer la technique de représentation de densité utilisée (transparence, échelle de couleurs, *etc.*) mais aussi par la précision de l'œil humain à distinguer deux valeurs proches dans cette représentation.

Des variations moins fréquentes des coordonnées parallèles se sont plutôt orientées sur une modification de l'agencement spatial des axes. On retrouve ainsi les *coordonnées radiales*, où les dimensions sont organisées autour d'un point central. Une nouvelle technique, appelée *many-to-many Paralleles Coordinates Plot* est présentée dans [90] (voir figure I.23a). Cette technique se caractérise par une duplication des dimensions associée à un positionnement dans l'espace en polygones complexes. Cela permet de s'affranchir de l'un des principale inconvénient des coordonnées parallèles : l'agencement linéaire ne permettant de comparer les dimensions que deux-à-deux.

On peut également mentionner les *TimeWheel*, introduites par [129], tech-

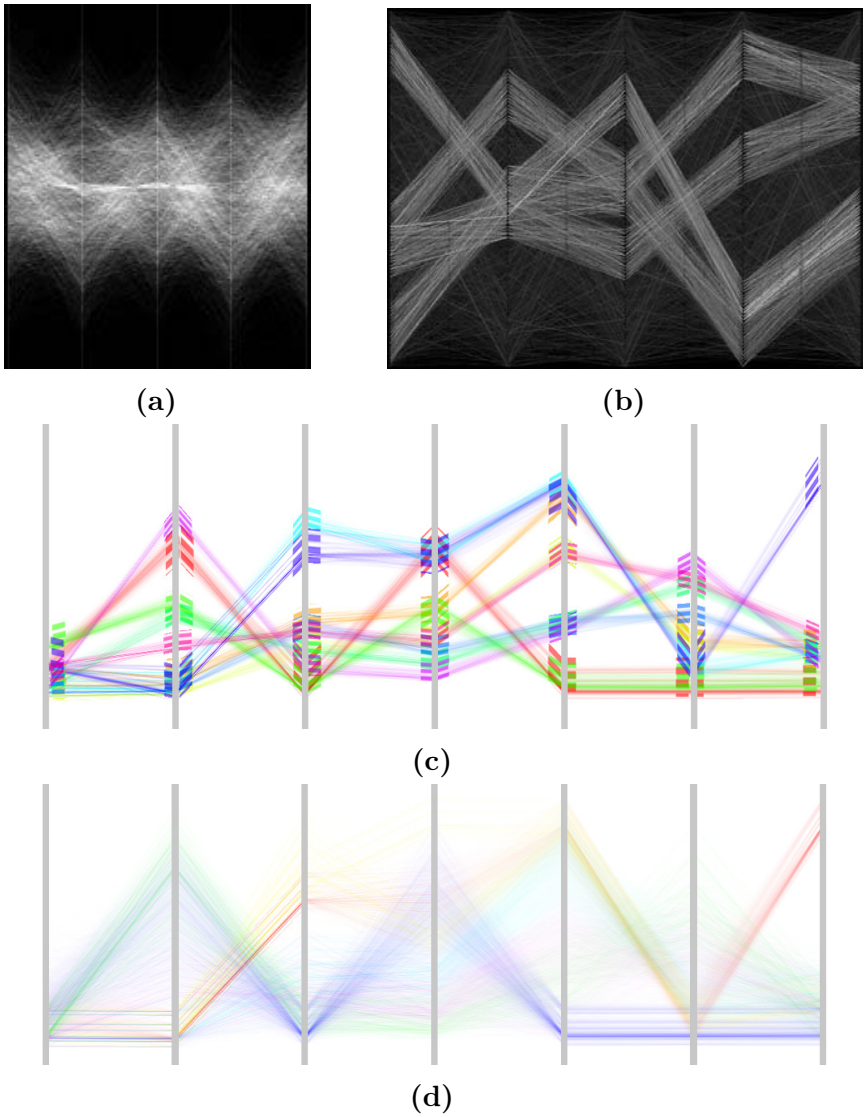


FIGURE I.21: Exemples de Coordonnées parallèles utilisant une fonction de densité pour mettre en avant les zones denses du diagramme. Ici, l'utilisation de la transparence permet de mettre en évidence les zones de forte densité. (a) technique de Wegman et Luo [143], (b) technique de Artero *et al.* [12], (c) et (d) technique de Johansson *et al.* [74].

niques permettant de mettre en relation les dimensions du diagramme par un arrangement circulaire de *n-1* dimensions autour de la dernière qui est ainsi considérée comme la dimension de *focus*.

Enfin, Claessen et Van Wijk [28] présentent une nouvelle technique permettant un libre positionnement des axes dans l'espace : les *Flexible Linked Axes* (voir figure I.23). Ainsi la technique permet à l'utilisateur de positionner

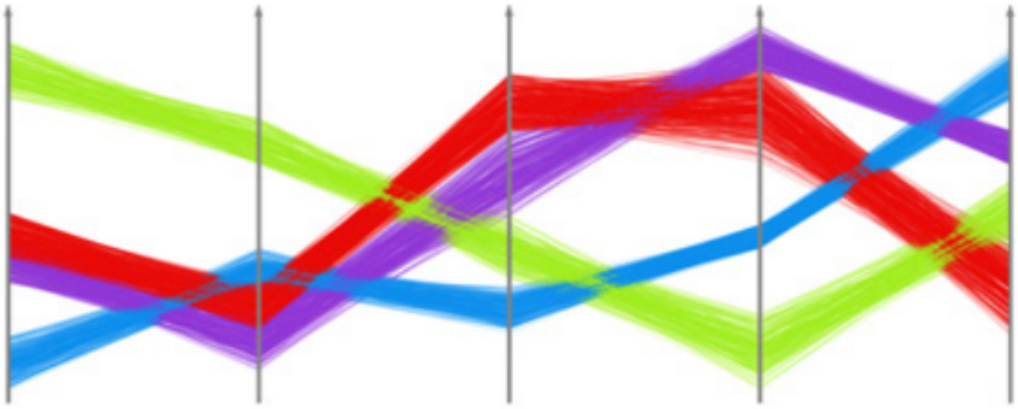


FIGURE I.22: La technique présentée dans [152] utilise le faisceautage des arêtes afin de mettre en avant les tendances majeures dans les données.

librement les dimensions dans le plan, soit de façon complètement déconnectée, soit organisée de façon à créer un polygone. Ensuite, l'utilisateur peut définir le type de représentation à utiliser entre deux dimensions proches *représentation ligne*, les coordonnées parallèles, ou *représentation points*, nuages de points.

2.3 Coordonnées parallèles abstraites

Une autre approche consiste à utiliser une représentation abstraite afin de réduire l'encombrement visuel. Cette abstraction se fait généralement par l'agrégation des données, et on considère généralement deux approches : agrégation des éléments toutes dimensions confondues et agrégation des éléments différemment pour chaque dimension. L'échantillonnage serait également une approche envisageable mais ne sera pas considéré ici [1].

Agrégation des éléments

Dans [42] les auteurs décrivent une technique d'exploration de données utilisant une approche d'agrégation hiérarchique des éléments. La technique décrit également une approche de coloration utilisant la hiérarchie résultante afin d'attribuer une couleur semblable aux éléments proches dans la hiérarchie. McDonnell et Mueller [95] mettent en avant dans leur représentation un faisceautage des arêtes par un algorithmes d'agrégation et une représentation par des enveloppes superposées (voir figure I.24). Ces enveloppes représentent, sur chacune des dimensions, les intervalles de valeurs contenant les éléments

1. on ne considère pas l'échantillonnage comme de l'abstraction de données mais comme le remplacement du jeu de données par un sous-ensemble représentatif

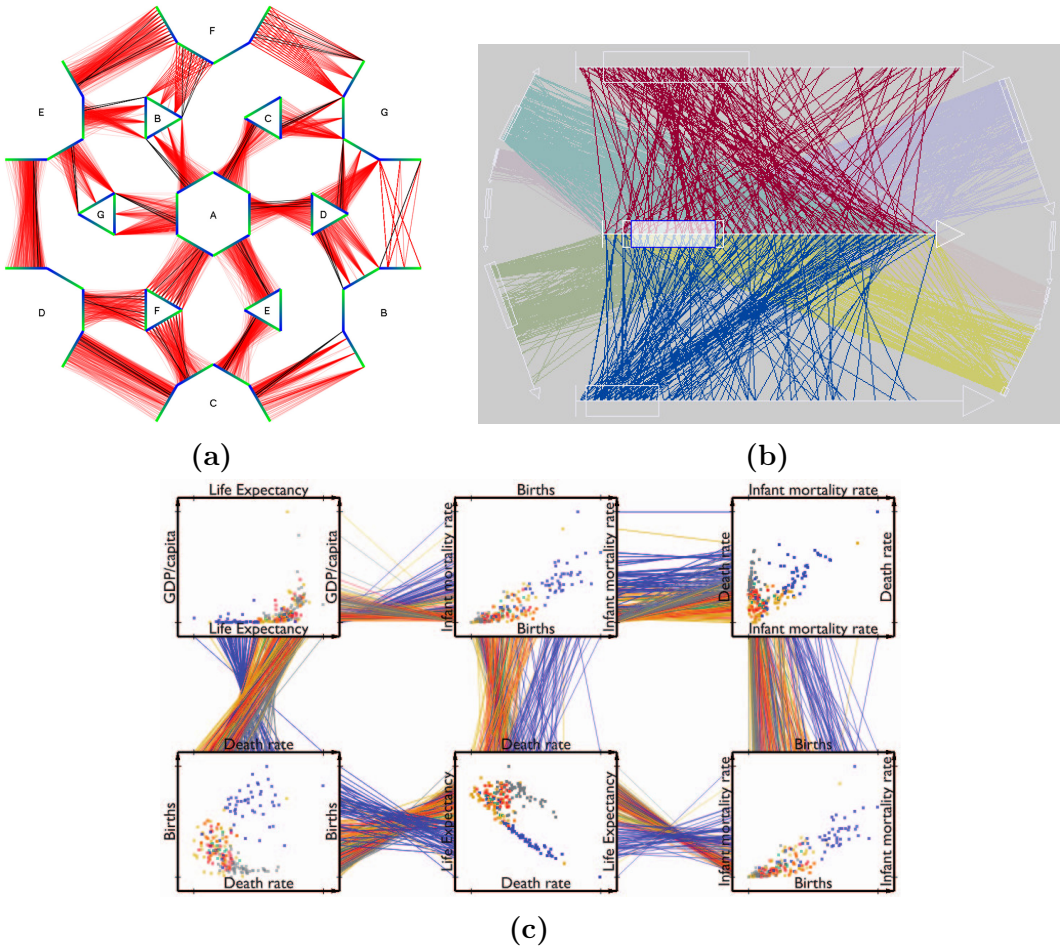


FIGURE I.23: Exemples de représentation inspirées des coordonnées parallèles utilisant un positionnement des axes non linéaire. (a) Many-to-many PCP introduit par [90], (b) Timewheel présenté par [129] et (c)*Flexible Linked Axes* introduites par [28].

du groupe. Ils s'appuient également sur une représentation orientée densité pour représenter la distribution des éléments entre et par dimension. Pour cela, ils utilisent différentes approches, telles que la transparence et les ombrages (changement de luminance) pour représenter des histogrammes par exemple.

Agrégation des éléments par dimension

D'autres se sont plutôt intéressés à une agrégation des éléments différente pour chaque dimension du jeu de données. Andrienko et Andrienko [8] s'intéressent notamment à la visualisation d'éléments regroupés par classes, calculées pour chaque dimension, avec une représentation par enveloppes. Afin de faciliter l'exploration des données, ils mettent en place une représentation des données regroupées dans des ellipses en fonction des classes créées précédemment, ou

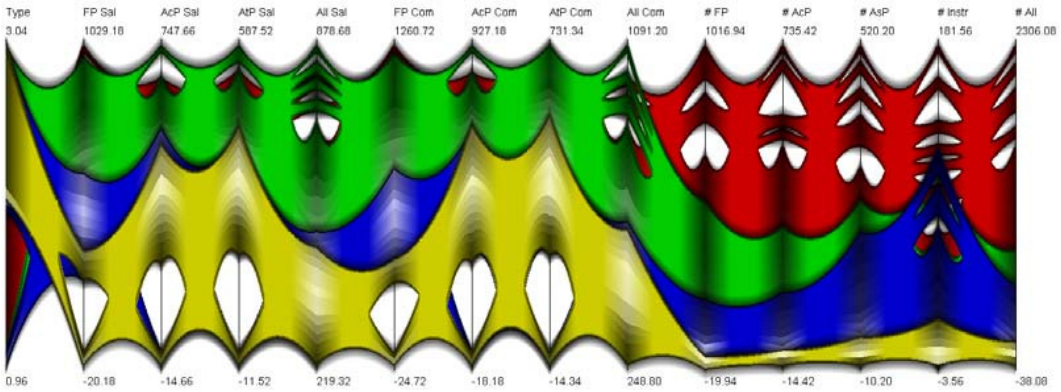


FIGURE I.24: Technique de coordonnées parallèles par agrégation des éléments, représentées par des enveloppes superposées.

en groupes de tailles égales.

Palmas *et al.* [102] utilisent quant à eux une technique d'agrégation des éléments pour chaque dimension et agrègent les segments entre agrégats pour mettre en avant les tendances majeures. Pour cela, ils utilisent les différentes propriétés des agrégats telles que les valeurs minimales, maximales et moyennes (voir figure I.25) et représentent les éléments par des surfaces courbées à l'aide de *splines*. Les extrémités sur les axes de ces surfaces représentent les valeurs minimales, maximales ainsi que la moyenne par le positionnement des connexions sur les agrégats (voir figure I.25).

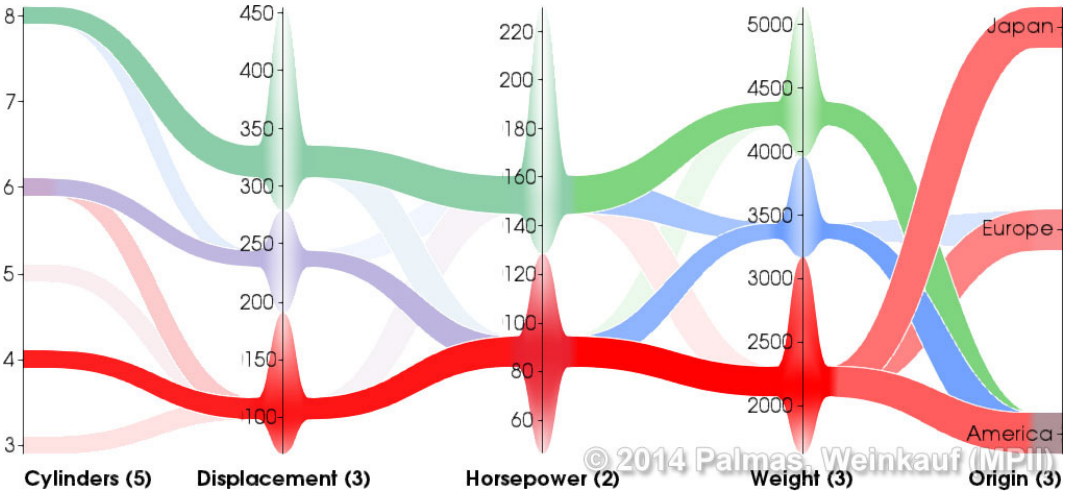


FIGURE I.25: L'utilisation des surfaces permet de représenter sur les axes les valeurs minimales, maximales et moyennes des regroupements tandis que l'utilisation des *splines* permet de réduire les surfaces entre les axes pour réduire l'encombrement visuel et améliorer la lisibilité du diagramme.

D'autres techniques s'appuient sur une agrégation des données en classes permettant de grouper les éléments entre deux axes. Ainsi, Novotny et Hauser [101] présentent une technique s'appuyant sur une méthode dite *output-oriented* pour permettre la visualisation de données multivariées s'appuyant sur de l'abstraction de données par *binning*. Le processus de traitement décrit sépare les données en deux groupes, les tendances et les valeurs aberrantes (*outliers* en anglais). Cette séparation permet d'agréger les données tout en réduisant le *bruit* provoqué par ces *outliers*. Les deux groupes sont ensuite réunis dans la représentation permettant ainsi de maintenir la visibilité des valeurs anormales tout en mettant en avant la représentation des tendances générales (voir figure I.26).

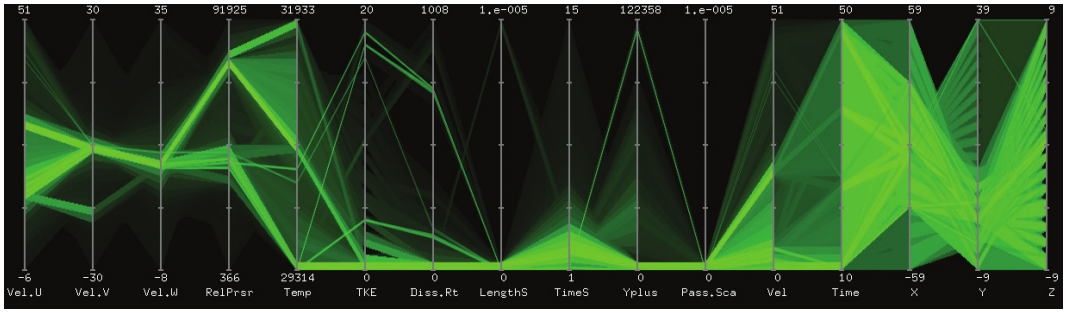


FIGURE I.26: Exemple de coordonnées parallèles utilisant une agrégation par binning, ici la technique présentée par [101].

Une amélioration de leurs travaux est présentée dans Rubel *et al.* [116] présentant notamment une modification de la technique de *binning* permettant de regrouper les données en classes d'intervalles de taille différentes.

Une technique de visualisation et d'analyse dédiée aux données catégorielles est présentée dans [19, 86]. Avec cet outil, chaque catégorie est représentée par un rectangle dont la taille représente le nombre d'éléments dans la catégorie. En complément, des histogrammes sont placés à l'intérieur même des rectangles afin de mettre en avant le degré d'indépendance entre chaque catégorie et ses catégories voisines (dans la figure I.27, représentés sur l'axe central). Plusieurs outils d'interaction sont associés à la représentation afin de permettre la sélection, la modification manuelle des catégories ou le produit croisé de plusieurs dimensions.

3 Exploration multi-échelle

Les sections 1.3 et 2.3 ont montré que la représentation abstraite permettait de réduire la quantité de données à représenter. Pourtant cette réduction induit une perte d'information, par exemple la valeur d'un élément précis. La technique la plus souvent utilisée pour corriger cette perte d'information consiste en la

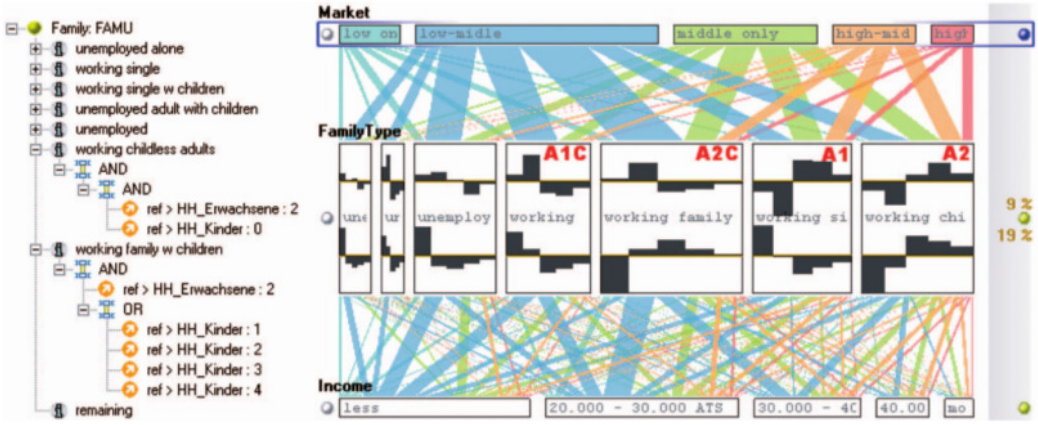


FIGURE I.27: ***ParallelSets*** **présenté par [86] présentant la représentation de données catégorielles et fournissant des informations complémentaires au sein de la représentation.**

mise en place d'outils d'interaction afin de permettre l'exploration des données à différents niveaux de détails. Cela permet notamment à l'utilisateur d'obtenir des informations de plus en plus précises sur les partitions d'intérêt. Nous avons vu en section 1.3 que le partitionnement des données et plus particulièrement la hiérarchisation des données permettait de montrer différents niveaux de détails en effectuant une coupe de l'arbre des partitions. Il existe de nombreuses techniques de dessin de graphes qui s'appuient sur cette technique[9, 44, 53], mais toutes ne permettent pas l'exploration interactive qui nous intéresse. Elles utilisent effectivement l'arbre de partitionnement mais uniquement pour construire un unique dessin qui peut ensuite être analysé. Si on s'attache à l'exploration interactive, deux approches peuvent être identifiées : celles qui s'appuient sur un positionnement pré-calculé pour l'ensemble des données et celles qui utilisent un positionnement calculé lorsque les partitions sont représentées. Dans le premier cas, on retrouve notamment les travaux de Eades et Feng [33] déjà présentés plus haut. Dans la même veine, on retrouve également la technique introduite dans [131] qui s'intéresse à la représentation interactive de graphes petit monde en utilisant un zoom sémantique et géométrique basé sur la distance avec les éléments d'intérêt. On peut retrouver plusieurs techniques utilisant cette approche de distorsion des distances, généralement associée à une distorsion géométrique (on parle ainsi de l'effet *fish-eye*). Ainsi, dans [123], la technique est essentiellement géométrique mais dans d'autres cas elle peut également utiliser la topologie du graphe (technique de Gansner *et al.* [46] par exemple). Enfin dans [4], cette distorsion est associée à une représentation treemaps pour l'exploration de l'arbre de partitionnement et à un DNL pour la représentation de la donnée. Si on considère le second cas, il existe là aussi plusieurs exemples dans la littérature. Dans [69] la représentation est basée sur

le dessin d'un graphe quotient permettant d'*ouvrir* les méta-nœuds, c'est-à-dire de le remplacer par les éléments qu'il représente. La technique présentée dans [112] consiste en une amélioration des travaux de [128] permettant notamment l'ouverture et la fermeture des méta-nœuds en minimisant le déplacement des éléments non impactés par ces opérations. Dans [3], la technique présentée s'appuie sur le calcul d'agrégation hiérarchique permettant l'exploration interactive d'un graphe via un diagramme nœud-lien. La technique d'agrégation présentée est basée sur la limitation du nombre d'éléments contenus dans chaque agrégat. Ainsi, le contenu d'un méta-nœud peut-être représenté par un sous-graphe de même taille que celui de plus haut niveau. Dans [11], les auteurs décrivent une technique de représentation de graphes permettant la représentation de portions de graphe en fonction de leurs topologies.

Bien que souvent associée au DNL, cette approche se retrouve également dans d'autres techniques de représentation ou dans des techniques qui combinent des représentations pour permettre l'exploration multi-échelle. On peut ainsi en trouver des exemples avec la technique de Elmqvist *et al.* [36] qui utilise une représentation DOM associé à des glyphes pour compenser la perte d'informations due à l'abstraction de données. Ces glyphes peuvent par exemple représenter un histogramme ou encore les valeurs minimales, moyennes et maximales de la partition. On peut également trouver des approches multi-échelles pour les coordonnées parallèles. Par exemple, la technique de Fua *et al.* [42, 43] permet d'éditer la coupe de l'arbre de partitionnement et d'afficher différents niveaux d'abstraction des données. Des [132], la technique s'appuie sur une combinaison entre le positionnement radial de l'arbre et des coordonnées parallèles circulaires pour permettre la représentation multi-échelle des données. Enfin, la technique de Zhao *et al.* [150] s'appuie sur la combinaison entre treemap pour l'exploration de l'arbre de partition et les DNL pour l'analyse détaillée d'une partition.

4 Visualisation et infrastructure *Big Data*

Lorsqu'on considère la visualisation de données massives, plusieurs problèmes majeurs se posent et peuvent se résumer à ces trois questions : 1. Comment stocker les données lorsqu'elles dépassent les capacités de stockage d'un ordinateur classique ? 2. Comment parcourir les données en un temps raisonnable ? 3. Comment effectuer des calculs sur les données lorsqu'elles ne peuvent pas être contenues dans la mémoire vive de l'ordinateur ?Pour répondre à ces problématiques, différentes architectures spécialisées ont été mises au point. Ainsi, à partir des années 1960 apparaissent les *super-ordinateurs*, des machines conçues pour permettre des calculs hautes performances. Ces ordinateurs ont donc une infrastructure hautement spécialisée permettant d'obtenir des vitesses de calculs hors normes, et sont ainsi utilisés dans différents domaines scientifiques, notamment en météorologie ou en physique. Le principal inconvénient de ces infrastructures est le coût, s'élevant à plusieurs dizaines de millions d'euros. Bien que généralement issues de collaborations dans le but de partager cette ressource, il reste néanmoins très difficile d'accéder à ce type d'infrastructure. Plus récemment, au début des années 2000, apparaissent les infrastructures de type *Cloud* : des infrastructures de stockage et de calculs distantes et parallélisées, partagées entre utilisateurs ; on parle ainsi d'*Infrastructure en tant que Service (IaaS en anglais*). Comme l'explique Holliman et Watson [66] dans leur article, ces infrastructures présentent un réel intérêt pour la visualisation en temps réel de grande masses de données. Cela vient notamment du fait qu'elles sont à l'heure actuelle beaucoup plus simples d'accès, grâce à la démocratisation et la multiplication des entreprises proposant ce type de services et pour un coût beaucoup plus faible que celui des super-ordinateurs. Les travaux dans ce domaine restent cependant très limités, de part sa très récente émergence. Pourtant des entreprises proposent des services *cloud* et s'appuient sur le paradigme *MapReduce* pour permettre le calcul sur de grand jeux de données. Ce modèle de conception permet de distribuer des calculs entre plusieurs machines en découpant une tâche en un ensemble de sous tâches (partie *Map*) et en agrégeant les résultats (partie *Reduce*). L'un des environnements de développement les plus connus faisant usage de ce paradigme est *Hadoop* développé par Apache [41] mais il en existe d'autres.

En ce qui concerne les travaux sur la visualisation utilisant ce type d'environnement, on peut les classer en deux catégories. D'un côté, ceux utilisant des techniques de réduction de données afin de limiter les temps de transferts. De l'autre, ceux qui favorisent l'accès à la donnée brute et utilisent une approche incrémentale pour transférer progressivement les données et affiner la réponse au fur et à mesure des traitements.

4.1 Approches par réduction de données

L'approche par réduction de données est celle la plus décrite dans la littérature mais on peut la séparer en différents groupes. Tout d'abord, les approches généralistes, non dédiées à un certain type de données ou à une architecture matérielle particulière. Liu *et al.* [92] présentent un logiciel de visualisation de points d'intérêt sur une carte géographique. Ils utilisent une technique d'agrégation par *binning*, sélectionnent un représentant dans cet espace, lui attribuent une valeur de pondération et pré-calculent ensuite des *tuiles de données multi-variées* (construction basée sur les cubes de données). Ces données réduites seront ensuite intégralement transférées sur le client de visualisation et représentées en mettant à profit les capacités des processeurs graphiques pour paralléliser et accélérer l'accès aux données. Cette technique présente cependant l'inconvénient de créer des artefacts visuels ainsi que le transfert de l'ensemble des données après l'étape de réduction par binning provoquant une perte de détails si la réduction est importante.

Dans [77, 78] la technique utilise une approche d'agrégation orientée *visualisation* qui optimise les requêtes en fonction de la représentation utilisée. Elles sont ainsi ré-écrites afin d'obtenir uniquement les résultats correspondant à ce que la représentation peut fournir comme information, évitant ainsi le transfert de données inutiles. Cependant, ils ne s'intéressent qu'à des données temporelles et pour des représentations peu gourmandes en pixel par donnée (histogrammes, nuages de points et courbes). On peut également mentionner les travaux de Perrot *et al.* [104] qui présentent un système de visualisation avec infrastructure Cloud utilisant des fonctions de densité (carte de chaleurs plus précisément). Pour cela, ils s'appuient sur leur nouvel algorithme d'agrégation multi-échelle distribué, l'utilisation de l'environnement *Hadoop-HBase-Spark* et une visualisation sur client léger utilisant les technologies WebGL (une version d'OpenGL dédié aux navigateurs). Ils arrivent ainsi à représenter des jeux de données de plus de 2 milliards d'éléments.

Enfin, Jonker *et al.* [76] présentent une technique pour la représentation de grands graphes de données relationnels se basant sur une architecture similaire. Leur approche calcule le positionnement des nœuds du graphe coté serveur et découpe le résultat en tuiles de données qui seront transférées et visualisées sur un client léger. De cette manière, ils effectuent toutes les opérations à distance, et ne transfèrent que les informations nécessaires à l'utilisateur. Enfin, la technique présentée dans [22] consiste en une plate-forme dédiée à la visualisation interactive de grands graphes utilisant une architecture de type *cloud* organisée en trois parties : le client de visualisation, le centre de calcul et la base de données. Les données sont intégralement pré-calculées et stockées sous forme d'objets géométriques utilisant un R-tree, une structure optimisée pour l'indexation de données spatiales.

D'autres ont orienté leurs travaux en utilisant des techniques dédiées aux

données à analyser. On retrouve généralement des techniques dédiées aux données géographiques (des coordonnées par exemple) et dans certains cas, avec une dimension temporelle associée. Ainsi, Lins *et al.* [91] décrivent une structure de données basée sur le *cube de données* pour la visualisation de données spatio-temporelles multi-variées. Le côté compact de cette structure de données la rend particulièrement intéressante pour les requêtes d'agrégation sur de grandes bases de données. La technique se base également sur le pré-calcul et le stockage de l'ensemble des entités afin de permettre un accès rapide aux données.

Dans [34], les auteurs utilisent le paradigme *MapReduce* pour l'exploration de données spatio-temporelles et utilisent un environnement de développement dédié à ce type de données, spatialHADOOP [35]. Leur système est séparé en deux parties avec l'interface des données (nettoyage, indexation et agrégation en deux parties, données temporelles et données spatiales) et l'interface utilisateur (gestion des requêtes et représentation par carte de chaleur). Ce système est cependant conçu pour la représentation de données issues d'une base spécifique, le *LP DAAC* (*Land Process Distributed Active Archive Center* en anglais), fournie par la *NASA* et intègre les processus de nettoyage des données obtenues de cette base.

Parmi ces différents travaux, certains utilisent des techniques bien spécifiques permettant d'accélérer les temps d'indexation, de recherche et d'accès aux données. Par exemple, les travaux présentés dans [92, 116] ont utilisé des bibliothèques d'indexation par *bitmap* tandis que dans [91] c'est une technique inspirée des cubes de données qui est utilisée.

4.2 Approches incrémentales

L'approche incrémentale est basé sur le principe que, lorsqu'on analyse des données massives, les temps de calculs sont trop longs pour effectuer une requête sur l'ensemble du jeu de données. La technique consiste ainsi à fournir des résultats partiels et à affiner la réponse, c'est-à-dire augmenter la précision de la réponse, au fur et à mesure que le calcul progresse. Les travaux de Fisher *et al.* [40] utilisent ce type d'approche permettant d'obtenir des résultats partiels. Ils fournissent également à l'utilisateur un degré de confiance des résultats obtenus afin d'indiquer à quel point l'échantillon de données sur lequel est basé le résultat courant est représentatif du jeu de données complet. Im *et al.* [70] présentent un système de visualisation, *VisReduce*, qui utilise une adaptation du modèle *MapReduce*. Leur approche consiste en une modification du paradigme *MapReduce* afin de pouvoir retourner des résultats intermédiaires. Ces résultats intermédiaires sont transmis indépendamment et permettent d'obtenir une réponse partielle à la requête. Ainsi, la représentation est complétée et son exactitude augmente au fur et à mesure que les réponses sont récupérées.

Enfin, d'autres utilisent une architecture matérielle spécialisée permettant

d'obtenir de très bonnes performances. La technique présentée dans [116] pour l'exploration de données massives avec des coordonnées parallèles utilise un super-ordinateur (on parle de calculs hautes performances), associé à une bibliothèque d'indexation des données en bitmap. Il existe bien d'autres travaux utilisant cette approche mais la spécificité du matériel les rend hors de notre champ de recherche donc nous ne développerons pas plus ces techniques.

Chapitre II

La relation dans les représentations matricielles : conception et évaluation

Avant même de considérer les abstractions de données relationnelles dans le cadre de données massives, il nous semble primordial de considérer le problème depuis ses origines : qu'est ce qu'une donnée relationnelle ? Comment la notion de relation entre des entités est-elle transmise visuellement ? Quelles sont les techniques utilisées et décrites dans la littérature ? Sont-elles efficaces ? et si oui, pourquoi ? Dans ce chapitre nous tenterons de déterminer, laquelle des deux techniques les plus connues et les plus utilisées, est la plus efficace, dans quelles conditions et ce qui lui confère cette efficacité. Une partie de ces travaux a déjà été publiée et présentée durant une conférence internationale [120].

1 Introduction - État de l'art

L'analyse visuelle des relations de type réseaux sociaux est devenue de plus en plus populaire. Que ce soit en biologie, en épidémiologie, en sociologie ou en marketing par exemple, les besoins en techniques et outils de visualisation efficaces sont croissants et déterminer l'existence d'un lien entre des entités et/ou des groupes est devenu une tâche fréquente. Une des façons de modéliser de telles relations consiste à utiliser le modèle graphe dans lequel les entités sont représentées par des nœuds et les relations entre entités par des arêtes entre les nœuds correspondants. La suite de cette section est un rappel de l'état de l'art des techniques de représentations de données relationnelles et des comparaisons de certaines de ces techniques. la section 1.3 présente les contributions ainsi que l'organisation générale du chapitre.

1.1 Techniques de représentations

Les approches les plus fréquentes pour explorer visuellement de telles données relationnelles font usage soit des *Diagrammes nœuds-Liens* (que nous appellerons DNL(s)), soit des *Diagrammes Orientés Matrices* (DOM(s) Dans un DNL, chaque entité (nœud) est représentée par une forme géométrique (appelée glyphe), tandis qu'une relation entre deux nœuds est représentée par une ligne (un lien) reliant les deux glyphes. Cette ligne peut-être droite ou brisée, c'est-à-dire constituée d'une suite de segments joints. On parle alors de ligne polygonale, ou plus simplement de polyligne. Les DOM quant à eux sont des matrices carrées dans lesquelles les entités sont représentées deux fois, en abscisse et en ordonnée. À chaque ligne et colonne, correspond une et une seule entité et les relations entre entités sont représentées par des glyphes aux intersections des lignes et colonnes des entités concernées.

De manière générale, l'amélioration de ces deux diagrammes (DNL et DOM) s'effectue par l'optimisation de critères esthétiques, certains étant communs aux deux techniques tandis que d'autres sont spécifiques à l'une ou à l'autre.

Diagrammes nœud-lien

La communauté du dessin de graphe par exemple se concentre sur la création d'algorithmes de dessin efficaces, par exemple la réduction du nombre de croisements d'arêtes ou du nombre maximal de brisures par arêtes. Ces deux derniers paramètres ont été déterminés comme étant parmi les critères les plus importants pour l'amélioration des DNL [107, 139]. Il existe différentes méthodes de dessin de DNL, certaines sont spécialisées pour la représentation de classes de graphes spécifiques (arbre, DAG, *etc.*) tandis que d'autres sont plus générales. On peut ainsi citer des exemples d'algorithmes par modèle de forces [44, 53, 88] qui s'appuient sur l'équilibrage de forces inspirées de modèles physiques pour positionner les nœuds et arêtes les uns par rapport aux autres. D'autres algorithmes utilisent l'algèbre linéaire [55, 83] ou des modèles de contraintes (*constraint-based*) [31, 32]. Une autre approche consiste à diviser le graphe en sous-graphe plus simples à représenter, on parle alors de décomposition topologique du graphe [9, 127]. D'autres encore tentent d'optimiser l'occupation de l'espace en utilisant par exemple des courbe dîtes *space-filling* (courbe de Peano par exemple) [13, 97]. L'optimisation de tels critères reste cependant complexe ; réduire les croisements d'arêtes a ainsi été prouvé comme étant un problème NP-complet [33, 49] et est donc généralement résolu par des approches heuristiques.

Diagrammes orientés matrices

Pour les DOM, l'amélioration de l'ordre des nœuds correspond au problème d'arrangement linéaire minimum connu pour être un problème NP-difficile [48].

La communauté s'attache, là aussi, à la création de méthodes heuristiques, avec par exemple les algorithmes de minimums locaux (hill-climbing heuristics) [82, 115], les approches spectrales (spectral sequencing) [80, 84], une combinaison des deux [105] ou encore de l'ordonnancement partiel inspiré des techniques de partitionnement [23, 130]. Dans [98] les auteurs ont comparé les effets de différents algorithmes d'ordonnancement des nœuds sur la lisibilité des DOM pour des graphes relativement grands (jusqu'à 1000 nœuds). Ils en déduisent que l'un des aspects esthétiques majeurs pour les DOM consiste à rapprocher autant que possible dans la représentation les entités proches en termes de distance dans le graphe. D'autres approches tentent de minimiser les croisements d'arêtes en s'appuyant sur des approches heuristiques [18].

Techniques hybrides

D'autres encore, au lieu de s'intéresser à l'amélioration et l'optimisation d'une technique particulière, se sont plutôt orientés sur des méthodes *hybrides*, qui essaient de tirer le meilleur des deux représentations en les combinant (voir figure II.1). Shen et Ma [125] représentent les chemins entre entités sur les DOM en s'appuyant sur la représentation DNL des connexions. Ils utilisent des polylignes orthogonales apposées sur la matrice pour représenter les chemins possibles entre deux nœuds. La figure II.1a montre les chemins possibles entre les deux nœuds *input* et *output* grâce à des polylignes orthogonales colorées, chaque couleur représentant un chemin différent. La représentation *matlink* introduite dans [60] utilise des arcs semi-transparents pour connecter les nœuds dans une matrice d'adjacence. Cela permet notamment d'accentuer le degré de chaque sommet ainsi que la répartition des nœuds qui lui sont adjacents (voir figure II.1b). Un nœud avec un fort degré aura donc beaucoup de courbes pour le connecter à ses voisins ; un amas d'arcs très blancs permet donc d'identifier les nœuds les plus connectés. Les mêmes auteurs présentent aussi dans *Nodetrix* [61] une nouvelle approche pour mettre en avant les communautés dans un réseau et en favoriser l'exploration. Ils utilisent plusieurs DOM afin de représenter les communautés et les connectent à l'aide de *liens* propres à la représentation NLD. Il en résulte un DNL dont les *nœuds* sont des représentations matricielles de communauté (voir figure II.1c).

1.2 Comparaisons des techniques

Les deux techniques DOM et DNL sont des représentations fondamentalement différentes et pour autant véhiculent la même information. Certains travaux [15, 51, 60, 81] ont donc comparé expérimentalement ces deux représentations afin de déterminer quand favoriser l'une ou l'autre. Dans [51] les auteurs comparent les DNL et les DOM en utilisant, pour les premiers un algorithme par modèle de forces pour le positionnement des nœuds et un

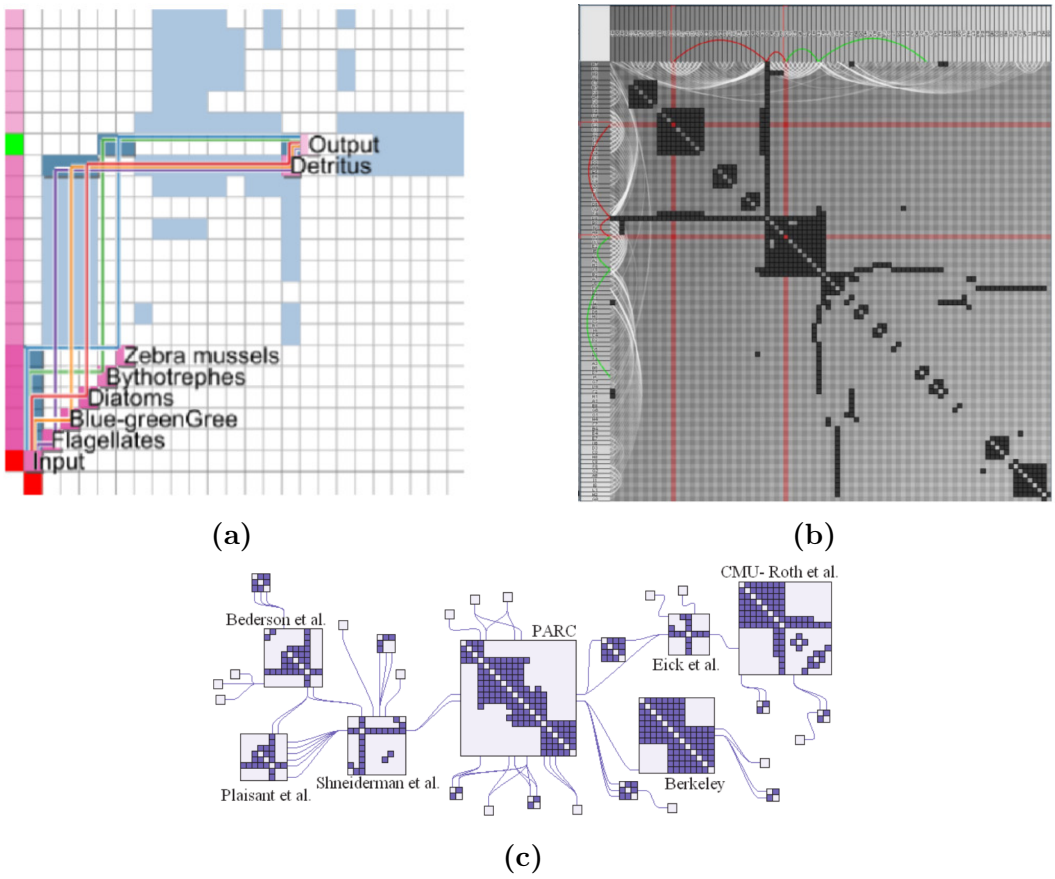


Figure II.1: Exemples de techniques hybrides de représentation de graphe. (a) technique de [125], (b) Matlink présenté par [60] et (c) Nodetrix introduit dans [61]

ordonnancement alphabétique des entités pour les seconds. Si on considère des graphes grands et denses, leur évaluation utilisateur a montré que les DOM étaient plus performants que les DNL pour presque toutes les tâches testées, c'est-à-dire : estimer le nombre de nœuds et d'arêtes, trouver un nœud, le nœud le plus connecté ou le voisin d'un nœud. Les DNL ont été montrés comme statistiquement plus performants que les DOM lorsqu'on considère une tâche de recherche de chemin entre deux entités ou lorsqu'il s'agit de représenter de petits graphes ou des graphes peu denses. Keller *et al.* [81] ont confirmé ces résultats en comparant les DNL et les DOM avec des paramètres expérimentaux proches. Dans [60] les auteurs présentent une évaluation utilisateur de trois représentation :*matlink*, une représentation hybride décrite dans l'article, et les DOM et DNL. Ils confirment ainsi que les représentations matricielles sont plus efficaces que les représentations nœuds-liens pour des tâches simples (dites bas-niveaux). Ils obtiennent également de meilleures performances globales avec la représentation hybride qu'avec une matrice traditionnelle pour des

tâches liées à de la recherche de chemins. Les représentations nœuds-liens restent cependant plus efficaces pour les tâches nécessitant une vue générale des données. Bae et Watson [15] ont de leurs cotés comparé dans une tâche de recherche de chemin la représentation *Quilt* sous ses différentes formes et également avec les représentations nœuds-liens et matricielle. Ils montrent ainsi que dans le cadre de leur évaluation, les *Quilts* sont plus adaptés que les deux autres représentations pour de la recherche de chemins dans un graphe orienté acyclique. Ils confirment aussi partiellement les résultats des deux évaluations précédentes indiquant que les DOM étaient moins efficaces que les DNL pour de la recherche de chemin. Alper *et al.* [6] présentent une évaluation entre DOM et DNL pour la visualisation et la comparaison de graphes pondérés. Ils effectuent une évaluation expérimentale montrant que les DOM sont plus efficaces que les DNL pour la représentation de la pondération et pour la comparaison de deux états d'un graphe. Ces études apportent des réponses intéressantes pour l'amélioration des performances utilisateur, cependant il est difficile de tirer une conclusion franche sur la supériorité d'une technique de représentation en particulier. En effet, la façon dont les entités et les relations sont visuellement encodées dans les DNL et les DOM introduit une limitation dans de telles études puisqu'aucun algorithme de dessin des entités/relations n'est commun aux deux techniques. La position des nœuds et leur ordre induisent une variation dans la qualité des deux représentations (par exemple le nombre de croisements d'arêtes ou la longueur moyenne/maximale des arêtes) et ces variations peuvent véritablement influencer les résultats expérimentaux et l'analyse qui en sera faite.

1.3 Contribution

Notre contribution consiste à proposer trois encodages visuels des arêtes afin de progressivement transformer l'encodage visuel des arêtes de DOM en celui de DNL tout en préservant l'encodage visuel des nœuds ainsi que leur ordre. Deux évaluations expérimentales ont été conduites afin de comparer l'efficacité de l'encodage visuel des arêtes pour les DOM et les DNL tout en évitant le biais du positionnement/ordre des nœuds. La première a pour objectif d'évaluer ces encodages visuels sur les 4 tâches simples, introduites dans d'autres évaluations [50, 81] : identification du nœud de plus fort degré, identification d'une arête reliant deux nœuds, identification d'un voisin commun à deux nœuds et estimation de la densité d'un graphe. La seconde a le même objectif mais dans le cas d'une tâche complexe plus proche de l'utilisation réelle d'une représentation de graphe : la recherche d'un chemin entre deux nœuds d'un graphe. À notre connaissance, ces évaluations sont les premières à permettre la comparaison des DOM et des DNL tout en évitant le biais inhérent au choix de l'algorithme de dessin des entités pour les DNL et d'ordonnancement pour les DOM.

La suite de ce chapitre sera donc structurée comme suit : en section 2, nous introduirons tout d'abord les trois évolutions de l'encodage visuel des arêtes dans les DOM ainsi que nos hypothèses de départ en section 3. Ensuite nous présenterons en section 4 l'organisation et la mise en place des évaluations expérimentales afin de comparer l'efficacité de ces trois encodages pour les différentes tâches envisagées. Les sections 5 et 6 fourniront les détails spécifiques à chaque évaluation ainsi que les résultats obtenus et l'analyse que nous en faisons. Nous discuterons ensuite de façon globale les résultats obtenus dans la section 7. Finalement nous présenterons nos conclusions en section 8.

2 Modification de la représentation matricielle

Dans les DOM, une relation entre deux entités est traditionnellement représentée par un glyphe, généralement un carré ou un cercle, à l'intersection des lignes et colonnes correspondant aux deux entités. Dans *NodeTrix* [61] les auteurs considèrent que ces lignes et colonnes font partie de l'encodage visuel des entités, tandis qu'une arête est simplement représentée par un glyphe à leur intersection. Nous émettons ici l'hypothèse inverse et considérons que la ligne et colonne font au contraire partie de l'encodage visuel des arêtes, c'est-à-dire qu'une arête est représentée par deux rectangles (un pour chaque ligne et colonne) ainsi que par un glyphe à leur intersection. Le fait que ce processus ressemble fortement au suivi d'un lien dans un DNL où l'œil suit la polyligne qui représente l'arête pour trouver l'entité à son extrémité conforte cette hypothèse. L'idée principale consiste à réduire autant que possible l'espace nécessaire pour encoder les arêtes et par conséquent à réduire la complexité visuelle. Pour cela nous proposons trois variations de l'encodage visuel des arêtes pour transformer progressivement l'encodage traditionnel des DOM en encodage *lien* grâce à différentes étapes de simplification (figures II.2 à II.5).

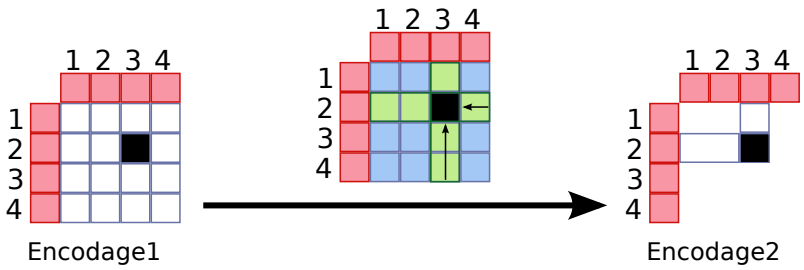


FIGURE II.2: 1re étape de réduction de l'encodage visuel des arêtes : Les rectangles de l'encodage visuel sont raccourcis en leurs parties essentielles, c'est-à-dire jusqu'à la position de leurs intersections lorsqu'il y a une arête, sinon sont retirés de la représentation.

Pour la première transformation, nous retirons les lignes et colonnes non utilisées et réduisons au maximum les longueurs des deux rectangles représentant

une connexion. La longueur des rectangles n'est donc plus égale à la taille de la matrice mais s'arrête à l'intersection de la ligne et colonne de l'arête. On peut voir sur la figure II.2 que la taille des rectangles verts est réduite. Dans un DOM traditionnel, la grille formée par les intersections des rectangles contient des cellules/cases vides (en dehors des matrices de graphes complets) ce qui fournit une information de type *absence de relation* entre les entités correspondantes. En réduisant la taille des rectangles en leurs parties essentielles, nous éliminons une partie de ces cases : le retrait des rectangles ne contenant aucun glyphe et le raccourcissement de ceux qui en contiennent réduit le nombre de traits et par conséquent réduit la complexité visuelle.

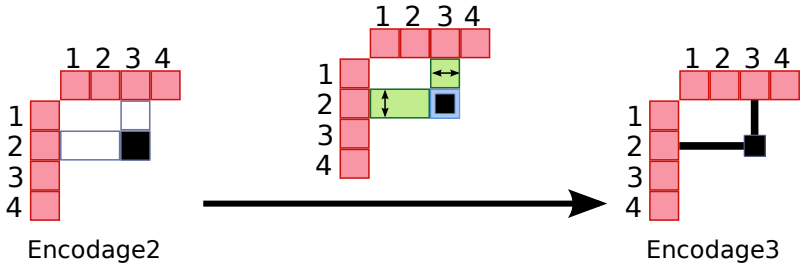


FIGURE II.3: 2e étape de réduction de l'encodage visuel des arêtes : Les rectangles (en vert) sont réduits jusqu'à devenir une simple polyligne et le glyphe (en bleu) est réduit pour faciliter l'interprétation.

Pour la deuxième transformation nous remplaçons les deux rectangles par une polyligne avec une brisure orthogonale à leur intersection (voir figure II.3). Pour permettre la différenciation entre les brisures, représentant les connexions, et les croisements, nous utilisons un glyphe, tel qu'utilisé par Kornaropoulos et Tollis [85] dans la technique de représentation *DAGview*. Cette technique utilise un positionnement orthogonal des nœuds et des arêtes d'un DNL. Les nœuds sont positionnés sur une matrice en utilisant deux ordres différents (un pour chaque axe) évitant ainsi que les entités ne se retrouvent systématiquement sur la diagonale de la matrice. Les arêtes suivent un chemin orthogonal et sont marquées par un glyphe rond servant à distinguer les croisements de liens des brisures de polylignes. Les entités sont donc représentées une seule et unique fois et les arêtes forment une grille très semblable à une matrice d'adjacence.

Le glyphe précédemment utilisé pour symboliser l'existence d'une arête est conservé et sa taille est réduite afin de maintenir un espacement vide tout autour permettant la distinction de la polyligne entre deux glyphes contigus. Cette réduction est nécessaire lorsqu'on considère une portion importante du graphe fortement connectée, une clique par exemple, où des glyphes non réduits constituent un bloc unique et réduisent fortement la lisibilité (voir figure II.4).

La dernière transformation consiste à retirer le glyphe indiquant l'existence d'une connexion et à représenter la brisure par une courbure de la polyligne (voir figure II.5). La courbure est nécessaire afin d'empêcher toute ambiguïté

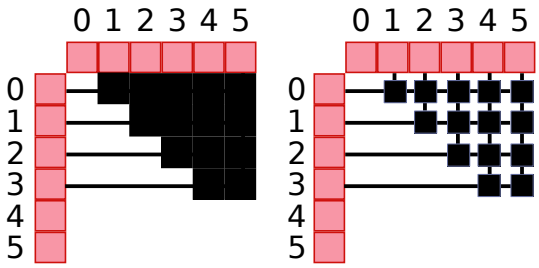


FIGURE II.4: La réduction de la taille du glyphe permet de maintenir un espace entre les glyphes et donc de suivre les polylignes entre glyphes contigus.

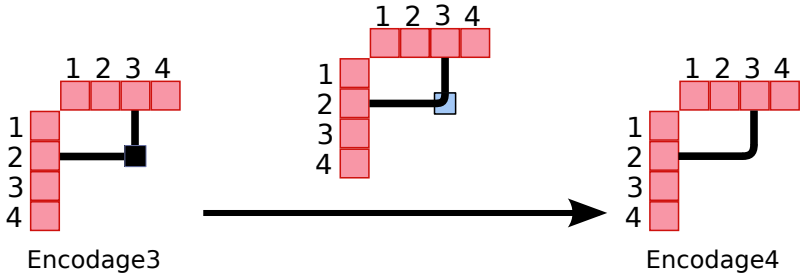


FIGURE II.5: 3e étape de réduction de l'encodage visuel des arêtes. Les arêtes sont représentées par un encodage *Liens* tel que présent dans les DNLs traditionnels.

avec les croisements d'arêtes (voir figure II.6). Ce dernier encodage visuel a déjà été mentionné dans [61] mais comme un potentiel état transitoire lors d'une transformation animée d'un DOM en DNL.

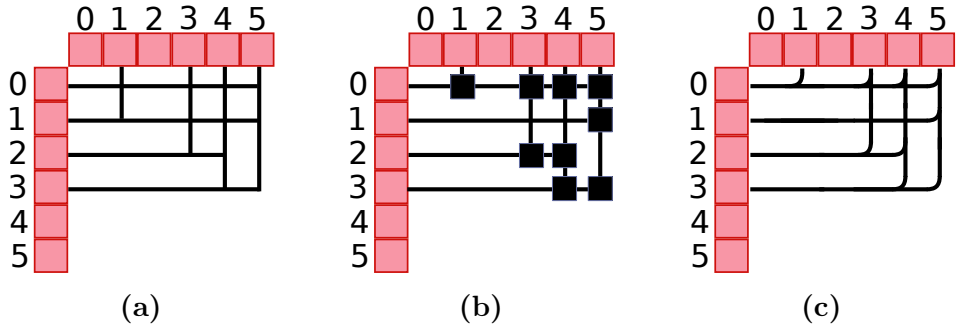
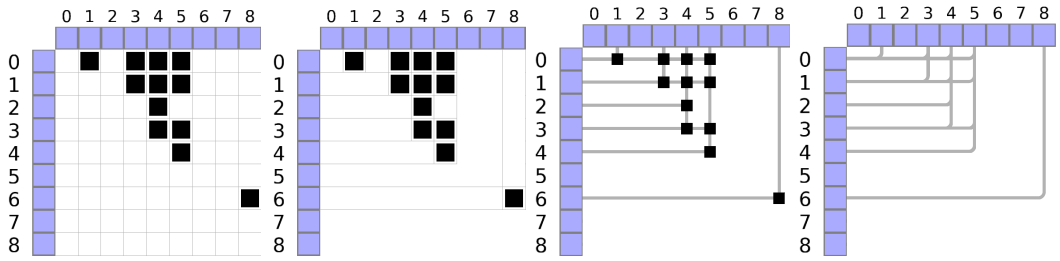


FIGURE II.6: Distinction entre arêtes et croisements d'arêtes sans forme distinctive (a), en utilisant un glyphe (b) ou avec une courbure de la polyligne (c).

On peut remarquer que les quatre variations (voir figure II.7) sont visuellement semblables puisque l'encodage visuel des entités a été conservé à chaque étape de transformation, seules les arêtes évoluent. Cependant, on peut d'ores

et déjà considérer les deux derniers encodages comme étant des encodages nœuds-Liens puisque les arêtes sont caractérisées par des polylignes reliant des entités connectées. Cette constatation est le point de départ de la comparaison de l'encodage visuel des arêtes entre DOM et DNL sans biais de positionnement des entités. Afin de faciliter l'identification des différentes variations, nous nous référerons à l'encodage traditionnel comme Encodage 1 et aux différentes variations respectivement par Encodage 2, Encodage 3 et Encodage 4.



(a) encodage 1 **(b) encodage** 2 **(c) encodage** 3 **(d) encodage** 4

Figure II.7: Simplification de l'encodage visuel des arêtes : en trois étapes, il est possible de progressivement transformer l'encodage visuel d'un DOM en encodage visuel d'un DNL. Ainsi on peut considérer avec les encodage 3 **et** 4 **que les entités sont reliées par des liens si elles sont connectées, tout comme pour un DNL traditionnel, la différence résidant dans la duplication des entités et la présence, pour l'encodage** 3 **d'un glyphe supplémentaire.**

3 Hypothèses préliminaires

Notre hypothèse préliminaire est qu'il est possible d'améliorer les performances utilisateurs pour diverses tâches, quelles soient simples ou complexes, en réduisant l'encombrement visuel et en simplifiant les règles d'interprétation par l'utilisation d'un encodage d'arête hybride. L'un des résultats attendus est que l'Encodage 3 induira une différence majeure dans les performances utilisateurs. Cette hypothèse est due au fait qu'il semble plus simple de suivre une simple ligne comme pour les encodages 3 et 4, que d'essayer de suivre un chemin lorsque celui-ci n'est distinguable que par ses contours (les contours des rectangles). On pourrait appliquer ce raisonnement pour l'Encodage 4, cependant la distinction entre les courbures et les croisements d'arêtes parait moins aisée avec cet encodage-ci. La présence d'un glyphe pour signifier l'existence d'une brisure ajoute une discontinuité visuelle dans la polyligne, facilitant cette distinction. Ainsi, il semble logique de s'attendre à des résultats moins bons lors de l'utilisation de cet encodage, par exemple avec un taux d'erreur plus important. Pour synthétiser, un ordre de performance croissant peut-être

donné : Encodage 4, Encodage 1, Encodage 2 et Encodage 3.

4 Éléments de protocole communs

Nous avons construit nos protocoles d'évaluation en tâchant de suivre les recommandations de Purchase [109] sur la conduite d'évaluations expérimentales. Pour cela nous avons organisé nos deux évaluations comme suit :

1. Introduction de l'évaluation aux participants : présentation des termes de l'expérience (anonymat, exploitation des résultats, etc),
2. Présentation de la ou des tâche(s) et de l'interface d'évaluation : tout d'abord sur support statique (papier) puis sur support interactif (interface de présentation similaire à celle de l'évaluation) avec essais multiples et explication par une personne référente,
3. Évaluation expérimentale, sans interaction entre les participants et le référent, utilisée pour le recueil des résultats quantitatifs,
4. Questionnaire anonyme que le participant remplit sans intervention du surveillant sur ses préférences et retours d'expériences notamment, utilisé pour le recueil des résultats qualitatifs, et
5. Entretien avec le participant : retour d'expérience global, avis et remarques.

La phase expérimentale (phase 3) constitue la phase de recueil des données quantitatives de l'évaluation. Elle est divisée en *tâches expérimentales* correspondant aux questions auxquelles doit répondre l'évaluateur. Chaque tâche expérimentale est divisée en plusieurs *essais*, correspondant aux différentes combinaisons des variables évaluées.

Comme présenté dans [109], lors d'une évaluation utilisateur, les participants suivent une courbe de progression similaire à celle présentée en figure II.8.

Malgré une présentation de l'évaluation durant laquelle nous avons introduit et détaillé l'interface ainsi que les tâches à accomplir sa capacité à interagir avec l'interface et sa compréhension des mécanismes à adopter ne sont pas encore optimales. On décompose la courbe de performance des utilisateurs en trois parties :

— La phase de montée en compétence (en rouge),

— La phase d'optimum de performance (en bleu), et

— La phase de baisse de l'attention (en vert).

L'objectif est de parvenir à recueillir les données expérimentales lorsque l'utilisateur est dans la phase d'optimum. Si on utilise les données dès le début de l'évaluation, durant la montée en compétence, les performances ne seront pas optimales et l'utilisateur continuera à progresser, induisant une variation de ses résultats entre le début et la fin de l'évaluation. La solution proposée dans

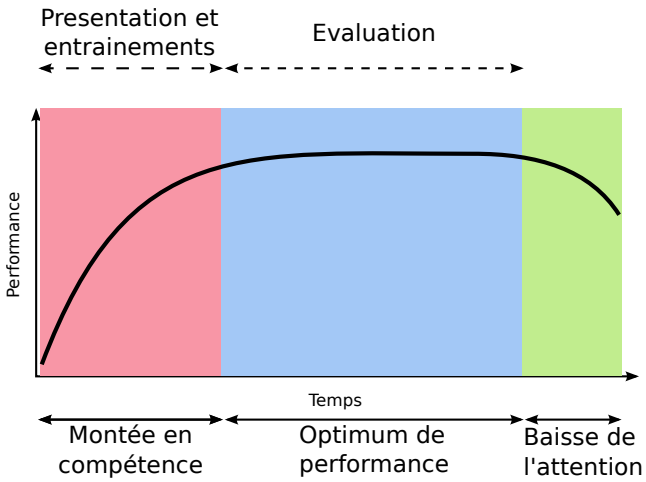


FIGURE II.8: Illustration de la courbe de performance des participants durant une évaluation. La présentation des tâches et de l'évaluation initie la montée en compétence. Afin d'obtenir les meilleurs résultats possible pour chaque participant, les essais de la phase expérimentale doivent se produire durant l'optimum de performance, et avant d'entrer dans la phase de baisse de l'attention.

[109] consiste à séparer l'évaluation expérimentale en deux phases : l'une dite phase d'entraînement, l'autre étant la phase expérimentale à proprement parlé. Du point de vue des participants, la phase d'entraînement est indistincte de la phase expérimentale. Les essais de la phase d'entraînement sont les premiers à apparaître et sont les mêmes pour tous les participants. Ils servent à s'assurer que l'utilisateur a correctement pris en main l'interface et à initier sa montée en compétence. L'objectif est de maximiser le nombre des résultats de la phase expérimentale obtenus durant l'optimum de performance.

Il existe principalement deux méthodes d'ordonnancement des essais lors d'une évaluation : par blocs ou non. Lors d'une organisation par blocs de tâches, on propose à l'utilisateur de compléter d'abord tous les essais d'une tâche avant de changer de tâche expérimentale. Dans la seconde organisation au contraire, on répartit uniformément les essais de chaque tâche sur l'ensemble de l'évaluation.

Dans [110], l'auteur préconise de porter attention sur l'effet d'apprentissage, qui induit une amélioration des performances utilisateurs entre le début et la fin d'une évaluation. Cet effet concerne les deux méthodes mais nous pensons que la répartition des essais sur l'ensemble de l'évaluation ralentira cet effet d'apprentissage et avons donc choisi la seconde méthode.

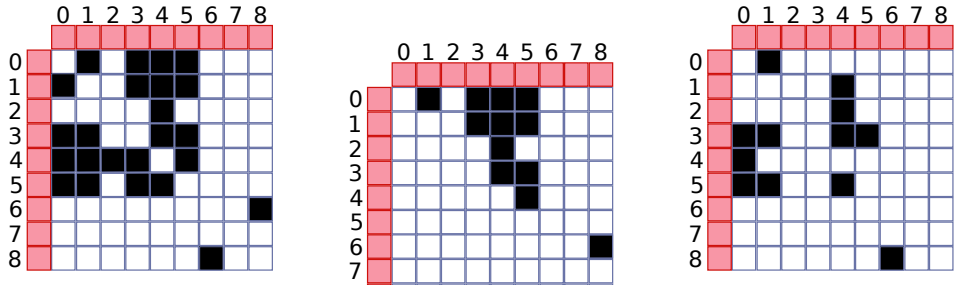
Une limite de temps pour compléter chaque essai a aussi été imposée aux participants. Le temps restant est affiché dans le coin supérieur droit de l'interface. Cette limite sert à la fois à encourager les participants à compléter les essais aussi vite que possible mais aussi à empêcher une éventuelle trans-

formation de la tâche à accomplir. Lors d'une tâche de recherche de chemin par exemple, l'absence de limite temporelle permettrait de chercher un plus court chemin ce qui pourrait influencer les résultats. Afin de permettre aux participants de se reposer, de courtes pauses séparant les essais et des pauses sans limites de temps sont proposées de façon régulière.

4.1 Présentation générale des deux évaluations

4.2 Orientation

Dans nos évaluations nous considérons la visualisation de graphes non orientés ; ceux-ci sont généralement représentés avec les DOM de deux manières : soit avec un double affichage où chaque arête est dupliquée, ce qui résulte en une matrice symétrique (voir figure II.9a), soit avec un affichage simple où les arêtes sont représentées une et une seule fois, et qui résulte en une demi-matrice (voir figure II.9b). Il existe une troisième façon de positionner les arêtes dans la matrice (voir figure II.9c), avec un simple affichage des arêtes mais réparties sur les deux demi-matrices. Cette méthode est généralement utilisée dans le cas de graphes orientés. Étant donné que nous utiliserons des graphes non orientés et que cette répartition peut diminuer la visibilité des communautés, nous ne l'utiliserons pas. Les évaluations pilotes ont montré qu'un double affichage des arêtes, malgré son efficacité à mettre en avant les communautés présentes dans le graphe, augmentait l'encombrement réduisant ainsi la lisibilité de la matrice. Par conséquent, nous avons choisi un affichage simple en une seule demi-matrice telle que présentée sur la figure II.9b.



(a) matrice complète **(b) demi-matrice** **(c) demi-matrice orientée**

FIGURE II.9: Effet de la duplication des arêtes sur la lisibilité de la matrice

4.3 Programme d'évaluation et implémentation

L'interface d'évaluation est constituée d'un affichage interactif dans un navigateur internet en mode *plein écran* (utilisant l'un des quatre encodages

visuels précédemment décrits). Un rappel de la question expérimentale est affiché dans un bandeau en haut de l'interface d'évaluation et, en-dessous, une vue interactive de la matrice ainsi que deux boutons. Le premier bouton (placé à gauche) permet de centrer la représentation dans l'écran et le second (à droite) permet de valider la tâche (voir figure II.10).

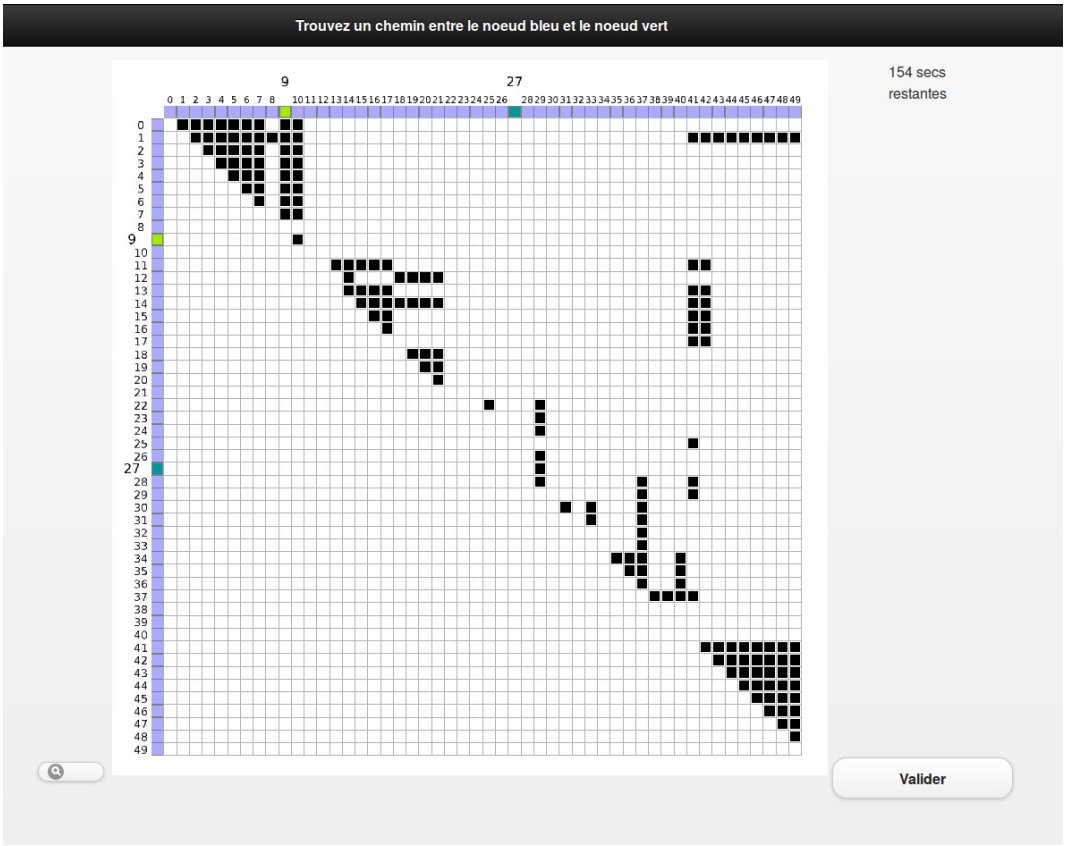


Figure II.10: Capture d'écran de l'interface d'évaluation.

Trois outils d'interaction ont été ajoutés pour effectuer la tâche : le premier est un outil classique d'agrandissement/rétrécissement et de déplacement (*zoom-and-pan* en anglais). Le deuxième est un outil de sélection permettant de répondre aux questions que nous détaillerons en sections 5.1 et 6.1. Le troisième est un outil de mise en surbrillance d'une ligne et/ou colonne lorsque le participant survole un nœud ou une arête avec la souris. Ces outils d'interaction permettront une interaction très limitée, l'objectif étant de minimiser le biais des interacteurs dans les résultats obtenus.

5 Évaluation dans le cas de tâches basiques

5.1 Protocole

Description des tâches

Les tâches pour cette évaluation sont au nombre de quatre et sont les suivantes :

tâche 1 : Estimation de la densité d'un graphe (voir définition 24) de façon approximative (peu, moyennement ou très dense) ;

tâche 2 : Identification du nœud de plus fort degré, c'est-à-dire ayant le plus grand nombre de voisins ;

tâche 3 : Identification d'une arête reliant deux nœuds ;

tâche 4 : Identification d'un voisin commun à deux nœuds.

Afin de répondre à ces tâches, les utilisateurs ont accès à un ou plusieurs outils d'interaction suivant la tâche en cours (voir tableau II.1).

Tableau II.1: Tâches et outils d'interaction disponibles.

	tâche1	tâche2	tâche3	tâche4
Zoom et déplacement	✓	✓	✓	✓
Sélection de nœud		✓		✓
Sélection d'arête			✓	
Mise en surbrillance		✓		✓

Les tests pilotes ont révélé que 45 secondes maximum par essai était un bon compromis permettant de fournir aux participants suffisamment de temps pour compléter la majeure partie des essais tout en maintenant acceptable le temps total requis pour effectuer l'évaluation.

Jeux de données

Dans cette évaluation, nous voulons évaluer des graphes de différentes catégories allant de petit et peu dense à grand et dense. Pour cela, nous établissons trois tailles de graphes (20, 50 ou 100 nœuds) et trois densités (20, 40 ou 60%). Nous avons ainsi généré un graphe pour chaque combinaison de taille/densité possible, soit 9 au total, à l'aide d'un générateur aléatoire de graphe simple nécessitant deux paramètres : le nombre de nœuds et le nombre d'arêtes. Cet algorithme sélectionne aléatoirement deux nœuds du graphe et crée une arête entre ces deux nœuds tout en s'assurant qu'il n'en existe pas encore. Chacun de ces graphes associé aux 4 encodages visuels ne seront utilisés qu'une fois pour chacune des tâches afin de limiter le temps total de l'évaluation. Afin de déterminer l'ordre des nœuds dans la matrice, nous

utilisons l'identifiant des nœuds, attribué suivant leur ordre de création par l'algorithme de génération.

Population

Les participants à l'évaluation sont des étudiants de l'enseignement supérieur (licence, master ou doctorat), enseignants-chercheurs ou ingénieur de recherche dans le domaine de l'informatique. Chaque participant avait une certaine connaissance des notions de graphes, représentations nœuds-liens et matricielles mais ne manipulent pas ces concepts et outils de façon régulière. Ces notions sont d'ailleurs ré-introduites durant la phase de présentation afin de s'assurer de la bonne compréhension des participants. Au total, 30 personnes ont participé à l'évaluation dont 6 pour les tests pilotes et 24 pour l'évaluation expérimentale.

5.2 Résultats quantitatifs

Les résultats quantitatifs permettent de définir statistiquement si les performances des participants dépendent de l'encodage visuel des arêtes, et ce en prenant en compte différents niveaux de précision dans l'analyse si nécessaire. Nous avons ainsi observé deux paramètres : le taux d'erreur ainsi que le temps de réponse. Les résultats obtenus sont de tailles variables (retrait des essais non validés dans le temps imparti) et ne suivent pas une loi de distribution normale. Nous devons donc utiliser deux tests de significativité non-paramétriques afin d'évaluer la robustesse [1] des résultats obtenus. Le *test de Friedman* permet d'évaluer les résultats pour les valeurs de taux d'erreur (données complètes) et le *test de Kruskal-Wallis* pour les temps de réponse. Nous avons utilisé un seuil de significativité α standard fixé à 0.05 afin de déterminer si les différences observées sont significatives ou non. Ainsi, une valeur-p (*p-value* en anglais) inférieure au seuil indique que les différences sont significatives, tandis qu'une valeur supérieure révèle l'absence de significativité.

Lorsque le test indique une valeur-p inférieure au seuil, nous effectuons un *test de Mann-Whithney-Wilcoxon* pour effectuer une comparaison deux-à-deux des groupes testés. Ce seuil est sujet à variation par application d'une *correction de Bonferroni* lorsqu'on compare des échantillons multiples. Par exemple, un premier test comparant 4 groupes dans une population aura pour seuil de significativité $\alpha = 0.05$. Une comparaison deux-à-deux de ces 4 groupes nécessite de corriger le seuil de significativité. Comme 6 comparaisons deux-à-deux doivent être effectuées, nous utiliserons $\alpha = 0.05/6$. Nous présenterons ici les résultats analysables (synthétisés en tableau II.2) mais l'ensemble est accessible en annexe A.

Les mesures effectuées sont représentées par des *boîtes à moustaches* présentant différentes informations : médiane, moyenne, écart-type, 1er et 3e quartiles

1. un test de significativité vérifie que les résultats obtenus ne sont pas dû au hasard.

Tableau II.2: Comparaisons des encodages visuels en fonctions des différents paramètres évalués. (A C) < B signifie que A et C sont moins bons que B

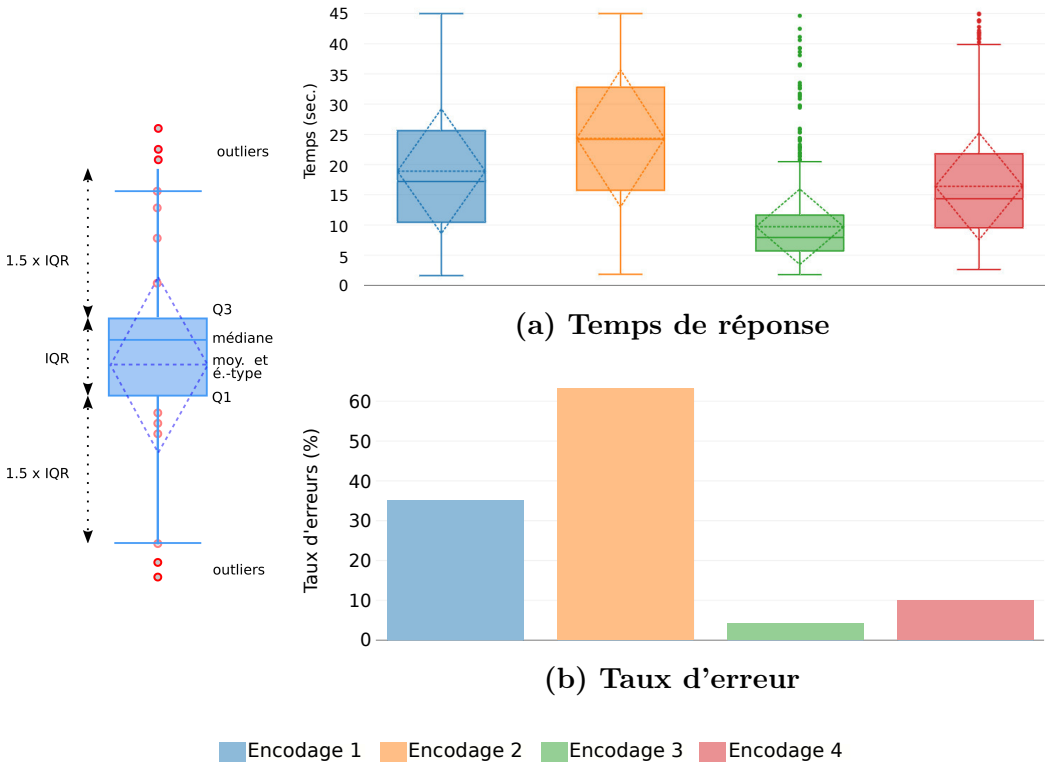
	Tâche	Encodage	Taille (T.)		Densité (D.)	
Temps de réponse	1	-	-		-	
	2	(1 2 3) <4	T. 50 : (1 2 3) <4	T. 100 : (1 3) <4	D. 0,4 : 4 <3	D. 0,6 : 4 <(1 2)
	3	-	T. 20 : 4 <2	T. 50 : (1 2) <3	-	
	4	4 <(1 2 3)	-		D. 0,4 : 4 <(1 2 3)	
Taux d'erreurs	1	-	T. 20 : -	T. 50 : 4 <3	-	
	2	-	T. 20 : 4 <1		-	
	3	-	-		-	
	4	-	-		-	

et outliers. L'annexe 6 présente un schéma de ce diagramme ainsi que le guide de lecture. Ce schéma sera également représenté lorsque nécessaire dans le manuscrit.

Comparaisons des tâches.

Deux analyses de hauts niveaux peuvent être effectuées en considérant soit les tâches de l'évaluation, soit les encodages visuels.

L'analyse par tâche consiste à déterminer si certaines tâches induisent des variations dans les résultats indépendamment de l'encodage visuel ou du graphe utilisé. Les *valeurs-p* des comparaisons des tâches deux-à-deux étant toutes inférieures au seuil de significativité (voir annexe), les différences observables sur la figure II.11 sont statistiquement significatives. On peut ainsi ordonner les tâches en fonction du temps nécessaire pour les compléter ou du taux d'erreur par tâche : dans les deux cas, la tâche 3 est la plus rapide à compléter et avec le plus faible taux d'erreur, puis vient la tâche 4, la tâche 1 et finalement la tâche 2 avec le plus long temps de réponse et le plus grand taux d'erreur.



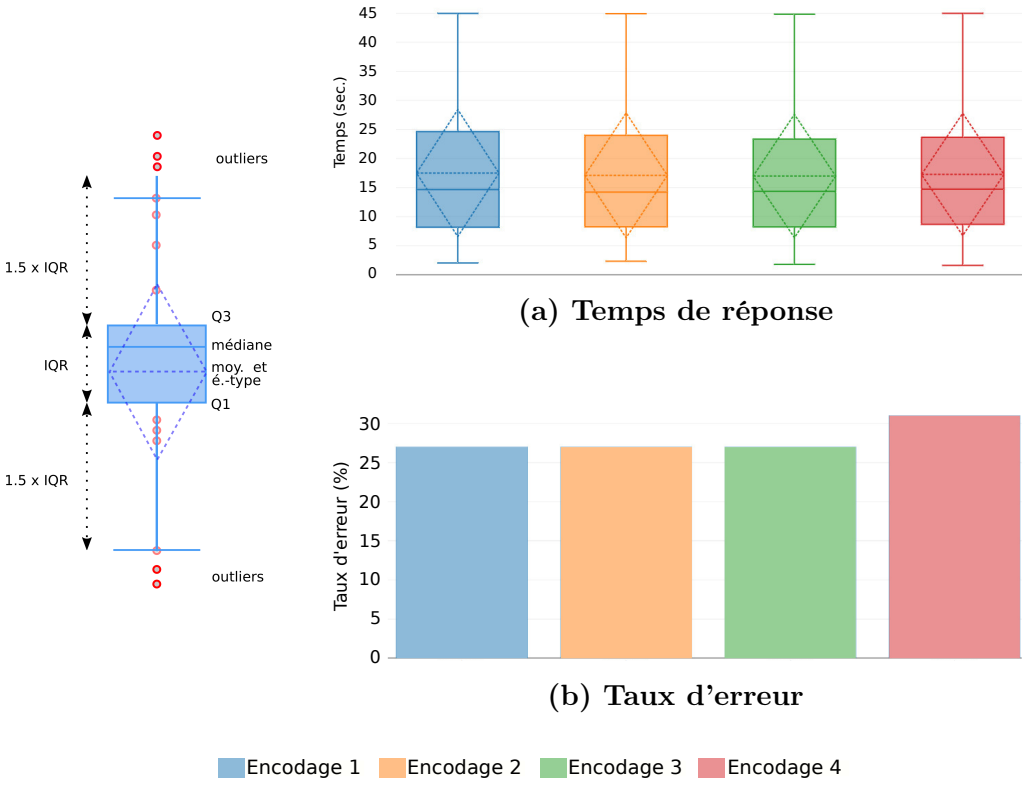
(a) Temps de réponse

(b) Taux d'erreur

FIGURE II.11: Temps de réponse et taux d'erreur pour chaque tâche. Les résultats aux tests de significativités sont fournis en annexe.

Comparaison des encodages visuels

L'analyse par encodage visuel, de façon assez similaire, consiste à déterminer si certains encodages visuels induisent des variations dans les résultats, indépendamment de la tâche considérée ou du graphe utilisé. Les tests de significativité étant tous supérieurs au seuil α (voir annexe), aucune comparaison entre encodage ne peut être effectuée. De plus, les résultats obtenus (voir figure II.12) ne montrent pas de différence permettant de déduire des tendances [1] entre encodages, que ce soit sur les temps de réponse ou les taux d'erreur. Il est donc nécessaire de faire une analyse à un niveau de granularité plus bas. Ces résultats ne sont pas surprenants, les écarts de temps de réponse entre tâches vus précédemment étant relativement importants, il nous parait logique de ne pas détecter de différences entre encodages visuels toutes tâches confondues.



(a) Temps de réponse

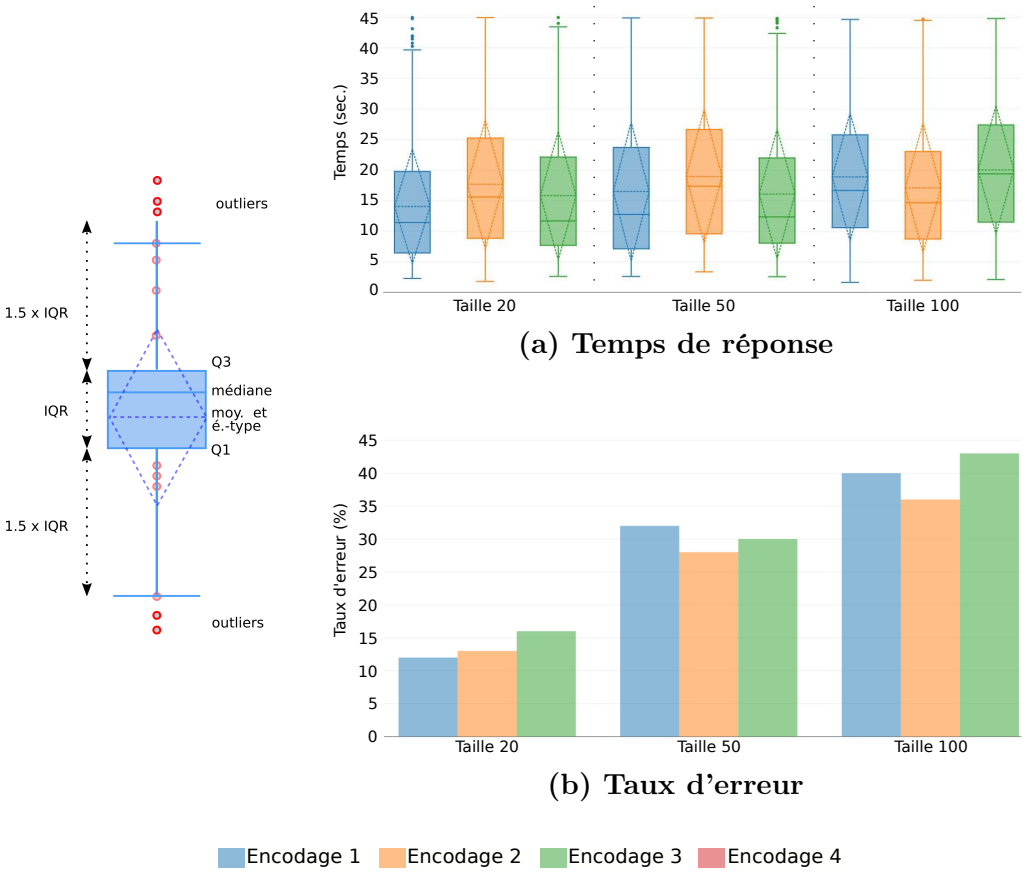
(b) Taux d'erreur

FIGURE II.12: Temps de réponse et taux d'erreur pour chaque encodage visuel. Les résultats aux tests de significativités sont fournis en annexe.

1. on entend par tendances, des différences visibles dans les résultats mais statistiquement non significatives

Comparaison des graphes

Une analyse en fonction des catégories de graphes utilisées (combinaison tailles et densités) a été effectuée. Le tableau II.3 montre qu'une partie des différences observables sont statistiquement significatives. On peut ainsi faire quelques observations à partir de la figure II.13 : 1. les temps de réponse, semblent peu influencés par les catégories de graphes, 2. la taille du graphe semble avoir un impact fort sur le taux d'erreur mais 3. la densité semble l'impacter relativement peu.



(a) Temps de réponse

(b) Taux d'erreur

Figure II.13: Temps de réponse et taux d'erreur en fonction des graphes. Voir tableau II.3 pour les résultats aux tests de significativité.

Tableau II.3: Valeurs-p des tests de significativité pour les Temps de réponse (en haut) et les taux d'erreur (en bas) pour l'analyse en fonction des graphes. (les deux tests globaux étant largement inférieurs au seuil $\alpha = 0.05$, voir annexes). le seuil α de significativité est fixé à 0.0013 pour les comparaison deux-à-deux.

Taille		20		50			100		
	Densité	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
20	0.2	**<10^{-4}**	0.0079	0.0035	**<10^{-4}**	0.0021	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**
	0.4		0.0039	0.0192	0.0589	0.0099	0.0445	0.1983	**0.0007**
	0.6			0.4012	**<10^{-4}**	0.275	**<10^{-4}**	0.0195	**<10^{-4}**
50	0.2				**0.0002**	0.3727	**<10^{-4}**	0.0564	**<10^{-4}**
	0.4					**0.0001**	0.4423	0.0092	0.0417
	0.6						**<10^{-4}**	0.0511	**<10^{-4}**
100	0.2							0.0038	0.0564
	0.4								**<10^{-4}**

Taille		20		50			100		
	Densité	0.4	0.6	0.2	0.4	0.6	0.2	0.4	0.6
20	0.2	0.3713	0.0862	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**
	0.4		0.1499	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**
	0.6			**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**	**<10^{-4}**
50	0.2				0.1522	0.3479	0.0097	0.0847	**0.0008**
	0.4					0.2623	**0.0004**	0.0082	**<10^{-4}**
	0.6						0.0032	0.0388	**0.0002**
100	0.2							0.1667	0.2098
	0.4								0.0381

Comparaison des encodages visuels pour la tâche 1 : Estimation de densité

Les résultats aux tests de significativité (voir tableau II.4) nous indiquent qu'il n'est pas possible de comparer les temps de réponse obtenus en fonction des encodages visuels et ce, quelle que soit l'analyse effectuée. En effet, que l'on effectue l'analyse en considérant l'encodage visuel, la taille ou la densité utilisés, aucune des différences que l'on pourrait observer entre les temps de réponse n'est statistiquement significative. Si l'on considère les taux d'erreur, les seuls différences significatives apparaissent lorsqu'on considère les graphes de tailles 20 et de taille 50. Dans le cas des petits graphes, seul le test global est significatif, aucune comparaison deux-à-deux n'est significative (voir figure II.14). On peut cependant remarquer que l'encodage 3 semble induire un plus fort taux d'erreur (10 % plus élevé), mais ce n'est qu'une tendance qui n'est pas confirmée par les tests. Pour les graphes de taille moyenne, seule la comparaison entre les encodages 3 et 4 est significative et les résultats semblent indiquer que l'encodage 4 induit un taux d'erreur beaucoup plus important. Même si peu de résultats apparaissent significatifs, les résultats tendent à indiquer que pour les graphes de taille moyenne, l'encodage 3 induit moins d'erreurs que les autres encodages et que l'encodage 4 en induit plus.

Tableau II.4: Valeurs-p des tests de significativité pour les taux d'erreur pour l'analyse de la tâche 1 pour la comparaison des encodages visuels en fonction des tailles de graphes. Le seuil α de significativité est fixé à 0.05 pour les tests globaux et 0.0083 pour les comparaison deux-à-deux. Les tests deux-à-deux ne sont effectués que lorsque le résultat du test global est en-dessous du seuil.

		Tâche 1					
Taille	global	E.1- E.2	E.1- E.3	E.1- E.4	E.2- E.3	E.2- E.4	E.3- E.4
20	**0.0244**	0.2816	0.0393	0.4269	0.01	0.2223	0.0576
50	**0.0052**	0.499	0.1255	0.0421	0.1255	0.0421	**0.0022**
100	0.646	-	-	-	-	-	-

Comparaison des encodages visuels pour la tâche 2 : Identification du nœud de plus fort degré

Pour cette tâche, la comparaison directe des encodages visuels nous fournit des résultats significatifs lors de la comparaison entre l'encodage 4 et les autres encodages (voir tableau II.5). Ainsi, l'encodage 4 induit des temps de réponse plus courts que ceux obtenus avec les autres encodages (voir figures II.15 et II.16). Une analyse plus détaillée en fonction des tailles de graphes ou des densités ne présente pas de différence significative en ce qui concerne les autres encodages.

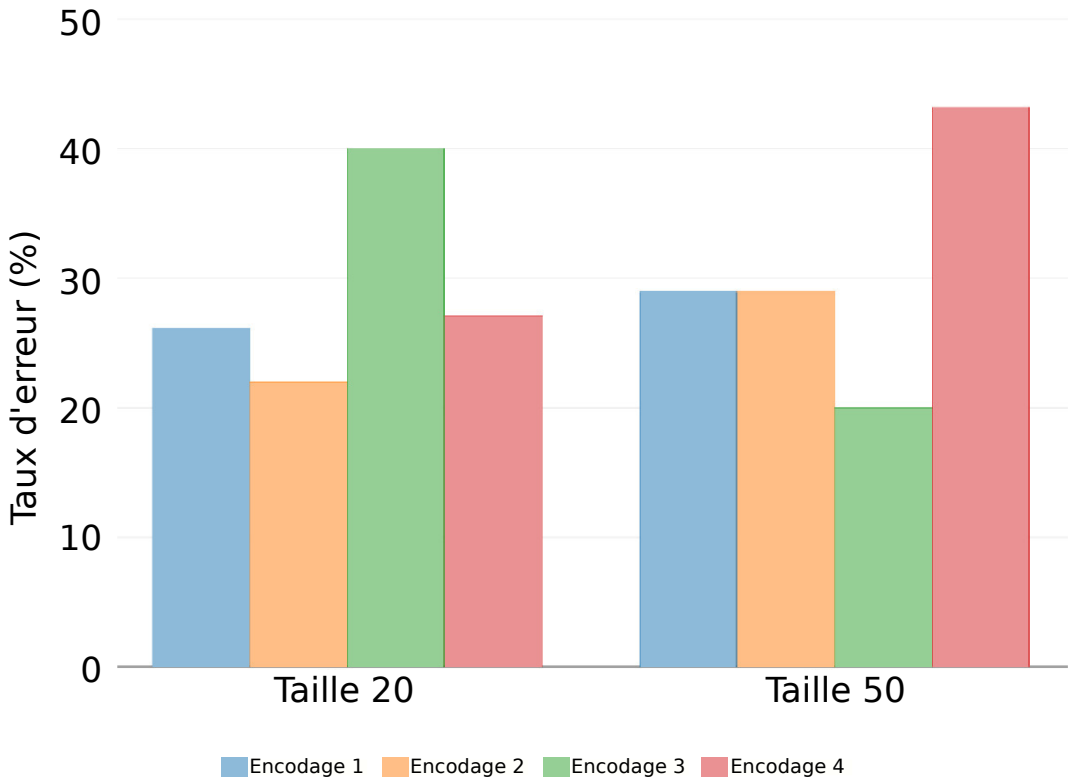


Figure II.14: taux d'erreur pour la tâche 1 pour la comparaison des encodages visuels en fonction des tailles de graphes. Ici ne sont fournis que les résultats statistiquement significatifs (graphes de tailles 20 et 50), voir tableau II.4 pour les résultats aux tests de significativité.

En effet, les seules différences significatives apparaissent lors de comparaisons entre l'encodage 4 et les autres encodages et seulement dans quelques cas particuliers. De plus, les différences que l'on peut observer ne semblent pas suffisantes pour supposer l'existence de tendances. Les taux d'erreur obtenus ne permettent pas non plus d'isoler de différences dues à l'encodage visuel et l'analyse détaillée nous fournit relativement peu d'informations. Seul le test comparant les encodages dans le cas de graphes de taille 20 sont au-dessous du seuil α mais les comparaisons deux-à-deux nous indiquent que seul l'encodage 1 peut être comparé à l'encodage 4. Dans ce cas, l'encodage 1 induit des taux d'erreur plus faibles.

Tableau II.5: Valeurs-p des tests de significativité pour les temps de réponse et taux d'erreur pour la tâche 2.

			Tâche 2						
			global	**E1-E2**	**E1-E3**	**E1-E4**	**E2-E3**	**E2-E4**	**E3-E4**
Temps de réponse	**encodages**		**0,0002**	0,4995	0,1155	**0,0001**	0,1239	**0,0001**	**0,0034**
	Tailles	**20**	0,0865	-	-	-	-	-	-
		50	**0,0032**	0,1405	0,4849	**0,007**	0,1382	**0,0003**	**0,0048**
		100	**0,0104**	0,1649	0,4142	**0,0022**	0,2392	0,0126	**0,0019**
	Densité	**0,2**	0,7292	-	-	-	-	-	-
		0,4	**0,0489**	0,3091	0,436	0,0091	0,2369	0,043	**0,0055**
		0,6	**0,0001**	0,1583	0,121	**0,0007**	0,0121	**<0,00001**	0,0143
Taux d'erreurs	**encodages**		**0,0857**	0,4225	0,3118	**0,0177**	0,384	**0,0282**	0,0532
	Tailles	**20**	**0,0131**	0,0568	0,1319	**0,0016**	0,3167	0,074	0,0279
		50	0,1163	-	-	-	-	-	-
		100	0,4502	-	-	-	-	-	-
	Densités	**0,2**	**0,0224**	0,3707	0,4991	0,0646	0,3707	0,0324	0,0646
		0,4	0,9008	-	-	-	-	-	-
		0,6	0,3638	-	-	-	-	-	-

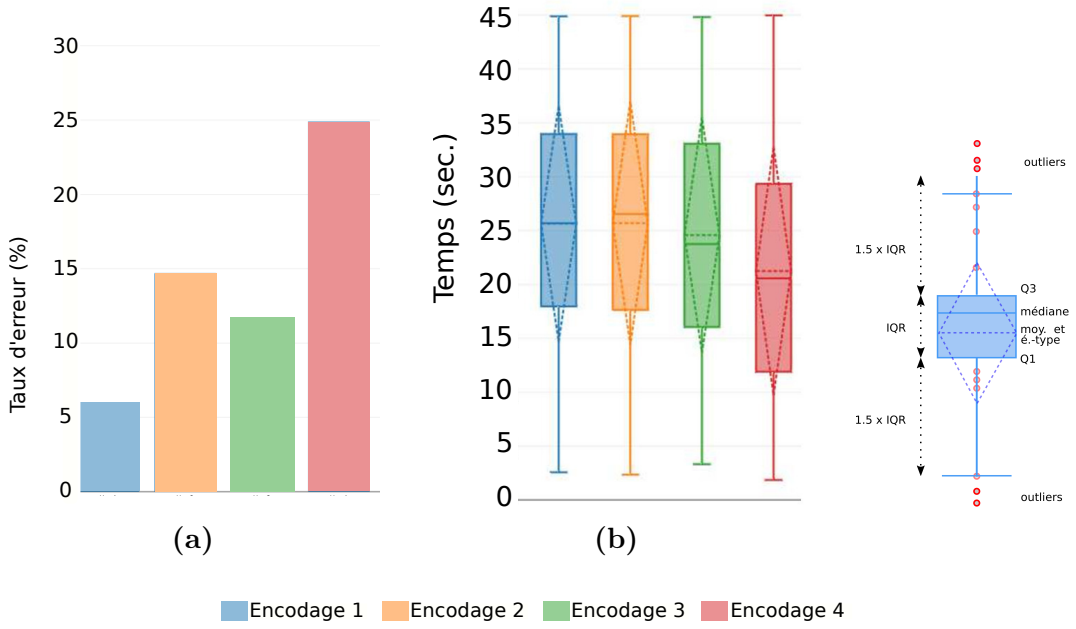
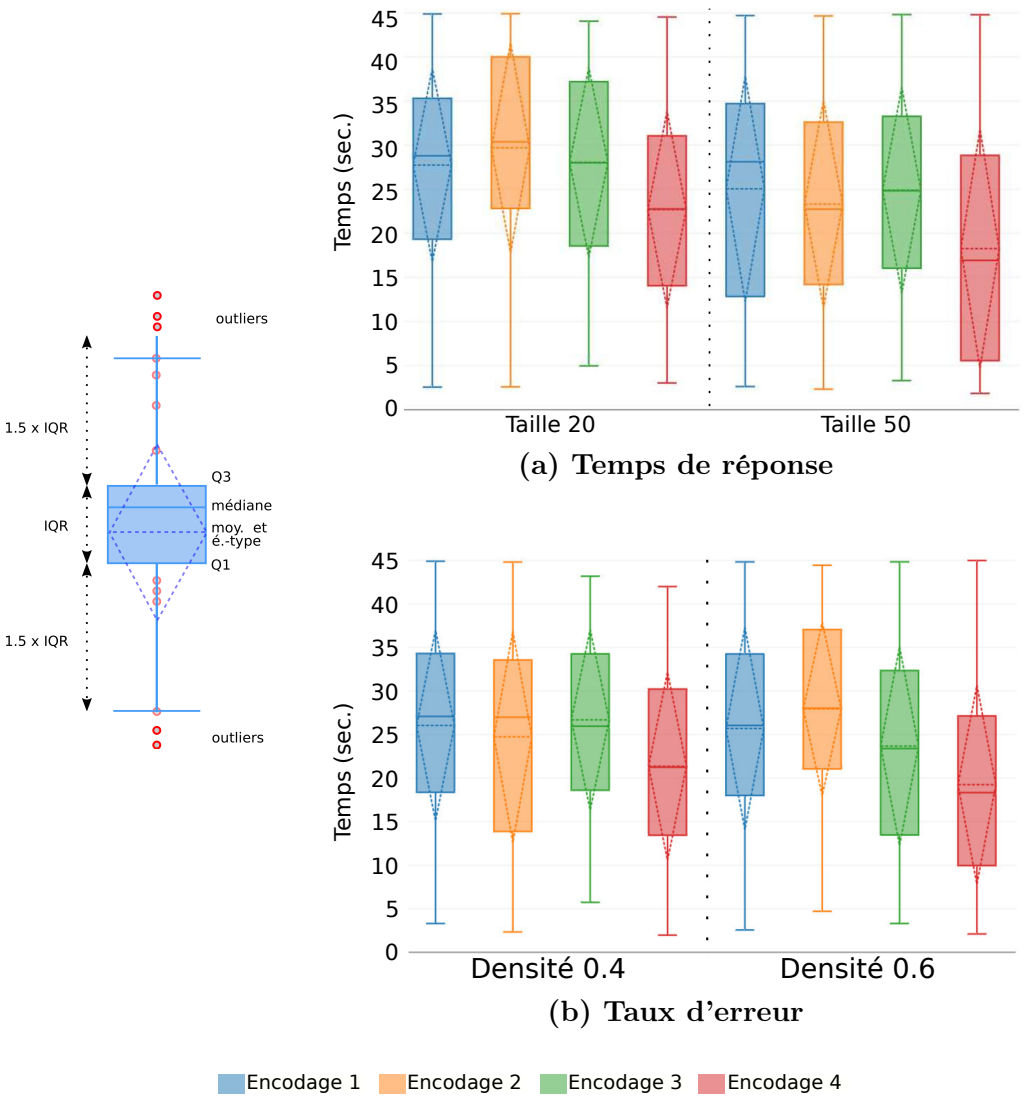


FIGURE II.15: Temps de réponse et taux d'erreur pour la tâche 2 pour la comparaison des encodages visuels. Seuls les résultats statistiquement significatifs sont fournis. (a) Taux d'erreur en fonction de la taille des graphes (taille 20) et (b) comparaison des temps de réponse en fonction des encodages.



(a) Temps de réponse

(b) Taux d'erreur

FIGURE II.16: Temps de réponse pour la tâche 2 pour la comparaison des encodages visuels. Seuls les résultats statistiquement significatifs sont fournis. (a) comparaison des temps de réponse en fonction des tailles de graphes (tailles 50 et 100) et (b) en fonction des densités de graphes (densités 0.4 et 0.6).

Comparaison des encodages visuels pour la tâche 3 : Identification d'une arête reliant deux nœuds

Pour l'analyse des résultats obtenus pour la tâche 3, très peu de résultats répondent positivement aux tests de significativité (voir tableau II.6). La plupart des résultats aux tests sont supérieurs au seuil α, indiquant que l'on ne peut pas comparer les encodages entre eux, que ce soit pour les temps de réponse ou les taux d'erreur. L'analyse détaillée ne fournit que peu d'informations complémentaires. Les tests ne sont inférieurs au seuil que dans deux cas (voir figure II.17) :

— dans le cas de petits graphes (taille 20), l'encodage 2 permet d'obtenir des temps de réponse plus courts qu'avec l'encodage 4, et

— dans le cas de graphes de taille 50, l'encodage 3 permet d'obtenir des temps de réponse plus courts qu'avec les encodages 1 et 2.

Ici aussi, les résultats observés ne permettent pas de supposer l'existence de tendances.

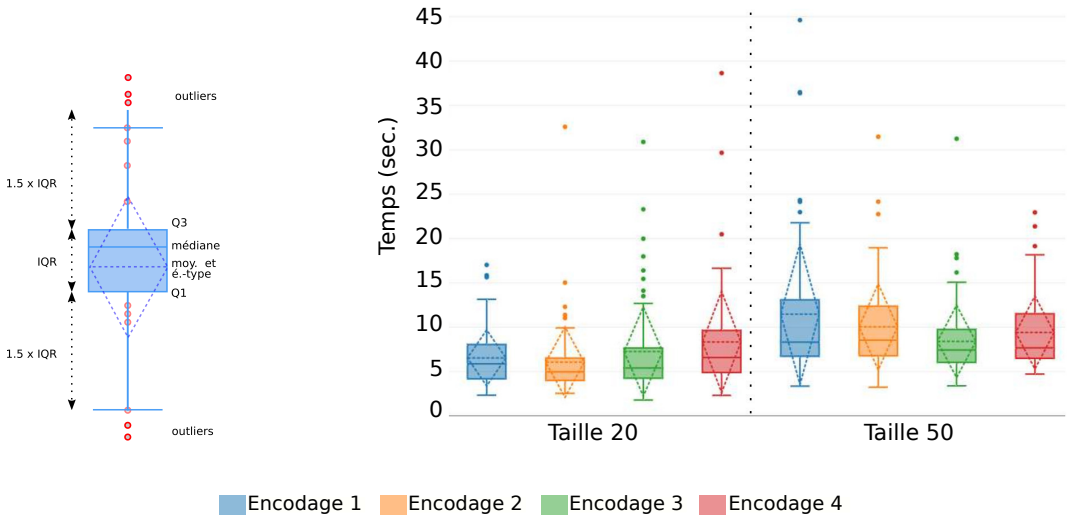


FIGURE II.17: Temps de réponse pour la tâche 3 pour comparaison des encodages visuels en fonction des tailles de graphes. Seuls les résultats statistiquement significatifs sont fournis.

Comparaison des encodages visuels pour la tâche 4 : Identification d'un voisin commun à deux nœuds

Pour la tâche 4, il est possible de comparer les temps de réponse obtenus avec certains encodages visuels. Les tests permettant la comparaison avec l'encodage 4 sont en-dessous du seuil α pour les graphes de densité 40 (voir tableau II.7). Ainsi, la figure II.18 nous montre que l'utilisation de l'encodage 4

Tableau II.6: Valeurs-p des tests de significativité pour les temps de réponse pour la tâche 3.

		Tâche 3						
		global	**E1-E2**	**E1-E3**	**E1-E4**	**E2-E3**	**E2-E4**	**E3-E4**
Tailles	**20**	**0,0058**	0,0827	0,4686	0,0193	0,0682	**0,0001**	0,0327
	50	**0,0495**	0,3997	**0,0083**	0,1693	**0,0079**	0,2387	0,0557
	100	0,4614	-	-	-	-	-	-

induit des temps de réponse plus longs qu'avec les autres encodages. Quasiment aucune des autres analyses effectuées ne donne de résultats significatifs complémentaires, que ce soit sur les taux d'erreur ou à différents niveaux d'analyse.

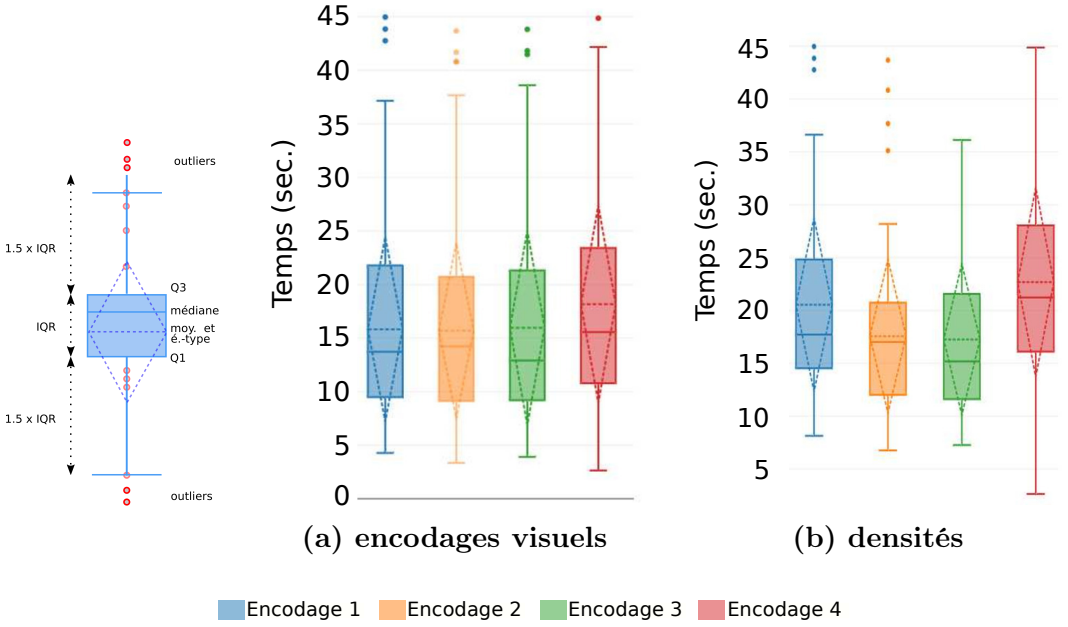


FIGURE II.18: Temps de réponse pour la tâche 4 en fonction de l'encodage visuel. Seuls les résultats statistiquement significatifs sont fournis. (a) Comparaison des encodages visuels et (b) comparaison des densités (densités 0.4)

5.3 Résultats qualitatifs

Les résultats qualitatifs correspondent à l'ensemble des remarques des participants récupérées dans les questionnaires. Ces résultats permettent de mettre en avant une éventuelle influence de l'encodage visuel des arêtes pour ce qui est du confort d'utilisation. Comme précisé en section 4 nous avons

Tableau II.7: Valeurs-p des tests de significativité pour les temps de réponse pour la tâche 4.

		Tâche 4						
		global	**E1-E2**	**E1-E3**	**E1-E4**	**E2-E3**	**E2-E4**	E3-E4
Encodages		0,0088	0,4893	0,4534	0,0031	0,4292	0,0036	0.0017
	0,2	0,0647	-	-	-	-	-	-
Densités	**0,4**	**0,00001**	0,0095	**0,0028**	0,0605	0,3628	**0,0001**	**0.00001**
	0,6	0,2791	-	-	-	-	-	-

demandé aux participants de remplir un questionnaire comprenant différentes questions, et notamment quel encodage visuel ils avaient préféré en fonction des tâches (voir figure II.19). Ainsi, les préférences des participants permettent d'ordonner les encodages visuels toutes tâches confondues avec, par ordre croissant : 1. Encodage 1, 2. Encodage 3, 3. Encodage 2 et, en dernière position Encodage 4.

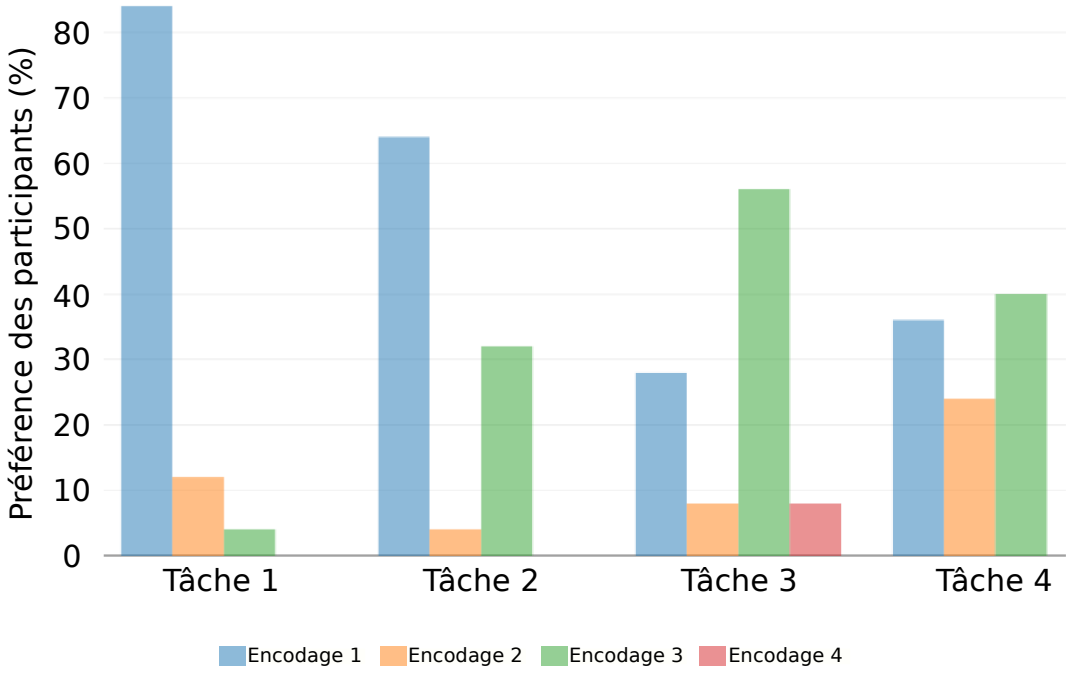


Figure II.19: Pourcentage de préférence utilisateur pour chaque tâche.

Nous constatons que l'encodage 1 est largement majoritaire pour les tâches 1 et 2. L'encodage 3 est majoritaire pour les deux autres tâches même si le résultat est plus nuancé pour la tâche 4. On peut observer que l'encodage 4 est très peu apprécié, quelle que soit la tâche considérée. On remarque ainsi que pour les tâches où il est essentiel d'avoir une vision plutôt globale du graphe, telles que

la tâche 1 (estimation de densité) ou la tâche 2 (recherche du nœud de plus fort degré), l'encodage 1 emporte le suffrage des participants. Pour les autres tâches impliquant une vision plus précise et localisée des nœuds et de leurs connexions, même si l'encodage 1 est fortement apprécié, c'est l'encodage 3 qui devient majoritaire. En ce qui concerne les tâches, la presque totalité des participants (92%) a trouvé la tâche 3 extrêmement simple tandis que la tâche 2 apparaît comme la plus difficile à réaliser dans le temps imparti, ce qui est en adéquation avec les résultats fournis plus haut (voir figure II.11 page 69).

5.4 Discussion

Les résultats quantitatifs obtenus de l'analyse des temps de réponse et taux d'erreur pour chaque tâche permettent de valider le protocole mis en place pour l'évaluation. En effet, on observe des tâches de difficulté variable avec, par ordre croissant, les tâches 3, 4, 1 et 2. Cela correspond aux résultats obtenus lors de l'analyse qualitative sur la difficulté des tâches selon les participants. De même, les résultats obtenus relatifs aux graphes correspondent à des graphes de difficulté variable avec une influence de la taille des graphes sur les taux d'erreur. De façon surprenante, la densité d'arêtes ne semble pas avoir une grande influence sur les résultats obtenus, que ce soit sur le temps de réponse ou les taux d'erreur. Les autres résultats issus d'analyses détaillées des tâches en fonction de l'encodage visuel sont un peu moins marqués et il est difficile d'en extraire un comportement général même si quelques différences significatives apparaissent. Cela nous amène à penser que les différences observables proviendraient en majeure partie de la structure des graphes utilisés, en particulier leurs tailles, et non l'encodage visuel comme nous le suggérait notre hypothèse. On peut cependant observer que l'analyse à haut niveau des encodages visuels donne deux résultats qu'il semble intéressant d'exploiter : l'encodage 4 est meilleur que les autres pour la tâche de recherche du nœud de plus fort degré, qui nécessite une vue générale de la matrice. Au contraire, pour la tâche de recherche de voisin commun qui nécessite une vue locale des relations entre éléments de la matrice, l'encodage 4 devient moins bon que les autres encodages. Les données récoltées durant cette évaluation permettent difficilement d'étendre cette analyse mais c'est un comportement intéressant que nous utiliserons en section 6.

On remarque que malgré le manque d'impact statistiquement significatif des encodages visuels sur les performances utilisateurs, l'analyse qualitative indique qu'ils ont un impact sur le confort d'utilisation. Ces résultats permettent ainsi de nuancer notre hypothèse initiale qui voulait que la simplification de l'encodage visuel des arêtes permettrait d'améliorer les performances utilisateurs ; dans le cadre de cette étude, elles influencent essentiellement son confort. On peut ainsi supposer que des différences plus significatives pourraient émerger dans le cadre d'une utilisation plus longue (heures ou jours).

6 Évaluation dans le cas d'une tâche complexe

Dans l'évaluation précédente, nous avons remarqué que l'encodage 4 semble moins bon que les autres encodages visuels pour des tâches nécessitant une approche locale de la matrice et meilleur dans les cas nécessitant une vue plus globale. De plus, des différences dans les préférences utilisateurs ont pu être mises en évidence. Nous avons donc tenté de révéler des différences significatives dans les résultats quantitatifs en augmentant l'influence de l'encodage visuel des arêtes sur le confort d'utilisation et sur une tâche nécessitant à la fois une analyse globale et à la fois une approche locale de l'information représentée : la recherche d'un chemin entre deux entités. Il s'agit d'une tâche complexe, nécessitant une observation plus longue de la représentation et à différents niveaux de précision mais l'objectif et les hypothèses restent les mêmes que celles de l'évaluation précédente.

6.1 Protocole

Description de la tâche

La tâche évaluée lors de cette évaluation consiste à rechercher un chemin entre deux nœuds mis en surbrillance dans la représentation ; nous appelons ces nœuds *nœuds de départ*. La tâche consiste ainsi à sélectionner les arêtes une à une afin de construire un chemin entre les deux nœuds de départ. Le participant a la possibilité de construire son chemin en commençant par l'une ou l'autre des extrémités, voire les deux. S'il décide de construire le chemin depuis les deux nœuds de départ, la tâche consiste à réunir les deux sous-chemins afin de créer un chemin unique, permettant ainsi de relier les deux nœuds de départ. Pour garantir un niveau de difficulté raisonnable, trois règles sont fixées :

— il est possible de sélectionner une arête seulement si elle permet d'étendre l'un des sous-chemins,

— il est impossible de créer des boucles, et

— si le participant crée deux sous-chemins, alors ceux-ci ne peuvent se réunir que par les extrémités n'étant pas un nœud de départ.

La tâche ne peut être considérée comme terminée que dans deux cas : soit l'utilisateur a validé sa sélection dans le temps imparti, soit le temps alloué s'est écoulé. La validation du chemin par l'utilisateur est une condition *sine qua non* pour que la tâche soit considérée comme complétée. Si le participant a terminé la construction du chemin mais n'a pas eu le temps de valider sa réponse dans le temps imparti, la tâche est considérée comme invalide. Les tests pilotes ont cette fois-ci révélé que 3 minutes maximum par essai était un bon compromis pour cette tâche.

Jeux de données

La tâche étant plus complexe et s'approchant d'une tâche réelle, il nous semble opportun d'utiliser un jeu de données réelles. Les données expérimentales ont été extraites d'un réseau social et correspondent au jeu de données issu du *Concours InfoVis 2007* [87]. Dans ce graphe, les nœuds représentent les acteurs et ceux-ci sont connectés lorsqu'ils apparaissent ensemble dans un film. Chaque sous-graphe est extrait par un algorithme de parcours en largeur (appelé *BFS* pour *Breadth First Search*) avec un nœud initial sélectionné aléatoirement. Nous avons mis en place deux paramètres d'extraction, n et k, le premier correspondant au nombre de nœuds désiré dans le graphe induit et le second au nombre de voisins parcouru à chaque étape du BFS. Le paramètre k peut être interprété comme une approximation de la taille des communautés dans le graphe induit. L'algorithme s'arrête lorsque le nombre de nœuds sélectionnés atteint n et le graphe induit par les sommets visités est extrait. Afin d'obtenir des graphes de complexité variable, nous avons utilisé deux valeurs différentes pour ces deux paramètres n (fixées à 50 et 100) et k (fixées à 5 et 10) ce qui nous fournit 4 catégories de graphes. Deux instances de graphes sont extraites pour chaque catégorie. Là aussi, la limite de 100 nœuds a été fixée pour maintenir un seuil de lisibilité de la représentation dans son intégralité acceptable. Pour chacun de ces graphes, nous avons sélectionné aléatoirement deux nœuds reliés par au moins un chemin de taille minimale 4.

Les tests préliminaires ont révélé qu'un ordre pseudo-aléatoire des nœuds comme celui utilisé précédemment rendait la tâche trop complexe et difficile à réaliser dans le temps imparti. Pour remédier à cela, nous avons choisi d'utiliser l'ordre d'apparition des nœuds durant le processus d'extraction permettant de regrouper les nœuds appartenant à une communauté. En effet, la tâche étant plus complexe, il nous a semblé nécessaire de mettre en avant dans la représentation les communautés afin de simplifier la compréhension de la structure du graphe et par conséquent, faciliter la navigation dans le graphe. Afin d'éviter un biais dans notre évaluation, nous utilisons le même ordre des nœuds dans la matrice quel que soit l'encodage visuel utilisé.

Population

Les participants à l'évaluation sont des étudiants de l'enseignement supérieur (licence, master ou doctorat), des enseignants-chercheurs ou des ingénieurs de recherche dans le domaine informatique. Tous savent comment interpréter un graphe ainsi que les représentations nœuds-liens et matricielles mais ne manipulent pas ces concepts et outils de façon régulière. Ces notions sont ré-introduites durant la phase de présentation afin de s'assurer de la bonne compréhension des participants. Au total, 33 personnes ont participé à l'évaluation dont 6 pour les tests pilotes et 27 pour l'évaluation expérimentale.

6.2 Résultats

Comme présenté dans la section 4, nous analysons ici les résultats de l'évaluation sous deux aspects : tout d'abord quantitatif puis qualitatif. Tout d'abord, nous avons essayé de déterminer si les résultats des participants permettent de mettre en avant des variations statistiques dépendant de l'encodage visuel des arêtes, et ce en prenant en compte si nécessaire les catégories de graphes étudiés. Pour cela, nous avons observé quatre paramètres lors de cette évaluation :

— le taux d'erreur,

— le temps de réponse pour compléter une tâche,

— la longueur du chemin résultant, et

— le nombre total d'arêtes sélectionnées pour chaque essai.

Ensuite nous avons considéré les résultats obtenus lors de l'entretien avec les participants afin d'évaluer l'influence de l'encodage visuel des arêtes sur les participants, en termes de perception et de confort d'utilisation.

Résultats quantitatifs

Nos résultats ne suivant pas une loi de distribution normale, nous avons utilisé les deux tests de significativité non-paramétriques introduits dans l'évaluation précédente afin d'évaluer la robustesse des résultats obtenus. Ainsi le *test de Friedman* est utilisé pour le taux d'erreur (données complètes) et le *test de Kruskal-Wallis* pour le temps de réponse, les longueurs de chemins, et le nombre d'arêtes sélectionnées. Ici aussi nous avons défini un seuil de significativité standard $\alpha = 0.05$, que nous avons ajusté par une *correction de Bonferroni* pour la comparaison de plusieurs groupes. Nous présenterons ici les résultats significatifs mais l'ensemble est accessible en annexe 10.4.

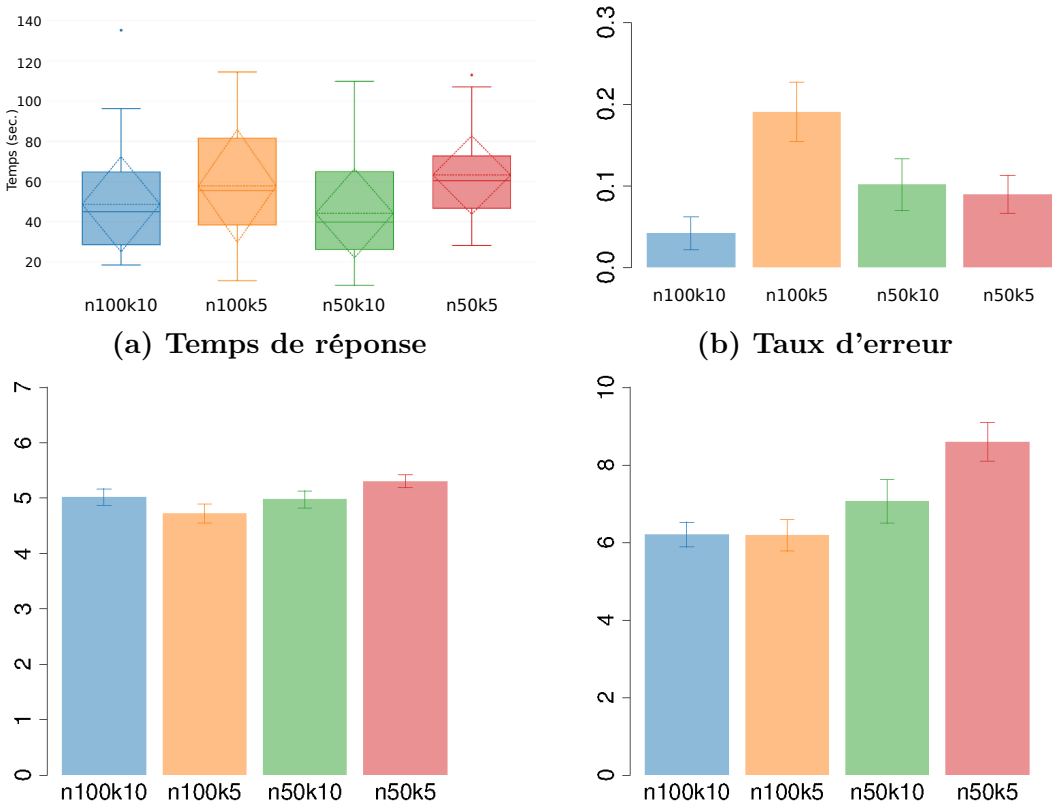
Certains des participants ont montré des taux d'erreur très importants durant l'évaluation, leurs résultats dépendant non pas des conditions de l'étude mais plutôt d'une réelle difficulté à comprendre et à compléter la tâche. Leurs résultats ont généré par conséquent un nombre important de valeurs aberrantes et ont pu influencer les conclusions de l'étude ; il nous a donc semblé préférable de retirer les résultats de ces participants. Ainsi 7 participants ont été retirés, 4 ayant un taux d'erreur avoisinants les 50 %, les 3 autres étant supérieurs à 30 %.

Analyse de résultats par catégorie de graphes La comparaison des catégories de graphes montre que tous les tests de significativité effectués donnent une valeur-p inférieure au seuil $\alpha = 0.05$ (voir tableau II.8). On peut ainsi voir sur la figure II.20 que les temps de réponse des graphes dont $k = 5$ sont plus grands et que les graphes dont $n = 100$ impliquent des tailles de chemins et un nombre total de sélections plus faible que ceux dont $n = 50$.

Ainsi il semblerait que la structure du graphe ait une influence notable sur les performances utilisateurs sur les paramètres mesurés.

catégories	taux d'erreur	nombre d'arêtes sélectionnées	temps de réponse moyen	longueur de chemin moyen
n100k10	0.8614	0.4746	0.1013	0.2539
n100k5	0.078 64	0.64	0.1221	0.1525
n50k10	0.1213	0.5738	0.1994	0.2821
n50k5	0.6444	0.1751	0.61	0.4469

Tableau II.8: Valeurs-p des tests de Friedman et de Kruskal-Wallis pour les taux d'erreur moyens, Temps de réponse, longueur de chemin et nombre d'arêtes sélectionnées pour l'analyse par catégories de graphes et par encodages visuels.



(a) Temps de réponse **(b) Taux d'erreur**

(c) longueur de chemin moyen **(d) nombre de sélections moyen**

FIGURE II.20: Résultats par catégories de graphes. Les noms utilisés correspondent à la combinaison utilisée pour l'extraction, n pour le nombre de nœuds et k pour la taille maximale des communautés.

Comparaison des encodages visuels : Analyse globale et détaillée. L'analyse globale des résultats obtenus pour chaque encodage visuel ne montre aucune différence significative (voir résultats aux tests en annexe), que ce soit sur les mesures de temps de réponse, taux d'erreur, longueur de chemin ou nombre de sélections (voir figure II.21). Il en est de même lorsqu'on considère une comparaison des encodages visuels par groupe de graphes (voir figure II.22). Il est surprenant de ne pas avoir de résultats statistiquement significatifs, l'observation des graphiques semblant indiquer qu'il existe des différences entre encodages visuels. Cette absence de significativité peut s'expliquer par de très grands écarts-types, empêchant la détection statistique de différence entre les groupes.

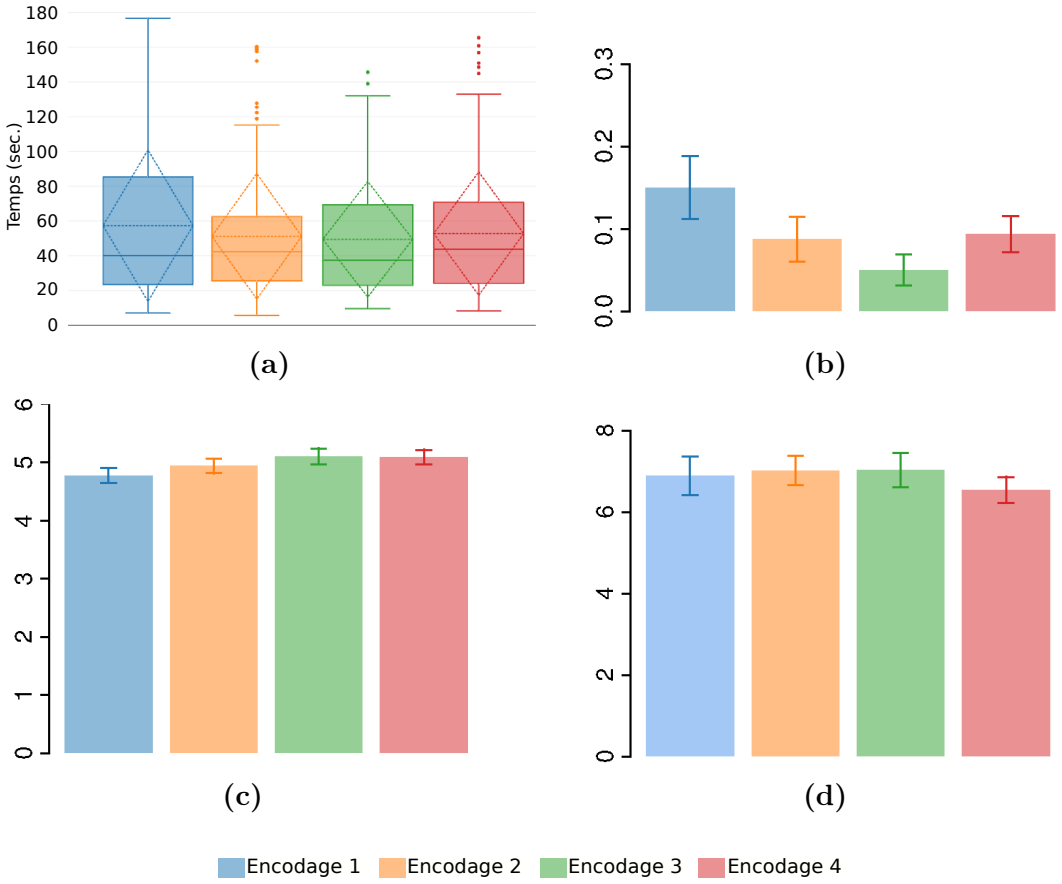
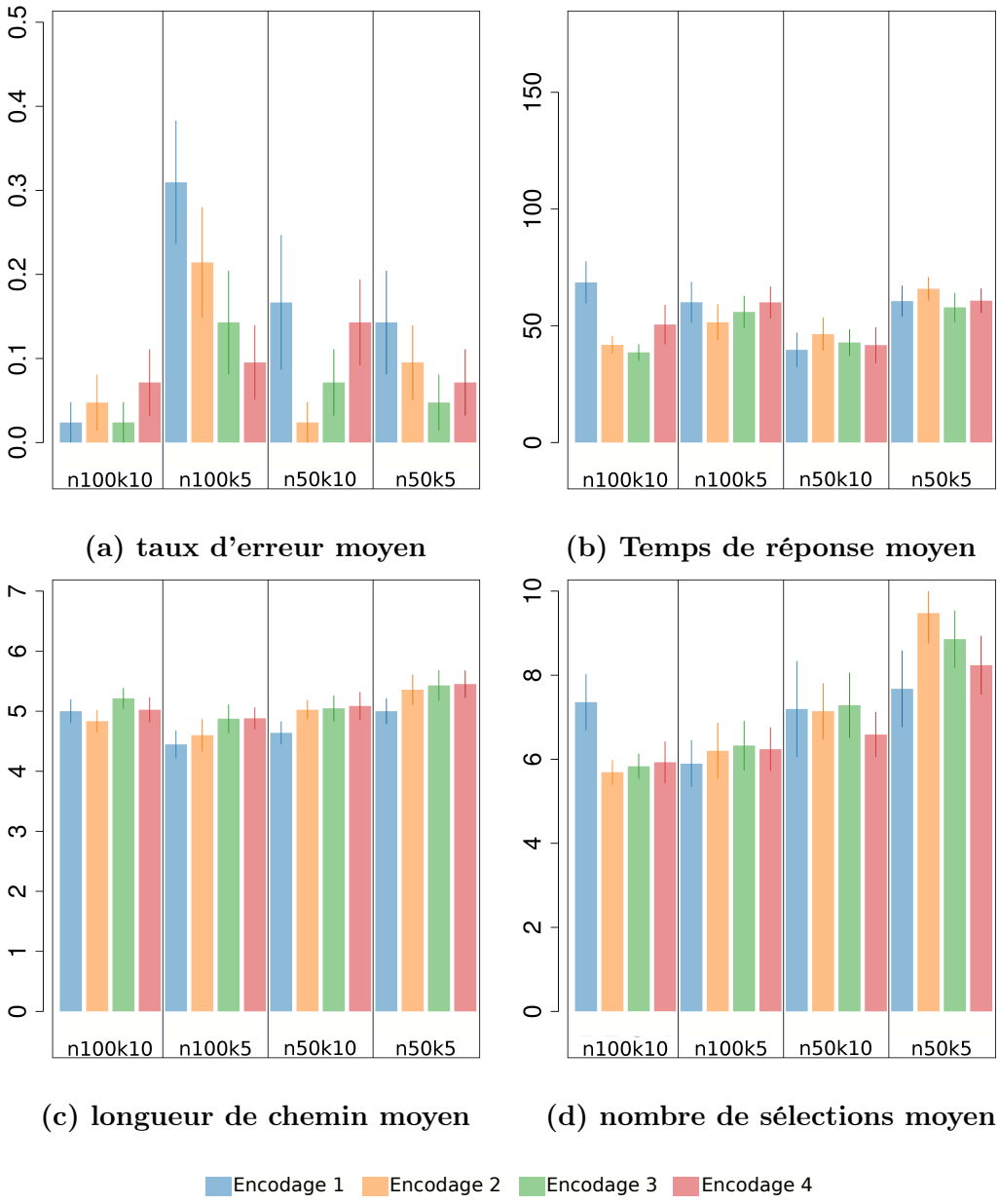


FIGURE II.21: Résultats par encodages visuels. (a) Temps de réponse. (b) Taux d'erreur. (c) Longueur de chemin. (d) Nombre de sélections.



(a) taux d'erreur moyen **(b) Temps de réponse moyen**

(c) longueur de chemin moyen **(d) nombre de sélections moyen**

FIGURE II.22: **Résultats par catégories de graphes et encodages visuels.**

Résultats qualitatifs

Lors de l'entretien avec les participants, nous leurs avons demandé quel encodage visuel ils avaient préféré, nous avons ainsi pu ordonner les encodages visuels en fonction de leurs réponses. Nous pouvons constater sur la figure II.23 une nette préférence pour les encodages 3 (environ 47 %) et 2 (environ 21 %). Les encodages 1 et 4 quant à eux ne sont appréciés que par une minorité avec approximativement 7 % et 5 % des participants. Approximativement 20 % des participants n'ont pas su déterminer de préférence parmi les encodages visuels.

L'analyse des commentaires révèle par ailleurs que l'encodage 1 apparaît généralement comme plus intuitif et plus simple d'utilisation pour des graphes de petites tailles. La majorité des participants ont trouvé l'encodage 2 meilleur que l'encodage 1 en facilitant la lisibilité des arêtes et des communautés. L'encodage 3 semble être le meilleur en ce qui concerne la lisibilité des arêtes, cependant cette lisibilité semble se faire, selon les avis des participants, au détriment de la détection des communautés et de l'interaction lors de la sélection. L'encodage 4 quant à lui était le moins apprécié : mal adapté pour les grands graphes, zone d'interaction réduite, peu intuitif, la distinction entre les croisements et les courbures d'arêtes est qualifiée de difficile et rend complexe l'identification des communautés.

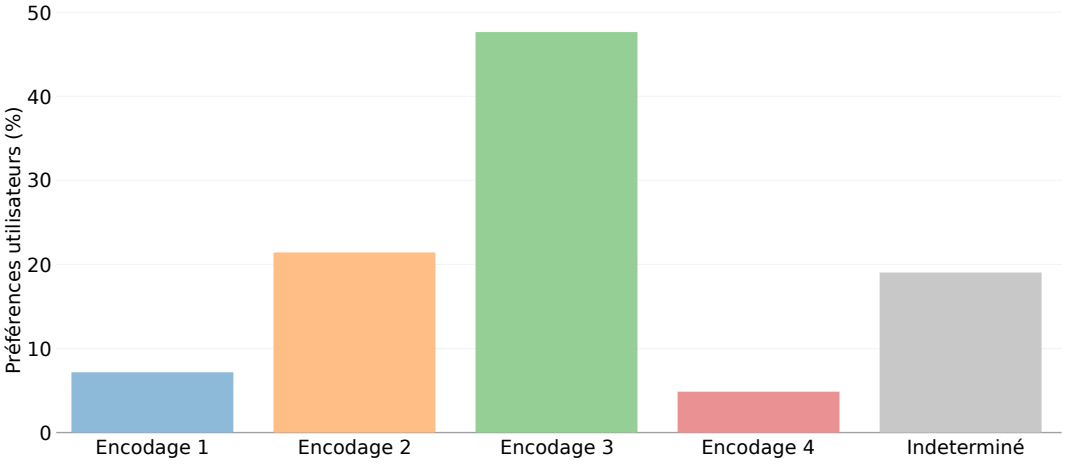


Figure II.23: Même si les résultats quantitatifs ne sont pas significativement différents, les préférences tendent à indiquer que l'encodage visuel choisi pourrait influencer le taux d'erreur.

6.3 Discussion

L'analyse quantitative des résultats indique qu'il n'y a pas de différences significatives dans les performances des participants entre les 4 encodages visuels, et cela même si l'on prend en compte les catégories de graphes. Nous

supposons que cela doit être dû à un plafond induit par la difficulté des tâches et le temps imparti, empêchant la détection de variations dans les résultats utilisateurs en fonction des encodages visuels. Il est surprenant de voir des taux d'erreur relativement faibles, pouvant indiquer que la tâche était trop simple à résoudre tandis que certains des participants n'ont pas réussi à finir tous les essais dans le temps imparti. Il est difficile de trouver une raison logique à ce comportement.

L'absence de différence significative dans les résultats lorsqu'on considère les encodages visuels est surprenant, cependant, on peut remarquer que des tendances semblent se dégager. Ainsi, il semble que la simplification de l'encodage visuel impliquerait une diminution des taux d'erreur dans la recherche de chemins (voir figure II.21b). Il devient donc possible, suite à cette constatation, d'ordonner les encodages visuels suivant les taux d'erreur qu'ils induiraient, donnant l'ordre suivant (des plus forts taux d'erreur aux plus faibles) : encodage 1, encodage 4, encodage 2 et encodage 3. Ces résultats tendent à favoriser les encodages 3 et 2 par rapport aux encodages 1 et 4 ce qui correspond aux avis des participants. En effet, la majorité d'entre eux ont classé l'encodage 3 comme étant le meilleur et l'encodage 4 comme le moins bon, ainsi qu'une légère préférence pour l'encodage 2 par rapport à l'encodage 1. Cette constatation concorde avec nos hypothèses préliminaires exprimées en section 3.

7 Discussion générale et comparaison à l'existant

Comme introduit au début de ce chapitre, notre travail vise à comparer les encodages visuels des arêtes dans les représentations matricielles et nœuds-liens. Pour cela nous avons mis en place trois évolutions de l'encodage visuel des arêtes dans les matrices permettant de transformer une matrice d'adjacence en un diagramme nœuds-liens dont les nœuds auraient été dupliqués. Les encodages 3 et 4 peuvent ainsi être considérés comme étant des encodages nœuds-liens standards avec duplication des nœuds. Cette transformation nous a permis d'effectuer une comparaison entre les deux représentations tout en nous soustrayant du biais du positionnement que l'on retrouve dans d'autres évaluations utilisateurs.

Les résultats obtenus ne permettent pas de mettre en évidence une quelconque influence de l'encodage visuel des arêtes sur les performances utilisateurs. Suite aux résultats obtenus, il apparaît que la correction de Bonferonni, bien que très souvent utilisée puissent être une cause de rejet de nos résultats. En effet, la lecture de discussion sur la puissance statistique des tests [99] semblent indiquer que cette correction soit trop forte dès lors que l'on considère l'analyse de plus de 4 ou 5 classes. Cependant les résultats obtenus, même s'ils ne sont pas statistiquement significatifs, nous amènent à penser qu'une diminution

extrême de la complexité visuelle a pour effet de rendre difficile la lecture des arêtes. Ainsi, il semblerait que l'utilisation d'une simple polyligne (encodage 4) complique l'identification et le suivi des arêtes.

Si l'on considère les retours utilisateurs, l'encodage 3 se retrouve en 2^{e} et 1re position des préférences sur les deux évaluations. Dans l'évaluation pour des tâches basiques, il emportait le suffrage des participants pour les tâches nécessitant une analyse plus précise des relations du graphe. Les retours utilisateurs pour l'évaluation de tâches complexes, nécessitant une vue globale et locale des données, indiquent que la qualité de la vue locale est plus importante que la vue globale, l'encodage 3 se retrouvant en 1re position des préférences utilisateurs. L'encodage 2 suit un schéma similaire en passant de la 3^{e} à la 2^{e} position entre les deux évaluations. Cette évolution associée à l'absence de variation dans les autres paramètres analysés durant les évaluations semble indiquer qu'une simplification raisonnable de la complexité visuelle facilite l'usage des matrices d'adjacence en terme de confort sans affecter les performances utilisateurs.

Les résultats obtenus dans le cadre de ces deux évaluations sont plutôt surprenants comparés à ceux présentés dans la littérature [15, 50, 60, 81]. En effet, ceux-ci présentent des résultats concordants, il est donc légitime de se demander quelle est l'origine d'une telle différence entre leurs résultats et les nôtres. On peut tout d'abord souligner une différence majeure entre leurs protocoles et le nôtre : la représentation des entités. En effet, dans notre étude, les nœuds sont dupliqués pour les représentations nœuds-liens et leur positionnement et représentation sont identiques dans les deux diagrammes. D'après nos résultats et les leurs, cette distinction permettrait de lisser les différences de performances obtenues pour les deux diagrammes. L'effet de la duplication des entités dans l'efficacité et la lisibilité des représentations de graphes a d'ailleurs déjà été partiellement étudiée dans [58] via une représentation hybride mixant DNL et DOM pour la représentation de communautés dans un graphe. Les auteurs montrent que la duplication de certaines entités donne plus d'amplitude à l'algorithme de dessin et améliore la perception d'appartenance des entités aux communautés.

Se pose par ailleurs le problème de la reproductibilité des résultats. Pour des tailles d'échantillons similaires à celles des évaluations présentées dans la littérature (entre 10 et 40 personnes) de telles différences sont surprenantes. On peut ainsi se demander si le nombre de participants utilisé est réellement représentatif et s'il permet de tirer des conclusions et de garantir la reproductibilité des résultats obtenus.

8 Conclusion

Dans ce chapitre nous nous sommes centrés sur l'étude de l'encodage visuel des arêtes et son influence pour la transmission visuelle de l'existence d'une

relation entre deux entités. Après avoir présenté les techniques couramment utilisées, nous avons tenté de déterminer pour les deux techniques que sont les DOM et les DNL, ce qui leur conférait cette efficacité. Nous avons ainsi effectué une transformation progressive de l'encodage visuel des arêtes caractéristique des DOM en un encodage nœuds-liens, et cela sans changer la représentation des autres éléments. Cette transformation est la base de deux évaluations utilisateurs permettant de comparer les deux techniques tout en s'abstrayant du biais de positionnement/ordonnancement que l'on peut retrouver dans d'autres évaluations présentes dans la littérature. Les résultats présentés ont ainsi montré que l'encodage visuel des arêtes apporte peu de variation dans les performances utilisateurs pour un usage tel que défini dans nos protocoles. Cependant dans le cas de tâches complexes ou nécessitant une vision locale et précise des connexions, l'encodage 3 emporte les faveurs des participants. On peut ainsi supposer que pour une utilisation régulière et intensive, il permettrait d'apporter un confort d'utilisation et une amélioration des performances, mais c'est une supposition qui mériterait un travail complémentaire.

Chapitre III

Visualisation de données hiérarchiques orientées et pondérées

1 Introduction - État de l'art

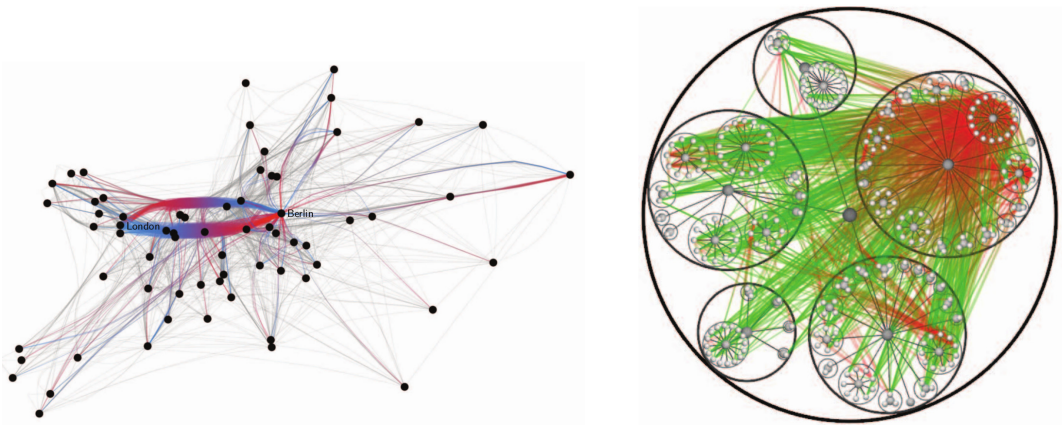
Dans de nombreux secteurs, analyser la façon dont les informations transitent dans un système est une préoccupation majeure pour surveiller, comprendre et prédire les variations de ce système. Dans l'analyse de réseaux sociaux par exemple, comprendre la façon dont les rumeurs se propagent dans et entre les communautés et identifier les personnes qui démarrent ou arrêtent ces rumeurs sont des questions importantes. En étude d'audience d'un site Web, avoir des informations sur la navigation des utilisateurs dans un site internet est important pour identifier des tendances ou des habitudes d'achats. L'analyse de ces informations permet de déterminer ce qui incite un consommateur à valider un panier d'achats ou au contraire ce qui le refrène. La taille et la complexité de ces données peuvent cependant atteindre un seuil empêchant le traitement et la représentation par des techniques habituelles. La représentation de ces données regroupe ainsi plusieurs aspects : mettre en évidence des information relationnelles, orientées et pondérées tout d'abord mais également prendre en considération la masse de données que cela peut représenter et l'encombrement visuel qu'une telle quantité de données peut induire. Les travaux que nous présenterons dans ce chapitre ont été publié et présenté dans une conférence internationale [121] et ont été récompensés par le prix du meilleur article de la conférence (*Best paper Award*). La suite de cette section est un rappel de l'état de l'art des techniques de représentations de données relationnelles pondérées et des techniques d'abstraction et exploration multi-échelles. la section 1.3 présente les contributions ainsi que l'organisation générale du chapitre.

1.1 Représentation de données relationnelles

L'approche fréquemment utilisée consiste à modéliser ces données avec un graphe orienté pondéré dans lequel les nœuds représentent les entités du réseau et les arêtes et les poids qui y sont associés représentent l'information qui transite entre ces entités et leur importance. Différentes techniques ont été proposées au cours des dernières décennies pour visualiser ce type de structure. Les techniques les plus connues et les plus utilisées sont les diagrammes nœuds-liens (DNLs) et les diagrammes orientés matrices (DOMs).

Diagrammes nœuds-liens

Différentes techniques ont déjà été présentées afin de représenter la direction et la pondération des arcs d'un graphe dans un DNL.



(a) **(b)**

FIGURE III.1: Exemple de visualisation d'un graphe orienté avec un diagramme nœuds-liens utilisant la coloration pour indiquer l'orientation des arcs : illustrations extraite de (a) [124] et (b) [67].

La technique introduite dans [67] présente l'utilisation d'un code couleur pour représenter la direction des arcs (voir figure III.1b). Coloré en rouge du coté de la source de l'arc, la couleur est progressivement transformé jusqu'à devenir verte au fur et à mesure que l'on se rapproche de la destination. De la même manière, la technique présentée dans [124] met en avant les transferts entre éléments du graphe par des regroupements d'arêtes en fonction de leurs orientations. On peut voir sur la figure III.1a que seules les arêtes ayant la même direction sont regroupées ensemble, les couleurs servant à identifier l'orientation des connexions.
Dans [85] une technique de dessin est introduite positionnant les nœuds sur une grille et utilisant des arcs orthogonaux pour la visualisation de graphe orienté acyclique associé à un tri topologique des nœuds (voir figure III.2).

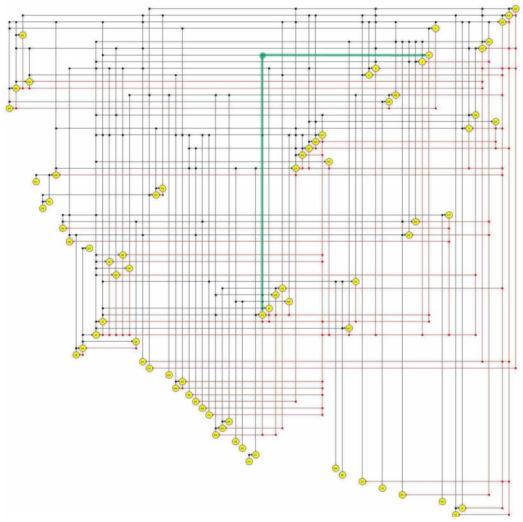


FIGURE III.2: Visualisation de graphe orienté avec un diagramme nœuds-liens : la technique introduite dans [85] détermine un ordre des nœuds spécifique afin de les positionner dans le plan pour que à chaque arc, le nœud cible soit dans le quadrant supérieur droit de l'espace par rapport au nœud source.

Leur algorithme de tri des nœuds permet, si deux nœuds sont connectés par un arc, de positionner le nœud destination dans le quadrant supérieur droit de la matrice par rapport au nœud source.

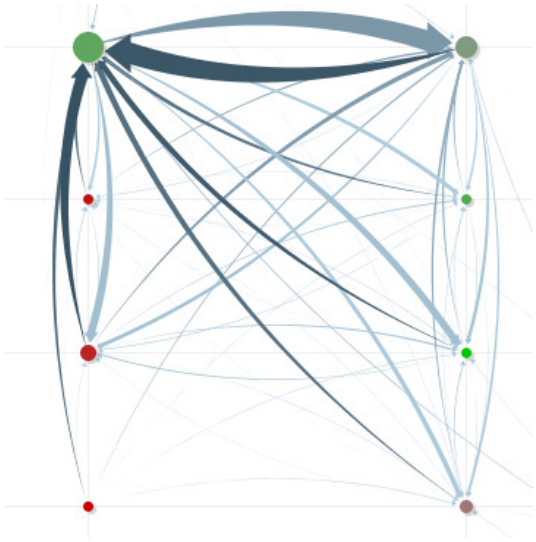
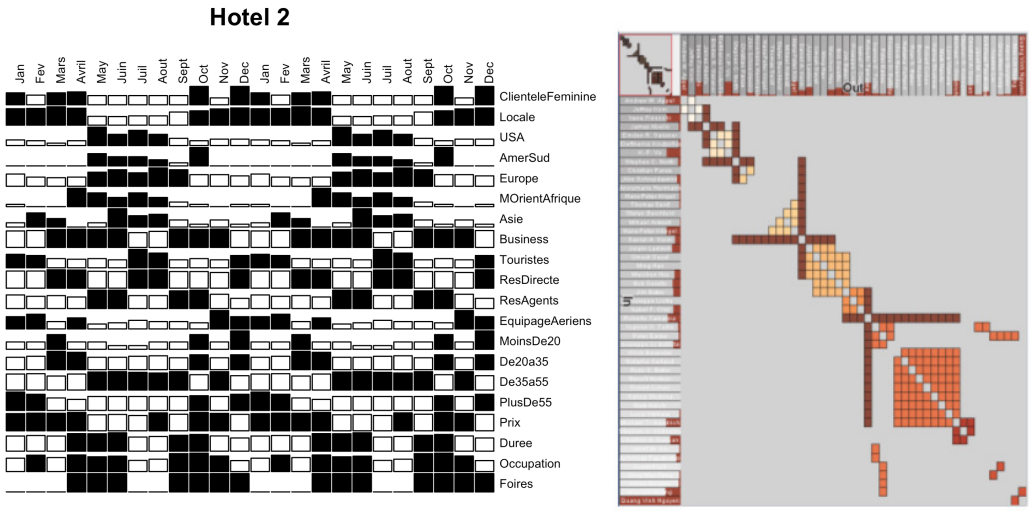


FIGURE III.3: Exemple de visualisation de graphe pondéré avec un diagramme nœuds-liens : Exemple d'une technique graphe pivot introduites dans [142] utilisant la taille des nœuds et des arcs pour représenter leurs poids.

Dans [142] les auteurs présentent une technique de visualisation de graphe multi-varié par des graphes pivot figure III.3. La technique qu'ils décrivent utilisent des arcs en forme de flèches pour indiquer leurs directions.
Dans [68] les auteurs présentent une évaluation utilisateur comparant différentes techniques afin de représenter la direction des arcs d'un graphe. Ils mettent en évidence l'efficacité d'une représentation par diminution de la largeur des arcs entre les sources et destinations ou à défaut, l'utilisation d'une variation de la couleur (de sombre vers clair).

Diagrammes orientés matrices



(a) **(b)**

Figure III.4: Exemple de visualisation de graphe pondéré avec un diagramme orienté matrice. En (a) les matrices telles qu'introduites dans [20] utilisant des histogrammes pour représenter la pondération. En (b)la technique décrite dans [59] utilisant la coloration pour indiquer la pondération des connexions entre éléments.

Pour les DOMs, on peut également trouver différentes techniques afin de véhiculer l'information de poids et d'orientation des connexions entre les éléments. Dans une matrice l'ordre des éléments est généralement le même sur les deux axes, afin de faciliter l'identification des éléments lors de l'analyse. Lorsque la matrice est non orientée, les éléments étant représentés deux fois (sur l'axe vertical et horizontal), les connexions peuvent elles aussi être représentées deux fois. Cette duplication des connexions a pour effet de rendre la matrice symétrique. La principale technique pour représenter la direction des arcs d'un graphe orienté se base sur l'utilisation d'un axe comme source des connexions et l'autre comme destination et de positionner les arêtes dans l'une ou l'autre

des deux demi-matrices en conséquence. Les connexions n'étant représentées plus qu'une fois, cela a pour effet de *casser* la symétrie de la matrice. Les matrices pouvant être considérées comme des tableaux, les valeurs d'arêtes pondérées peuvent être inscrites textuellement dans la matrice. Dans [20], Bertin introduit la technique de représentation visuelle de matrice et permet notamment d'utiliser des histogrammes à la place des valeurs textuelles (voir figure III.4a). Cette idée est reprise dans [36] pour la visualisation interactive de grands graphes. Une adaptation plus visuelle de ces matrices pondérées consiste en l'application d'une coloration des *cases* de la matrice en fonction d'une échelle de couleur pour caractériser le poids des relations entre éléments. Les techniques décrites dans [2, 59] utilisent notamment cette coloration (voir figure III.4b). Lorsque toute la matrice est colorée, on peut considérer le résultat de cette coloration comme étant une carte de chaleur permettant d'identifier les différentes valeurs dans la matrice. Pour un peu plus d'informations sur les différentes évolutions de cette technique, nous recommandons la lecture de [144].

Autres techniques

Les DNLs et les DOMs sont les techniques les plus couramment utilisées mais d'autres techniques ont été décrites dans la littérature.

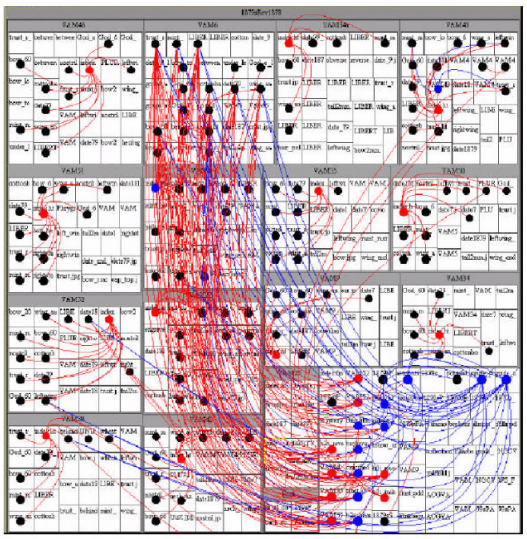
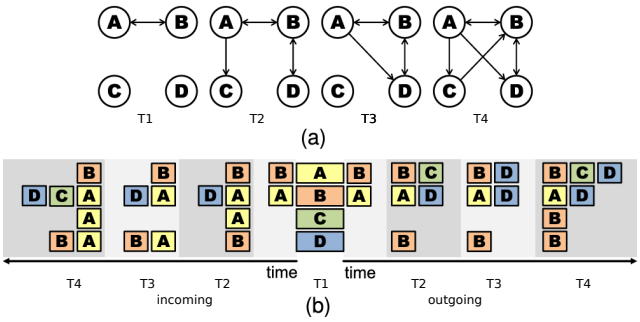


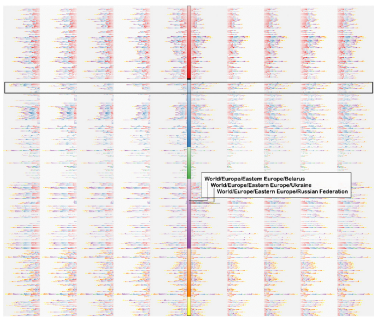
Figure III.5: Visualisation de liens orientés avec un diagramme nœuds-liens : Dans [39] la courbure de l'arc indique l'orientation. Une forte pente indique que l'on est proche de la source et une pente douce indique que l'on est proche de la cible.

Par exemple, la technique présentée dans [39] utilise la courbure des arcs pour représenter leurs directions dans des treemaps (voir figure III.5). Afin de

montrer les arcs d'un graphe en utilisant une treemaps, les auteurs utilisent des liens courbés et l'intensité de la courbure indique l'orientation. Une pente forte indique la source de l'arc et une pente douce dirige vers la cible de l'arc.



(a) Construction de la représentation en liste d'adjacence



(b) Exemple de liste d'adjacence visuelle

FIGURE III.6: Exemple de visualisation de graphe orienté pondéré : Liste d'adjacence visuelle pour les graphes dynamiques décrite dans [64].

La technique introduite dans [64] présente l'utilisation des listes d'adjacences pour la visualisation de graphes orientés et pondérés (voir figure III.6). Les nœuds sont alignés en colonne et de part et d'autre de chaque nœud de cette colonne, sont représentés les nœuds voisins à distance 1. La position à gauche ou à droite du nœud central (dans la colonne) est déterminée par l'orientation des arcs reliant le nœud de la colonne à ceux à droite ou à gauche. En effet, si le nœud dans la colonne est la cible de l'arc, alors le nœud source est placé à sa gauche ; inversement, s'il est la source, la cible est placée à droite. Cette technique utilise la taille des nœuds pour indiquer les informations de pondération. On peut également citer l'exemple des arcs diagrammes tels qu'introduits dans [141], où les entités sont alignées et des arcs de cercles relient les entités connectées. Ces liens peuvent être pondérés et dans ce cas représentés par des arcs de cercle de taille ou de transparence variable. Une approche par fractionnement de matrice pour les graphes orientés acycliques appelée *Quilt* est décrite dans [140] (voir description en chapitre I). La représentation résultante se *lisant* de haut en bas, le sens de lecture nous fournit le sens des connexions (voir figure III.7).

Une autre approche pour représenter efficacement la pondération fait références aux méthodes s'inspirant des diagrammes de flux. Depuis les premiers diagrammes décrits par Minard [96] et Sankey [119] différents travaux ont été décrits afin d'apporter certaines fonctionnalités (interactivité notamment), améliorer le positionnement ou réduire l'encombrement visuel [5, 114, 136] (voir figure III.8). Ces techniques permettent de mettre en avant les tendances majeures tout en représentant les quantités associées et facilitent l'identification des contributions majeures dans l'ensemble du flux. Cependant, elles souffrent

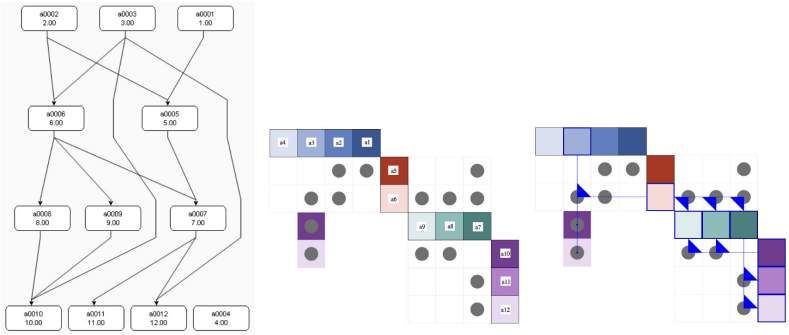


Figure III.7: Fractionnement et quadrillage d'une matrice d'adjacence (*quilting* en anglais) introduite par [140].

de gros problèmes de mise à l'échelle et d'encombrement lorsque le nombre d'éléments à représenter augmente.

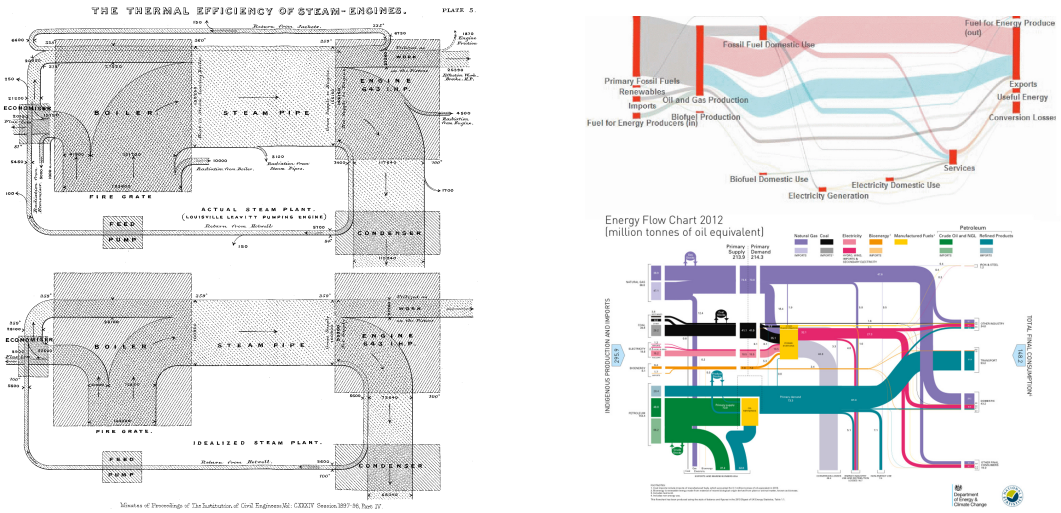


Figure III.8: Diagrammes de Sankey : à gauche, le diagramme présenté par Sankey[119], à droite des versions plus modernes (en haut, illustration extraite de [5] et en bas, les flux d'énergie du Royaume-Uni publiés en 2012 par le DECC).

D'autres encore ont présenté des représentations dites hybrides ayant pour objectifs de mixer et de profiter des avantages de plusieurs techniques[60, 61, 125]. Dans [61], Les auteurs présentent une technique de fractionnement de matrices qui mélange les approches nœuds-liens et matricielles. Leur approche consiste à fractionner une matrice en différentes matrices de petites tailles correspondant à des regroupement d'entités (communautés) et à considérer ces petites matrices comme des nœuds. Ces *matrices-nœuds* sont ensuite positionnées dans le plan comme pour un DNL classique. Les connexions entre éléments de différentes matrices sont ensuite représentées par des liens. Cette

technique permet notamment de mettre en avant les différents regroupements d'éléments et de fractionner des matrices de grande tailles pour facilite l'analyse des relations entre communauté. Dans [60, 125] ce sont des approches par juxtaposition de lien par dessus une matrice qui sont décrites afin de mettre en avant soit le degré des éléments [60], soit des chemins entre éléments de la matrice [125].

1.2 Abstraction de données et exploration multi-échelle

Même si les techniques mentionnées plus haut sont utilisables pour des tailles de plusieurs millions d'éléments, trois défauts majeurs restent présents : 1. l'encombrement visuel, 2. le manque d'efficacité pour représenter la pondération dans le réseau et/ou 3. un temps de rendu relativement long.L'encombrement visuel, généralement provoqué par les chevauchements et croisements de nœuds et d'arêtes, augmente avec le nombre d'arêtes et de nœuds. Plus le graphe gagne en complexité, plus les représentations deviennent difficilement interprétables. Ces effets sont par ailleurs accentués lorsqu'on veut représenter la pondération des arêtes. Finalement, dessiner un grand nombre d'éléments a pour effet de ralentir le temps de rendu et les délais que cela engendre est un frein à l'exploration interactive.

Une méthode fréquemment utilisée pour réduire l'encombrement et augmenter la vitesse de rendu consiste à construire une abstraction du graphe original. L'intérêt réside ici dans la mise en avant des différents groupes issus de la hiérarchisation des éléments du graphe ainsi que des relations dans et entre ces groupes. La structure hiérarchique peut être soit inhérente aux données, soit calculée avec un algorithme de partitionnement. On peut, à partir d'un graphe partitionné et de l'arbre de partitionnement correspondant, extraire une coupe maximale de l'arbre et la représenter par un graphe quotient (voir définitions 34 et 35). Les sommets de ce graphe quotient sont appelés *méta-nœuds*. Si l'on considère deux groupes du graphe, l'ensemble des arêtes reliant ces deux groupes (arêtes inter-groupes, voir définition 33) est représenté dans le graphe quotient par une unique arête appelée méta-arête. Les arêtes intra-groupes quant à elles ne sont pas représentées. Il est possible de créer un graphe quotient pour chaque coupe maximale de l'arbre. On peut ainsi effectuer une coupe à différents niveaux de l'arbre hiérarchique et générer des abstractions des différents niveaux de l'arbre. Cette technique permet ainsi de guider l'utilisateur dans son analyse en lui facilitant l'étape d'analyse et d'identification des portions intéressantes du réseau et l'aide à concentrer sa recherche sur les éléments important. En effet, cette opération induit une réduction du nombre d'entités et d'arêtes affichées et permet donc de réduire l'encombrement visuel tout en améliorant la vitesse de rendu. Cette méthode a déjà largement été présentée depuis une dizaine d'années et a été utilisée à de multiples occasions en visualisation de données (*e.g.* Abello *et al.* [3], Archambault *et al.* [10], Balzer et Deussen [16]),

là encore, principalement via l'une des deux représentations DNLs et DOMs (voir figure III.10).

Cette approche est particulièrement utilisée lorsqu'on veut permettre l'exploration multi-échelle du graphe, c'est-à-dire, l'exploration interactive à différents niveaux d'abstraction du graphe. Depuis un niveau d'abstraction du graphe, l'utilisateur peut demander l'affichage de plus ou moins de détails pour certains méta-nœuds (voir figure III.9). L'ouverture d'un méta-nœud se caractérise par la modification de la coupe de l'arbre afin d'obtenir les descendants du méta-nœud ouvert. L'opération inverse s'appelle fermeture et provoque le remplacement d'un ensemble de nœuds ou méta-nœuds ayant un ancêtre commun par cet ancêtre.

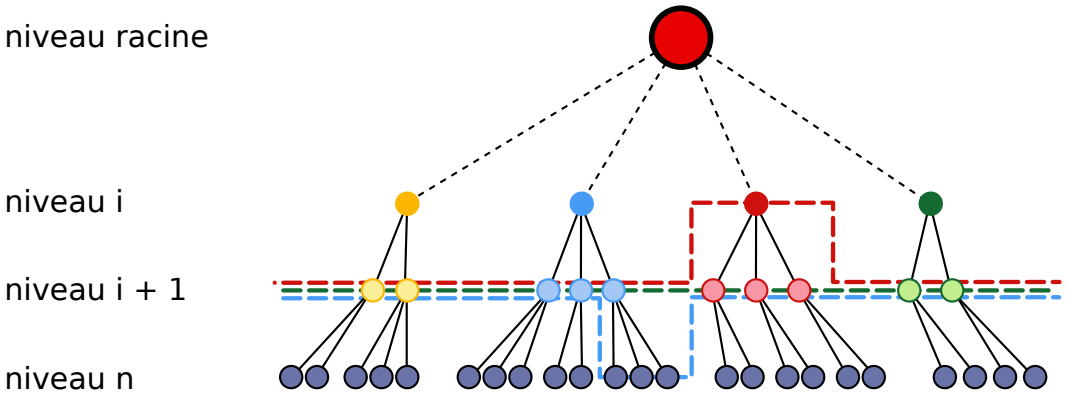
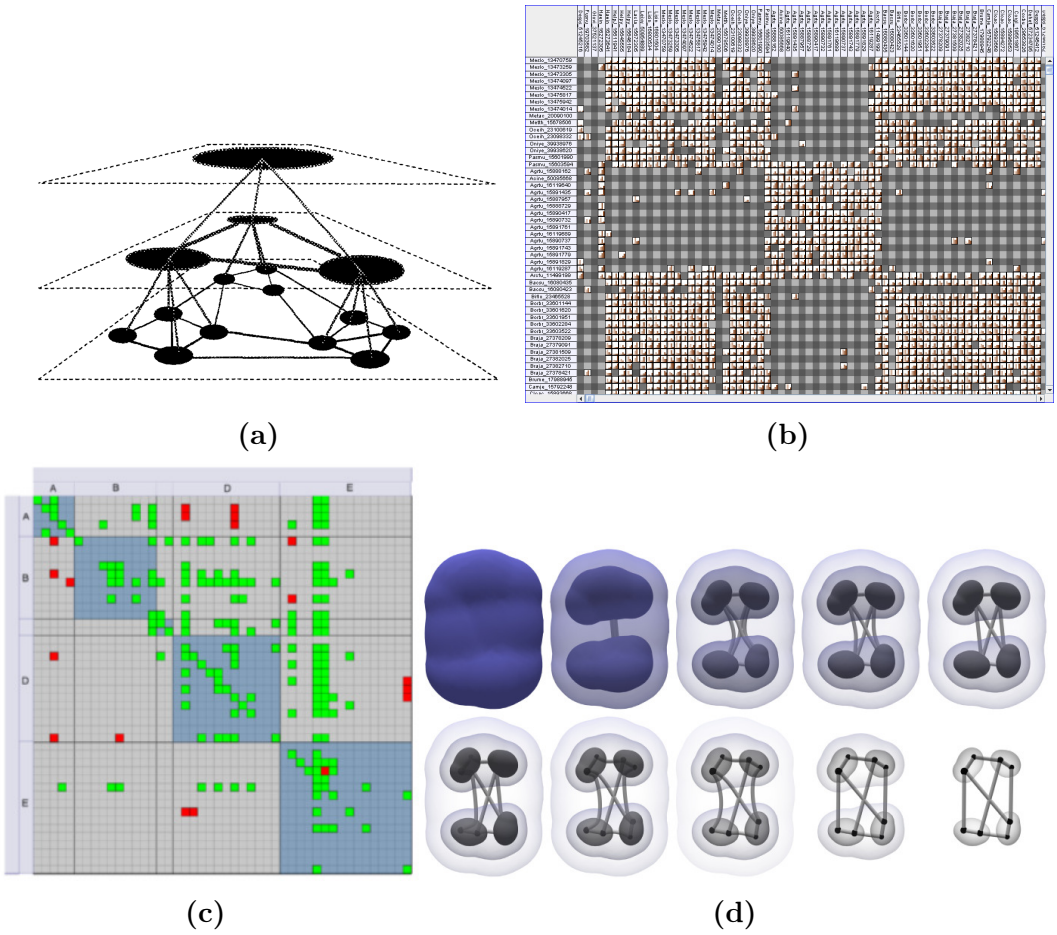


Figure III.9: Exploration multi-échelle avec arbre hiérarchique et graphe quotient.La ligne pointillée verte indique la visualisation courante, il s'agit de la coupe maximale pour le niveau i+1 dans l'arbre hiérarchique. La ligne rouge montre la modification de cette coupe lors de l'ouverture du méta-nœud bleu. La ligne jaune représente quant à elle la coupe de l'arbre lors de la fermeture des méta-nœuds rouge du niveau i+1.

La technique présentée dans [33] permet la visualisation de graphes partitionnés dans un espace tri-dimensionnel. Ainsi chaque niveau dans la hiérarchie est représentable par un niveau de profondeur dans l'espace 3D. Dans [36] les auteurs présentent une technique permettant l'agrégation multi-niveaux pour les matrices d'adjacence et à l'exploration interactive de la représentation obtenue. La technique présentée dans [16] permet elle aussi d'affiner le niveau de détails du graphe en utilisant une représentation par enveloppes des méta-nœuds. L'opacité des enveloppes permet d'obtenir plus ou moins de détails sur les éléments du graphes. Dans [54], la technique décrite permet la visualisation et l'exploration multi-échelle de graphes en utilisant des DOMs.



(a) **(b)**

(c) **(d)**

FIGURE III.10: Exemple de visualisation de données multi-échelle. (a et b) visualisation par DOM introduit dans [36](b) ou dans [54](c). En (c), une représentation par DNL décrite dans [33] attribuant à un niveau dans l'espace 3D un niveau de l'arbre de partitionnement. En (d), une visualisation avec différents niveaux de détails (opacité des enveloppes) tel que décrite dans [16] .

1.3 Contribution

Nous l'avons montré, les problématiques de visualisation de données relationnelles ont déjà largement été traitées dans la littérature. Il y a beaucoup de travaux qui s'intéressent à la mise en évidence de l'orientation des arcs d'un graphe, de la pondération des éléments voire de la structuration hiérarchique et de son utilisation pour explorer les données. Pour autant, très peu s'intéressent à une représentation efficace de ces trois propriétés de façon conjointes avec pour objectif la visualisation et l'exploration de données massives. C'est dans cet objectif que nous présenterons ici *Adjasankey* une nouvelle technique de visualisation interactive dédiée à la représentation de graphes orientés pondérés

avec partitionnement hiérarchique. L'objectif visé étant d'aider les utilisateurs à répondre aux questions suivantes :

— Quelles sont les entités connectées aux flux les plus importants ?

— Est-ce que l'information transite essentiellement dans ou entre certains groupes (intra- ou inter-groupes) ?

— Combien de flux démarrent ou s'arrêtent depuis une entité ou un groupe d'entités ?

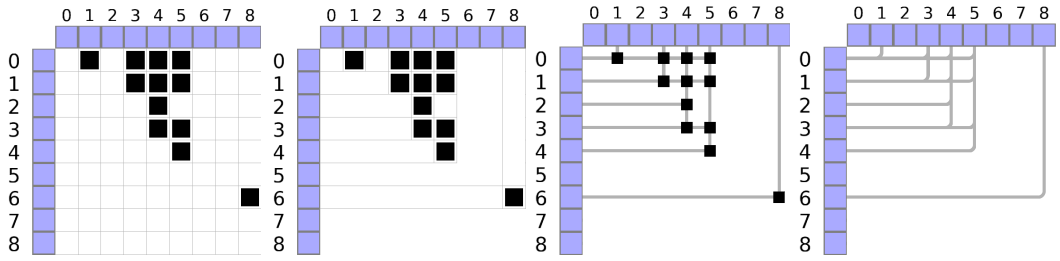
Pour cela, la représentation que nous allons décrire devra notamment permettre l'identification des entités importantes, identifier les sources et destinations des relations et l'importance de ces relations dans le réseau.

La suite de ce chapitre sera structuré comme suit : En section 2, nous présenterons comment, depuis les résultats obtenus dans le chapitre II nous construisons une nouvelle technique de visualisation de données relationnelles orientées et pondérées avec structuration hiérarchique. Ensuite, en section 3 nous présenterons les différentes parties concernant la conception d'Adjasankey. Cette section fournira les détails sur le modèle de donnée utilisé pour répondre aux problèmes de temps d'accès et de rendu et la section 4 présentera un cas d'utilisation de notre outil de visualisation. Finalement, nous présenterons nos conclusions et les travaux envisageables suite aux résultats obtenus en section 5.

2 De l'évaluation à la pratique

Les différents travaux sur la visualisation de graphes présentés en chapitre II ont mis en avant l'efficacité des Diagrammes Orientés Matrices (DOMs) pour la majorité des tâches classiques en analyse de graphes. Cependant d'autres études montrent aussi que pour des tâches de recherche/suivi de chemins, les Diagrammes Nœuds-Liens (DNLs) sont efficace lorsque le graphe est de taille raisonnable. Nous nous sommes ainsi orientés sur la question d'optimisation des matrices d'adjacences pour la représentation de pondération et nous sommes donc appuyés sur les différentes transformations introduites dans la section 2 du chapitre II. Les résultats obtenus dans les deux évaluations ont montré que la modification de l'encodage visuel des arêtes pouvait dans certains cas améliorer le confort des utilisateurs sans pour autant avoir un impact sur leurs performances dans la réalisation des tâches. Cela nous laisse donc libre dans le choix de l'encodage visuel d'arête, présenté en figure III.11.

Afin d'ajouter l'information de pondération des éléments, il faut lui attribuer une des variables visuelles de l'espace de conception. Celles-ci sont au nombre de huit : la *taille*, la *couleur*, le *glyphe*, l'*orientation*, la *texture*, la *position*, la *luminosité* et le *mouvement*. L'utilisation d'une étiquette textuelle indiquant le poids des éléments était également envisageable. Cependant, comme expliqué dans [137, sect. 8.3.1 et 4.3] l'utilisation d'un texte pose les problèmes de



(a) encodage 1 **(b) encodage** 2 **(c) encodage** 3 **(d) encodage** 4

Figure III.11: Résultats de la simplification de l'encodage visuel d'arête dans les matrices d'adjacences.

positionnement, de taille et de chevauchements des étiquettes ayant une influence sur la lisibilité du texte et semblait peu adaptée. Parmi ces variable, les plus adaptées pour des valeurs ordinales sont : les textures, la taille et la luminosité. Nous verrons dans la suite de ce chapitre que nous utiliserons la couleur et la luminosité pour véhiculer d'autres informations. L'utilisation conjointe de la couleur, la luminosité et des textures risquant d'être complexe à interpréter, nous préférons utiliser la taille pour représenter le poids des éléments.

L'option choisie consiste à adapter la taille des éléments du graphe afin de représenter leur pondération et permettre d'identifier rapidement les éléments ayant une place prépondérante dans le graphe. Nous présentons ici les transformations visuelles à apporter sur les tailles d'éléments afin de fournir l'information de pondération.

2.1 Adaptation de l'encodage visuel des arêtes

Comme le montrent les figures III.12 à III.14, un nombre limité de transformations permet d'ajouter l'information de pondération des éléments dans une matrice d'adjacence, tout en limitant l'encombrement visuel dans la représentation. Les différentes étapes qui suivront seront détaillées avec une matrice traditionnelle mais la même démarche peut être faite avec les autres encodages visuels présentés.

La première étape consiste à adapter la taille des nœuds et arcs en fonction de leurs poids. La taille du glyphe représentant un arc est modifiée afin de représenter son importance dans le graphe : plus son poids est grand, plus la surface du glyphe est grande (voir figure III.12).

Afin de réduire l'encombrement visuel, les lignes/colonnes vides sont retirées de la matrice. Comme le montre la figure III.13, les nœuds *F, H et I* (resp. *A, C, G, et H*) n'ayant pas d'arcs sortant (resp. entrant) sont retirés de la matrice. La largeur des lignes et colonnes est ensuite adaptée pour correspondre à la taille de l'arc de plus grand poids qu'elle contient. La taille du nœud est modifiée en conséquence pour maintenir l'alignement entre un nœud et sa ligne/colonne

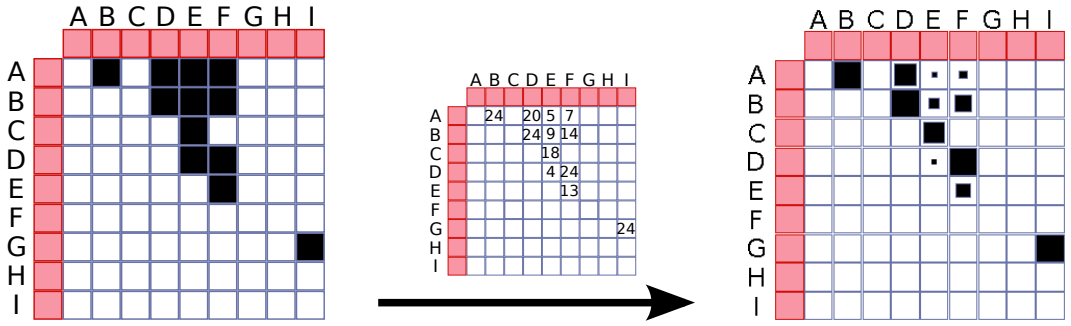


Figure III.12: Transformation d'une matrice d'adjacence pour associer la pondération des arêtes à leurs tailles. (a) une matrice non pondérée, (b) étiquetage des arêtes pour indiquer la pondération, (c) affichage avec adaptation de la taille des glyphes.

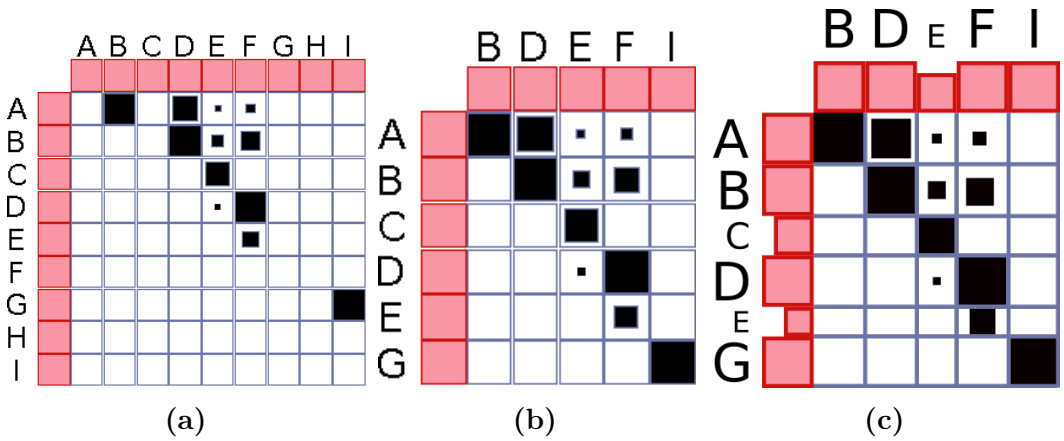
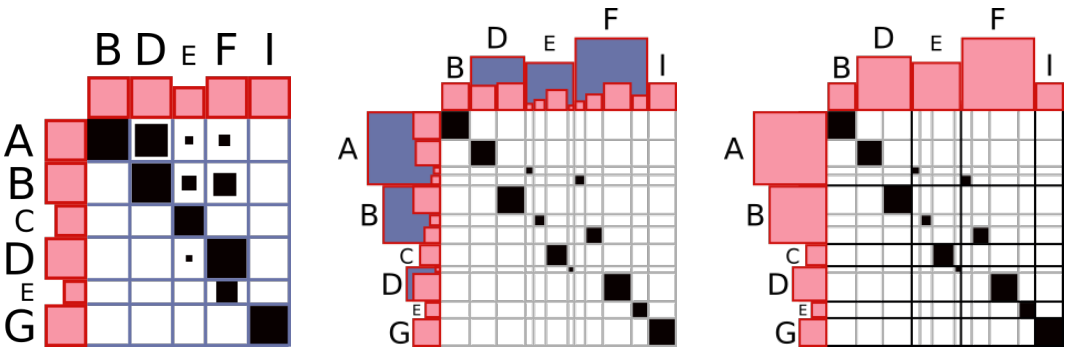


Figure III.13: Retrait des colonnes/lignes n'ayant pas d'arêtes ($poids = 0$) et adaptation des largeurs de lignes et colonnes. (a) avant transformation (b) retrait des lignes et colonnes vides (c) adaptation des largeurs de lignes et colonnes en fonction de l'arc de plus fort poids qu'elles contiennent.

tout en conservant les proportions pour qu'il reste carré.

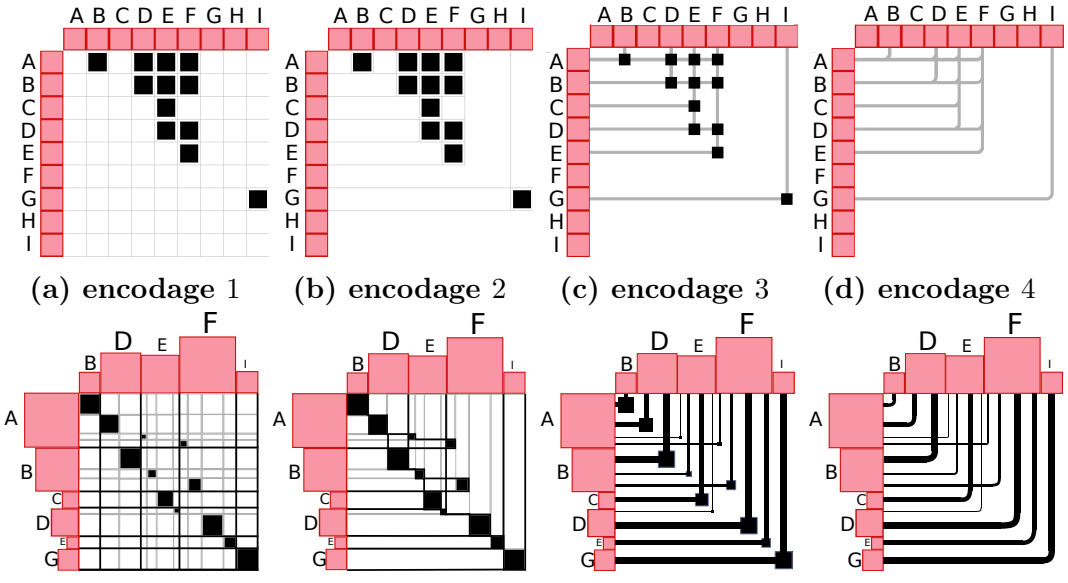
Ensuite les colonnes et lignes sont dupliquées pour chacun des arcs (voir figure III.14). Chaque arc disposant ainsi de ses propres ligne et colonne et la taille du nœud est adaptée afin de correspondre à son degré pondéré. Ces transformations permettent représenter les éléments en fonction de leurs poids respectifs. Pour les nœuds, cela implique de maintenir proches les arcs ayant une extrémité commune. Afin d'éviter les croisements, nous ordonnons les arcs afin de respecter l'ordre des nœuds de la matrice. Nous avons ainsi pu transformer une matrice d'adjacence pondérée afin de représenter visuellement la pondération des arcs et leur impact sur les nœuds. Ces transformations ont



(a) (b) (c)

FIGURE III.14: Attribution d'une ligne et colonne pour chaque arc du graphe. Les nœuds sont tout d'abord dupliqués pour chaque arête puis agrégés. (a) État initial résultant de l'étape précédente. (b) Les lignes et colonnes sont dupliquées pour chaque arc connecté à un nœud. (c) Les nœuds résultants de la duplication sont ré-agrégés. La taille d'un nœud correspond ainsi au poids pondéré de ses arcs incidents.

été appliquées aux 4 encodages visuels des arêtes décrits et évalués dans le chapitre II. La figure III.15 montre les représentations correspondantes.



(a) encodage 1 (b) encodage 2 (c) encodage 3 (d) encodage 4

(e) Encodage 1 (f) Encodage 2 (g) Encodage 3 (h) Encodage 4

FIGURE III.15: Pondération des encodages visuels. En haut, les 4 encodages visuels tels que décrits dans le chapitre II sans information de pondération. En bas, les mêmes encodages visuels avec la transformation permettant de mettre en avant la pondération.

Dans la section suivante, nous allons tirer partie des résultats obtenus lors de nos évaluations afin d'adapter ces transformations et générer une représentation simple d'interprétation et facilitant l'interaction.

2.2 Hybridation des encodages visuels pondérés

Les résultats de nos évaluations montrent que l'encodage visuel des arêtes a peu d'impact sur les performances utilisateurs mais un impact non négligeable sur le confort des utilisateurs. Ces résultats mettent également en évidence l'importance d'un encodage visuel efficace pour l'analyse locale du DOM mais également l'importance de la taille des glyphes pour l'analyse globale. Il est difficile d'envisager utiliser l'un ou l'autre en l'état, tous souffrant de plus ou moins d'inconvénients compliquant la tâche que l'on souhaite permettre. Le tableau III.1 récapitule les avantages et inconvénients des 4 encodages visuels en leur attribuant une note allant de 4 (mauvais) à 1 (bon) sur différents critères. On peut voir que les encodages 1 et 2, utilisant des rectangles vides pour afficher la matrice et contenant des glyphes de grandes tailles, sont critiquables sur l'encombrement visuel mais très efficace pour l'interaction. Au contraire, les encodages 3 et 4, qui utilisent des polylignes, ont un encombrement visuel très réduit et semblent plus agréable à utiliser que les deux autres pour l'identification du nombre d'arêtes dans le graphe ainsi que pour le suivi d'une arête mais ces deux encodages souffrent d'une surface réduite pour les interactions. Il semble intéressant d'utiliser ces deux caractéristiques, polylignes pour le suivi et grande surface d'interaction, et de créer une version hybride de l'encodage visuel des arêtes afin de combiner les avantages respectifs des encodages visuels. Pour cela nous définissons une nouvelle représentation des arcs du graphe par une représentation hybride combinant l'approche polyligne de l'encodage 4 et la surface des rectangles de l'encodage 2. Ce nouvel encodage est défini par un rectangle dont l'épaisseur est fonction du poids de l'arc connectant les deux nœuds dans le graphe. Ainsi, les éléments importants du graphe, que ce soit des nœuds ou des arcs sont aisément identifiés par l'importance qu'ils prennent dans la représentation. La figure III.16 présente les deux encodages visuels utilisés ainsi que le résultat de l'hybridation. L'encodage visuel hybride présente des arcs représentés par des rectangles d'épaisseurs variables et connectant deux nœuds dans la matrice d'adjacence. Ainsi, un nœud ayant des arcs incidents nombreux ou de poids importants occupera un espace plus important qu'un nœud de faible incidence, permettant ainsi de mettre en avant les nœuds tenant un place prépondérante dans le réseau et facilitant ainsi leur identification. On peut voir une certaine ressemblance entre notre version hybride et les diagrammes de Sankey tels que montrés en section 1. Dans les deux cas, les entités et les flux entre entités sont de tailles variables, permettant de montrer leurs importances relatives, ici en fonction de leur pondération.

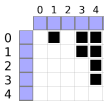
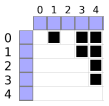
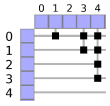
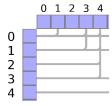
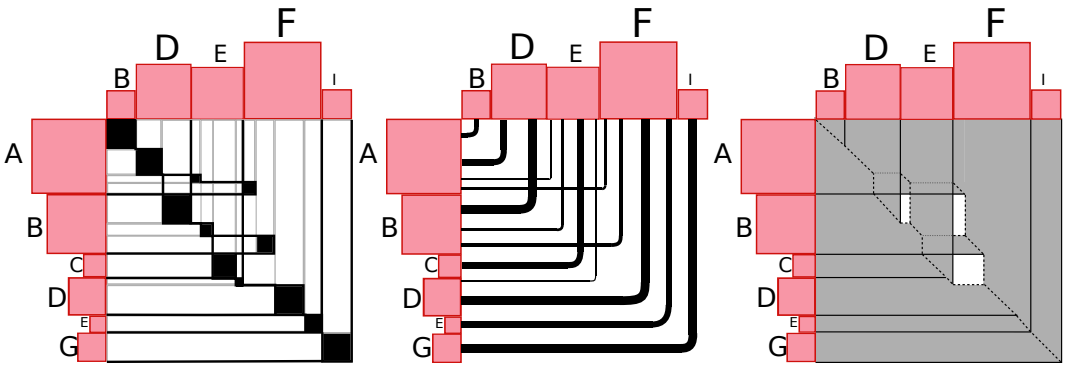
				
Encombrement visuel	4	3	2	1
Vue globale	3	2	1	2
Interaction	1	1	2	4
Ident. nombre d'arêtes	2	2	1	1
Suivi d'arête	3	2	1	1

Tableau III.1: Comparaison des encodages visuels sur 5 critères. Les encodages visuels sont notés sur une échelle allant de 1 (efficace) à 4 (pas efficace).

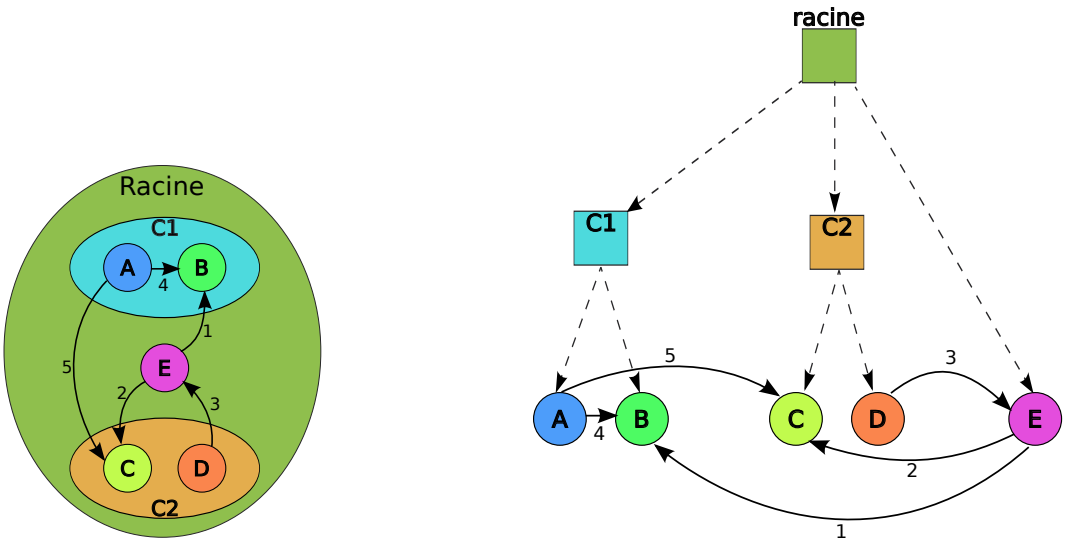


(a) Encodage 2 **(b) Encodage 4** **(c) hybride**

FIGURE III.16: Hybridation des encodages visuels 2 et 4. La version hybride décrite ici reflète une version intermédiaire de la représentation et est difficile à interpréter, notamment à cause de la coloration uniforme des arêtes.

3 Conception

Comme précisé précédemment, nous considérerons les données initiales comme étant un graphe orienté et pondéré associé à un partitionnement hiérarchique.



(a) jeu de données initial et son partitionnement

(b) hiérarchisation en utilisant le partitionnement

Figure III.17: Données utilisées comme paramètres d'entrées pour Adjasankey : (a) Le graphe orienté et pondéré initial partitionné et (b) l'arbre hiérarchique associé.

La figure III.17 présente un exemple de graphe orienté, pondéré et partitionné ainsi que l'arbre de décomposition correspondant tel qu'utilisé pour notre représentation. Ainsi, on peut voir sur la figure que les nœuds A et B sont regroupés dans le groupe *C1* et les nœuds C et D dans *C2*. Ces deux groupes *C1* et *C2* sont représentés par deux méta-nœuds dans la hiérarchie. Cet exemple servira d'illustration tout au long de cette section.

3.1 Coloration

Afin de mettre en valeur le partitionnement hiérarchique du graphe, nous avons mis en place un algorithme qui vise à attribuer des couleurs proches aux groupes parents tout en permettant de les distinguer aisément. L'idée principale de cet algorithme consiste à diviser l'espace de couleurs Teinte-Saturation-Valeur (*TSV*, *HSV* en anglais) afin d'attribuer à chaque nœud et méta-nœud de l'arbre un intervalle de teinte en fonction de son degré pondéré. Pour cela nous attribuons l'intervalle complet à la racine de l'arbre et divisons cet intervalle

de valeur pour chacun de ses descendants en fonction de leurs degrés pondérés. Cette opération est répétée récursivement en divisant l'intervalle attribué à un nœud en fonction du degré pondéré de ses descendants.

De façon plus formelle, on définit T l'intervalle de teinte dans l'espace de couleurs TSV compris entre une valeur de départ d et une valeur de fin f. On définit $T_n = [d_n, f_n[$ l'intervalle de valeur d'un nœud n ayant S descendants et pour chaque nœud s parmi ses descendants ayant un degré pondéré w_k :

$$T_s = [d_s, f_s[$$

tel que

$$d_s = \frac{\sum_{k=0}^{s-1} w_k}{\sum_{k=0}^{S} w_k} \times (f_n - d_n) + d_n$$

et

$$f_s = d_s + \frac{w_s}{\sum_{k=0}^{S} w_k} \times (f_n - d_n)$$

La valeur de teinte finalement attribuée à un nœud est ainsi définie par

$$t_n = (f_n - d_n)/2$$

Ainsi, comme le montre la figure III.18a, l'espace $[0, 360[$ du nœud racine est divisé entre les deux groupes C1, C2 et le nœud E en fonction de leurs degrés pondérés respectifs. Les intervalles de valeurs T_{C1} et T_{C2} sont à nouveau divisés pour chacun de leurs descendants (resp. (A, B) et (C,D)).

Afin de faciliter l'identification de la source et de la destination d'un arc, nous colorons les arcs en fonction des nœuds à ses extrémités. Pour déterminer rapidement et aisément si les connexions sont intra- ou inter- agrégats, nous avons mis en place une interpolation des couleurs pour chaque arc, comme dans [67, 124]. Cette interpolation permet une modification progressive de la couleur de l'arc depuis la couleur du nœud à une extrémité vers la couleur du nœud à l'autre extrémité. Dans le but de faciliter l'identification du nœud à une extrémité de l'arc depuis l'autre extrémité, nous inversons les couleurs des deux extrémités de l'arc. Par exemple, la figure III.18b, montre que les arcs sortants du nœud E sont émis vers les représentations de deux nœuds : B et C. La coloration permet de distinguer les deux arcs sortant du nœud E et peut, lorsque les couleurs de destination sont suffisamment distinctes, faciliter la distinction des proportions (approximativement 1/3 et 2/3 dans le cas présent).

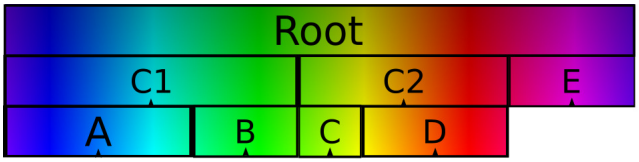
Cette interpolation inversée permet également de fournir un histogramme cumulé des sources/destinations à coté de chaque entité (voir figure III.18b).

Afin de garantir la visibilité des histogrammes, un espace sans interpolation est maintenu près des entités, garantissant l'affichage d'un histogramme non transformé à proximité des nœuds.

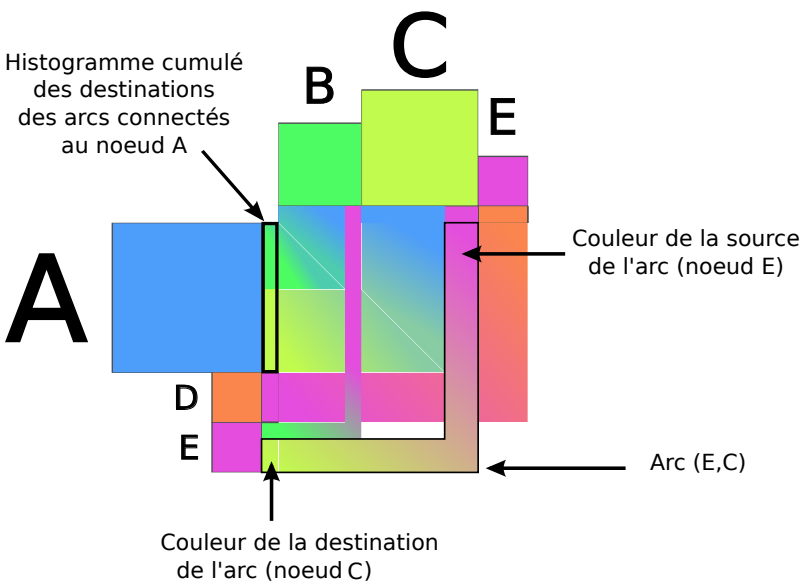
L'algorithme de coloration, bien qu'efficace pour observer la distribution des sources et cibles des nœuds n'est pas optimum. En effet il souffre de deux inconvénients majeurs :

— deux nœuds éloignés dans l'arbre peuvent avoir une couleur proche (voir nœuds B et C dans la figure III.18a) et

— plusieurs nœuds dans le même chemin de parenté (descendance/ascendance) peuvent avoir exactement la même couleur (les couleurs du nœuds D et du méta-nœud C2 dans la figure III.18a sont très proches).



(a) Distribution des couleurs par nœud sur l'échelle de teinte de l'espace *TSV*.



(b) Coloration des arêtes

FIGURE III.18: (a) Distribution des couleurs par nœud sur l'échelle de teinte de l'espace colorimétrique *TSV* (pour *Teinte-Saturation-Valeur*). (b) Interpolation progressive et inverse des couleurs des arêtes.

3.2 Rendu

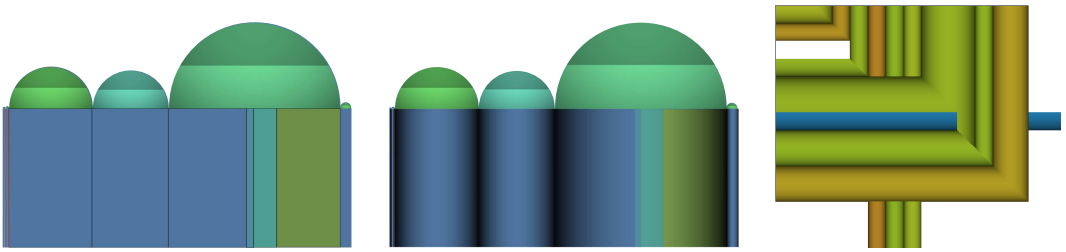
Rendu des arcs

Afin de corriger les inconvénients de la technique de coloration présentée ci-dessus, nous avons utilisé un effet de luminance pour donner un aspect tubulaire aux arcs appartenant à un même faisceau (ayant une source ou une cible commune). Cet effet de rendu a été introduit dans [134] afin de mettre en évidence les différents niveaux hiérarchiques et les relations d'appartenance des éléments dans une treemap. On peut voir sur les figures III.19a et III.19b qu'il est plus simple de déduire l'appartenance des arcs à un faisceau sur la figure III.19b, l'effet de luminance permettant de déduire aisément si deux arcs appartiennent au même faisceau ou non. Non seulement l'effet de regroupement est accentué mais cela permet aussi de faciliter la distinction des croisements et des courbures d'arcs (voir figure III.19c). En effet, l'identification des croisements sans cet effet tubulaire peut-être extrêmement difficile, voir impossible même lorsque les bordures d'arcs sont représentées.

Cette technique a cependant l'inconvénient de modifier l'intensité des couleurs pour un même groupe d'arcs. Ainsi, un faisceau d'arcs n'aura pas exactement la même couleur en fonction de sa position dans le tube. Les faisceaux présents sur les bords du tube auront une couleur transformée vers le noir, tandis que les faisceaux au centre du tube n'auront pas cette transformation. Cela constitue un inconvénient pour les utilisateurs désirant comparer les couleurs des faisceaux.

Rendu des nœuds

Nous l'avons vu auparavant, l'importance des nœuds dans le graphe est représentée par leur aire. L'importance est donc amplifiée de façon quadratique par rapport à la largeur du nœud. Afin de réduire cette amplification et dans un soucis d'optimisation de l'occupation de l'espace (réduction de l'encombrement visuel), nous réduisons leur hauteur de moitié. Cependant, d'autres informations concernant les éléments du graphe peuvent avoir une certaine importance. Afin de permettre l'ajout d'information complémentaire sur les nœuds, nous mettons en place une variation de la saturation de la couleur permettant de représenter un *seuil* sur les nœuds et méta-nœuds du graphe. On peut voir ce seuil sur les nœuds des figures III.19a et III.19b. Ce seuil peut avoir différents rôles, nous en présenterons un dans la section 4. Afin de maintenir une cohérence graphique, nous utilisons également l'effet de luminance utilisé sur les arcs pour les nœuds et modifions la forme des nœuds afin qu'ils forment un arrondi aux extrémités des arcs.

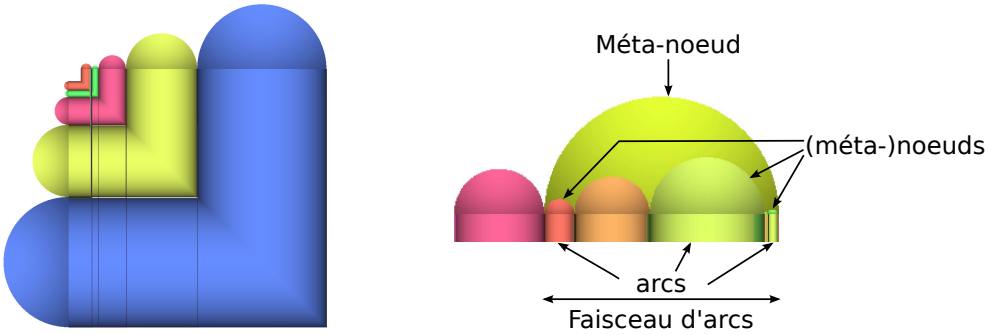


(a) Sans effet de luminance (b) Avec effet de luminance (c) Distinction des arêtes contiguës

Figure III.19: (a) Méta-arcs sans effet de luminance. (b) Méta-arcs avec effet de luminance. (c) L'effet de luminance permet de différencier les brisures de polylignes des croisements d'arcs par la rupture que cela provoque sur l'effet tubulaire. (a) et (b) présentent également un *seuil* sur les entités du graphe pouvant indiquer diverses informations.

3.3 Interaction

La première image affichée à l'utilisateur est le plus haut niveau d'abstraction fourni par le partitionnement hiérarchique en dehors de la racine (voir figure III.20a). Afin de faciliter la navigation dans la représentation, un outil d'interaction permettant de modifier le niveau de détail est intégré à Adjasankey. Grâce à cet outil, l'utilisateur peut demander l'affichage de plus ou moins de



(a) Visualisation d'un graphe avec Adjasankey. (b) ouverture d'un méta-nœud.

Figure III.20: (a) La première représentation fournie par Adjasankey correspond au niveau d'abstraction le plus élevé, quelques groupes représentent l'ensemble des entités du graphe. (b) À gauche en rose, un méta-nœud fermé, à sa droite en jaune, un méta-nœud ouvert. L'ensemble des nœuds qui le composent entre dans l'espace qui lui est réservé, ainsi l'ouverture n'implique pas de modification de l'agencement spatial des entités dans la représentation.

détails sur un groupe particulier. La demande d'ouverture d'un groupe provoque l'apparition des entités qui le composent (nœuds et/ou méta-nœuds) et la séparation des méta-arcs en leurs connexions respectives (arcs et/ou méta-arcs). Les nœuds/méta-nœuds sont représentés par-dessus le méta-nœud dont on vient de demander l'ouverture, permettant de maintenir le contexte d'appartenance de chaque entité nouvellement affichée.

On peut ainsi voir en figure III.20b que le méta-nœud ouvert jaune reste représenté en arrière-plan et les éléments qu'il représentait sont dessinés par-dessus. En conservant l'affichage du méta-nœud ouvert, l'utilisateur sait à tout instant à quels groupes les entités affichées appartiennent.

Cette interaction est compatible avec notre technique de positionnement des arêtes qui attribue à chaque (méta-)arête et (méta-)nœud un espace qui lui est propre dépendant de son poids. Il est donc possible de subdiviser un élément afin de représenter les éléments qui le constitue sans pour autant devoir modifier l'agencement spatial dans la représentation.

En complément à cet interacteur, nous fournissons aussi des outils classiques : zoom et déplacement dans la représentation, et identification des voisins d'un nœud ou des extrémités d'un arc par mise en surbrillance.

4 Cas d'étude

Nous présentons ici les résultats obtenus à partir de données de navigation d'utilisateurs sur un site internet. Après nettoyage, ces données représentent 46.500.000 hyperliens utilisés dans un site de vente entre particuliers contenant approximativement 50.000 pages internet et 823.000 hyperliens. Les nœuds représentent les pages du site internet et les arcs représentent les hyperliens qui permettent de naviguer d'une page à l'autre. L'orientation du graphe est donc fournie par les hyperliens et la pondération par le nombre d'utilisation (clic) de ces hyperliens. L'arbre hiérarchique associé au graphe nécessaire à l'utilisation d'Adjasankey est fourni par le partitionnement hiérarchique du site, et plus précisément par les catégories et différents niveaux de sous-catégories contenus dans l'url des pages.

La première image présentée par Adjasankey (voir figure III.21) représente le plus haut niveau d'abstraction du jeu de données, c'est-à-dire les catégories et hyperliens cliqués entre les catégories les plus généralistes. La première observation permet d'identifier trois points distincts :

- le site contient 5 catégories principales,
- il n'y a pas de changement de couleur des arcs (l'interpolation se fait entre deux nœuds identiques, et donc de même couleur) indiquant que les relations sont essentiellement intra-catégories, et
- il existe quelques arcs inter-catégories de très faible poids, visibles par la présence de tracés verticaux et horizontaux qui *rompent* les arcs.

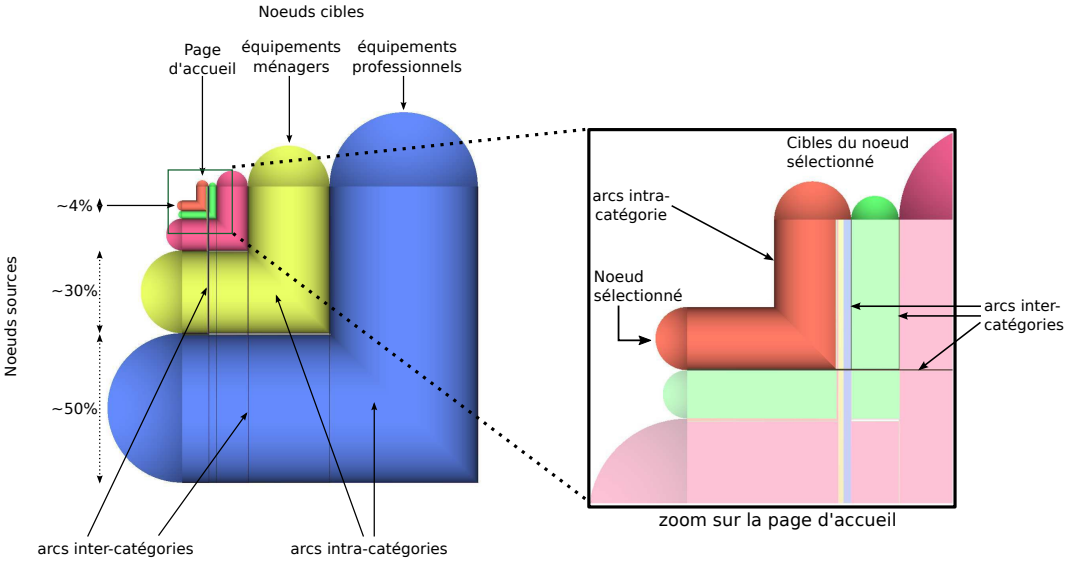


FIGURE III.21: Vue du plus haut niveau d'abstraction d'adjasankey avec zoom et mise en surbrillance de la page d'accueil. La sélection d'un nœud entraîne la mise en surbrillance de ce nœud ainsi que des différents arcs (inter- et intra- catégories) et du nœud à leurs extrémités. Malgré la faible épaisseur des arcs inter-catégories (désignés par les flèches à droite du zoom), ceux-ci restent distinguables grâce à la mise en surbrillance.

Parmi ces 5 catégories, deux d'entre elles représentent plus des 3/4 des arcs pondérés. L'une (en bleu dans la figure) représentant les *équipements professionnels* centralise approximativement 50% des pages visitées et l'autre (en jaune) représentant les *équipements ménagers* et concerne 30% des pages visitées du jeu de données.

Le nœud représentant la page d'accueil (voir zoom figure III.21) fournit des informations relativement surprenantes. En effet on peut observer quatre points importants :

- La page d'accueil du site de vente ne représente que 4% des pages visitées.
- La présence d'une arête majeure représentant des connexions depuis la page d'accueil vers la page d'accueil.
- Peu de connexions entrantes inter-catégories.
- Peu de connexions sortantes inter-catégories.

En ce qui concerne le premier point, deux explications sont plausibles pour justifier ces résultats : tout d'abord, il est possible que beaucoup d'utilisateurs utilisent un moteur de recherche pour accéder directement à une catégorie spécifique du site. Ensuite les utilisateurs passent relativement peu de fois par la page d'accueil comparé aux autres catégories/pages du site, il s'agit d'un

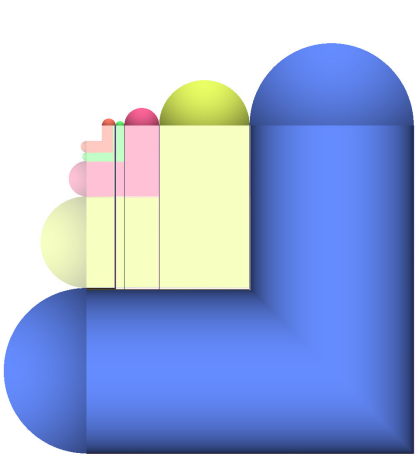
point d'entrée mais il y a peu de raison d'y aller de façon récurrente. Cela se remarque notamment par le peu de flux inter-catégories ayant pour destination le nœud de la page d'accueil. Pour le deuxième point, il pourrait provenir de personnes voulant actualiser la page ; pour tester le bon fonctionnement de leur connexion ou la bonne mise à jour du site par exemple. Le peu de connexions entrantes peut se justifier de la même manière que pour le point sur l'importance de la page d'accueil, il y a peu de personnes qui retournent sur la page d'accueil durant leur visite. Le dernier point cependant est très surprenant et une explication pourrait être l'arrivée sur le site internet via un moteur de recherche qui amène directement à la page concernée sans passage par la page d'accueil. Une autre supposition met en cause l'intégrité du jeu de données utilisé. Certaines visites ayant dû être retirées, il est fort probable que le retrait de ces données provoque ce genre d'incohérences.

Si on observe la catégorie *matériels professionnels* de façon plus précise (voir figures III.22a et III.22b), une sous-catégorie majeure est visible, les *équipements agricoles* (en jaune). Cette sous-catégorie représente approximativement 50% des pages visitées de la catégorie *matériels professionnels* et 25% de l'ensemble des (méta-)nœuds. Les autres sous-catégories de *matériels professionnels* sont de tailles beaucoup plus réduites et représentent soit d'autres types d'équipements professionnels, soit une division géographique (région ou département). On peut ainsi observer deux types de comportements utilisateurs :

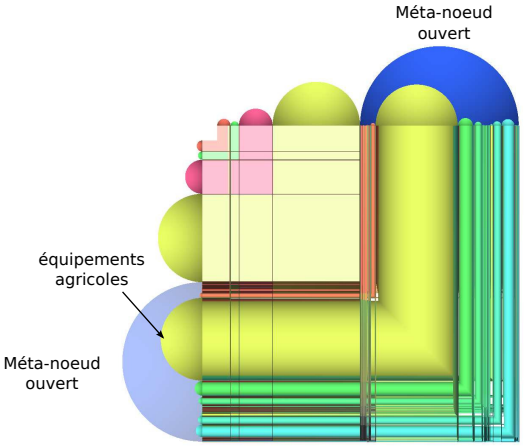
- Les utilisateurs qui utilisent de préférence les sous-catégories correspondant aux divisions territoriales, et
- les utilisateurs qui naviguent dans les sous-catégories sémantiques pour ensuite préciser la division territoriale qui les intéresse.

Ce comportement révèle donc une distinction entre les utilisateurs qui favorisent la proximité géographique ou la précision sémantique dans leurs recherches. Puisqu'une page ne peut apparaître dans deux catégories différentes, cela indique aussi qu'en utilisant l'une ou l'autre des stratégies de recherche, l'utilisateur n'obtiendra pas les même résultats lors de sa recherche. Cela permet donc de révéler un problème dans l'architecture du site internet. Enfin, la plupart des hyperliens utilisés parmi les sous-catégories des *équipements professionnels* sont une fois encore intra-catégories, c'est-à-dire qu'après avoir choisi une catégorie, peu d'utilisateurs décident de modifier ce choix. Cela peut traduire le fait que le site soit correctement structuré, les utilisateurs ne se trompent que rarement de catégories pour trouver un produit, et donc ne ressentent pas le besoin d'en changer.

Si l'on considère maintenant la catégorie des *équipements ménagers* (voir figure III.23), on peut observer deux sous-catégories principales correspondant au *jardinage* (en vert) et aux *chaussures* (en orange), une catégorie de taille intermédiaire, les *habillements premiers âges* et deux sous-catégories de plus petite taille, *bagagerie* et *outillage*.



(a) Section des Matériels professionnels



(b) Ouverture et focus du méta-nœud le plus important (cible et source)

FIGURE III.22: (a) La catégorie dédiée au *Matériels professionnels* est la plus importante du jeu de données étudié. La présence d'un méta-arc reliant ce méta-nœud en source et en destination indique que l'essentiel des connexions sont intra-catégories. (b) Ouverture et focus des méta-nœuds source et cible, révélant un méta-nœud majeur et plusieurs méta-nœuds de taille moins importante. Là aussi les connexions sont essentiellement intra-catégories même si quelques connexions inter-catégories peuvent être observées grâce au croisements des tubes.

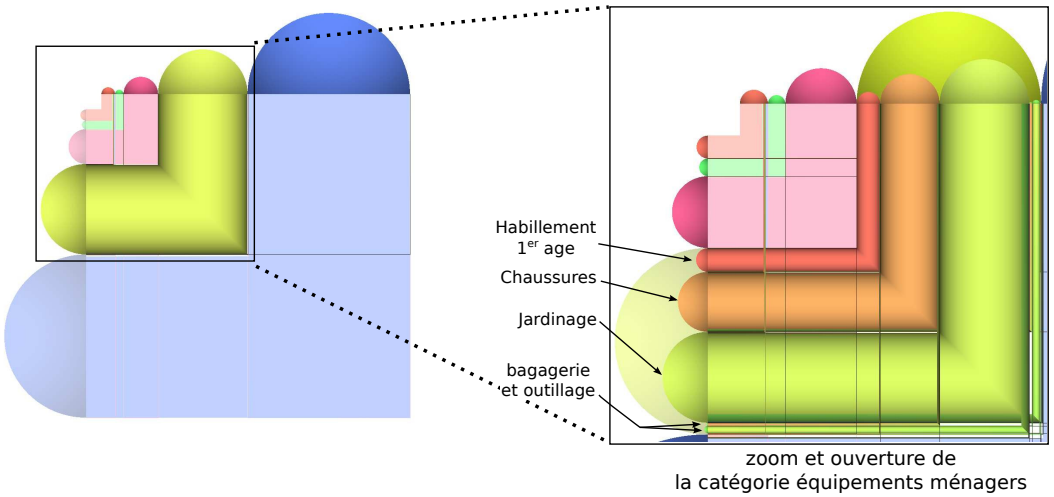


FIGURE III.23: La catégorie dédiée aux *Équipements ménagers* est la seconde plus importante du jeu de données étudié. Elle aussi présente essentiellement des connexions intra-catégories. L'image montre l'ouverture et le focus du méta-nœud source et cible, révélant trois méta-nœuds de taille relativement importante et quelques-uns de taille plus réduite. Là aussi les connexions sont essentiellement intra-catégories avec quelques connexions inter-catégories.

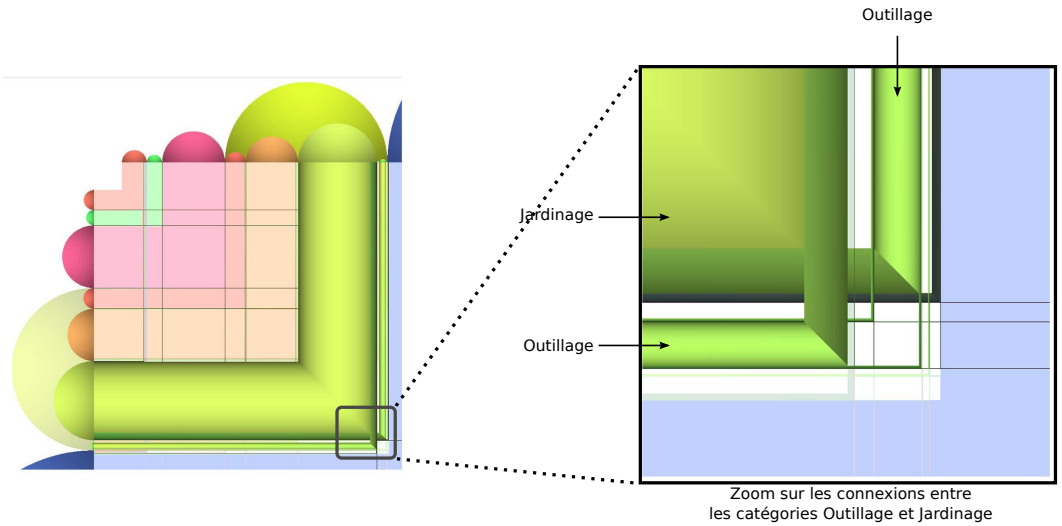


FIGURE III.24: Mise en évidence du transfert entre les méta-nœuds représentant les sections *jardinage* et *outillage*.

Tandis qu'ici aussi la plupart des hyperliens cliqués sont intra-catégories, deux comportements intéressants peuvent être observés :

1. la quasi totalité des hyperliens cliqués connectés à la catégorie *outillage* proviennent ou se dirigent vers la catégorie *jardinage* (voir figure III.24) ,

et de façon similaire,

2. la quasi totalité des hyperliens cliqués connectés à la catégorie *bagagerie* proviennent ou se dirigent vers la catégorie *chaussures*.

On peut voir en figure III.24 qu'approximativement 90% des hyperliens cliqués depuis la section *outillage* redirigent vers la section *jardinage*. Notre hypothèse suggère que ces déplacements résultent d'une mauvaise classification des offres. En effet, la plupart des annonces contenues dans la section *jardinage* font très probablement référence à des outils de jardinage, la présence de deux catégories distinctes apportant de la confusion dans la navigation des utilisateurs. La section *jardinage* devrait donc faire partie intégrante de la section *outillage* en tant que sous-section. Le même raisonnement s'applique en ce qui concerne le cas présenté des hyperliens cliqués entre les sections *bagagerie* et *chaussures*.

5 Conclusion

Nous avons présenté dans ce chapitre Adjasankey, une nouvelle technique de représentation conçue pour la visualisation de graphe orienté pondéré avec partitionnement hiérarchique. À partir des résultats obtenus lors d'évaluation utilisateurs, nous avons développé une nouvelle méthode de visualisation. Cette méthode associe la conception visuelle des diagrammes de Sankey, efficace pour transmettre l'information de pondération et son évolution, au positionnement orthogonal des entités caractéristique des matrices d'adjacences. Ainsi, Adjasankey permet de transmettre de manière efficace l'information de connexions entre des entités mais aussi celle de la pondération des connexions qui les relient, tout en optimisant la lisibilité globale du réseau. Nous avons également intégré des outils d'interactions permettant de naviguer dans le partitionnement hiérarchique, fournissant ainsi aux utilisateurs la capacité d'explorer des réseaux multi-échelles, et donc d'obtenir différents niveaux de détails et de précision sur les éléments (entités et connexions) qui composent le graphe et l'information qu'il contient.

Chapitre IV

Les Coordonnées parallèles abstraites

1 Introduction - État de l'art

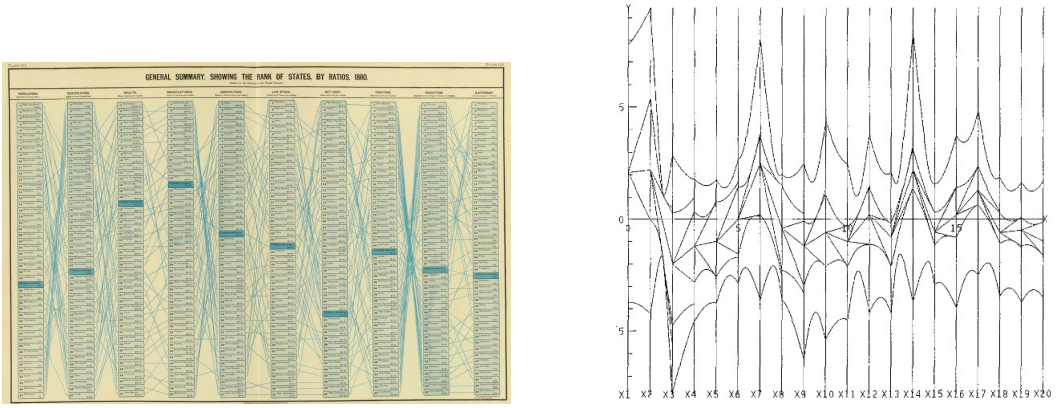
On appelle données multivariées ou données multi-dimensionnelles un ensemble d'objets ayant plusieurs attributs. Ce type de données est utilisé dans de nombreux domaines ayant recours à l'analyse de données, que ce soit dans les domaines des sciences (biologie, médecine, physiques), de l'économie, social, *etc.* L'évolution des technologies a par ailleurs pour effet de faciliter l'acquisition de ce type d'information. Le problème ne réside cependant pas dans l'acquisition mais bien dans l'exploitation : la quantité de données mais aussi la complexité des données augmentent et il devient difficile de les explorer et d'en extraire les informations qu'elles contiennent. Les représentations interactives jouent ainsi un rôle prépondérant pour les étapes d'explorations et d'analyses, notamment par la mise en évidence de corrélations, anti-corrélations ou motifs divers.

La suite de cette section est un rappel de l'état de l'art des techniques de représentations utilisant des coordonnées parallèles et des approches utilisées pour réduire l'encombrement visuel. La section 1.3 présente les contributions ainsi que l'organisation générale du chapitre.

1.1 Les coordonnées parallèles

Quelques techniques dédiées à la visualisation de données multivariées ont été décrites dans la littérature en dehors des représentations classiques qui peuvent efficacement représenter deux à trois dimensions (nuages de points, histogrammes de fréquences, *etc*). Les travaux de [52] et plus récemment l'étude de [25] présentent la majorité de ces techniques comme les matrices de nuages de points, hyperboxes, histogrammes multi-dimensionnels, cartes de chaleurs, diagrammes en étoiles, *etc.* Cependant, ces techniques souffrent soit d'une limite en ce qui concerne la taille de jeu de données utilisable, soit d'une faible lisibilité

lorsque le nombre d'attributs devient important. Ainsi, une des techniques les plus utilisées et prenant en compte ces paramètres est celle des coordonnées parallèles. Une façon de se représenter les coordonnées parallèles est d'imaginer un tableau à plusieurs lignes et colonnes. Les colonnes représentent les attributs des données et les lignes les éléments. Chaque case correspond donc à la valeur d'un attribut pour un élément. Cette technique représente les attributs par des axes verticaux parallèles (des dimensions) et les éléments des données par des polylignes. Chaque polyligne croise les axes à la valeur de l'élément pour l'attribut concerné. Ce sont les travaux de [72] et [71] qui ont permis de diffuser la technique et l'ont rendue connue mais les premières apparitions de cette représentation sont cependant antérieures avec notamment la représentation de [45] (voir figure IV.1).



(a) **(b)**

Figure IV.1: (a) Première présentation des coordonnées parallèles par [45] dans le cadre d'un classement des états nord-américains. (b) utilisation des coordonnées parallèles par [72].

Depuis sa première présentation, les coordonnées parallèles ont souvent été utilisées et décrites, on peut ainsi se référer à la revue de la littérature de [57] pour une description détaillée. Les auteurs y présentent notamment les divers travaux effectués utilisant les coordonnées parallèles, que ce soit sur la technique de représentation orientée géométrie ou densité, les transformations apportées aux axes, des présentations de cas d'utilisation, les techniques d'interaction, *etc.* En effet de nombreux travaux s'appuient sur cette technique car elle présente l'avantage de fournir une visualisation de l'intégralité des données. Cependant, de part sa capacité à exposer l'intégralité des attributs numériques et catégoriels des données, cette technique souffre d'importantes limites lorsqu'on considère la visualisation de jeu de données de taille importante. Rapidement, les représentations résultantes souffrent d'encombrement visuel et de ralentissement du temps de réponse, que ce soit pour les calculs, le dessin ou l'interaction. Puisque les éléments croisent les axes indépendamment les uns des autres, cela

provoque de nombreux croisements de polylignes, ce qui réduit la lisibilité et empêche la détection de tendances ou corrélations. De plus, les temps nécessaires aux calculs, dessin et interaction empêche de considérer l'utilisation de cette technique dans sa version de base lorsque le jeu de données atteint des millions ou milliards d'éléments.

1.2 Réduction de l'encombrement visuel

Une première approche pour prendre en compte le problème d'encombrement visuel consiste en l'utilisation de la transparence ou des cartes de chaleurs pour mettre en évidence les zones d'importances (*e.g.* Artero *et al.* [12], Wegman et Luo [143], Zhou *et al.* [151]), on parle ainsi d'approches orientées densités.

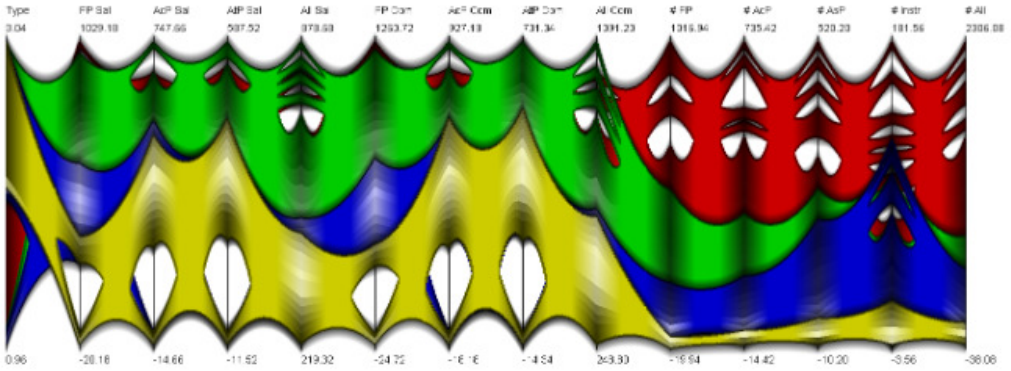
Malgré l'efficacité de ces approches pour réduire l'encombrement visuel, elles ne sont cependant pas adaptées pour prendre en charge des jeux de données à grandes échelles. En effet, plus le nombre d'entités à représenter augmente, plus les temps nécessaires pour calculer et afficher la représentation augmentent. Ces délais provoquent ainsi des latences durant l'utilisation de l'outil notamment pendant la phase de chargement des données et les phases d'interaction ce qui est un frein à son utilisation. Bien qu'utile, cette technique est cependant limitée par la définition maximale que peut atteindre la technique de densité utilisée [1] et la capacité de distinction de l'œil humain. Ces deux facteurs laissent peu de possibilités pour la distinction de nombreuses valeurs différentes.

Une autre approche consiste à construire une abstraction du jeu de donnée initial. Cette abstraction est généralement construite en agrégeant des éléments qui partagent une ou plusieurs caractéristiques. Dans ces techniques, les connexions similaires sont agrégées afin de mettre en valeur les tendances majeures et de réduire l'encombrement. Cette approche par abstraction a déjà été présentée auparavant (voir chapitre III) où nous présentions une agrégation des éléments en fonction de leurs parentés dans un arbre hiérarchique. Contrairement aux techniques présentées ci-dessus, l'abstraction a plusieurs avantages : en réduisant la taille du jeu de données à représenter, elle réduit le nombre d'entités à afficher et donc le risque d'encombrement visuel tout en accélérant les temps de rendu. Par exemple, McDonnell et Mueller [95] mettent en avant dans leur représentation une agrégation des polylignes par des enveloppes superposées. Comme le montre la figure IV.2a, les éléments sont agrégés en groupes, chacun d'une couleur spécifique, et ces groupes sont représentés par des enveloppes. Ces enveloppes représentent, sur chacune des dimensions, les intervalles de valeurs contenant les éléments du groupe. Ils s'appuient également sur la mise en évidence des zones d'importance par une échelle de couleurs (de blanc à noir) pour représenter la distribution des éléments entre et par

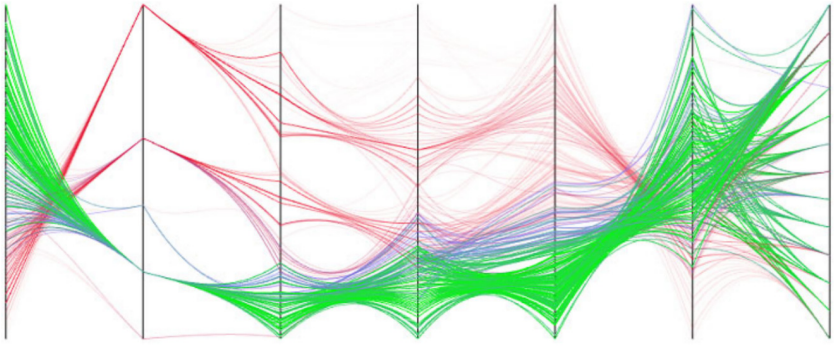
1. une échelle de couleur monochromatique et la transparence par exemple sont basées sur un intervalle de 255 valeurs

dimension.

Zhou *et al.* [152] proposent de leur coté une technique qui agrège les connexions en utilisant un modèle d'interaction de forces pour réduire les interférences visuelles (voir figure IV.2b). Pour cela, ils s'appuient sur une technique de faisceautage d'arêtes dans un graphe similaire à celle décrite dans [67]. Palmas *et al.* [102] utilisent quant à eux des *splines* pour mettre en avant les connexions majeures et les différentes propriétés des agrégats utilisés telles que les valeurs minimale et maximale et la zone de forte densité d'arêtes (voir figure IV.3b).



(a)



(b)

FIGURE IV.2: Différentes techniques de coordonnées parallèles présentées dans la littérature. (a) McDonnell et Mueller [95] et (b) Zhou *et al.* [151].

Par ailleurs, comme le montrent Johansson *et al.* [74] et Zhou *et al.* [151] dans leurs articles, cela permet d'aider l'utilisateur à concentrer son analyse sur les parties importantes en mettant en avant les éléments intéressants des données. De nombreux travaux explorant cette piste peuvent être trouvés dans la littérature ; certains se concentrent plus particulièrement sur l'agrégation des éléments pour l'intégralité du jeu de données. On peut notamment citer

Fua *et al.* [42], Johansson *et al.* [74], McDonnell et Mueller [95]). D'autres se sont plutôt intéressés à une agrégation des éléments différentes pour chaque dimension du jeu de données avec notamment Andrienko et Andrienko [8] et Palmas *et al.* [102] déjà présentés plus haut (voir figure IV.3). Andrienko et Andrienko [8] s'intéressent notamment à la visualisation d'éléments regroupés par classes, calculées pour chaque dimension, avec une représentation par enveloppes. Afin de permettre l'exploration des données en fonction de leurs distributions, ils proposent en complément une représentation des données par des ellipses qui peuvent être de deux types. La première représentation montre les différentes classes utilisées précédemment tandis que la seconde montre une agrégation des données en groupes de tailles égales. Ces deux types de regroupement peuvent ensuite être utilisés par l'utilisateur pour son analyse.

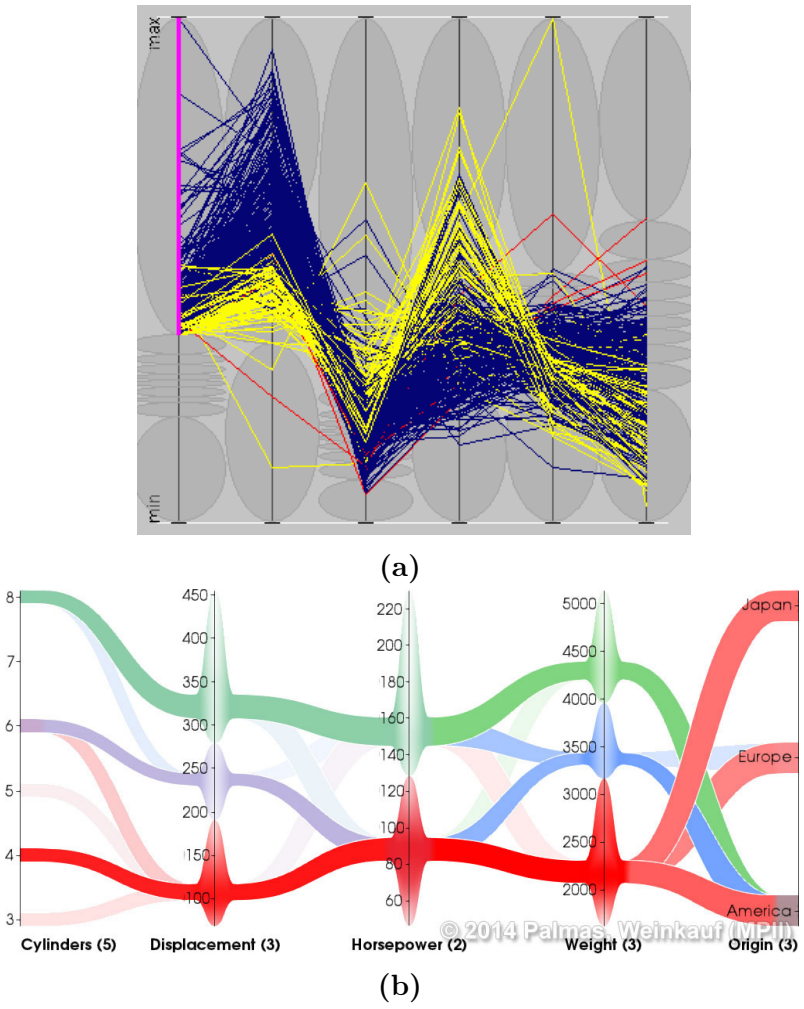
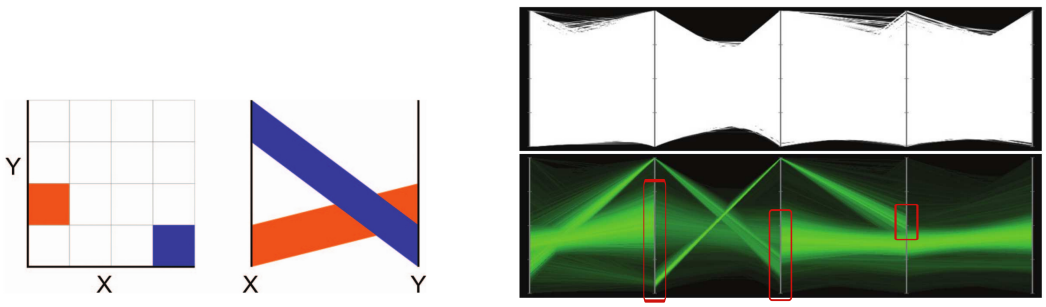


FIGURE IV.3: Techniques de coordonnées parallèles utilisant l'agrégation des éléments. (a) Andrienko et Andrienko [8] et (b) Palmas *et al.* [102].

D'autres techniques encore s'appuient sur des approches d'agrégations par regroupement des données par classe (*binning* en anglais) permettant d'agréger les éléments entre deux axes (voir figure IV.4a) tels que Novotny et Hauser [101] ou une amélioration de leurs travaux par Rubel *et al.* [116]. Le *binning* consiste à diviser un intervalle en n parts et à y répartir les éléments en fonction de leur valeur. De part la notion d'intervalle pondéré par le nombre d'éléments inhérent à la technique, Novotny et Hauser [101] considèrent cette technique d'agrégation comme étant une approche orientée *histogramme*. Les auteurs utilisent cette forme pour utiliser des techniques de détection d'anomalies propres à l'analyse de traitement de signal. Cependant, l'inconvénient majeur de leur technique réside dans les discontinuités que cela peut provoquer dans la représentation. On peut voir cet effet sur la figure IV.4b, où la figure en haut montre les coordonnées parallèles traditionnelles et dessous, la représentation obtenue par *binning*. La représentation abstraite est clairement plus lisible et on détecte facilement les grandes tendances dans les données, cependant les ruptures que l'on peut voir au niveau des axes encadrés en rouge rompent la linéarité usuelle de la représentation et peuvent être très perturbantes et représenter un frein à l'analyse pour l'utilisateur.



(a) **(b)**

FIGURE IV.4: Agrégation par binning entre deux axes telle que présentée par [101]. (a) Détails de la technique d'agrégation. (b) Exemple de coordonnées parallèles, il y a bien agrégation des éléments, permettant de voir les tendances entre chaque couple d'axes mais cela provoque des discontinuités dans la représentation pouvant perturber l'utilisateur dans son analyse.

Bendix *et al.* [19] et Kosara *et al.* [86] présentent eux un outil de visualisation et d'analyse dédié aux données catégorielles. Avec cet outil, chaque catégorie est représentée par un rectangle dont la taille représente le nombre d'éléments dans la catégorie, représenté sur des axes horizontaux sur la figure IV.5. En complément, des histogrammes sont placés à l'intérieur même des rectangles afin de mettre en avant le degré d'indépendance entre chaque catégorie et ses dimensions voisines (dans la figure, représenté sur l'axe central). Plusieurs

outils d'interaction sont associés à la représentation afin de permettre la sélection, la modification manuelle des catégories ou le produit croisé de plusieurs dimensions.

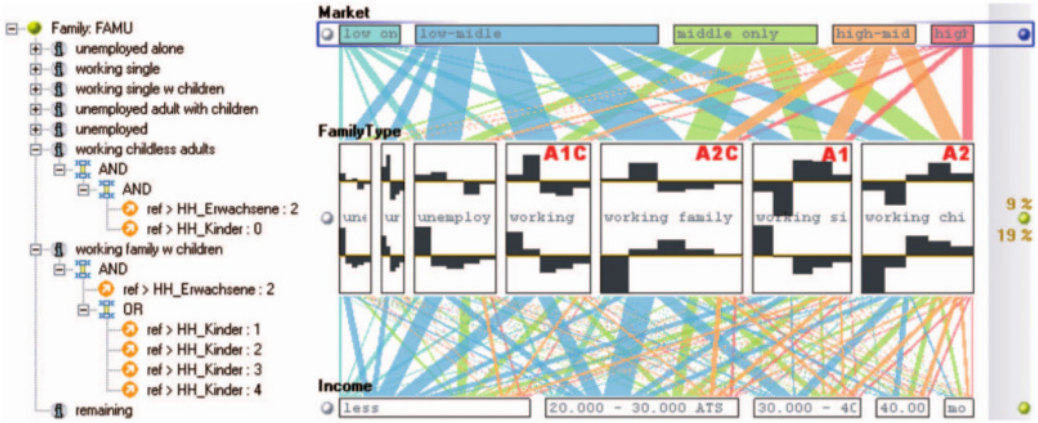


Figure IV.5: .
ParallelSets présenté par [86]

1.3 Contributions

Nous décrirons ici une technique de visualisation, exploration et interaction de données avec des coordonnées parallèles dédiées aux très grandes masses de données. Pour cela, nous nous appuierons sur les technologies et techniques de gestion et traitement de données massives, telles que les architectures déportées (appelée *architecture cloud*), de la parallélisation des processus et de l'abstraction de données. Nous montrerons ainsi qu'il est possible de fournir un outil de visualisation et d'interaction en temps réel pour visualiser et manipuler de très grands jeux de données. Une étape de pré-calcul des données utilisant l'architecture *cloud* que nous présenterons dans le chapitre V associée au client léger de visualisation présenté ici rend ainsi possible la visualisation interactive de coordonnées parallèles abstraites en temps réel. En effet, l'utilisation d'abstraction de données permet de représenter l'ensemble des éléments et des connexions dans les données de façon synthétique. De plus, la taille réduite des données abstraites associée à une technique de rendu bas-niveau permet de fournir un système d'interaction réactif comme support à l'exploration pour l'utilisateur.

La suite de ce chapitre suivra l'organisation suivante : en section 2, nous introduirons les objectifs globaux ainsi qu'une présentation générale de la technique. Ensuite, nous présenterons les détails relatifs à l'abstraction de données en section 3 ainsi qu'à la technique de représentation et du client de visualisation en section 4. Nous présenterons également les différents outils

d'interaction mis en place pour permettre l'exploration de données et finirons avec la présentation d'un cas d'étude en section 5 afin de montrer l'efficacité de la technique. Pour finir nous présenterons nos conclusions et travaux futurs en section 6.

2 Principe de fonctionnement

Comme précisé en introduction, nous nous focaliserons ici sur des données dont le volume est tellement important qu'il est impossible d'envisager de les stocker ou de les utiliser sur un simple ordinateur. Dans ce contexte, nous proposons un système complet combinant une infrastructure *Big Data* pour les pré-calculs et un client léger pour la visualisation. C'est sur ce dernier que se concentrera ce chapitre, l'infrastructure *Big Data* étant décrite de façon plus exhaustive dans le chapitre V. Cette association nous permet de fournir un outil de visualisation permettant la représentation de données massives avec une interaction en temps réel.

La première étape du système consiste en l'abstraction du jeu de données initial. Cette opération a déjà été présentée et a montré son efficacité à réduire l'encombrement visuel dans les coordonnées parallèles dans les travaux de Palmas *et al.* [102]. Pour cela, nous effectuons une étape de création des agrégats pour chaque dimension du jeu de données. Plusieurs algorithmes existant sont envisageables, à condition que les intervalles de valeurs couverts par les agrégats ne s'intersectent pas. Nous avons fait le choix de mettre en place trois algorithmes de partitionnement : l'algorithme des k-moyennes (*k-means* en anglais) présenté par [93] et deux algorithmes de *binning*. L'algorithme de *k-moyennes* crée k sous-ensembles de points de façon à minimiser les distances entre les points de chaque sous-ensemble. Les algorithmes de *binning* séparent l'intervalle de valeurs en k parties et répartissent les données dans ces parties. Le premier effectue une répartition par distribution uniforme des éléments (le même nombre d'éléments dans chaque partie) tandis que le second en intervalles uniformes (intervalles de mêmes tailles et répartition des éléments dans les intervalles correspondant à leurs valeurs). L'abstraction résultante est ensuite utilisée en fonction des besoins. Ainsi, la création d'un jeu de données abstrait, et par conséquent de taille réduite, rend possible le transfert de données sur le réseau et la visualisation sur un client léger déporté. Ce client est donc consacré à la représentation des coordonnées parallèles abstraites. Afin de permettre des temps de rendu courts et des animations fluides, celui-ci s'appuie sur des techniques de rendu bas-niveau. L'avantage principal de notre technique de visualisation est de permettre l'utilisation des coordonnées parallèles ainsi que quelques optimisations présentées dans la littérature pour l'exploration de données massives.

3 Agrégation des données

3.1 Cas général

Nous considérons ici qu'une dimension comporte de manière générale, beaucoup de valeurs distinctes. Cela est généralement vrai lorsqu'on considère des attributs de valeurs réelles mais cela peut aussi être le cas avec des valeurs entières. La première étape de notre système consiste donc à abstraire toutes ces valeurs en un faible nombre de représentants ou d'agrégats.

Comme mentionné précédemment, cela se fait par un algorithme permettant d'agréger les valeurs proches pour chaque attribut indépendamment les uns des autres. Pour cela, de nombreux algorithmes peuvent être trouvés dans la littérature, nous avons essentiellement utilisé l'algorithme des *k-means* présenté dans [93] car le nombre d'agrégats à générer est paramétrable, permettant de limiter la taille de l'abstraction. D'autres cependant sont aussi utilisables, à la condition que les intervalles couverts par les agrégats aient une intersection vide. On peut ainsi citer les algorithmes de [94], [149], [38] ou [63] par exemple.

Pendant le processus de création de l'abstraction, différentes informations sont perdues. Pour pallier à cette perte, nous calculons et stockons : le nombre d'éléments appartenant aux agrégats ainsi que les valeurs minimales, maximales et la distribution des valeurs dans les agrégats. Un calcul similaire est effectué pour les connexions, toutes les connexions entre deux agrégats sont regroupées et remplacées par une unique connexion pondérée. Afin d'accélérer les temps de réponse pendant les interactions utilisateurs, nous effectuons ces agrégations dans l'espace de données et transférons les données d'intérêt dans l'espace visuel. Cela signifie que les connexions existantes entre chaque agrégat de chaque dimension sont agrégées et les résultats stockés ; ce qui permet de retrouver rapidement les connexions entre deux dimensions consécutives quel que soit l'ordre des attributs visualisés. Si l'on observe la figure IV.6, chaque combinaison d'ordre des attributs dans les coordonnées parallèles est calculée et stockée (voir figure IV.6a), ainsi on peut fournir à la demande de l'utilisateur ou durant une interaction, les connexions représentantes quel que soit l'ordre des dimensions visualisées (voir figures IV.6b, IV.6c et IV.6d).

3.2 Cas particuliers

Deux cas particuliers peuvent être rencontrés dans ces processus d'abstraction : le cas des valeurs ordinales et le cas où peu de valeurs distinctes sont présentes pour l'attribut. En ce qui concerne les valeurs ordinales, l'étape d'agrégation doit prendre en compte la sémantique de ces valeurs pour produire une abstraction qui aura un sens. Nous considérons que dans ce cas particulier, cette information est fournie en tant que paramètre afin de pouvoir agréger les valeurs ayant un sens proche. Dans le cas où peu de valeurs distinctes

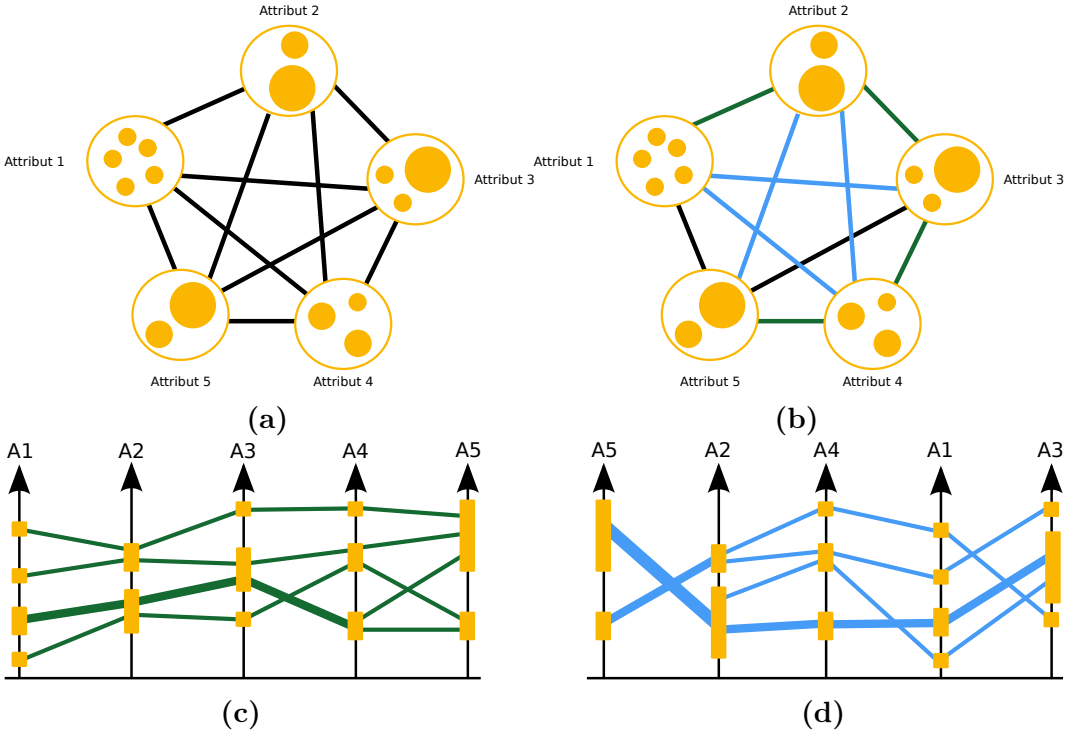


FIGURE IV.6: (a) Les agrégations de connexions sont calculées pour chaque combinaison d'ordre des attributs. La figure (b) présente deux chemins différents dans le graphe complet : en vert avec, dans l'ordre, Attribut 1 (A1), 2, 3, 4 et 5, en bleu avec l'ordre Attribut 5, 2, 4, 1, 3. Les figures (c) et (d) présentent les coordonnées parallèles de ces deux chemins respectivement.

existent pour un attribut, et ce quel que soit le type de ces valeurs (entier, réel), l'abstraction est construite en agrégeant simplement les éléments ayant l'exacte même valeur. Cela permet de s'assurer de ne pas séparer des valeurs identiques uniquement dans le but de respecter la condition du nombre d'agrégats à l'issue de l'algorithme d'agrégation.

4 Client léger

Dans cette section nous nous intéresserons essentiellement au client de visualisation de notre système. Ce client s'occupe du positionnement et du rendu visuel des agrégats d'éléments et de connexions. Il prend en charge également plusieurs outils d'interaction et de nombreux paramètres de rendu que nous décrirons par la suite.

4.1 Encodage visuel

Les agrégats d'éléments sont symbolisés par des rectangles pleins dont la hauteur est définie par le nombre d'éléments qu'ils représentent. On retrouve cet encodage visuel dans différentes techniques de représentation abstraite d'éléments, notamment les matrices d'adjacences, les diagrammes nœuds-liens, les histogrammes cumulés ou encore dans certaines versions présentées en section 1 de coordonnées parallèles. Cela permet d'interpréter rapidement le nombre approximatif d'éléments dans un agrégat et de le comparer aux autres. La figure IV.7 montre deux axes (A1 et A2) avant et après agrégation des éléments et des connexions. Pour l'axe A1 (resp. A2), l'agrégat bleu (resp. rouge) représente plus d'éléments que les autres agrégats, il est donc plus grand.

En utilisant cette représentation de la pondération, il devient très simple de détecter quels sont les agrégats les plus importants pour une dimension donnée et s'ils sont connectés à des agrégats de taille plus réduite ou au contraire de taille plus importante.

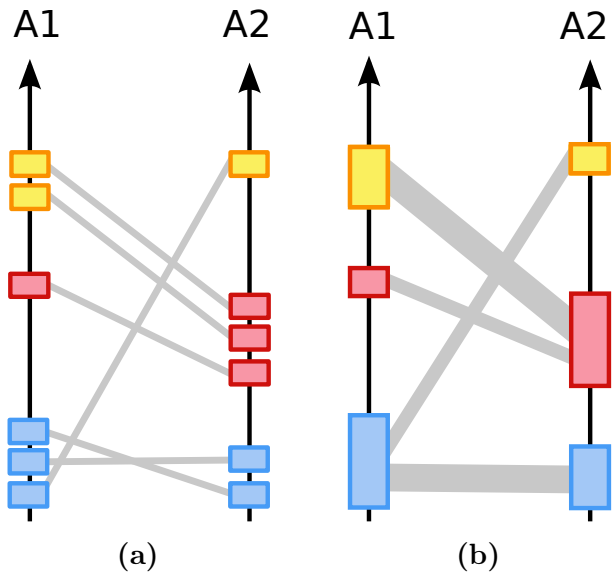
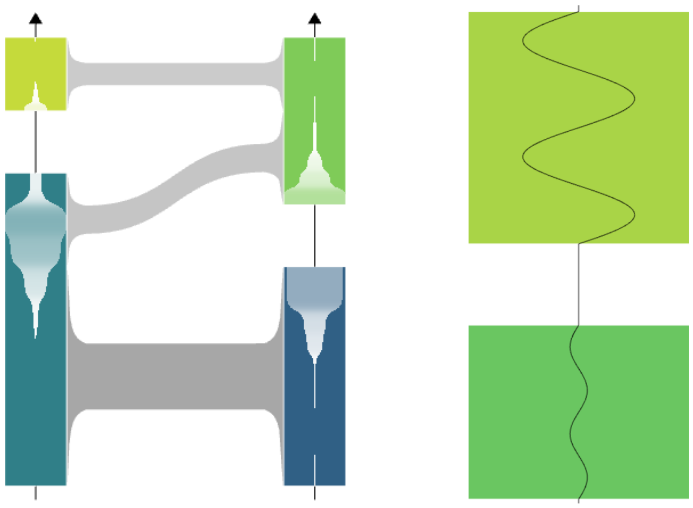


Figure IV.7: Agrégations des éléments et des connexions pour les coordonnées parallèles. (a) deux axes avec éléments et connexions non agrégés. (b) agrégation des éléments et des connexions. Les agrégats sont représentés par un rectangle dont la hauteur est définie par le nombre d'éléments qu'ils représentent. L'épaisseur des connexions représente le nombre d'éléments communs à deux agrégats. Les connexions incidentes à l'agrégat bleu de l'axe A1 sont réordonnées afin d'éliminer les croisements d'arêtes partageant une extrémité.

Afin de mettre en avant la distribution des valeurs dans un agrégat, un histogramme de fréquence lissé est affiché au centre des agrégats (voir figure IV.8a).

Cet histogramme montre la distribution des éléments au sein de chaque agrégat ; plus l'histogramme est large (s'approche des bords de l'agrégat), plus cette plage de valeurs est dense en éléments.



(a) (b)

FIGURE IV.8: Illustration des histogrammes et courbes de fréquences(a) L'histogramme montre la distribution des éléments dans chaque agrégat. (b) Dans la *vue intervalle*, l'axe représentant un attribut est compressé pour représenter la taille de l'intervalle auquel les éléments appartiennent. Ici, les éléments de l'agrégat du haut sont compris dans un intervalle de valeurs plus grand que pour ceux du bas.

La taille étant utilisée pour mettre en avant le nombre d'éléments représenté par chaque agrégat, il est impossible d'obtenir l'information de l'intervalle de valeurs représentées. Si cela n'est pas très important lorsque le nombre de valeurs distinctes est faible, cela peut avoir un réel impact avec des données dont la répartition n'est pas uniforme, et donc pour lesquels il y aura une grande variation des intervalles de valeurs de chaque agrégat. En effet, un agrégat représentant un grand nombre d'éléments peut être contenu dans un petit intervalle de valeur tandis qu'un agrégat représentant peu d'éléments pourra couvrir un intervalle beaucoup plus étendu. Pour résoudre ce problème, nous avons ajouté une représentation permettant de montrer à quel point un intervalle a été compressé pour pouvoir être compris dans la taille de l'agrégat. Pour cela, nous représentons à la place de l'histogramme de distribution une courbe dont l'amplitude est déterminée par un facteur de compression. On définit le facteur de compression F pour une dimension D composée d'agrégats A par

$$F(D) = min(F(A, D))$$

où

$$F(A, D) = \frac{T_a}{T_i}.$$

, avec T_a le nombre d'éléments et T_i la taille de l'intervalle de valeurs couvert par ces éléments. Ce facteur de compression est utilisé comme plus petite unité de fréquence et est représenté par un segment droit de ligne pour cette dimension. La figure IV.8b montre cette courbe pour deux agrégats différents appartenant à une même dimension, on peut voir que l'amplitude de la courbe est différente pour les deux agrégats.

L'opération d'agrégation est également appliquée aux connexions ; elles sont regroupées et représentées par une unique connexion abstraite pondérée. L'épaisseur des connexions est fixée par leurs poids afin de mettre en avant les agrégats partageant un grand nombre d'éléments. Afin de limiter l'encombrement visuel dans la représentation, la largeur des connexions est réduite dans la partie centrale et les connexions sont courbées à l'aide d'une courbe de Bézier à deux points de contrôle. Nous avons appliquons ensuite une translation en ordonnée de la courbe de Bézier (voir figure IV.9). Cette technique a cependant le défaut d'accentuer l'amincissement des connexions si la pente de la courbe est très forte mais sans qu'il n'y ait de réelle perte d'information (l'épaisseur des connexions est maintenue à proximité des agrégats). Une solution aurait été d'utiliser le vecteur normal à la courbe pour attribuer l'épaisseur mais cette approche n'a pas encore été intégrée.

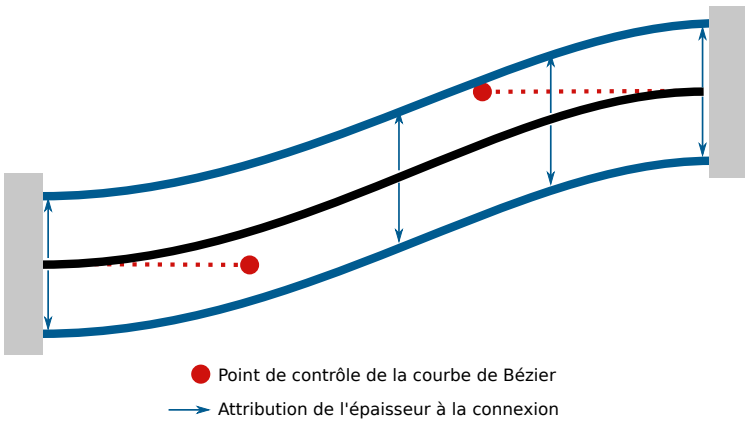


FIGURE IV.9: Dessin d'une connexion par une courbe de Bézier, définie par deux points de contrôles (en rouge) et épaissie par translation verticale (en bleu). Ce schéma ne montre pas l'amincissement de la partie centrale de la connexion.

4.2 Positionnement des agrégats et des connexions

Notre technique de visualisation de coordonnées parallèles permet de visualiser la distribution des éléments au sein d'une dimension, mais aussi de visualiser la distance relative entre agrégats d'une dimension. Pour cela, une proportion de chaque axe est réservée pour représenter cet espacement. Durant nos expérimentations, nous avons réservé un espace correspondant à 20% de la taille totale de l'axe, mais cet espacement est un paramètre pouvant être modifié de manière interactive (*cf* section 4.3). La hauteur de l'axe est séparée en deux parties : la partie réservée aux espacements et celle réservée aux agrégats. Ces deux parties sont ensuite divisées afin de représenter proportionnellement les espacements entre agrégats et les tailles d'agrégats. Dans la figure IV.10, les éléments sont regroupés en quatre agrégats et 20% de l'axe est attribué aux espacements entre agrégats ; nous avons réduit la quantité d'espace vide (espacements entre agrégats) afin de mettre en avant les agrégats mais les distances entre agrégats sont toujours représentées.

Afin d'éviter tout croisement inutile de connexions entre deux axes, les connexions incidentes à un agrégat sont ordonnées en fonction de la position de leur extrémité dans l'axe voisin. De plus, comme les tailles des agrégats et des connexions sont dépendantes du nombre d'éléments qu'ils représentent, cela empêche tout croisement de connexions lorsqu'elles ont une extrémité commune. Cet ordre permet également de créer un regroupement naturel des connexions incidentes d'un agrégat. Par exemple, sur la figure IV.7b, même si plusieurs connexions sont reliées à un même agrégat, aucune d'entre elles ne se croisent. Les seuls croisements pouvant apparaître sont ceux entre connexions n'ayant aucun agrégat en commun, quelle que soit l'extrémité considérée, limitant l'encombrement visuel et augmentant la lisibilité de la représentation.

4.3 Outils d'interaction

Notre système permet d'interagir grâce à divers outils que nous distinguons en deux catégories en fonction de la nécessité ou non de manipuler les données pour modifier le rendu visuel.

Interaction sans accès aux données

Les outils d'interaction ne nécessitant pas d'accéder aux données permettent de modifier la représentation visuelle des données en changeant les paramètres de rendu et peuvent être faits en temps réel. Ils permettent notamment de personnaliser la représentation en fonction de l'analyse que l'on veut effectuer, en mettant en avant les agrégats ou les connexions par exemple.

Largeur des axes La modification de la largeur des axes permet d'augmenter ou de réduire l'espace écran occupé par les agrégats et ainsi de modifier la

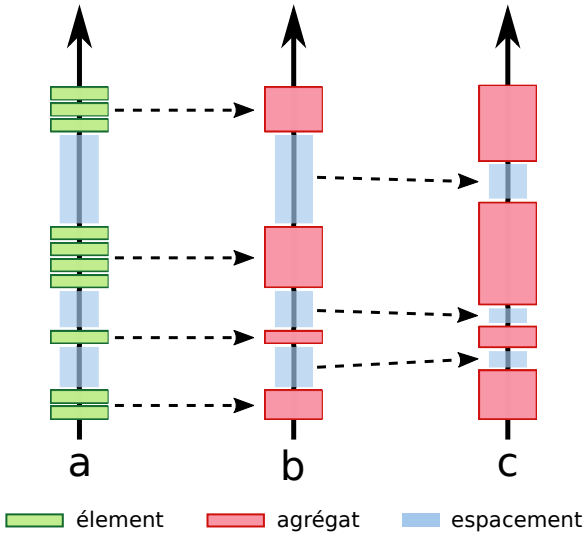


Figure IV.10: Représentation des distances inter-agrégats. (a - b) Agrégation des éléments en quatre agrégats (par soucis de clarté, les éléments représentés ont tous des valeurs différentes), les proportions agrégats/espacements ne sont pas encore modifiés. (b - c) Changement des proportions, l'espacement entre agrégats et réduit (ici 20% de l'axe est attribué aux espacements et 80% aux agrégats).

visibilité des histogrammes qu'ils contiennent.

Largeur des arêtes Par défaut, les arêtes sont représentées amincies dans leurs parties centrales afin de réduire l'encombrement visuel mais cet amincissement réduit la visibilité des arêtes et de leurs proportions sur cette partie centrale. En permettant de modifier cette épaisseur, nous fournissons la possibilité d'augmenter la visibilité des arêtes et des poids associés.

Espacements inter-axes L'espace entre deux axes consécutifs peut être modifié afin d'augmenter ou de réduire l'espace écran alloué aux connexions entre agrégats. Le choix d'augmenter ou de réduire cet espacement dépend de l'information que l'utilisateur veut mettre en avant. S'il veut effectuer une analyse fine des agrégats et de leurs répartitions sur les axes, il réduira ces espacements. Au contraire, s'il veut analyser les connexions entre agrégats, il définira un espace plus grand entre les axes.

Espacements intra-axes La modification de l'espace intra-axes se traduit par la modification du pourcentage alloué aux espacements. Augmenter cet espace permet de mettre l'accent sur la distance entre agrégats et d'analyser la distribution des agrégats sur un axe. Au contraire, en la réduisant, ce sont les proportions des agrégats qui seront mises en avant.

Comme le montre la figure IV.11, ces deux espacements (intra- et inter- axes) peuvent être modifiés en fonction des besoins de l'utilisateur et de l'analyse qu'il veut mener et permet d'obtenir une large gamme de visualisation différentes. Si on réduit ces espacements au maximum, on ne montre que les proportions prises par les agrégats au sein de chaque axe et les distributions au sein de chaque agrégat ; les informations de distances relatives entre agrégats d'un même axe et de connexions entre agrégats d'axes consécutifs sont masqués. Cette représentation est équivalente à un histogramme cumulé par attributs. Au contraire, en augmentant ces espacements, on génère la figure montrée en haut à droite, qui s'approche de la représentation classique des coordonnées parallèles.

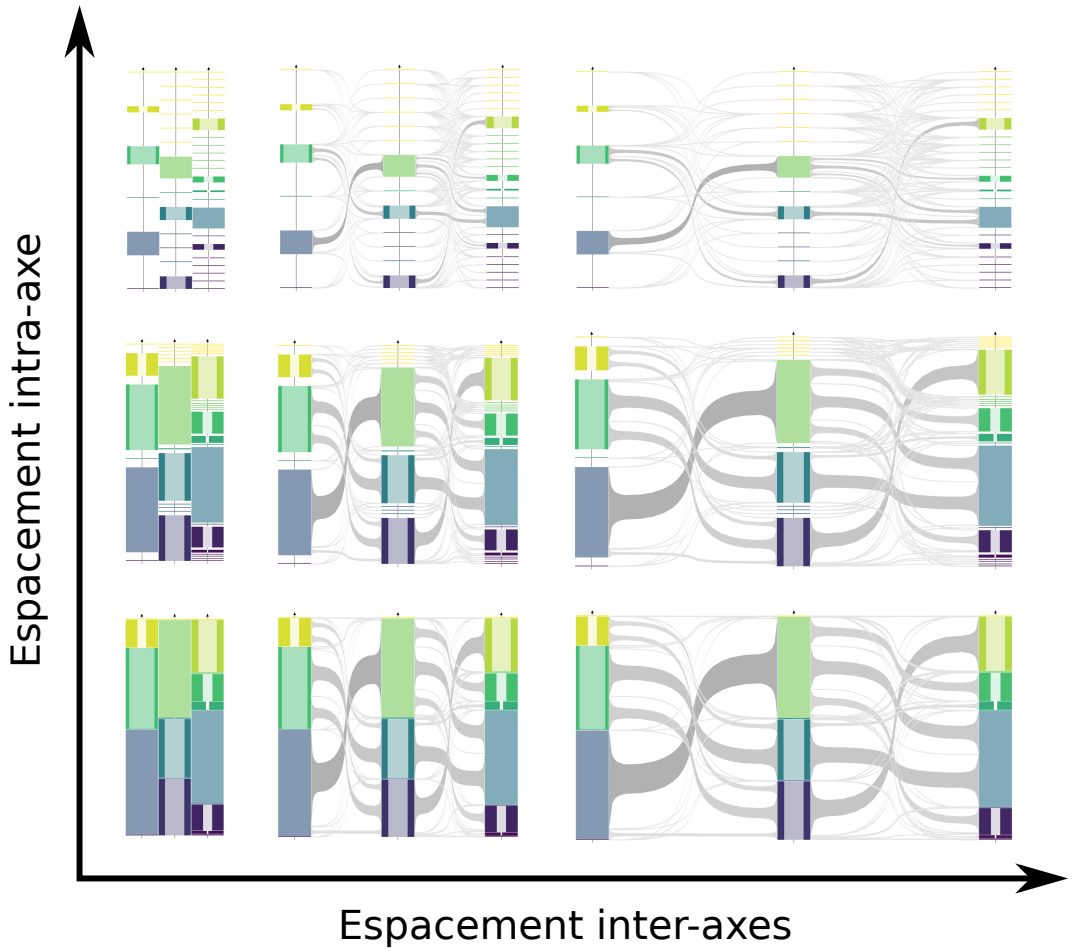


Figure IV.11: La modification des espacements intra- et inter-axes permet d'obtenir une large gamme de coordonnées parallèles. Ainsi, la figure en bas à gauche, correspondant à des valeurs nulles pour les deux paramètres correspond à un histogramme cumulé.

Inversion d'axe L'une des particularités des coordonnées parallèles est leur capacité à mettre en avant les corrélations et anti-corrélations dans les données (voir figure IV.12). Dans le cas de corrélation ou anti-corrélation parfaite, les deux résultats sont relativement simples à identifier, cependant, dans le cas de relation partielle, la distinction est plus difficile. L'inversion d'axe est un outil généralement associé aux coordonnées parallèles pour faciliter la détection de ces relations. L'ordre des éléments étant l'exact inverse entre deux attributs anti-corrélés, inverser l'un des axes permet de transformer visuellement une anti-corrélation en corrélation (et inversement). Puisque les agrégats changent de positions, les connexions sont elles aussi ré-ordonnées afin de retirer les croisements non nécessaires (voir section 4.2).

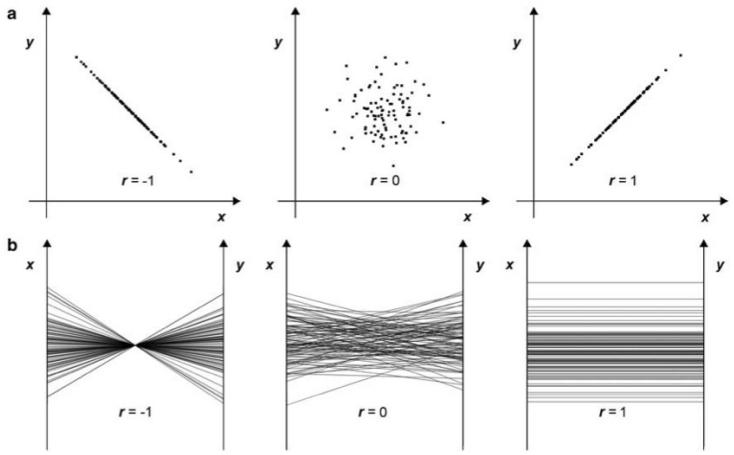


Figure IV.12: Figure extraite des travaux de [89] sur la représentation des anti-corrélations (à gauche), distribution aléatoire (au centre) et corrélations (à droite) sur les nuages de points (en haut) et les coordonnées parallèles (en bas). Une corrélation se caractérise par le même ordre des éléments pour deux axes différents, les connexions sont donc parallèles. Au contraire, l'ordre est inversé dans le cas d'une corrélation inverse les connexions se croisent donc toutes en leurs centres.

Interaction avec accès aux données

Suppression/insertion/ré-ordonnancement des axes L'ordre des attributs influence fortement la lisibilité de la visualisation et les informations que l'on saura en retirer. Cet inconvénient est connu et est lié au problème d'ordonnancement linéaire. Dans le cas des coordonnées parallèles, on retrouve deux cas de modification de l'ordre des axes, soit pour minimiser les croisements de connexions, soit pour correspondre à une logique particulière, liée à une expertise métier. Nous avons donc mis en place un outil d'interaction permettant

de modifier l'ordre des axes. Il est donc possible d'ajouter, de retirer ou de modifier les attributs en fonction de l'intérêt qu'on leur porte et de n'analyse que l'on veut effectuer.

Ces interactions modifient les informations à afficher ; si l'on prend l'exemple du retrait d'un axe de la représentation, en retirant l'axe i, le système doit afficher les connexions entre les axes $i-1$ et $i+1$. Il faut donc récupérer les informations nécessaires dans les abstractions pré-calculées, et les transférer depuis l'espace de données jusqu'au client de visualisation. Afin de maintenir le contexte de l'analyse, la modification de la représentation est animée en différentes étapes effectuées les unes après les autres afin de faciliter la compréhension du changement opéré [56, 153]. Dans le cas de la suppression d'un axe, l'animation se fait en trois étapes : retrait des éléments et connexions par fondu, apparition des nouvelles connexions par fondu puis rapprochement des deux axes. L'animation inverse est utilisée pour l'ajout d'une dimension : éloignement des deux axes, retrait des connexions existantes par fondu au blanc puis apparition du nouvel axe, agrégats et connexions par fondu. Le déplacement d'un axe correspond à la succession des deux animations précédentes : retrait de l'axe suivi de l'ajout de cet axe à une autre position. Cette interaction peut être effectuée sur plusieurs axes sélectionnés par une boite englobante afin de faciliter le déplacement du groupe d'axes.

Sélection : intersection, union et différences La sélection d'éléments est un outil d'interaction majeur pour les coordonnées parallèles. Il permet de mettre en évidence par surbrillance la répartition des éléments sélectionnés sur l'ensemble de la représentation. Les éléments non sélectionnés sont représentés en arrière-plan avec une coloration grisée et l'ensemble sélectionné est représenté en utilisant le même encodage visuel que décrit précédemment. Une jauge est ajoutée par effet de contraste sur chaque agrégat de la représentation (visible sur la figure IV.13) afin de mettre en évidence le pourcentage d'éléments que représente la sélection pour les agrégats représentés. Les histogrammes intra-agrégats sont mis à jour pour correspondre à la distribution de la sélection dans l'intervalle de valeur représenté par l'agrégat. Par ces modifications on peut identifier quels sont les agrégats concernés par la sélection, la taille qu'ils occupent dans la représentation générale, la proportion de la sélection dans cet agrégat et sa distribution.

Quatre interactions de sélection sont possibles :

- *sélection d'un agrégat* En sélectionnant un agrégat l'utilisateur demande la mise en avant de la valeur de l'ensemble sélectionné pour chaque attribut visualisé.
- *sélection d'une connexion* La sélection d'une connexion correspond à la sélection des éléments présents dans les deux agrégats à ses extrémités ; en termes d'ensembles, cela correspond à l'intersection des ensembles

représentés par les deux agrégats ($A \cap B$).

- *ajout d'un agrégat à la sélection* L'ajout d'un agrégat consiste à ajouter un ensemble à un ensemble préalablement sélectionné ; en termes d'ensembles cela correspond à l'*union* des ensembles représentés par les deux agrégats ($A \cup B$),
- *retrait d'un agrégat de la sélection* Le retrait d'un agrégat consiste à retirer un ensemble d'éléments de l'ensemble sélectionné ; en termes d'ensemble cela correspond à la différence des deux ensembles ($A \backslash B$).

Filtres de valeurs L'application d'un filtre sur les valeurs d'un axe permet d'effectuer une sélection sur l'ensemble des éléments de la représentation en spécifiant une plage de valeurs d'intérêt. Les éléments étant agrégés, le filtre se fait par agrégats complets. L'intervalle de valeur filtré est défini par deux curseurs initialement placés aux extrémités de l'axe que l'on déplace afin de définir une plage de valeurs d'intérêt. Si un agrégat est réparti sur un intervalle de valeurs et l'une des extrémités du filtre positionné dans cet intervalle, le filtre retirera l'agrégat dans son intégralité. Une fois les bornes de l'intervalle définies, les éléments compris dans le filtre sont récupérés depuis l'espace des données et représentés par le client de visualisation de la même façon que pour la sélection d'un ensemble. La figure IV.13 présente un filtre des valeurs pour l'attribut A3 et l'ensemble d'éléments résultant est mis en surbrillance sur l'ensemble de la représentation.

Informations complémentaires Notre outil étant destiné à une utilisation pour des données massives, notre objectif tout au long de la conception et de la mise en place de l'outil s'est placé dans l'interprétation de tendances et de comportements globaux ou anormaux. Dans ce contexte, nous ne portons pas une grande attention dans le fait de restituer visuellement une information précise des valeurs de chaque élément ou agrégat, ces valeurs ne sont d'ailleurs pas indiquées sur la représentation. L'utilisateur peut cependant vouloir récupérer des informations statistiques sur certains agrégats. Pour cela, nous fournissons un outil permettant d'afficher des informations statistiques pour un agrégat (telles que les valeurs moyennes, minimales et maximales). La figure IV.13 présente un exemple de données statistiques concernant un agrégat.

4.4 Détails techniques

De la même manière que dans le chapitre III, nous nous basons sur une technique de dessin bas-niveau pour représenter et interagir avec les coordonnées parallèles. Nous utilisons ainsi la librairie graphique *OpenGL* ainsi que le langage de programmation *GLSL* afin de communiquer nos instructions de

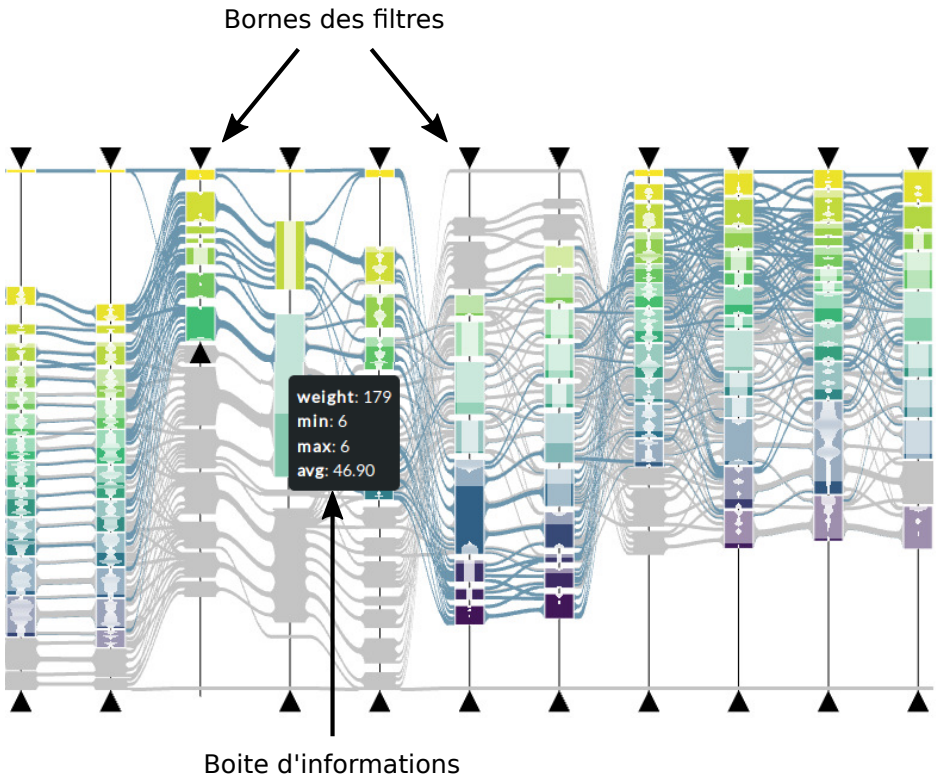


Figure IV.13: Application d'un filtre sur les éléments de l'axe A3 et affichage de la répartition de l'ensemble résultant pour les autres attributs. L'affichage d'une boite d'informations permet de fournir des statistiques : poids de l'agrégat, valeur minimale, maximale et moyenne.

rendu directement à la carte graphique, et bénéficier de ses capacités de parallélisation et d'optimisation de rendu. De même, nous nous sommes appuyés sur *Emscripten*[1] afin de transformer le code *C++* et *OpenGL* en code *JavaScript* et *WebGL*, rendant ainsi notre outil utilisable sur une interface web. Les interfaces graphiques nécessaires à l'utilisation des outils d'interaction ont eux également été implémentés en *JavaScript* et *HTML5-CSS3*.

5 Cas d'étude

Nous allons présenter ici un cas d'utilisation avec un jeu de données multivariés couramment utilisé : le jeu de données *cars* fourni dans [137]. Ce jeu de données présente des caractéristiques techniques de différents modèles de voitures ; différentes déclinaisons peuvent-en être trouvées avec un nombre variable de véhicules et d'attributs répertoriés. Celui que nous avons utilisé

1. compilateur bas-niveau par machine virtuel (Low Level Virtual Machine)

contient 387 véhicules avec 8 attributs booléens caractérisant le type de véhicule (*Pickup*, *Sportscar*, *etc.*) et le type de motorisation (propulsion ou 4-roues motrices) et 12 autres attributs numériques, entiers ou réels (prix d'achat, coût vendeur, taille du moteur, dimensions du véhicule, *etc.*).

Ce jeu de données est relativement petit et ne nécessite donc pas l'utilisation de notre outil de visualisation, conçu pour permettre la visualisation et l'interaction pour des données massives avec des coordonnées parallèles. Il est cependant couramment utilisé comme jeu d'essai et nous permettra de mettre en avant l'efficacité de notre technique de visualisation.

Dans les exemples qui suivront, nous avons utilisé un algorithme d'agrégation dit de *binning* avec en paramètres la génération de k agrégats par attributs (ici, nous avons choisi 10 agrégats). Cet algorithme divise l'intervalle de valeurs de chaque attribut en k partitions de même taille et les éléments se positionnent, en fonction de leurs valeurs pour l'attribut considéré, dans la partition correspondante.

On peut ainsi voir sur la figure IV.14b que l'on distingue très clairement les distributions des agrégats sur chacun des axes. Ainsi, alors qu'il est impossible de déterminer la distribution des éléments sur les axes représentant des attributs booléens (les 8 axes de gauche) sur les coordonnées parallèles traditionnelles (voir figure IV.14a), elle est clairement exprimée sur notre outil de visualisation. En effet, dans la première, chaque élément étant représenté indépendamment les uns des autres, les éléments ayant des valeurs communes se chevauchent les uns les autres, rendant difficile d'estimer leur nombre. Cette estimation devient même impossible lorsque ces éléments ont les mêmes valeurs sur les axes voisins puisqu'une seule ligne sera représentée. Pour obtenir cette information il serait nécessaire d'utiliser une technique représentant les densités d'éléments comme la transparence ou la carte de chaleur([12, 143, 151]). Ainsi, on peut voir sur la figure IV.14b que les éléments de la catégorie *SUV* (en (1) sur la figure) sont regroupés en deux agrégats, un grand et un petit. En observant un peu plus les proportions, on peut estimer que le plus petit représente environ 15% des éléments. L'échelle de couleur et la position sur l'axe nous indiquent que le petit agrégat représente des valeurs plus grandes que celle du grand agrégat. Sachant qu'il s'agit de valeurs booléennes, on peut en déduire que l'agrégat vert représente la valeur 0 et l'agrégat rouge la valeur 1. L'outil d'information complémentaire nous confirme ces déductions. Il est possible de faire une analyse similaire sur la figure IV.14a mais il est impossible d'obtenir la distribution des éléments à cause du chevauchement des polylignes. De façon similaire, on peut voir sur la figure IV.14a que tous les éléments passent par la valeur 0 de la catégorie *Pickup* (en (2) sur la figure). Cela est représenté sur la figure IV.14b par une agrégation en un seul agrégat. Notre représentation ne fournit pas directement la valeur mais on peut là aussi l'obtenir par déduction (il s'agit de booléens et la couleur verte nous indique que c'est la valeur basse) ou en utilisant l'outil d'information complémentaire qui indiquera textuellement la

valeur correspondante.

De la même manière on peut voir sur notre représentation la répartition des éléments sur les axes voisins et l'importance qu'ils représentent grâce aux connexions et à leur taille. Il est ainsi simple de s'apercevoir avec notre outil de visualisation qu'il y a peu d'éléments faisant partie des catégories *Sportscar*, *SUV*, *Wagon* et *Minivan* et beaucoup appartenant à la catégorie *SSCLS*[1] (une combinaison de différents types de berlines).

Si l'on regarde de plus près les deux représentations (voir agrandissement en figure IV.15) on peut observer des lignes parallèles indiquant une corrélation entre les attributs *Retail price* (prix de vente) et *Dealer cost* (coût vendeur) (voir figure IV.15a, axes encadré en vert). Cette corrélation peut aussi être déduite avec notre représentation figure IV.15b avec pour autant une nette amélioration de l'encombrement visuel entre ces deux axes.

Une analyse similaire peut être effectuée avec les catégories *Engine Size* (taille du moteur), *Cylinders* (nombre de cylindres) et *Horse power* (puissance en chevaux) (axes encadrés en rouge). Il semble logique de déduire l'existence d'une corrélation entre ces trois attributs, la puissance du véhicule étant, tout du moins partiellement, dépendante de la taille du moteur et du nombre de cylindre dans le moteur.

Par ailleurs, si l'on considère les axes encadrés en violet et en jaune sur les figures IV.15a et IV.15b, il est plus facile de voir sur notre représentation une anti-corrélation pour les valeurs proches des valeurs minimales et maximales des attributs (les éléments sur les parties hautes et basses des axes) entre la puissance du véhicule (*horse power*) et la consommation en ville (*cityMPG*) ainsi qu'entre la consommation sur autoroute (*HighwayMPG*) et le poids du véhicule (*weight*). Ces analyses sont confirmées lorsqu'on inverse les deux axes relatifs à la consommation des véhicules (*CityMPG* et *HighwayMPG*) tel que montré sur la figure IV.15c. En ce qui concerne la sélection, notre technique permet d'étendre les informations obtenues en comparaison à la technique classique des coordonnées parallèles. Par exemple, sur la figure IV.16 on peut voir la même sélection des véhicules ayant un nombre de cylindres supérieur à 8. Tandis que la figure IV.16a reste très encombrée avec beaucoup de chevauchement de polylignes, la version agrégée présentée en figure IV.16b met en avant par surbrillance les agrégats sélectionnés pour tous les attributs ainsi que la proportion qu'ils représentent dans les autres agrégats. On peut ainsi voir que la sélection représentant une faible proportion des éléments, est représentée par une petite zone mise en surbrillance dans les autres axes.

1. *Small,Sporty,Compact or Large Sedan*

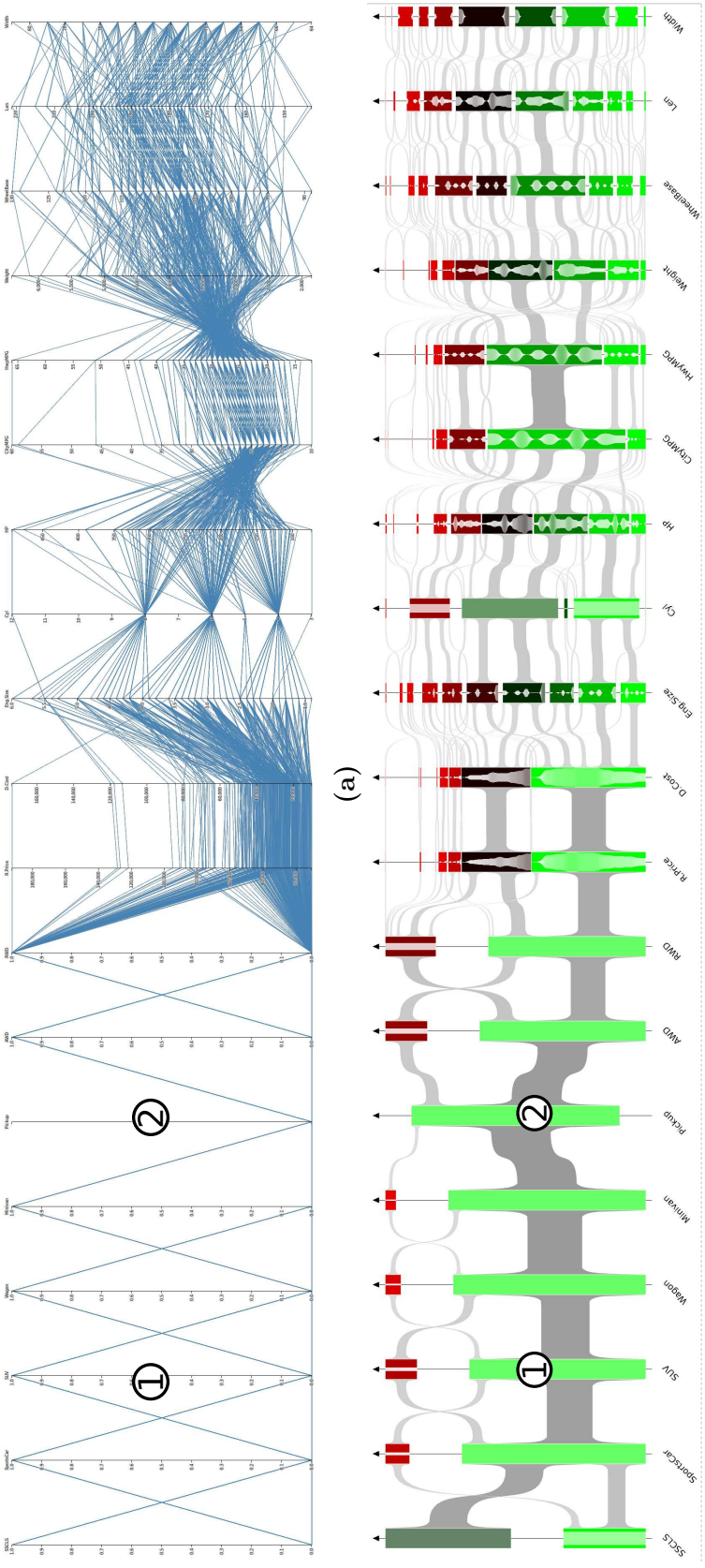


FIGURE IV.14: Visualisation générale du jeu de données *Cars* avec des coordonnées parallèles, (a) traditionnelles (sans optimisation spécifique) représentées avec la librairie graphique *D3JS* et (b) avec notre outil de visualisation (agrégation par *bucket* avec 10 agrégats par dimensions). En (1), la catégorie *Pickup* et en (2) les *SUV*.

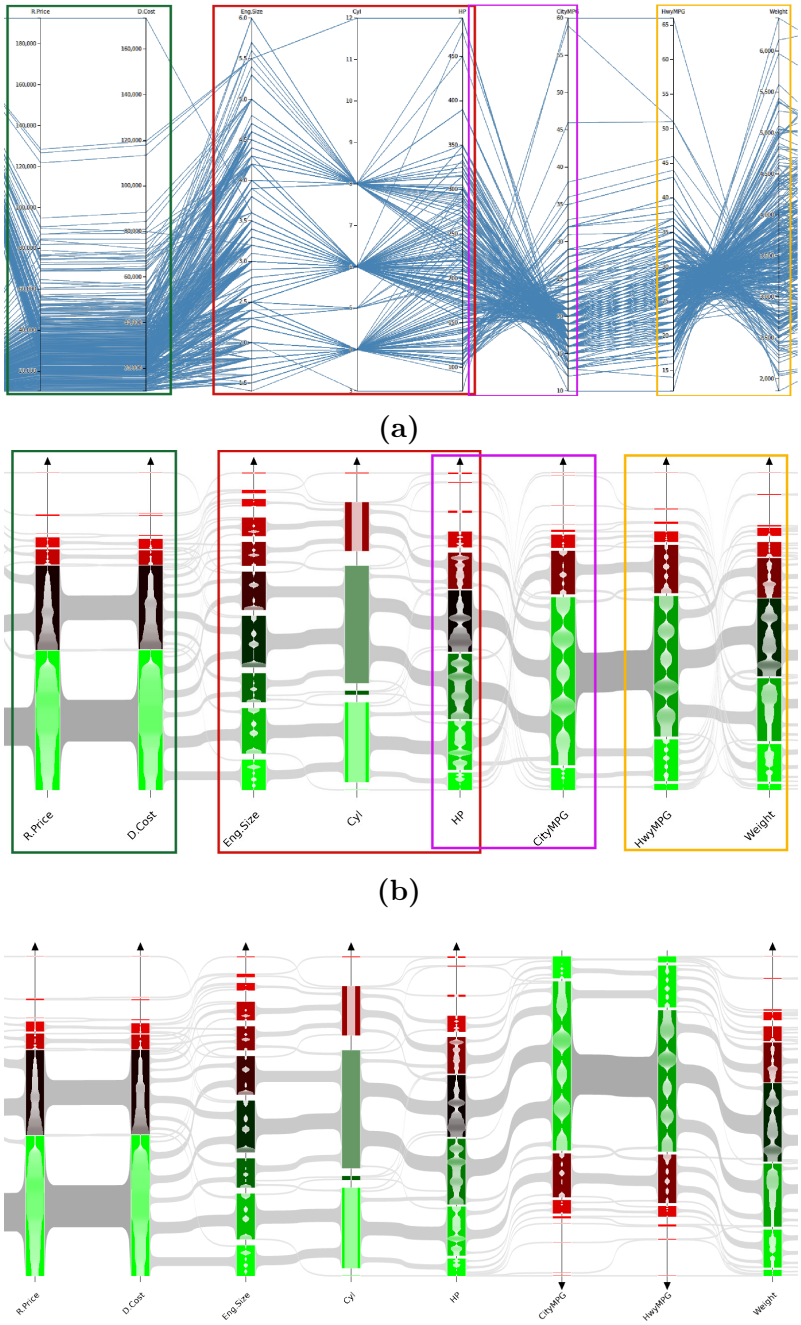


FIGURE IV.15: Agrandissement sur une région d'intérêt des coordonnées parallèles : les mêmes informations sont représentées sur les deux visualisations. (a) coordonnées parallèles traditionnelles, (b) notre technique, les corrélations (encadrées en vert et rouge) et anti-corrélations (encadrées en violet et jaune) sont identifiables. En (c) l'inversion de deux axes avec notre technique permet de confirmer la présence d'anti-corrélations.

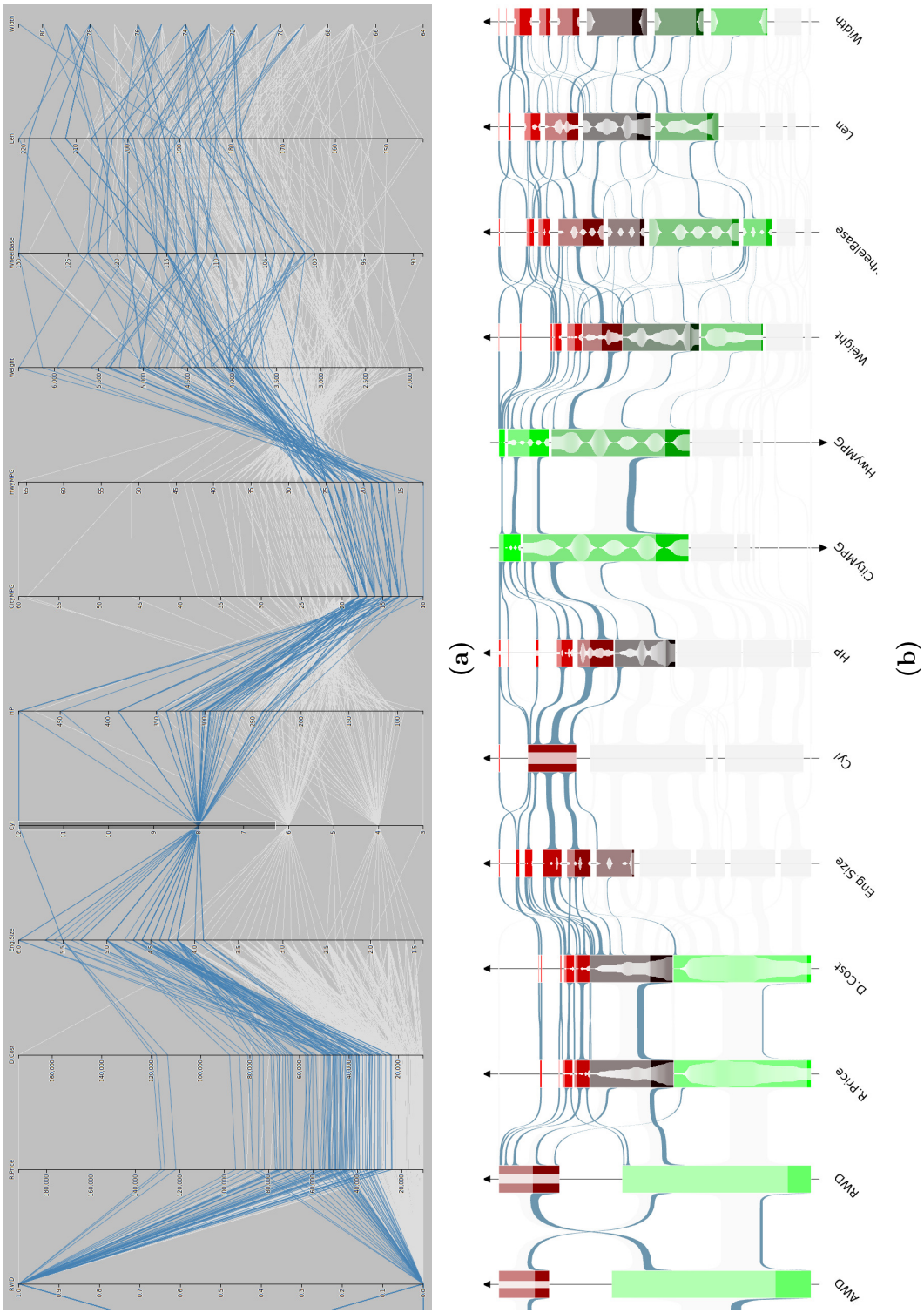


FIGURE IV.16: Sélection des éléments ayant un grand nombre de cylindres (supérieur à 8). (a) avec des coordonnées parallèles traditionnelles et (b) avec notre technique de représentation. Les axes indiquant la distance que l'on peut parcourir (en miles) avec une quantité d'essence fixe (ici, un gallon) sont inversés.

6 Conclusion

Nous avons présenté ici un nouvel outil de visualisation dédié au données massives utilisant des coordonnées parallèles. Cet outil permet non seulement de visualiser des données multivariées mais fournit aussi plusieurs outils d'interactions couramment utilisés avec des coordonnées parallèles. Nous nous sommes ainsi appuyés sur les travaux présentés dans l'état de l'art afin de mettre en place une représentation permettant les mêmes analyses que des outils dédiés à une utilisation plus classique. Notre technique permet donc la représentation et l'interaction sur des données abstraites représentant plusieurs millions d'éléments sans souffrir de ralentissements empêchant l'exploration des données. Pour cela, nous nous appuyons sur des techniques de représentation bas-niveaux, un outil de visualisation déporté sur navigateur web associés à de l'abstraction de données. Dans le prochain chapitre, nous présenterons l'architecture et les techniques utilisées permettant de fournir les données requises en des temps raisonnables, rendant possible l'exploration et l'interaction en temps réel sur des données massives. Nous verrons également que certaines des interactions possibles avec des jeux de données de taille raisonnable ont un coût computationnel très important rendant difficile leurs applications pour des données massives. Nous nous comparerons également aux travaux présentés dans la littérature permettant la visualisation de données multidimensionnelles massives, en particulier sur des aspects d'efficacité et de performance, et notamment avec les travaux de [116] qui offrent des performances intéressantes pour un objectif similaire.

Chapitre V

Les données massives

1 Introduction - État de l'art

Lorsqu'on considère l'analyse de données massives avec une infrastructure en nuage (cloud), relativement peu de travaux ressortent de l'état de l'art. Cela vient notamment du fait qu'il s'agit d'une technologie et d'un domaine d'étude relativement récents. Pourtant, quelques travaux décrivent des techniques permettant la représentation de grands jeux de données. Pour cela, deux approches sont généralement utilisées : la réduction des données à visualiser et l'utilisation d'une architecture matérielle et logicielle spécialisées.

La suite de cette section est un rappel de l'état de l'art des techniques de réduction de données et des techniques de visualisation de données massives. La section 1.3 présente les contributions ainsi que l'organisation générale du chapitre.

1.1 Réduction de données

Il existe différentes approches pour réduire la quantité de données à représenter. Une première approche est l'échantillonnage consistant en l'extraction d'un sous-ensemble caractéristique de l'ensemble des données qui est ensuite utilisé pour la représentation. Cette approche est utilisée par exemple dans [111] où l'échantillonnage sert à réduire la taille d'un graphe et en permettre la représentation. Ces travaux présentent notamment l'utilisation de l'augmentation ciblée de la taille de l'échantillon afin d'obtenir l'équivalent d'un zoom dans le réseau. La technique présentée dans [21] utilise également l'échantillonnage mais pour réduire cette fois l'encombrement visuel dans un nuage de points. Pour cela, ils utilisent notamment de l'échantillonnage uniforme et non-uniforme afin d'optimiser la préservation des informations visuelles.

Une autre approche consiste en l'agrégation des éléments (par binning ou clustering) afin de réduire les données à représenter. On peut distinguer deux groupes : les techniques qui utilisent ou s'inspirent des techniques de

représentation spatiale (visualisation et exploration de cartes) [91, 92] et celles qui au contraire propose une approche généraliste non dédiée à un type de données en particulier [3, 22, 78, 79].

Approches pour les données spatiales Lins *et al.* [91] présentent le *Nanocube*, une structure de données inspirée du *cube de données* pour manipuler des jeux de donnés spatio-temporelles comportant des milliards d'éléments. Un *cube de données* correspond à l'agrégation de l'ensemble des requêtes de regroupement qu'il est possible de faire sur une table de données. Cette structure de données est très compacte et permet d'accélérer les requêtes mais est connue pour son coût en stockage important. Leur technique se base sur le pré-calcul et le stockage de l'ensemble des requêtes d'agrégation et permet un accès rapide aux données grâce à cette structure de données. Les données sont ensuite transférées sur un client de visualisation afin d'être représentées par différentes techniques (carte de chaleur, coordonnées parallèles et histogrammes notamment). *Immens*, le logiciel présenté dans [92] utilise une agrégation par binning pour la représentation de points d'intérêts sur une carte. Pour leur approche, les auteurs partent du postulat que *la visualisation et l'interaction doivent être limitées par un choix de résolution des données et non par la masse de données*. Pour cela, ils s'appuient sur la réduction des données pour pouvoir les stocker là où elles seront visualisées et pré-calculent des *tuiles de données multi-variées* construites à partir de cubes de données et stockées sous formes de bitmaps. Ces bitmaps seront ensuite transférées sur le client et utilisées par le GPU en tant que textures. Ils mettent ainsi à profit les capacités des processeurs graphiques pour paralléliser et accélérer le traitement des données. Cette technique souffre cependant de deux inconvénients. Le premier est dû à l'algorithme de binning utilisé pour l'agrégation pouvant créer des artefacts visuels (amalgames ou sauts) qui ne seraient pas présents s'il n'y avait pas eu d'agrégation. Le second est l'utilisation du GPU comme outils de calcul et de requêtes dans les données, impliquant que l'ensemble des données nécessaires soit récupérées sur le client de visualisation.

Approches pour les données générales Abello *et al.* [3] décrivent *AskgraphView*, une technique qui s'appuie sur une architecture en deux parties : un serveur de calcul qui s'occupe du stockage et des pré-calculs et un client de visualisation pour la représentation visuelle des données. Les pré-calculs consistent en la construction d'un arbre hiérarchique à partir d'un graphe pondéré non orienté, en s'appuyant sur le partitionnement du graphe et sur la génération d'un graphe quotient. Ils construisent ensuite une anti-chaîne de l'arbre qu'ils utilisent pour représenter les données sur le client de visualisation.

Pour garantir les performances et la visibilité des éléments, l'agrégation est conditionnée par une limite en nombre d'éléments contenu dans chaque agrégat, bornée par le nombre d'éléments dans l'agrégation de plus haut niveau. Ainsi,

le contenu d'un méta-nœud peut-être représenté par un sous-graphe de même taille que celui de plus haut niveau. Afin de garantir le contexte d'exploration, la technique fournit également un système de vues multiples avec une liste des agrégats et un dendrogramme. Dans [77, 78] Jugel *et al.* présentent une approche d'agrégation orientée *visualisation*. Ils font le choix d'optimiser les requêtes dans l'espace de données en fonction de la représentation utilisée. Les requêtes effectuées par un utilisateur sont ainsi ré-écrites côté serveur afin de n'obtenir que les résultats correspondants à ce que la représentation peut fournir comme informations, évitant ainsi le transfert de données inutiles. Cependant, ils ne s'intéressent qu'à des données temporelles et pour des représentations peu gourmandes en pixel par donnée (histogrammes, nuages de points et courbes). Une des techniques les plus proches de celle que nous utilisons ici est celle présentée par Abello et Korn [1] qui présentent une technique de visualisation de multi-digraphes massifs. Pour cela, la technique s'appuie sur le calcul de coupes dans un arbre hiérarchique et propose une approche basée sur un client de visualisation lourd et d'un stockage distant. Le serveur distant calcule un arbre et stocke l'ensemble des nœuds internes et les feuilles de l'arbre ainsi que toutes les arêtes reliant des feuilles de l'arbre aux autres nœuds. Les données sont ensuite transférées à la demande de l'utilisateur en fonction de ses interactions et sont représentées à l'aide d'un client de visualisation. La technique décrite utilise des vues multiples afin de maintenir le contexte de visualisation lors de l'analyse. Dans [22], les auteurs présentent une plateforme dédiée à la visualisation interactive de grands graphes basée sur le partitionnement des données. Leur système fractionne le graphe en entrée en différents sous-graphes afin de réduire la taille des données à traiter. Par ce fractionnement, ils facilitent le calcul de position des éléments, et une fois ce calcul terminé, positionnent ces différentes partitions les unes par rapports aux autres. Les données sont ensuite abstraites afin de générer les différents niveaux de détails puis stockées en utilisant des tables de données différentes pour chaque niveau d'abstraction. Afin de réduire la quantité de données transmises au client de visualisation, ils utilisent également un système permettant de requêter uniquement les éléments visibles dans la fenêtre de navigation. Pour cela, ils utilisent un *R-tree*, une structure permettant d'indexer efficacement des données dans un plan 2D et d'identifier celles qui sont visibles dans l'espace écran.

1.2 Infrastructure spécialisée : cloud et super-ordinateurs

La technique présentée par Eldawy *et al.* [34] permet la représentation de données spatio-temporelles issues d'une base de données d'images satellites [1]. Leur système est ainsi complètement organisé autour de la représentation de données issues de cette base de données en particulier. Les auteurs s'appuient

1. le *Land Processes Distributed Active Archive Center*, une base de données de la NASA

sur une structure de données souvent utilisée pour la représentation de données spatiales, le *quad-tree*[118] qui permet de construire les différents niveaux de détails de la représentation et maintiennent deux index correspondant aux agrégations hiérarchiques des données temporelles et spatiales. Ils s'appuient notamment sur *spatialHADOOP*, une adaptation du paradigme mapreduce dédié aux données spatiales. Parmi les travaux utilisant une architecture *cloud* pour la représentation de données massives, on peut également citer les travaux de Perrot *et al.* [104]. Dans ces travaux, les auteurs associent une architecture de traitement de données massives *HADOOP* et une représentation déportée sur client léger pour la visualisation de données par carte de chaleur. Pour cela, leur technique s'appuie notamment sur un algorithme d'agrégation multi-échelle distribué inspiré de l'agrégation par canopée ainsi que sur le calcul en temps constant d'une estimation de fonction de densité avec un nombre constant d'éléments représentés. Jonker *et al.* [76] proposent une technique pour la représentation de grands graphes de données relationnels dans laquelle le partitionnement hiérarchique et le positionnement des nœuds du graphe sont pré-calculés coté serveur et le résultat est découpé en tuiles de données. Ces tuiles sont ensuite transférées au client de visualisation. L'ensemble des calculs fortement consommateurs en temps et en ressources est ainsi effectué sur la plateforme de calculs distribués et ne sont transférées que les informations nécessaires à l'utilisateur.

D'autres techniques, moins fréquentes utilisent des approches incrémentales, permettant d'exploiter les résultats au fur et à mesure que ceux-ci sont obtenus. Dans [40], les auteurs présentent une technique de visualisation utilisant cette approche. Leurs travaux se basent sur le fait que plus la taille des données augmente plus les temps nécessaires pour obtenir la réponse à une requête sont importants. L'une des conséquences de cette lenteur est l'abandon de tâches par un utilisateur qui estimera le coût en temps trop important par rapport à l'intérêt de la question. Pour contrer cela, ils présentent une technique qui s'appuie sur l'augmentation progressive de l'échantillon de données sur lequel sont effectuées les requêtes et sur l'envoi de résultats partiels. En se basant sur le principe que plus un échantillon est grand, plus il est représentatif des données, on peut associer un degré d'incertitude aux résultats obtenus dépendant de la taille de l'échantillon. L'incertitude de la réponse fournie est donc forte en début de requête (avec un petit échantillon) et se réduit au fur et à mesure que la taille de l'échantillon augmente. Dans leur exemple, les résultats sont représentés par des histogrammes et l'incertitude est représentée par des limites haute et basse de valeurs potentielles. Ces limites fournissent ainsi un intervalle de valeurs indiquant le degré de confiance que l'utilisateur peut avoir sur les résultats fournis. Au fur et à mesure que la taille de l'échantillon se réduit, les limites de l'intervalle se rapprochent permettant d'augmenter le degré de confiance dans le résultat indiqué.

Im *et al.* [70] présentent un système de visualisation utilisant une adaptation

du modèle *MapReduce* qu'ils appellent *VisReduce*. Leur technique présente une modification du paradigme *MapReduce* afin de pouvoir retourner des résultats intermédiaires qui seront ensuite représentés. Pour cela, ils émettent deux hypothèses : 1. le résultat est suffisamment petit pour être contenu dans la mémoire de l'ordinateur et être rapidement transféré sur le réseau, et 2. il est possible d'effectuer une opération inverse à l'agrégation afin de retirer des résultats partiels des résultats agrégés.Ces résultats intermédiaires sont transmis indépendamment et permettent d'obtenir une réponse partielle à la requête. Ainsi, la représentation est complétée et son exactitude augmente au fur et à mesure que les réponses sont récupérées. L'émission de résultats partiels empêche l'utilisation d'algorithmes nécessitant plusieurs parcours des données.

Une autre approche consiste à utiliser une infrastructure de calcul haute performance en utilisant un *super-ordinateur*. Les travaux de [116] présentent un système de visualisation de coordonnées parallèles s'appuyant sur ce type d'architecture. Leurs travaux constituent une extension des travaux de [101], notamment en ce qui concerne l'approche multi-échelle de la représentation ainsi qu'une agrégation des éléments définis par les utilisateurs. Ils souffrent cependant du même défaut visuel, c'est-à-dire d'une possible discontinuité dans les polylignes représentées due à l'agrégation par binning. Ils s'appuient notamment sur la bibliothèque d'indexation par bitmap *FastBit* [27] pour accélérer les processus de traitement d'accès, exploités notamment pour de l'exploration de données massives. L'utilisation de cette bibliothèque leur permet d'avoir un accès aux données et des temps de calcul très rapides. Dans leurs expériences, les auteurs s'intéressent notamment au suivi de particules au cours du temps. La technique leur permet, dans leurs conditions expérimentales et en utilisant 150 processeurs de calculs (nœuds), d'effectuer le suivi de 10000 particules en 2 secondes dans un jeu de données de plus de 1 TB. Il est difficile de réellement se comparer à ce type de travaux, notamment au vu de la différence de matériel utilisé [1], mais ils constituent, de part les objectifs que nous partageons, une référence qu'il fallait mentionner. Ils permettent cependant d'établir un objectif de performances : parcourir et explorer de très gros jeux de données (de l'ordre du Tera-octet) dans des temps très réduits.

1.3 Contributions

Nous avons orienté nos travaux afin de permettre la visualisation interactive de données massives avec une infrastructure cloud, accessible pour la majorité des utilisateurs potentiels. Or, l'utilisation d'une architecture de ce type pour la visualisation d'information interactive implique que les données soient nécessairement stockées sur une machine différente de celle où les utilisateurs

1. D'un côté un super-ordinateur, performant mais coûteux et difficile d'accès et de l'autre, une infrastructure distribuée de type *cloud* beaucoup plus accessible mais avec des performances moindres.

effectueront leurs analyses. Ainsi, les techniques de traitements et d'analyses ont dû s'adapter, et la visualisation d'information ne fait pas exception.

Tout d'abord en section 2 nous présentons les différents problèmes soulevés par l'utilisation de données massives en visualisation de données. Ensuite en section 3, nous présentons l'infrastructure logicielle utilisée pour permettre le traitement et la représentation de données massives. Ensuite en section 4, nous expliquerons pourquoi une approche de pré-calcul et stockage de l'intégralité des requêtes utilisateurs est envisageable avec Adjasankey. Nous montrons en section 5 que cette approche pose des problèmes de complexité avec les coordonnées parallèles et présentons une autre approche envisageable et comparons les deux approches. Finalement, en section 6 nous donnons nos conclusions sur les méthodes présentées.

2 Problématiques et défis

Même si le matériel informatique évolue constamment [1], les volumes de données à traiter augmentent bien plus rapidement et deviennent bien trop importants pour considérer l'utilisation d'ordinateurs personnels pour plusieurs raisons. Tout d'abord se pose le problème de l'espace de stockage. La plupart des ordinateurs personnels ne permettent pas le stockage de tels volumes de données et la mémoire vive disponible n'est pas suffisante pour permettre le traitement et la manipulation de ces données.

Cette contrainte impose l'utilisation de serveurs pour le stockage des données mais soulève également un autre problème. Si le lieu de stockage des données et celui où elles seront analysées n'est plus le même, cela induit le transfert de ces données pour en permettre l'analyse, or l'envoi de données entre machines nécessite du temps. Il devient donc impératif de prendre en considération le délai nécessaire à l'envoi et à la réception des données, et donc de limiter la taille des données qui seront transférées.

Ensuite se pose le problème du temps de traitement : les volumes de données à manipuler sont tellement importants que les délais nécessaires aux traitements avec des machines et des approches classiques sont beaucoup trop longs. Une solution souvent utilisée pour résoudre ce problème est d'utiliser une approche de calculs distribués, permettant de répartir un calcul sur plusieurs machines.

Cependant, cette approche pose également le problème des délais nécessaires au démarrage des traitements (temps de lancement des *jobs*) : la mise en route et la préparation du traitement ont eux aussi un coût en temps non négligeable qu'il faut prendre en considération. On s'aperçoit donc que la solution utilisée pour résoudre le problème de stockage des données ne fait que transformer le problème. On passe ainsi d'une contrainte de stockage à une contrainte de temps, traitements et transferts. Les problèmes soulevés ici sont d'autant

1. On parle souvent des *Lois de Moore*

plus vrais lorsqu'on veut permettre l'exploration interactive des données. En effet, la vitesse de calcul et de transferts des données devient primordiale pour permettre une interaction fluide et sans latence, surtout lorsque les interactions nécessitent de reparcourir les données.

Le défi de la visualisation de données massives consiste ainsi à trouver une solution prenant en compte ces deux aspects : le stockage et le temps. Pour cela, il est nécessaire d'identifier en amont les fonctionnalités que l'on veut fournir et les besoins en performances des utilisateurs afin d'établir les approches et les solutions à mettre en place.

3 Vue d'ensemble de l'architecture

La solution que nous proposons ici s'appuie sur la mise en place de deux composantes utilisant des techniques et structures adaptées pour permettre la représentation et l'interaction pour des données massives. D'un coté, une plateforme mutualisée associant stockage et calcul distribués pour des données massives, généralement appelée infrastructure *cloud* et d'un autre, un accès aux données *via* le réseau internet pour une représentation distante sur client léger.

La première composante (coté serveur) est mise en place afin de permettre le calcul et le stockage des données qu'il est possible de visualiser sur la représentation. Cette solution permet de concentrer l'ensemble des étapes fortement consommatrices en ressources (mémoire et calcul) telles que les calculs d'agrégation, calculs de poids, *etc.* en amont, puis de transférer une fraction des données sur le client de visualisation. À l'aide de cette architecture, nous avons mis en place une approche basée sur l'abstraction de données et le pré-calcul de l'ensemble des résultats de requêtes utilisateurs. L'ensemble de ces données seront ensuite stockées afin de les restituer à la demande. Cette approche permet de réduire les temps de réponse aux requêtes en les résumant à la récupération des données d'intérêt dans la base de données et au transfert jusqu'au client. Cette solution est cependant basée sur le postulat que dans le cas de données massives, il est plus simple d'obtenir de l'espace mémoire supplémentaire que de réduire le temps de traitement des algorithmes utilisés. Nous montrons en section 4 que cette solution est possible avec Adjasankey grâce aux propriétés des données représentées. En revanche, nous montrons en section 5 que cette approche pose un problème d'explosion combinatoire avec les coordonnées parallèles dû aux propriétés et outils d'interactions de la représentation et présentons une approche de calcul à la demande nécessitant peu de pré-calculs.

La seconde composante (client de visualisation) a également modifiée pour permettre la représentation de données massives (voir chapitres III et IV). Pour cela, nous utilisons des techniques de dessin dites *bas-niveaux*, permettant d'accélérer les temps de rendu. À cela, nous associons une interface accessible

sur navigateur internet permettant d'obtenir un client léger accessible via internet.

3.1 Composante de calcul

Nous avons utilisé l'architecture *Hadoop*, développée par *Apache*, dédiée au stockage et au traitement de données massives de façon distribuée. Nous nous sommes plus particulièrement appuyés sur quatre briques : *HDFS*[1] pour le stockage, *Hbase* et *ElasticSearch* pour la lecture/écriture/manipulation des données et *Spark* pour le traitement distribué des données (voir figure V.1). *Hbase* [135] est une implémentation libre du système de gestion de base de données non relationnelles distribuées BigTable [26] s'appuyant sur *HDFS*, le système de gestion de fichier distribué de *Hadoop*. L'environnement *ElasticSearch* est un service utilisé pour l'indexation et la recherche de données qui apporte notamment un moteur de recherche distribué. Ce service ne sera utilisé que pour l'approche de calcul à la demande (voir section 5). Enfin, *Spark* [147] est un environnement de développement permettant le calcul distribué sur un système de stockage distribué et s'appuie sur le modèle *MapReduce*. Ce modèle est constitué, dans sa version la plus simple, de deux étapes, *map* et *reduce* permettant de décomposer un problème en sous-problèmes traités indépendamment les uns des autres puis de combiner les résultats afin de résoudre le problème initial. Dans l'étape *Map*, les données sont reçues une à une à partir des blocs *HDFS* et sont découpées en association clé/valeur. Ces associations sont ensuite réparties sur chaque *reducer* qui pourra effectuer une opération sur les éléments partageant la même clé. Afin d'améliorer la distribution et la parallélisation des calculs, ces étapes peuvent être combinées sous différentes formes en mélangeant différentes étapes de *map* et de *reduce*. Un service web permet ensuite de gérer les demandes d'accès et les connexions à la base pour la transmission des données d'intérêt.

3.2 Composante de visualisation

Le client de visualisation ne fait aucun calcul sur les données mais se contente du rendu visuel et de la gestion des interactions utilisateurs. Afin d'améliorer les performances de rendu, nous nous sommes appuyés sur une technique de dessin permettant d'utiliser le processeur graphique (appelé *GPU*) de la machine. Celui-ci est optimisé pour les calculs spécifiques à l'affichage (graphisme 3D en particulier) grâce notamment à une forte parallélisation des calculs. Il présente cependant une plus grande complexité d'utilisation puisqu'il fait appel à des techniques de programmation dites bas-niveaux.

1. Hadoop Distributed File System

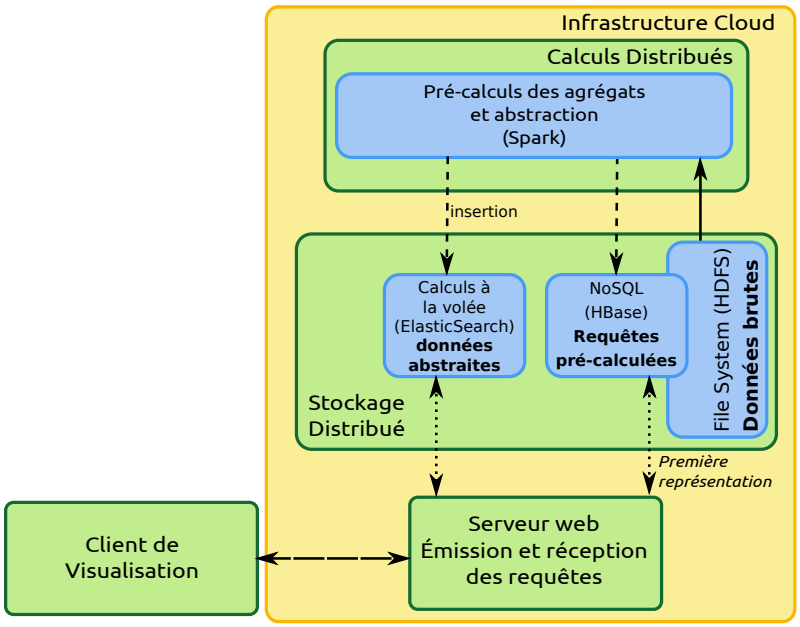


Figure V.1: Infrastructure logicielle permettant le calcul à la demande des résultats de requêtes utilisateurs pour la représentation de données massives. La première représentation est pré-calculée et stockée avec Spark et HBase. Les interactions nécessitant la réception de nouvelles données sont soit pré-calculées et stockées dans HBase soit calculées à la demande avec ElasticSearch.

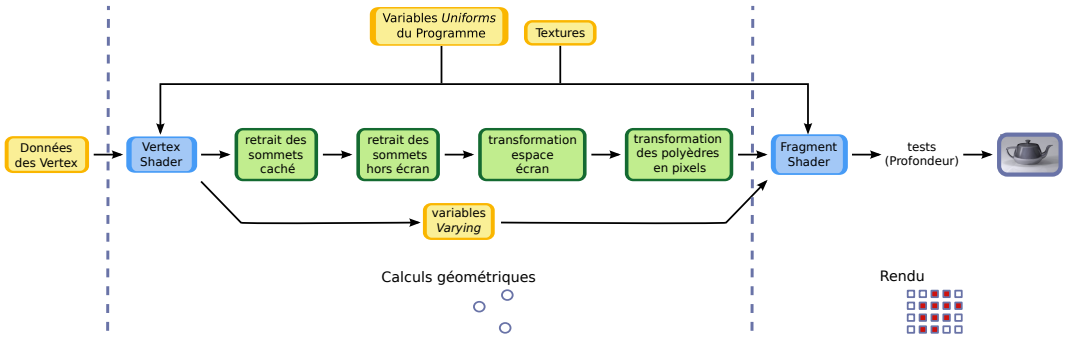


Figure V.2: Pipeline de traitement OpenGL.

Les sommets (*vertex* en anglais) qui caractérisent un polyèdre forment les données élémentaires et leurs coordonnées se placent dans un espace à trois dimensions (l'espace objet). Entre ces points sont tracés les contours et surfaces des objets à représenter par le processeur graphique. Le dessin au GPU est dirigé par des programmes nuanceurs (*shaders* en anglais) qui permettent de programmer les différentes opérations à effectuer pour obtenir la représentation (voir figure V.2). Ces programmes utilisent leur propre langage (Shading Language) et plusieurs ont été définis par les différents constructeurs :

le *GLSL* (openGL Shading Language) proposé par OpenGL (celui que nous avons utilisé), le HLSL par DirectX, Cg par NVidia ou encore MetalSL par Apple. Ces nuanceurs effectuent leurs calculs sur un unique sommet du polyèdre de façon isolée, la parallélisation des opérations sur le GPU interdisant l'accès aux informations des sommets voisins. Nous nous sommes principalement intéressés à deux nuanceurs du *GLSL* : le *Vertex shader* et le *Fragment shader* (ou *Pixel shader*). Le *vertex shader* permet de gérer les tâches de transformations géométriques. C'est grâce à ce programme que l'on détermine la projection de coordonnées des sommets à partir de l'espace objet tri-dimensionnel dans l'espace écran bi-dimensionnel. C'est dans ce programme qu'est effectuée la rastérisation (transformation des surfaces en pixels) en utilisant une *matrice de projection*. Ensuite, le *fragment shader* permet de gérer les transformations de rendu pour chaque pixel à l'écran. Son but est d'effectuer les calculs de couleurs de chaque pixel. En ayant une approche très bas-niveau des phases de rendu graphique, il est possible d'optimiser les calculs de dessin afin d'augmenter les performances de l'outil de visualisation et réduire les temps de calcul. L'outil de visualisation et les différents outils d'interaction ont ainsi été implémentés avec le langage *C++* associé à la bibliothèque de rendu *OpenGL* puis transformé en *JavaScript* et *WebGL* grâce au compilateur Emscripten [148] basé sur une machine virtuelle bas-niveau(LLVM, de l'anglais *Low-Level Virtual Machine*). Emscripten permet notamment de compiler du bitcode LLVM en JavaScript pour pouvoir l'exécuter dans n'importe quel navigateur web récent. Cette fenêtre de visualisation et d'interaction est ensuite intégrée dans le client léger, une page internet écrite en HTML-CSS et javascript.

4 Adjasankey : une approche stockage

4.1 Structure de données

La première image fournie par Adjasankey correspond au plus haut niveau de l'arbre de partitionnement (hormis la racine) puis l'utilisateur peut demander à visualiser plus ou moins de détails sur un méta-nœud. Le positionnement matriciel des éléments dans notre représentation implique que la sélection est faite soit sur la représentation du méta-nœud source des arcs, soit sur celle du méta-nœud cible des arcs. L'ouverture d'un méta-nœud implique donc de récupérer l'ensemble des descendants du méta-nœud dont les arêtes ont la même orientation que celui sélectionné. Il faut également pouvoir récupérer l'ensemble des arcs reliant ces descendants aux éléments déjà visualisés sur la représentation. Ces opérations, bien que relativement simples, ont un coût non négligeable dont on peut s'abstraire.

La première étape de pré-calcul que nous avons mise en place consiste à dupliquer le graphe et l'arbre de partitionnement en fonction de l'orientation

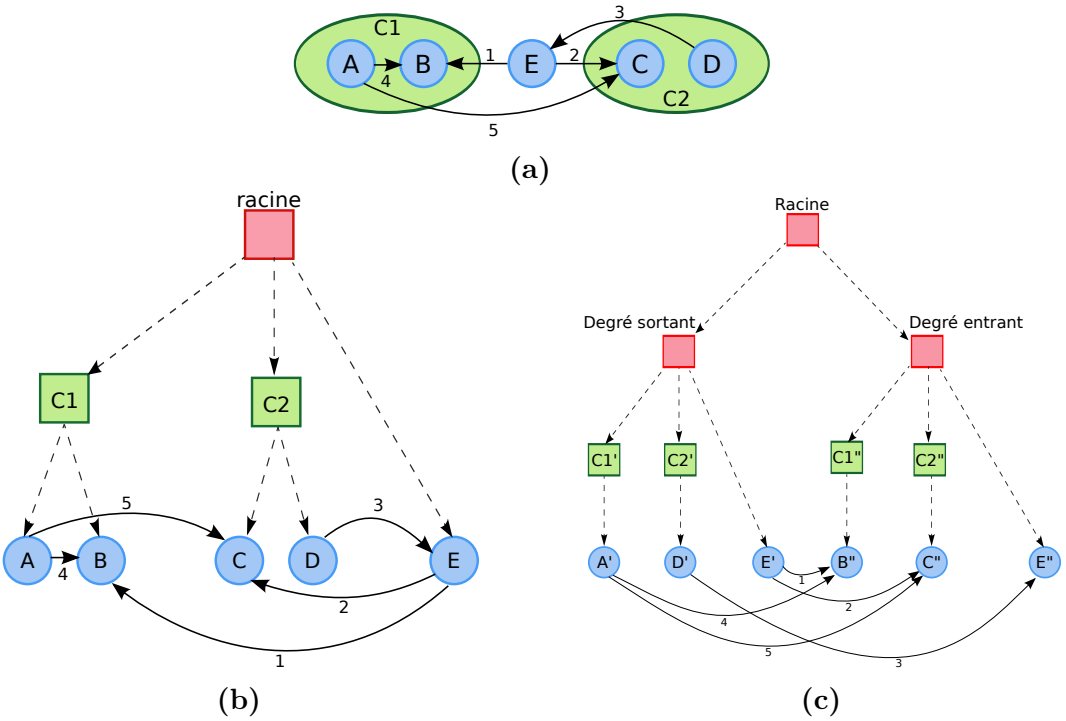


FIGURE V.3: Création de la structure de données pour la méthode de pré-calcul. (a) Le jeu de données initial partitionné et (b) l'arbre de partitionnement hiérarchique correspondant. Cet arbre est dupliqué afin de séparer les éléments en fonction de leur orientation et permet d'obtenir l'arbre hiérarchique (c). Dans un soucis de clarté, les dessins de graphes ne montrent pas les méta-arcs, uniquement les arcs entre les feuilles de l'arbre.

des éléments [1] (voir figure V.3). Cette duplication consiste ainsi à créer deux branches dans l'arbre depuis la racine, l'une qui contiendra l'ensemble des nœuds et méta-nœuds sources des arcs et méta-arcs, et l'autre les cibles. On peut ainsi, lorsqu'on sélectionne un méta-nœud, récupérer directement les descendants de même orientation, leurs voisinages ainsi que le poids des arcs adjacents sans devoir vérifier l'orientation de chaque élément. Chaque nœud et arc est ainsi référencé dans un système d'indexation par *clé/valeur* avec pour clé un identifiant de (méta-)nœud ou de (méta-)arc et en valeur, l'adjacence du nœud ou les extrémités de l'arc (voir figure V.4). Cette séparation des nœuds en fonction de leur direction permet également de potentiellement réduire la taille de l'index lors de la recherche d'un élément. Dans le meilleur des cas, on peut réduire par deux la taille de cet index, lorsque les (méta-)nœuds sont soit source soit destination mais jamais les deux en même temps. Cette réduction n'est

1. nous parlerons d'orientation ou de direction d'un nœud en faisant référence à l'orientation de ses arcs incidents

cependant pas garantie, le pire cas étant lorsque tous les (méta-)nœuds sont à la fois sources et cibles d'arcs. La figure V.3 présente l'étape de duplication, en figure V.3a le graphe d'origine et en figure V.3c l'arbre de partitionnement dupliqué résultant. On peut voir sur figure que le nœud E est dupliqué puisqu'il a des connexions à la fois entrantes et sortantes. Par récursivité de la méthode, les méta-nœuds représentant des nœuds cibles et sources de connexions sont dupliqués eux aussi.

La seconde étape consiste à construire les méta-arcs connectant chaque nœud de l'arbre de partitionnement dupliqué à l'ensemble de ses possibles voisins, c'est-à-dire les méta-nœuds du même niveau hiérarchique mais aussi aux nœuds et méta-nœuds des autres niveaux de l'arbre. On peut ainsi récupérer les voisins d'un nœud ou les extrémités d'une arête par un simple accès dans la table d'indexation.

A'	B" - C"
D'	E"
E'	B" - C"

B"	E' - A'
C"	E' - A'
E"	D'

(a)

C1'	B" - C"
C2'	E"
E'	C1" - C2"

C1"	E' - A'
C2"	E' - A'
E"	C2'

(b)

C1'	C1" - C2"
C2'	C1" - C2"

C1"	C1' - C2'
C2"	C1' - C2'

(c)

Figure V.4: L'arbre hiérarchique partitionné (voir figure V.3c) est stocké sous forme de liste d'adjacence en distinguant les éléments sources et cibles des arcs. Des listes équivalentes non montrées ici sont également stockées afin d'indexer les indices des (méta-)arcs et les (méta-)nœuds aux extrémités.

4.2 Démonstration de faisabilité

Cette solution n'est envisageable que grâce à la nature des données représentées dans Adjasankey. Ces données sont de type relationnel, des entités pondérées, reliées par des connexions elles aussi pondérées et organisées dans une structure hiérarchique sur un nombre restreint de niveaux. On peut ainsi déterminer le nombre maximum d'entités et de connexions que notre approche stockage impose de pré-calculer et de stocker. Soit $G' = (G, H)$ le graphe partitionné du graphe orienté pondéré $G = (V_G, A_G, p)$ sur l'arbre de partition $H = (V_H, A_H)$ de hauteur h. Les feuilles de l'arbre H correspondent ainsi dans le pire des cas à l'ensemble V_G pour chacune des deux branches de l'arbre, et la complexité en nombre d'éléments de cet ensemble est $O(2.|V_G|)$. De même, la taille de l'ensemble représentant les arcs entre les feuilles de l'arbre est $|A_G|$.

Le pire cas de figure avec notre approche correspond à l'absence d'agrégation des entités et des connexions entre les différents niveaux de l'arbre. On peut ainsi déterminer dans le pire cas la complexité en nombre d'éléments de V_H : $O(2.h.|V_G|)$. Cela implique que pour deux niveaux i et j de H, les méta-arcs de i vers j de l'arbre correspondent à l'ensemble A_G et contient donc $O(|A_G|)$ méta-arcs dans le pire des cas. Le nombre de combinaisons de niveaux dans l'arbre étant défini par h^2, on peut ainsi logiquement déduire que la taille maximale de l'ensemble A_H sera $O(h^2.|A_G|)$ Par cette méthode, nous pouvons pré-calculer, avec un coût en stockage restreint, l'ensemble des informations potentiellement demandées par un utilisateur.

On constate que tous deux sont dépendants de h. Cela constitue une limite à notre technique ; h doit impérativement rester petit pour limiter la quantité d'éléments à stocker. Cette technique est donc adaptée à la représentation de graphes dont la hauteur de l'arbre de partitionnement est relativement petite.

Le cas d'étude présenté en chapitre III section 4 montre l'utilisation d'Adjasankey pour la visualisation des visites utilisateurs dans un site internet. La structuration hiérarchique utilisée correspondait à celle des différentes catégories et sous-catégories du site internet, or ce type d'arbre de partitionnement a généralement une hauteur faible. Dans le cas où la nature des données à analyser ne permettrait pas d'extraire une hiérarchie, le choix de l'algorithme de partitionnement devra être guidé par la hauteur de l'arbre de partitionnement qui en résultera. Il faudra ainsi choisir le meilleur ratio entre le coût en stockage des pré-calculs et la hauteur de l'arbre pour l'exploration multi-échelle. À titre d'exemple, le jeu de données initial du cas d'étude présenté en chapitre III représentait environ 280000 paires de sommets distincts et, après pré-traitement, la structure de données utilisée contenait 700000 éléments (sommets et arcs) hiérarchisés en trois niveaux (hors racine).

Malgré un coût en stockage non négligeable mais que l'on considère comme acceptable, cette approche permet de limiter le délai entre la requête de l'utilisateur et la représentation de la réponse. Ce délai est ainsi réduit aux délais de recherche des éléments dans la base de données agrégées et aux délais de transferts.

5 Coordonnées parallèles et données massives

5.1 Approche par pré-calcul

Pour une utilisation avec les coordonnées parallèles, l'approche pré-calcul implique d'envisager l'ensemble des interactions utilisateurs et de stocker l'ensemble des réponses correspondantes. Nous présentons ici les estimations de complexité qu'impliquerait le pré-calcul de ces réponses.

Afin d'évaluer la taille totale des pré-calculs à effectuer pour stocker les résultats de toutes les requêtes possibles, nous allons commencer par évaluer le

nombre d'éléments sélectionnables dans la représentation. Ensuite nous évaluerons la taille en nombre d'éléments de la réponse d'une requête. La complexité en stockage des pré-calculs sera finalement obtenue par la multiplication des deux résultats intermédiaires.

Nombre d'éléments dans l'abstraction Soit un jeu de données J contenant n éléments avec d dimensions (représentées par des axes) et un algorithme d'agrégation qui agrège les éléments de J en maximum k agrégats, k étant fourni en entrée. Considérant le nombre d'agrégats, le meilleur cas est celui où chaque axe contient un seul agrégat. Il y a donc autant d'agrégats que d'axes, la complexité en nombre d'agrégats est donc $\Omega(d)$. Dans le pire cas, il y a au contraire k agrégats pour chacun des axes, donc avec une complexité $O(k \times d)$.

Le nombre de connexions entre deux axes est défini par, dans le meilleur cas $\Omega(1)$ (un agrégat par axe et une donc une unique connexion pour une paire d'axes) et dans le pire cas $O(k^2)$ (chacun des k agrégats des deux axes sont reliés). Pour déterminer le nombre de connexions à pré-calculer et à stocker, les connexions entre axes non successifs doivent également être prises en compte, celles-ci étant accessibles après permutation d'axes (voir chapitre IV section 4.3). Le nombre de paires d'axes distinctes étant égal à $(d(d-1))/2$, on obtient une complexité en nombre d'arêtes dans le meilleur des cas en $\Omega((d(d-1))/2)$, et dans le pire des cas en $O(k^2(d(d-1))/2)$.

La complexité en nombre d'éléments (agrégats et connexions) pouvant être sélectionné est donc dans le meilleur des cas en $\Omega(d + (d(d-1))/2)$ et dans le pire des cas en $O(kd + k^2(d(d-1))/2)$. Cela se simplifie en $\Omega(d^2)$ et $O(k^2d^2)$.

Taille du résultat d'une requête de sélection Lorsqu'un utilisateur sélectionne un élément de la représentation (agrégat ou connexion), il demande l'extraction d'un ensemble d'éléments appartenant à un certain intervalle de valeurs pour une dimension donnée (deux dans le cas de la sélection d'une connexion). Cet ensemble d'éléments est ensuite mis en valeur par surbrillance sur toute la représentation, c'est-à-dire sur tous les axes (voir section 4.3). Les interactions permettant l'accès à l'ensemble des permutations d'axes possibles, il faut considérer parmi les résultats potentiellement demandés par l'utilisateur l'ensemble des agrégats et connexions pouvant être visualisés, et cela quel que soit l'ordre des axes.

Si on considère le pire des cas, l'élément sélectionné représente un ensemble d'éléments qui se répartit sur l'ensemble des intervalles de valeurs des autres attributs (voir figure V.5). Cela signifie que pour les autres attributs, les éléments sélectionnés seront répartis sur tous les agrégats de chaque axe. La sélection implique également de récupérer les connexions reliant ces agrégats entre axes consécutifs. Ainsi, l'ensemble retourné par la requête de sélection correspond

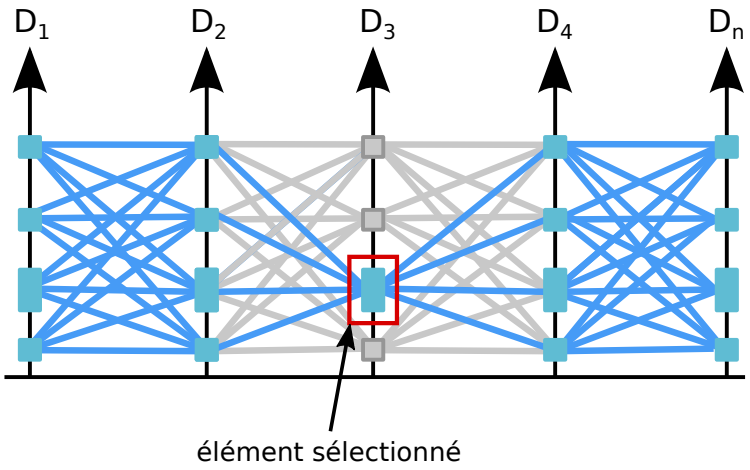


FIGURE V.5: Pire cas de sélection d'un agrégat : la sélection de l'agrégat sur l'axe D3 implique la mise en valeur de cet ensemble sur les autres axes de la représentation (en bleu sur la figure). Or, cet ensemble est réparti sur chaque connexion et agrégat de la représentation. Ce schéma ne montre pas les permutations d'axes mais celles-ci doivent être prévues dans le pré-calcul.

dans le pire des cas à la quasi-totalité [1] des agrégats et connexions de l'abstraction. La complexité en taille de cet ensemble a été montrée précédemment et est en $\Omega\left(d^2\right)$ et $O\left(k^2d^2\right)$.

Pré-calcul complet Les complexités obtenues ci-dessus montrent que si on reste dans l'objectif de pré-calculer l'ensemble des requêtes utilisateurs possible, il faut dans le pire des cas stocker pour chaque élément de la représentation (agrégat et connexion) la quasi-totalité des éléments de la représentation. La complexité en stockage totale se dénombre donc en multipliant le nombre d'éléments de la représentation par la taille de l'ensemble retourné par la sélection de ces éléments.

Le pré-calcul et le stockage de la réponse de l'ensemble des résultats de requête pour l'ensemble des arêtes sélectionnables ont donc un coût en mémoire dans le meilleur des cas en $\Omega((d^2)^2)$ et dans le pire des cas en $O((k^2d^2)^2)$. Cela se simplifie en $\Omega(d^4)$ et $O(k^4d^4)$.

Analyse des complexités On s'aperçoit ainsi que le coût en stockage est loin d'être négligeable, et cela en considérant uniquement les interactions de sélection simple. On imagine facilement que les interactions par filtres de valeurs induiront une augmentation très significative de ces complexités. En effet, il faudra envisager à chaque étape toutes les configurations possibles de filtres sur

1. les agrégats non retournés seront les agrégats de l'axe sur lequel a été fait la sélection

chaque axe de la représentation. Ces complexités montrent également que le nombre d'éléments n'a pas d'impact sur la complexité en mémoire, et également que ces complexités dépendent essentiellement du nombre d de dimensions du jeu de données et du paramètre k fourni à l'algorithme d'agrégation.

Ces résultats montrent que le choix du pré-calcul doit être fait en considérant deux facteurs : le temps nécessaire au pré-calcul et le coût du stockage. En effet, l'utilisation du pré-calcul est tout particulièrement intéressante lorsque d et k sont relativement petits. Cependant le coût en stockage ainsi que le temps nécessaire à l'ensemble de ces pré-calculs doivent être comparés au délai nécessaire à la réponse d'une requête sur les données agrégées. De plus, nous ne prenons en compte ici que les cas de sélection simple d'éléments mais les sélections complexes (intersection de trois ensembles par exemple [1]) ne sont pas accessibles et nécessiteraient d'être pré-calculées également.

5.2 L'approche par calcul à la demande

Présentation de la méthode

Afin de réduire l'impact mémoire qu'impose le pré-calcul complet, nous avons mis en place une approche permettant de calculer à la demande les résultats de requêtes utilisateurs. Comme pour l'approche pré-calcul, ce système utilise HDFS, Spark et HBase pour le calcul des agrégations de la première représentation mais s'appuie également sur l'environnement *ElasticSearch* pour la partie calcul des réponses aux requêtes utilisateurs. Cette approche nécessite également un pré-calcul, beaucoup moins long cependant, permettant d'indexer, pour chaque ligne des données brutes, les informations sur les agrégats auxquels elles appartiennent (agrégat, axe concerné et position relative dans l'intervalle de valeur de l'agrégat). Les réponses aux requêtes de sélections sont calculées *à la volée* de façon distribuée ; chaque nœud de calcul effectue un filtre sur les données et agrège les résultats, puis l'ensemble des réponses partielles sont agrégées puis envoyées au client de visualisation.

5.3 Comparaison des deux méthodes

Nous avons effectué un ensemble de tests afin d'évaluer les différences de performance entre les deux approches. Pour ces tests nous avons utilisé une infrastructure équipée de 15 exécuteurs, chacun équipé de 2 processeurs à 6 cœurs *hyper-threadés* cadencés à $2.1GHz$ et de $5GB$ de RAM. Le serveur permettant le lien entre l'interface utilisateur et l'infrastructure Hadoop est équipé d'un processeur à 4 cœurs cadencés à $3.3GHz$ et avec $4GB$ de mémoire vive.

1. le résultat de l'intersection de deux ensembles est déjà accessible par la sélection d'une connexion

Protocole de tests

Deux tests ont été effectués afin de comparer les deux approches, le temps de préparation de la base de données (pré-calculs et insertion) et le temps de réponse aux requêtes. Les tests de préparations de la base de données consistent à effectuer l'ensemble des pré-calculs nécessaires à la mise en place de la méthode et à insérer les résultats dans la base de données. Pour l'approche pré-calculs, cela correspond aux calculs d'agrégation, au pré-calcul et au stockage des résultats de l'ensemble des requêtes envisageables. Pour la méthode de calcul à la demande, cela correspond aux calculs d'agrégation et à l'insertion dans ElasticSearch. Le test sur le temps de réponse aux requêtes consiste à effectuer toutes les requêtes de sélections possibles (agrégats et arêtes) sur un jeu de données et à calculer le temps moyen de réponse. Afin de limiter le nombre de requêtes, nous utiliserons les requêtes possibles à partir de coordonnées parallèles représentées avec l'ordre initial des attributs et ne prendrons pas en compte les résultats issus des changements d'ordre des axes. Les temps mesurés correspondent au délai entre l'émission d'une requête par le client de visualisation jusqu'à la réception de la réponse par ce même client. Les tests sont effectués trois fois afin de pouvoir généraliser les résultats et de pouvoir identifier les comportements anormaux.

Dans les tests impliquant l'utilisation d'ElasticSearch, deux conditions expérimentales différentes ont été testées. En effet, ElasticSearch est équipé d'un système de mémoire cache permettant d'optimiser les temps de réponse pour des requêtes fréquentes. Nous avons donc effectué deux mesures utilisant ou non le cache mémoire. Dans le cas sans cache, la mémoire est vidée après chaque requête. Dans le cas avec mémoire cache, l'ensemble des requêtes sont effectuées sans être enregistrées afin de pré-charger le cache avec les paramètres par défaut d'ElasticSearch puis ré-effectuées avec enregistrements des temps de réponse. Les mesures obtenues ne donnant pas de différences notables entre les deux tests, nous ne présenterons les résultats utilisant le cache mémoire.

Les jeux de données évalués pour ces tests sont artificiels (non issus ou extraits d'un jeu de données réel) et ont un nombre de dimensions fixé à $d = 15$. Le choix de cette valeur est arbitraire mais a été effectué en ciblant un nombre de dimensions relativement grand tout en constituant un jeu de données utilisable en terme de représentation et d'exploration. Nous avons fixé le nombre maximal d'agrégats résultant de l'algorithme d'agrégation à $k = 15$ en suivant un raisonnement similaire. Trois types de données avec différents degré de corrélation ont été évaluées. Les données indépendantes ne présentent aucune corrélation entre les éléments et correspondent à une approximation du pire cas des pré-calculs. Le jeu de données corrélées comporte des corrélations entre les éléments, et le jeu de données *meilleur cas* pour lequel chaque élément a la même valeur pour chaque dimension. Différentes tailles de jeux de données seront également utilisées (10 tailles différentes au total) allant de 10^6 à 10^9

éléments.

Résultats

Résultats aux tests de pré-calculs La figure V.6 présente les résultats des tests de temps de pré-calculs. On peut constater deux points particulièrement intéressants :

— quelles que soient les conditions expérimentales, la méthode utilisant ElasticSearch est bien plus rapide que celle avec HBase,

— l'évolution du temps de pré-calcul par rapport au nombre d'éléments semble suivre une fonction linéaire.

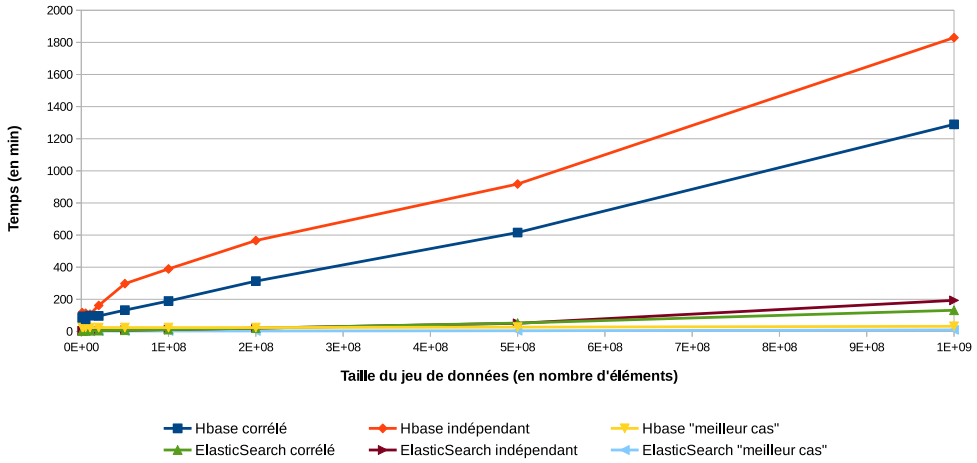


Figure V.6: Temps de pré-calculs pour les deux méthodes avec différents jeux de données de taille variable.

Le premier point s'explique par le fait que les pré-calculs nécessaires à l'utilisation de la méthode ElasticSearch sont plus simples et moins complets que ceux de la méthode HBase. Le second que la taille du jeu de données a un impact direct sur le temps nécessaire pour effectuer le pré-calcul, ce qui semble cohérent puisque le pré-calcul nécessite de parcourir l'ensemble des données. La figure V.7 nous montre l'impact du nombre de machines allouées aux pré-calculs de la méthode HBase sur le temps d'exécution. On peut observer que l'augmentation du nombre de machines induit une baisse du temps d'exécution. On observe également que la courbe de performance (*speedup*) suit une fonction linéaire. Ces résultats nous montrent que dans le cadre de l'expérience menée, l'augmentation du nombre de machines allouée au pré-calcul induit bien une augmentation de la puissance de calcul et par conséquent, une réduction du temps de traitement.

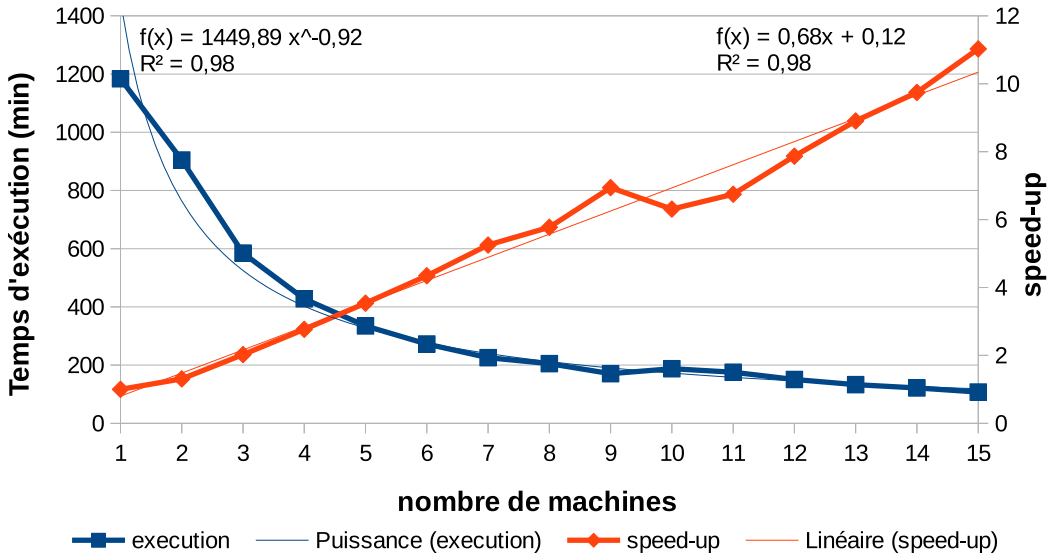


FIGURE V.7: Temps d'exécution et *speedup* en fonction du nombre de machines allouées aux pré-calculs pour un jeu de données de 10^7 éléments.

Résultats aux tests de requêtes Les mesures enfigure V.8 montrent les résultats obtenus avec HBase et ElasticSearch.

jeu de données *meilleur cas* Les résultats obtenus sont très faibles et relativement stables, les variations observables correspondant probablement à des latences réseaux. Malgré ces latences, les temps de réponse restent inférieurs au dixième de seconde.

jeux de données corrélées et indépendantes Les résultats montrent que les temps de réponse sont inférieurs à la seconde pour des données de moins de 5.10^8 éléments. Seules les requêtes impliquant une sélection de nœuds en utilisant ElasticSearch subissent une augmentation des temps de réponse (plus d'une seconde) pour des jeux de données dépassant 5.10^8 éléments. Quelles que soient les conditions, les temps de réponse obtenus avec la méthode de stockage de l'ensemble des pré-calculs (HBase) sont systématiquement inférieurs à ceux obtenus avec ElasticSearch. Les résultats montrent une évolution linéaire du temps de requêtes par rapport au nombre d'éléments du jeu de données, et ce quels que soient la taille ou le type du jeu de données envisagé.

La figure V.9 montre l'impact du nombre de machine sur le temps de réponse aux requêtes. On s'aperçoit que la courbe de performance n'est pas stable mais suit globalement un fonction linéaire. Là également, cela indique qu'augmenter le nombre de machine allouées à ElasticSeach améliore ses performances et permet de réduire le temps de réponse aux requêtes de sélection.

Discussion des résultats

Les résultats obtenus, que ce soit sur les temps de requêtes ou sur les temps de pré-calcul associés aux tests de scalabilité confirment que l'infrastructure supporte correctement la prise en charge de gros jeux de données. Ces résultats tendent à montrer que les performances de calculs de la plateforme sont directement liées au nombre de *nœud de calcul* qu'on lui a alloué. L'augmentation du nombre de nœuds de calculs induit une augmentation de la puissance de calculs, que ce soit du coté d'HBase pour les pré-calculs des données ou pour ElasticSearch pour le calcul de réponses aux requêtes à la demande. Ainsi, la plateforme que nous avons mis en place répond correctement au critère de scalabilité horizontale (lien entre puissance de calcul et nombre de machine allouée) propre à ce type d'infrastructure. Par ailleurs, les tests que nous avons effectués ne nous permettent pas d'identifier de limite ou de plafond à cette relation entre nombre de nœuds et puissance de calculs. Rien dans les résultats obtenus ne tendent à indiquer que nous approchions des limites des logiciels utilisés.

Ces tests nous confirment également que les deux approches présentées ont toutes deux des avantages et inconvénients. L'approche de pré-calcul et de stockage semble particulièrement intéressante si l'on considère l'utilisation de très grand jeux de données ou si l'utilisateur préfère favoriser l'optimisation des temps de réponse aux requêtes. Cela ne peut se faire cependant qu'au détriment de l'espace de stockage des données et au délai de préparation des données, tous deux importants. Au contraire, l'approche de calcul à la demande semble plus adaptée lorsque les jeux de données sont de taille plus raisonnable ou lorsqu'on veut réduire le temps de préparation des données et l'espace de stockage nécessaire. Cette approche aura cependant pour inconvénient de fournir des temps de réponse aux requêtes plus longs pour des grands jeux de données.

Une approche hybride Une autre approche pourrait consister à utiliser conjointement les deux techniques. Deux cas d'utilisation seraient alors possibles :

— utiliser l'approche de calcul à la demande comme solution initiale et stocker les réponses aux requêtes utilisateurs au fur et à mesure qu'elles sont effectuées, ou

— pré-calculer une portion des données, par exemple les requêtes qui seraient les plus longues à obtenir avec ElasticSearch, et calculer le reste à la demande.

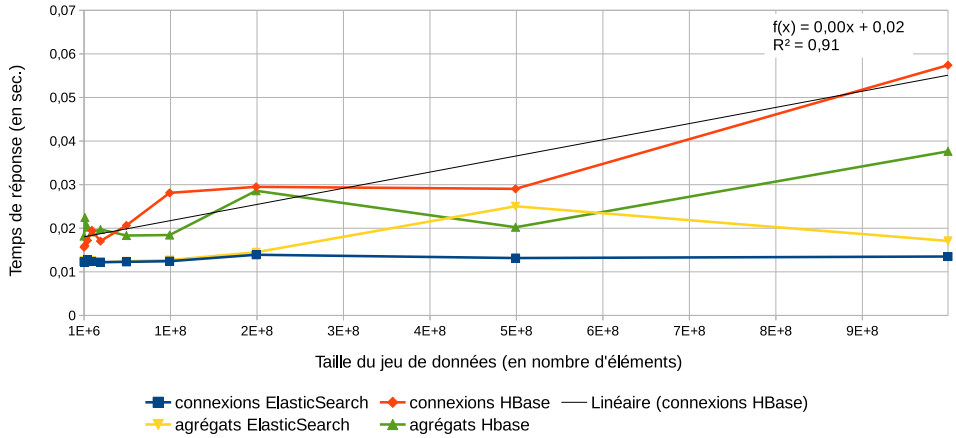
Le premier cas d'utilisation permet un accès rapide aux données les plus populaires, qui semble particulièrement intéressant pour une utilisation par plusieurs utilisateurs. Le second cas quant à lui consiste plutôt à optimiser l'espace disque disponible en utilisant la méthode de pré-calcul sur le maximum

de données et à utiliser la méthode de calcul à la demande pour les données qui ne peuvent être stockées faute de place.

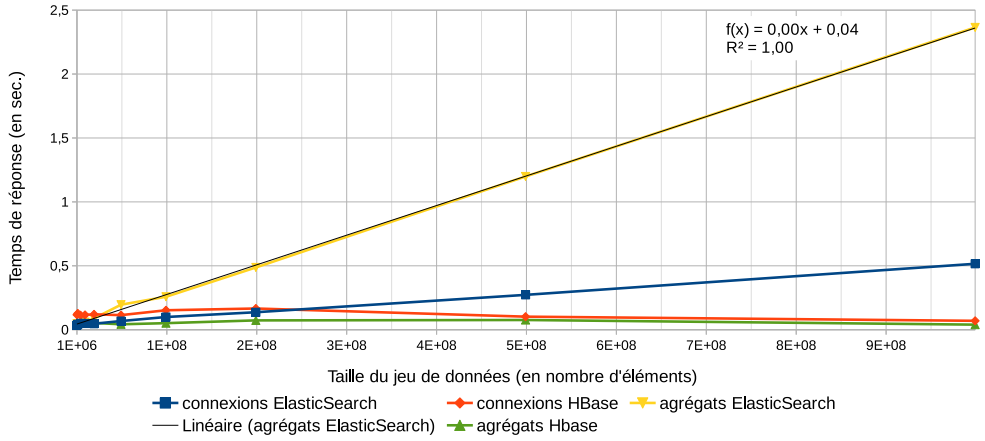
Une telle approche permettrait de limiter les inconvénients des deux approches (coût en stockage et temps de réponse) tout en optimisant les performances en fonction de l'infrastructure accessible.

6 Conclusion

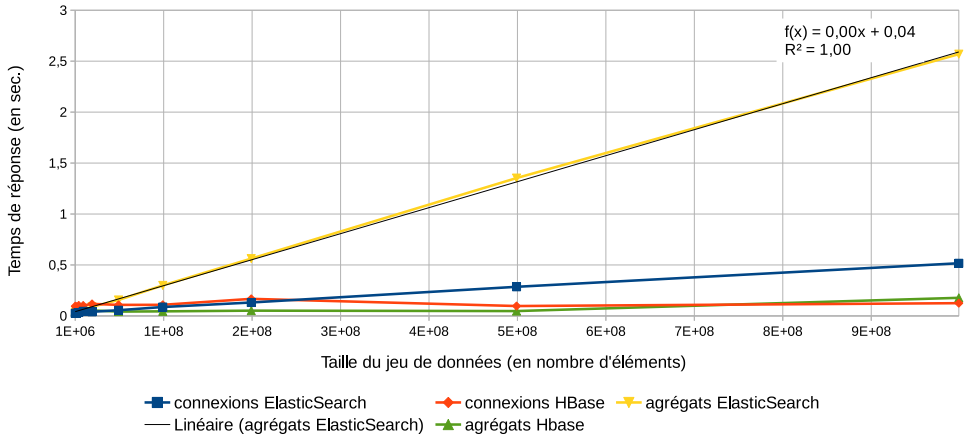
Nous avons présenté dans ce chapitre deux approches, orientées stockage ou calcul à la demande, afin de permettre la visualisation de données massives avec les outils décrits dans les chapitres III et IV. Pour cela, nous avons dû considérer le type de données utilisé afin d'évaluer la faisabilité et l'impact des deux approches sur le stockage, le temps de calcul ainsi que les temps de réponse aux requêtes lorsque les interactions nécessitent de transférer de nouvelles données. Nous nous sommes également appuyés sur une architecture *cloud* propre aux problématiques de données massives : un serveur distant partagé permettant les calculs distribués et parallélisés que l'on associe à un client de visualisation par internet. Ces deux points permettent la mise en place d'outils de visualisation dédiés aux données massives tout en considérant les contraintes de temps et d'espace associées aux traitements de données massives pour en permettre l'interactivité dans des délais limités. Ces travaux posent cependant différentes questions qu'il pourrait être intéressant d'approfondir. L'approche hybride décrite dans la discussion semble une piste intéressante, tout particulièrement pour la mise en place d'un algorithme automatique afin de déterminer quelles sont les données à pré-calculer et celles à calculer à la demande. Une approche multi-échelle pour les données massives permettant d'accéder à la donnée brute pourrait également être intéressante mais pose des problèmes en terme de masses de données à pré-calculer ou en temps de réponse lors de calculs à la demande.



(a)



(b)



(c)

Figure V.8: Temps de réponse moyen aux requêtes de sélection des éléments (agrégats ou connexions) en fonction de la taille du jeu de données. (a) données *meilleur cas*, (b) données indépendantes et (c) données corrélées.

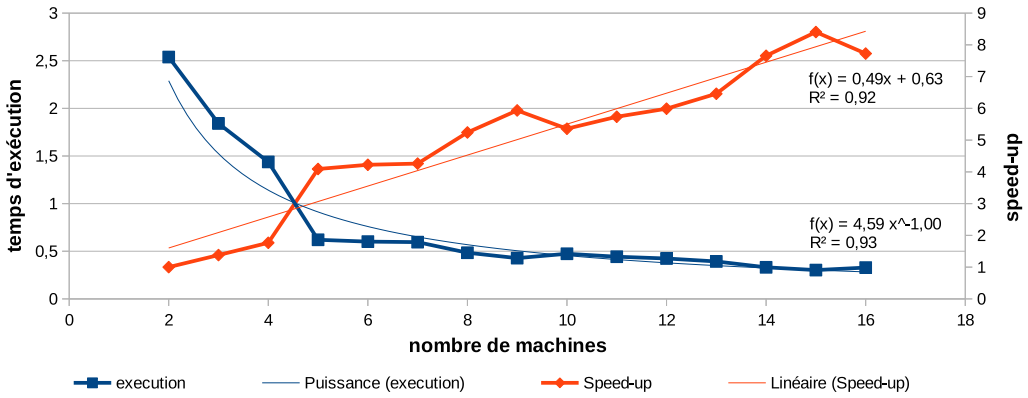


FIGURE V.9: Temps de réponse aux requêtes (sélection d'un nœud dans un jeu de données indépendant de 2.10^8 éléments) et *speedup* en fonction du nombre de machines allouées à ElasticSearch.

Conclusion

Conclusion

Nous avons abordé dans cette thèse la visualisation de données massives. Pour cela, nous nous sommes intéressés aux type de données souvent utilisés en représentation de données : les données relationnelles et les données multivariées. Nous avons élaboré nos recherches sur deux aspects : les techniques de représentations adaptées à ces données et la modification de ces techniques ainsi que la mise en place d'une infrastructure logicielle permettant la visualisation et l'interaction de données représentant des millions/milliards d'éléments.

Dans le chapitre II, nous avons tout d'abord étudié la transmission de l'information de l'existence d'une relation entre deux éléments pour deux représentations fréquentes de données relationnelles : la matrice d'adjacence et le diagramme nœud-lien. Pour cela, nous avons mis en place une évaluation utilisateur visant à identifier l'influence de l'encodage visuel des arêtes sur la résolution de tâches courantes en analyse de données relationnelles. L'analyse des temps de réponses et des taux d'erreurs de nos participants dans la résolution de tâches simples et d'une tâche complexe montre deux résultats. Le premier est qu'il n'y a pas d'influence statistiquement décelable dans le cadre de notre protocole d'évaluation et de nos analyses. Le second est la préférence des participants pour les encodages raisonnablement simplifiés. Ces résultats sont surprenant comparés à ceux d'autres évaluations que l'on trouve dans la littérature et qui montrent des résultats plus significatifs. Ces différences incitent à penser que le positionnement et la duplication des nœuds ont une forte influence sur la lisibilité des représentations.

Ces résultats ont ensuite été mis à profit pour la mise en place d'une nouvelle technique de visualisation adaptée à la représentation de données relationnelles, pondérées et permettant l'exploration multi-échelle décrite dans le chapitre III. Cette nouvelle technique, appelée Adjasankey, combine le concept visuel des diagrammes de Sankey et le positionnement orthogonal des éléments caractéristiques des matrices d'adjacences. En effet, les diagrammes de Sankey ont été créés et adaptés pour la représentation de flux pondérés dans un système tandis que les matrices d'adjacence ont été montrées comme étant particulièrement adaptées pour la représentation de grands graphes. Nous

avons également mis en place des outils d'interaction permettant l'exploration à différents niveaux de précision.

Dans le chapitre IV, nous nous sommes intéressés aux techniques de représentation de données multi-variées massives. Nous avons mis en place une technique de représentation basée sur les coordonnées parallèles pour la représentation de données abstraites. Cet outils met également à disposition de l'utilisateur différents outils d'interaction permettant de modifier la représentation mais aussi de filtrer les données représenter afin de mettre en évidence les schémas d'intérêt.

Ces deux techniques, Adjasankey et les coordonnées parallèles, ont été créées afin de les intégrer dans un système en deux composantes avec d'un côté le serveur de calculs et stockage et de l'autre le client de visualisation. Dans le chapitre V, nous avons montré qu'il était possible, tout particulièrement avec la représentation Adjasankey, d'avoir une approche de pré-calcul et de stockage des requêtes utilisateurs pour permettre l'exploration des données sans délais majeurs de calculs et de transferts. Nous montrons également que même si le stockage des pré-calculs est possible pour les coordonnées parallèles, cette approche implique un coût en temps de calculs et en capacité de stockage que l'on ne peut pas ignorer. Nous présentons ainsi une solution complémentaire permettant d'effectuer les calculs à la demande mais fournissant des performances qui baissent lorsque la taille des données devient importante. Ces deux solutions ne sont pourtant pas totalement antinomiques, et nous fournissons également des hypothèses de combinaisons des deux approches.

Perspectives

Les travaux et résultats présentés dans ce manuscrit offrent plusieurs perspectives.

Encodage visuel des arêtes

Les évaluations utilisateurs et les conclusions de ces études fournissent des pistes pour étendre l'analyse et améliorer notre compréhension des facteurs influençant l'efficacité d'une technique de représentation. Bien que le nombre de participants soit dans la norme du domaine, il pourrait être intéressant de poursuivre les évaluations afin d'augmenter la taille de l'échantillon et améliorer la puissance des tests.

Il serait également intéressant de mettre en place une évaluation permettant de comparer les encodages visuels des arêtes que nous avons présenté avec des techniques décrites dans la littérature : diagrammes nœuds-liens, NodeTrix, Matlink, quilts, *etc.* . En effet, on peut supposer qu'une telle évaluation permettrait de mettre en avant l'efficacité de certaines représentations pour

certaines tâches et pour certains graphes. Une telle mise en évidence pourrait nous permettre d'identifier le facteur visuel prépondérant dans l'efficacité de ces techniques et nous fournir une piste supplémentaire sur la construction d'une technique plus polyvalente ou au contraire à établir un classement des techniques de représentation par efficacité en fonction des tâches et des conditions d'utilisation (taille, densité des graphes, *etc.* .)

Représentation de données relationnelles pondérées

La technique de représentation Adjasankey offre plusieurs possibilités de travaux complémentaires. Nous avons présenté un cas d'utilisation d'Adjasankey pour la représentation de parcours utilisateurs dans un site internet. Hors, Adjasankey ne différencie à aucun moment les liens utilisés par les utilisateurs en fonction de leurs positions dans le chemin complet des utilisateurs. Une possibilité d'amélioration serait ainsi de combiner plusieurs représentations Adjasankey afin de permettre une telle distinction. On peut ainsi imaginer mettre en place une rotation et une distorsion de la représentation afin de suivre une courbe fractale (par exemple avec une courbe de Peano) et optimiser l'utilisation de l'espace.

L'ajout d'outils d'interaction tel que le filtre des éléments par d'autres facteurs que le simple passage par une page spécifique serait également une fonctionnalité intéressante à mettre en place mais soulève des problématiques similaires à celles que nous avons vues pour les coordonnées parallèles, notamment en terme de temps de traitement et de réponse ainsi que d'espace de stockage.

Les coordonnées parallèles

La technique des coordonnées parallèles a déjà largement été étudiée depuis sa création, pourtant notre technique de représentation soulève quelques pistes qui ont été peu étudiées à notre connaissance. En effet, notre technique s'appuie sur l'abstraction de données pour permettre la représentation de données massives, et cette abstraction nous permet notamment de mettre en place un système d'exploration multi-échelle des données abstraites. La représentation de données à différents niveaux de détails est une fonctionnalité essentielle dans l'analyse et l'exploration de données et permettre l'exploration de données massives depuis l'abstraction jusqu'à la donnée brute dans des délais permettant l'interaction en temps réel (inférieurs à la seconde) n'est pas possible avec les techniques actuelles. Par ailleurs les techniques de représentation imposent d'utiliser des algorithmes d'agrégation des données dans des ensembles non chevauchants. Il serait intéressant de développer une adaptation des coordonnées parallèles utilisant des représentations chevauchantes pour données massives ou au contraire d'améliorer les techniques d'agrégation par

l'utilisation d'algorithmes d'apprentissage.

Applications aux données massives

Enfin, l'application de ces techniques de représentation pour les données massives est un domaine encore émergent et les infrastructures et techniques sont vouées à évoluer rapidement dans les années à venir. Il sera évidemment nécessaire de suivre ces évolutions et les améliorations qu'elles fournissent afin d'augmenter les performances obtenues. La mise en application de l'approche hybride décrite dans le chapitre V mérite de plus amples réflexions, notamment sur son application pour les infrastructures limitées en espace de stockage. De plus, nos approches se basent sur le transfert à la demande des données mais il est possible d'envisager d'améliorer ce pan de nos travaux avec le transfert d'une fraction des données répondant aux probables requêtes d'un utilisateur. Cela permettrait de maintenir, côté client, un cache de données utilisable pour améliorer les performance lors de l'exploration, en remplaçant le transfert de données requête par requête par le maintien d'un cache des données *à proximité*.

Annexes

Annexe A

Analyse d'une boîte à moustaches

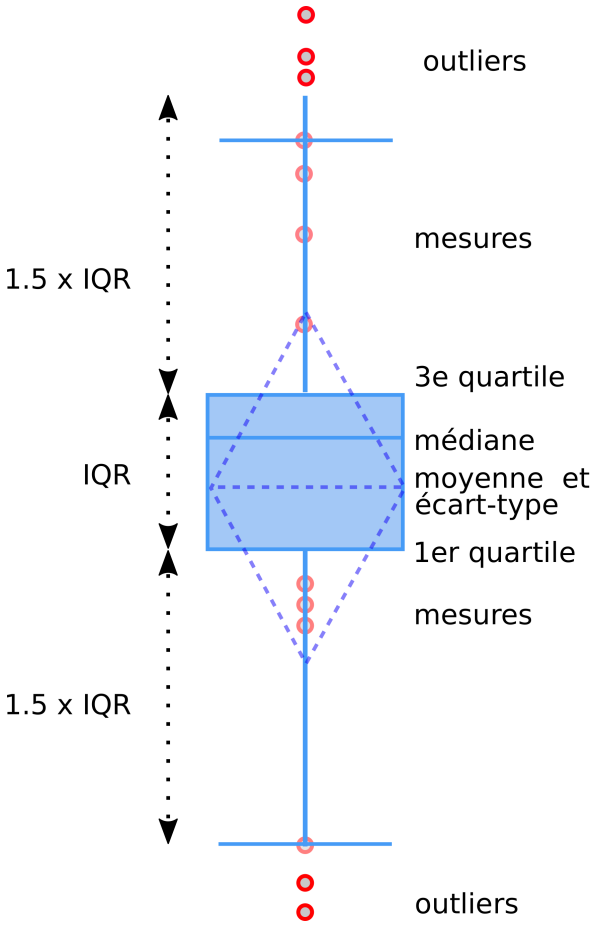


FIGURE A.1: Schéma d'une boîte à moustaches : la hauteur de la boite bleue représente l'espace inter-quartile (IQR) (intervalle entre les 1er et 3e quartiles) et la ligne pleine représente la valeur médiane. La ligne horizontale du losange (en pointillé) indique la valeur moyenne et ses deux extrémités haute et basse fournissent l'écart-type. La position des *moustaches* est déterminée par la mesure la plus haute ou la plus basse comprise dans l'intervalle 1,5 x IQR. Les mesures en dehors de ces seuils sont identifiées comme valeurs extrêmes (appelées *outliers*) et sont représentées sur le diagramme (points rouges plein) tandis que les autres mesures, contenues dans l'intervalle (points rouges transparents) ne sont pas représentées.

Annexe B

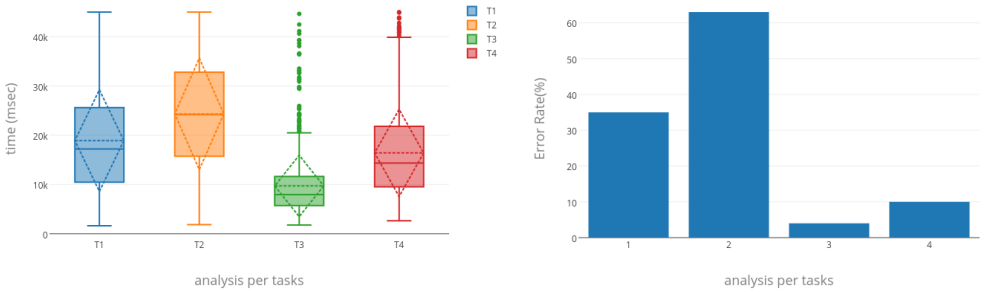
Rapport statistiques : évaluation tâches basiques

1 Introduction

Données statistiques de la seconde évaluation de l'encodage visuel des arêtes dans les matrices d'adjacences. les tâches évaluées sont :

1. Estimation de la densité du graphe
2. Trouver le noeuds de plus fort degré
3. Trouver une arête reliant les deux noeuds
4. Trouver un voisin commun aux deux noeuds

2 Analyse par tâche



(a) Temps de réponse par encodage **(b) Taux d'erreur**

Figure B.1: Analyse par tâche

2.1 Test statistique pour analyse du temps de réponse

Test de kruskal-wallis pour analyse global

$$\boxed{< 10^{-4}}$$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

Comparaison deux à deux des 4 tâches

	T2	T3	T4
T1	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
T2		$< 10^{-4}$	$< 10^{-4}$
T3			$< 10^{-4}$

2.2 Test statistique pour analyse des taux d'erreurs

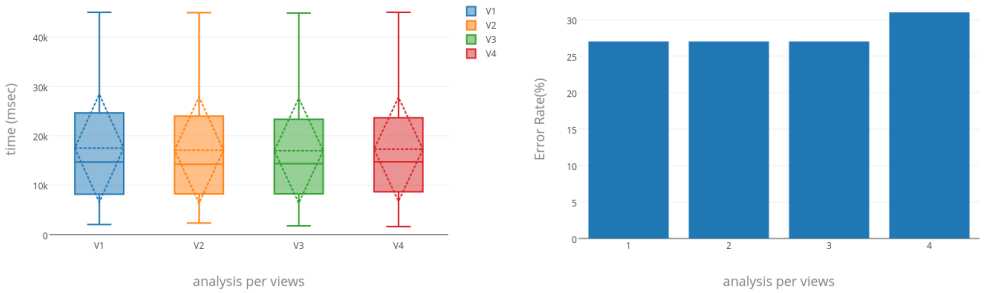
Test de Friedman pour analyse globale

$$\boxed{< 10^{-4}}$$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

	T2	T3	T4
T1	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
T2		$< 10^{-4}$	$< 10^{-4}$
T3			$< 10^{-4}$

3 Analyse par encodage



(a) Temps de réponse par encodage **(b) Taux d'erreur par encodage**

FIGURE B.2: Analyse par encodage

3.1 Test statistique pour analyse du temps de réponse

Test de kruskal-wallis pour analyse global

0.7728

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

Comparaison deux à deux des 4 tâches

	V2	V3	V4
V1	0.2456	0.2075	0.4885
V2		0.4688	0.2564
V3			0.2074

3.2 Test statistique pour analyse des taux d'erreurs

Test de Friedman pour analyse globale

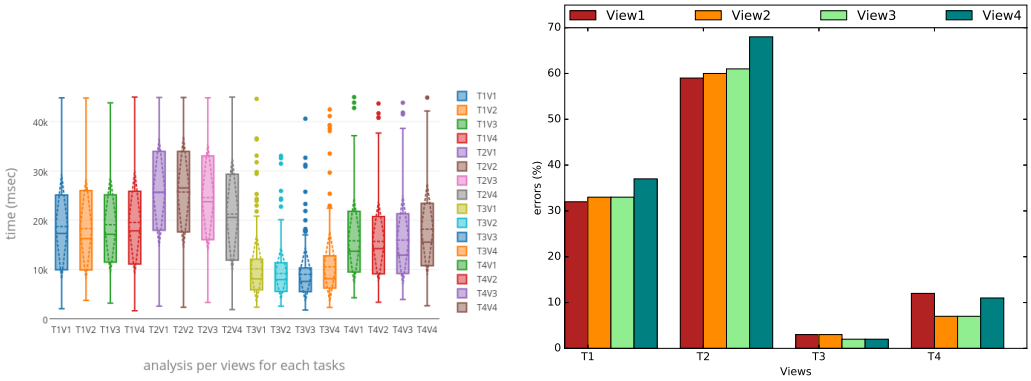
0.113

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

	V2	V3	V4
V1	0.2456	0.2075	0.4885
V2		0.4688	0.2564
V3			0.2074

4 Analyse par encodage pour chaque tâche

analyse des temps de réponse et des taux d'erreurs pour comparer les encodages entre elles, pour chacune des tâches. Autrement dit, une encodage est elle plus efficace qu'une autre pour une tâche particulière.



(a) Temps de réponse par encodage et par Tâche

(b) Taux d'erreur par encodage et par Tâche

FIGURE B.3: Analyse par encodage pour chaque tâche

4.1 Test statistique pour analyse du temps de réponse

Test de kruskal-wallis pour analyse global

T1	T2	T3	T4
0.5701	**0.0002**	0.1084	**0.0088**

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

tâche	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
t1	0.389	0.2648	0.1764	0.1618	0.1007	0.3897
t2	0.4995	0.1155	**0.0001**	0.1239	**0.0001**	**0.0034**
t3	0.2082	0.0763	0.1985	0.2819	**0.04**	**0.0108**
t4	0.4893	0.4534	**0.0031**	0.4292	**0.0036**	**0.0017**

4.2 Test statistique pour analyse des taux d'erreurs

Test de Friedman pour analyse globale

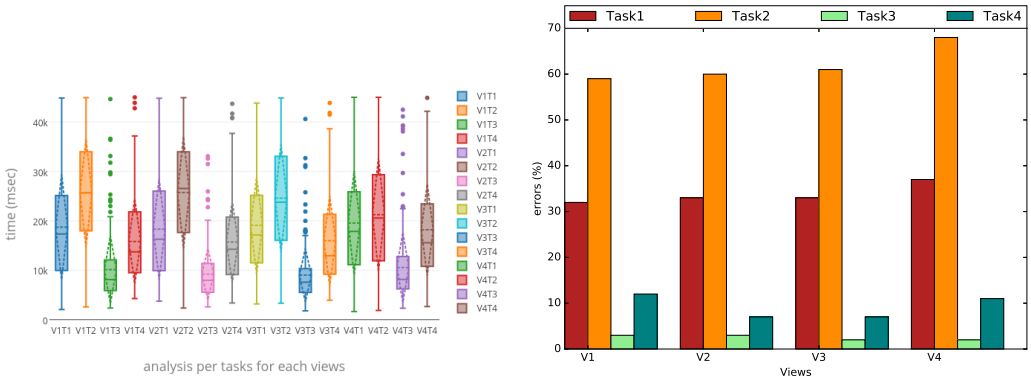
T1	T2	T3	T4
0.6466	0.0857	0.8415	0.247

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

tâche	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
t1	0.4595	0.4595	0.1572	0.4998	0.183	0.183
t2	0.4225	0.3118	**0.0177**	0.384	**0.0282**	0.0532
t3	0.3969	0.1997	0.2941	0.2798	0.3897	0.3807
t4	0.0744	0.0744	0.4411	0.4997	0.0973	0.0973

5 Analyse par tâche pour chaque encodage

Analyse des temps de réponse et des taux d'erreurs pour comparer les tâches entre elles, pour chacune des encodages. Autrement dit, une tâche est elle plus simple qu'une autre pour une encodage particulière.



(a) Temps de réponse par tâche et par encodage

(b) Taux d'erreur par tâche et par encodage

FIGURE B.4: Analyse par tâche pour chaque encodage

5.1 Test statistique pour analyse du temps de réponse

Test de kruskal-wallis pour analyse global

v1	v2	v3	v4
$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

Encodage	T1-T2	T1-T3	T1-T4	T2-T3	T2-T4	T3-T4
v1	$< 10^{-4}$	$< 10^{-4}$	**0.0054**	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
v2	$< 10^{-4}$	$< 10^{-4}$	**0.0108**	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
v3	$< 10^{-4}$	$< 10^{-4}$	**0.0002**	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
v4	0.0532	$< 10^{-4}$	0.0986	$< 10^{-4}$	**0.0025**	$< 10^{-4}$

5.2 Test statistique pour analyse des taux d'erreurs

Test de Friedman pour analyse globale

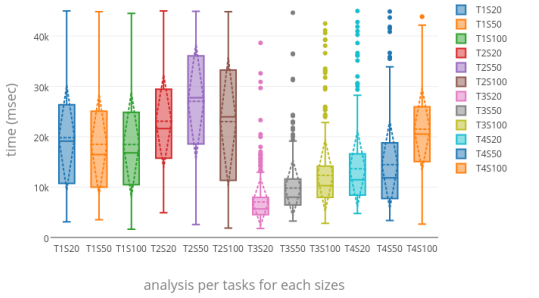
V1	V2	V3	V4
$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

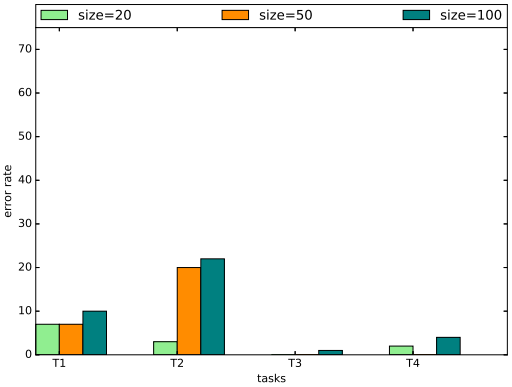
Encodage	T1-T2	T1-T3	T1-T4	T2-T3	T2-T4	T3-T4
v1	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	**0.0007**
v2	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	**0.018**
v3	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	**0.0044**
v4	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	**0.0002**

6 Analyse par taille pour chaque tâche

Analyse des temps de réponse et des taux d'erreurs pour comparer les tâches entre elles, pour chacune des tailles.



(a) Temps de réponse par taille pour chaque tâche



(b) Taux d'erreur par taille pour chaque tâche

FIGURE B.5: Analyse par taille pour chaque tâche

6.1 Test statistique pour analyse du temps de réponse par taille

Test de kruskal-wallis pour analyse global

S20	S50	S100
$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

taille	T1-T2	T1-T3	T1-T4	T2-T3	T2-T4	T3-T4
S20	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
S50	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
S100	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	0.0525	$< 10^{-4}$

6.2 Test statistique pour analyse du temps de réponse par tâche

Test de kruskal-wallis pour analyse global

t1	t2	t3	t4
0.192	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

tâche	S1-S2	S1-S3	S2-S3
T1	0.0651	0.052	0.4435
T2	$< 10^{-4}$	0.4353	**0.0003**
T3	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
T4	0.3589	$< 10^{-4}$	$< 10^{-4}$

6.3 Test statistique pour analyse des taux d'erreurs par taille

Test de Friedman pour analyse globale

S20	S50	S100
$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

taille	T1-T2	T1-T3	T1-T4	T2-T3	T2-T4	T3-T4
S20	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	**0.0068**	$< 10^{-4}$
S50	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	0.3164
S100	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

6.4 Test statistique pour analyse des taux d'erreurs par tâches

Test de Friedman pour analyse globale

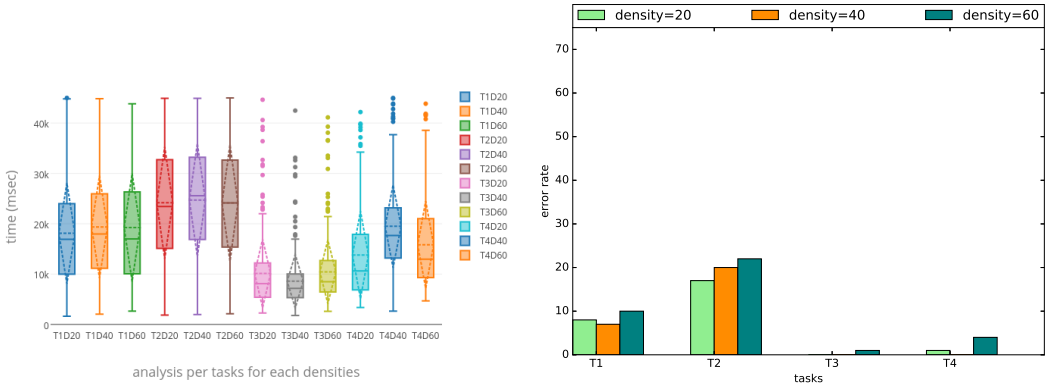
T1	T2	T3	T4
0.0004	$< 10^{-4}$	**0.0107**	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

tâche	S1-S2	S1-S3	S2-S3
T1	0.3581	**0.0003**	**0.0009**
T2	$< 10^{-4}$	$< 10^{-4}$	**0.0012**
T3	0.0643	**0.0021**	0.0684
T4	**0.0067**	**0.0004**	$< 10^{-4}$

7 analyse par densité pour chaque tâche

Analyse des temps de réponse et des taux d'erreurs pour comparer les densités entre elles, pour chacune des tâches.



(a) Temps de réponse par densité pour chaque tâche

(b) Taux d'erreur par densité pour chaque tâche

FIGURE B.6: analyse par densité pour chaque tâche

7.1 Test statistique pour analyse du temps de réponse par densité

Test de kruskal-wallis pour analyse global

d0.2	d0.4	d0.6
$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

densité	T1-T2	T1-T3	T1-T4	T2-T3	T2-T4	T3-T4
D0.2	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
D0.4	$< 10^{-4}$	$< 10^{-4}$	0.2344	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
D0.6	$< 10^{-4}$	$< 10^{-4}$	**0.0002**	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

7.2 Test statistique pour analyse des taux d'erreurs par densité

Test de Friedman pour analyse globale

D0.2	D0.4	D0.6
$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

densité	T1-T2	T1-T3	T1-T4	T2-T3	T2-T4	T3-T4
D0.2	**0.003**	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	**0.0018**
D0.4	$< 10^{-4}$	$< 10^{-4}$	**0.0002**	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
D0.6	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	**0.0146**

7.3 Test statistique pour analyse du temps de réponse par tâches

Test de kruskal-wallis pour analyse global

T1	T2	T3	T4
0.3073	0.6584	$< 10^{-4}$	$< 10^{-4}$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

tâche	D1-D2	D1-D3	D2-D3
T1	0.064	0.1537	0.3368
T2	0.2051	0.4723	0.2255
T3	**0.0034**	0.0648	$< 10^{-4}$
T4	$< 10^{-4}$	**0.0001**	$< 10^{-4}$

7.4 Test statistique pour analyse des taux d'erreurs par tâches

Test de Friedman pour analyse globale

T1	T2	T3	T4
$< 10^{-4}$	**0.0001**	0.8521	**0.026**

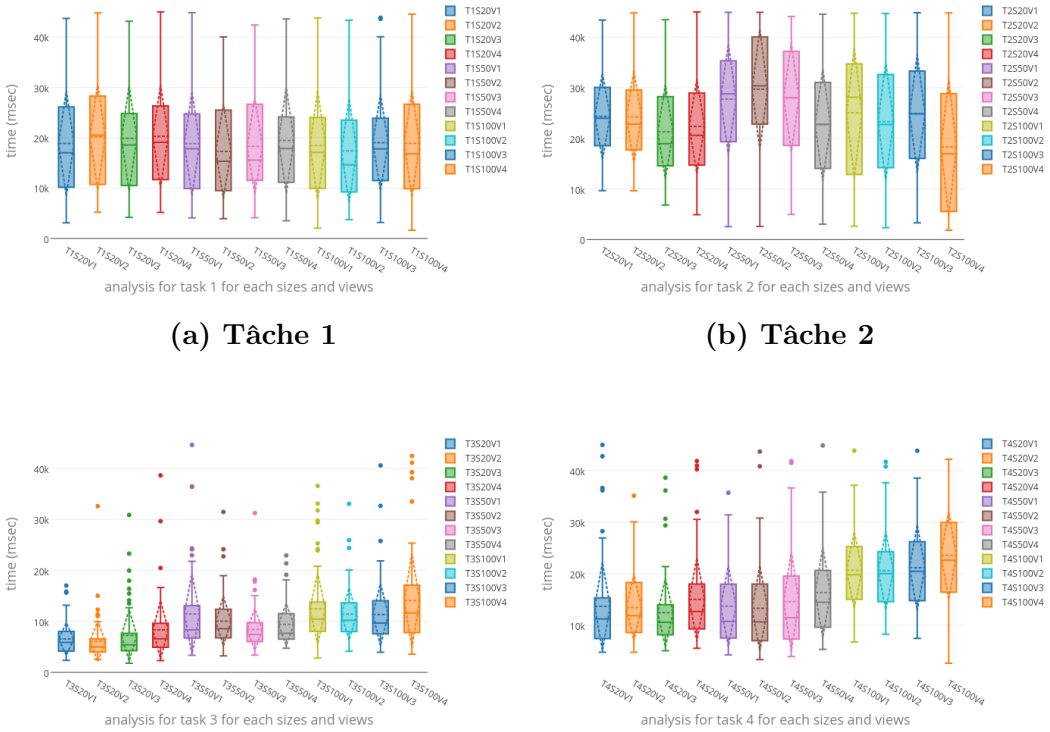
Test de Mann-Whithney-Wilcoxon pour analyse approfondie

tâche	D1-D2	D1-D3	D2-D3
T1	$< 10^{-4}$	**0.0066**	**0.0143**
T2	0.1031	$< 10^{-4}$	**0.0018**
T3	0.4996	0.3164	0.3164
T4	**0.0227**	0.3796	**0.0107**

8 Analyse par taille et encodage pour chaque tâche

Comparaison des temps de réponse par taille et par encodage avec séparation pour chaque tâche. objectif : essayer de voir si pour une tâche donnée, un variation apparait dans les performance en associant encodage et taille

8.1 Analyse du temps de réponse par taille pour chaque encodage et tâche



(a) Tâche 1

(b) Tâche 2

(c) Tâche 3

(d) Tâche 4

FIGURE B.7: Analyse du temps de réponse par taille pour chaque encodage et tâche

8.2 Test statistique pour analyse du temps de réponse

Test de kruskal-wallis pour analyse global

Tache	S20	S50	S100
T1	0.7977	0.7193	0.6336
T2	0.0865	**0.0032**	**0.0104**
T3	**0.0058**	**0.0495**	0.4614
T4	0.1321	0.0683	0.1264

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

test pour la tâche 1

Encodage	S1-S2	S1-S3	S2-S3
V1	0.475	0.3499	0.3754
V2	0.0638	0.0515	0.4432
V3	0.182	0.3716	0.3312
V4	0.291	0.232	0.3973

taille	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
S1	0.2234	0.2922	0.1678	0.3775	0.4395	0.3775
S2	0.2431	0.4432	0.3635	0.2501	0.1309	0.2826
S3	0.2922	0.2633	0.3693	0.0999	0.1698	0.4107

test pour la tâche 2

Encodage	S1-S2	S1-S3	S2-S3
V1	**0.0189**	0.2178	0.161
V2	**0.0006**	0.4038	**0.0012**
V3	**0.0001**	**0.025**	0.0806
V4	0.3789	**0.0124**	**0.0187**

taille	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
S1	0.4414	**0.0115**	0.0749	**0.0264**	0.1096	0.2351
S2	0.1405	0.4849	**0.007**	0.1382	**0.0003**	**0.0048**
S3	0.1649	0.4142	**0.0022**	0.2392	**0.0126**	**0.0019**

test pour la tâche 3

Encodage	S1-S2	S1-S3	S2-S3
V1	$< 10^{-4}$	$< 10^{-4}$	**0.0306**
V2	$< 10^{-4}$	$< 10^{-4}$	**0.019**
V3	**0.0002**	$< 10^{-4}$	$< 10^{-4}$
V4	**0.002**	$< 10^{-4}$	**0.0003**

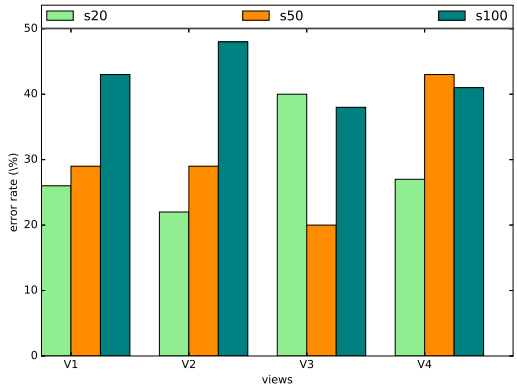
taille	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
S1	0.0827	0.4686	**0.0193**	0.0682	**0.0001**	**0.0327**
S2	0.3997	**0.0083**	0.1693	**0.0079**	0.2387	0.0557
S3	0.3423	0.1791	0.2223	0.3024	0.1356	0.0745

test pour la tâche 4

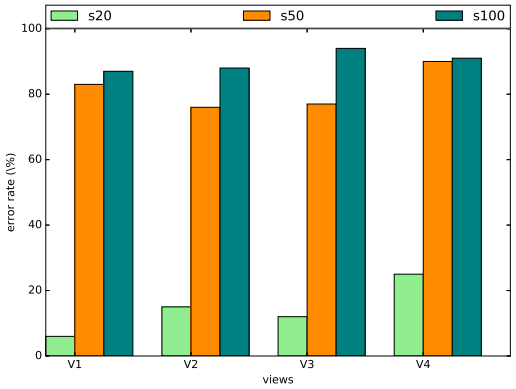
Encodage	S1-S2	S1-S3	S2-S3
V1	0.4382	$< 10^{-4}$	$< 10^{-4}$
V2	0.1859	$< 10^{-4}$	$< 10^{-4}$
V3	0.3119	$< 10^{-4}$	$< 10^{-4}$
V4	0.184	$< 10^{-4}$	$< 10^{-4}$

taille	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
S1	0.2651	0.3544	**0.0468**	0.1308	0.1419	**0.0099**
S2	0.2472	0.4462	**0.0211**	0.2744	**0.0086**	**0.0279**
S3	0.417	0.4043	**0.0248**	0.3782	**0.0183**	**0.0423**

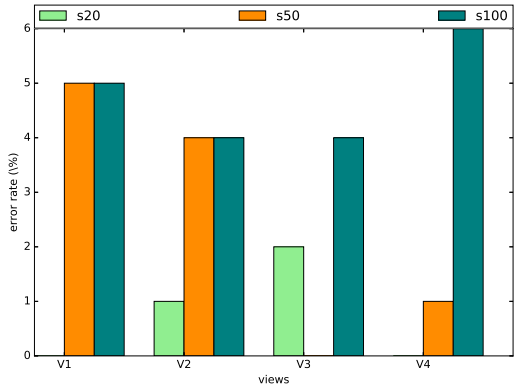
8.3 Analyse du taux d'erreur par taille pour chaque encodage et tâche



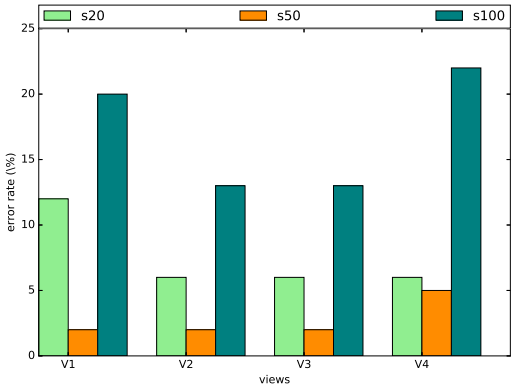
(a) taux d'erreur (%) par taille pour la tâche 1 pour chaque encodage



(b) taux d'erreur (%) par taille pour la tâche 2 pour chaque encodage



(c) taux d'erreur (%) par taille pour la tâche 3 pour chaque encodage



(d) taux d'erreur (%) par taille pour la tâche 4 pour chaque encodage

FIGURE B.8: Analyse du taux d'erreur par taille pour chaque encodage et tâche

8.4 Test statistique pour analyse des taux d'erreurs

Test de Friedman pour analyse globale

Tache	S20	S50	S100
T1	**0.0244**	**0.0052**	0.646
T2	**0.0131**	0.1163	0.4502
T3	0.2998	0.1414	0.8483
T4	0.4784	0.7325	0.3014

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

test pour tâche1

Encodage	S20-S50	S20-S100	S50-S100
V1	0.3563	**0.0183**	**0.0421**
V2	0.1717	**0.0005**	**0.0086**
V3	**0.0058**	0.4335	**0.0092**
V4	**0.0282**	**0.0408**	0.4342

taille	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
S20	0.2816	**0.0393**	0.4269	**0.01**	0.2223	0.0576
S50	0.499	0.1255	**0.0421**	0.1255	**0.0421**	**0.0022**
S100	0.2532	0.307	0.4342	0.1211	0.2027	0.3683

test pour tâche2

Encodage	S20-S50	S20-S100	S50-S100
V1	$< 10^{-4}$	$< 10^{-4}$	0.2412
V2	$< 10^{-4}$	$< 10^{-4}$	**0.0244**
V3	$< 10^{-4}$	$< 10^{-4}$	**0.002**
V4	$< 10^{-4}$	$< 10^{-4}$	0.3875

taille	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
S20	0.0568	0.1319	**0.0016**	0.3167	0.074	**0.0279**
S50	0.1509	0.2015	0.1106	0.4228	**0.013**	**0.0208**
S100	0.3998	0.0743	0.2085	0.1155	0.2889	0.2582

test pour tâche3

Encodage	S20-S50	S20-S100	S50-S100
V1	**0.022**	**0.022**	0.498
V2	0.1578	0.1578	0.4977
V3	0.0794	0.3273	**0.0413**
V4	0.162	**0.0119**	**0.0488**

taille	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
S20	0.162	0.0794	impossible	0.2832	0.162	0.0794
S50	0.3517	**0.022**	0.0878	**0.0413**	0.1578	0.162
S100	0.3517	0.3517	0.3676	0.4977	0.2357	0.2357

test pour tâche4

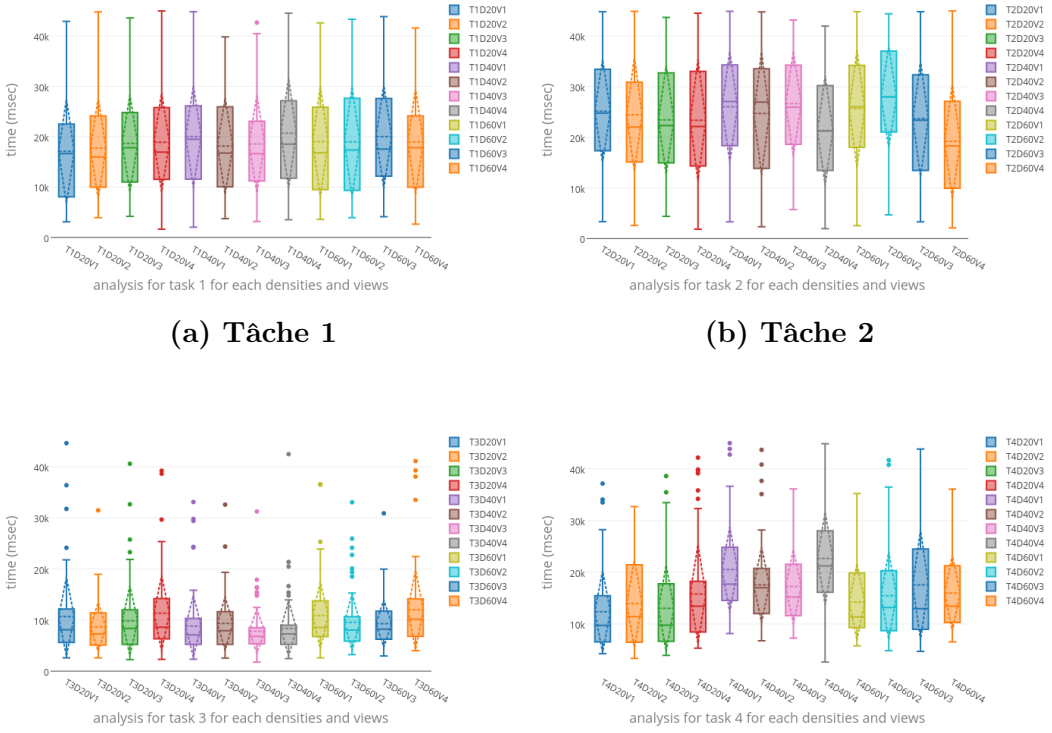
Encodage	S20-S50	S20-S100	S50-S100
V1	**0.0145**	0.0911	**0.0004**
V2	0.1244	0.0876	**0.0082**
V3	0.1244	0.0876	**0.0082**
V4	0.3676	**0.0049**	**0.002**

taille	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
S20	0.1319	0.1319	0.1319	0.4982	0.4982	0.4982
S50	0.4972	0.4972	0.2046	0.4972	0.2046	0.2046
S100	0.1372	0.1372	0.421	0.4987	0.0981	0.0981

9 Analyse par densité et encodage pour chaque tâche

Comparaison des temps de réponse par densités et par encodage avec separation pour chaque tâche. objectif : essayer de voir si pour une tâche donnée, une variation apparait dans les performance en associant encodage et densité, l'idée est donc de comparer les densités pour une encodage et une tâche ou les encodages pour une densité et une tâche.

9.1 Analyse des temps de réponse par densité pour chaque encodage et tâche



(a) Tâche 1 **(b) Tâche 2**

(c) Tâche 3 **(d) Tâche 4**

FIGURE B.9: Analyse des temps de réponse par densité pour chaque encodage et tâche

9.2 Test statistique pour analyse du temps de réponse

Test de kruskal-wallis pour analyse global

	D20	D40	D60
T1	0.4699	0.5029	0.8168
T2	0.7292	**0.0489**	**0.0001**
T3	0.1435	0.2887	0.076
T4	0.0647	$< 10^{-4}$	0.2791

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

test pour la tâche 1

Encodage	D1-D2	D1-D3	D2-D3
V1	**0.0248**	0.1598	0.2002
V2	0.3576	0.2922	0.3997
V3	0.4045	0.2696	0.2113
V4	0.2055	0.4191	0.176

densité	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
D1	0.2908	0.1072	0.0921	0.2221	0.182	0.4456
D2	0.1512	0.1754	0.4216	0.4231	0.0955	0.1623
D3	0.4992	0.2143	0.4574	0.2137	0.4432	0.2234

test pour la tâche 2

Encodage	D1-D2	D1-D3	D2-D3
V1	0.286	0.367	0.4206
V2	0.342	**0.0274**	0.0954
V3	**0.0388**	0.4679	0.0616
V4	0.1906	**0.0277**	0.0904

densité	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
D1	0.3162	0.1622	0.1873	0.2959	0.2933	0.4734
D2	0.3091	0.436	**0.0091**	0.2369	**0.043**	**0.0055**
D3	0.1583	0.121	**0.0007**	**0.0121**	$< 10^{-4}$	**0.0143**

test pour la tâche 3

Encodage	D1-D2	D1-D3	D2-D3
V1	0.0593	0.1566	**0.0026**
V2	0.2088	0.1541	0.4459
V3	0.052	0.3904	**0.0069**
V4	**0.0022**	0.2397	**0.0001**

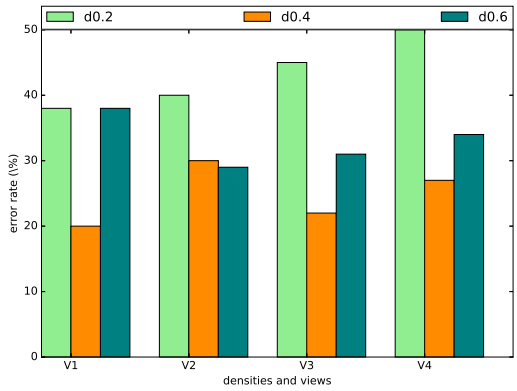
densité	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
D1	0.1024	0.3208	0.1873	0.242	**0.01**	0.0691
D2	0.0996	0.2494	0.4526	**0.0377**	0.0813	0.29
D3	0.0762	0.0796	0.2722	0.4566	**0.0146**	**0.0166**

test pour la tâche 4

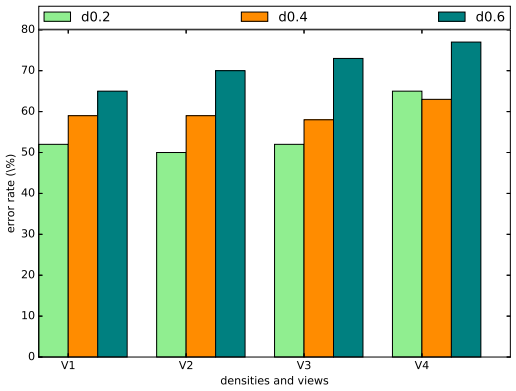
Encodage	D1-D2	D1-D3	D2-D3
V1	$< 10^{-4}$	**0.0122**	$< 10^{-4}$
V2	**0.0014**	0.0819	**0.0127**
V3	$< 10^{-4}$	**0.0018**	0.1106
V4	$< 10^{-4}$	0.1637	$< 10^{-4}$

densité	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
D1	0.2033	0.3129	**0.0044**	0.3307	0.0724	**0.0173**
D2	**0.0095**	**0.0028**	0.0605	0.3628	**0.0001**	$< 10^{-4}$
D3	0.2659	0.0748	**0.0305**	0.2125	0.1556	0.4108

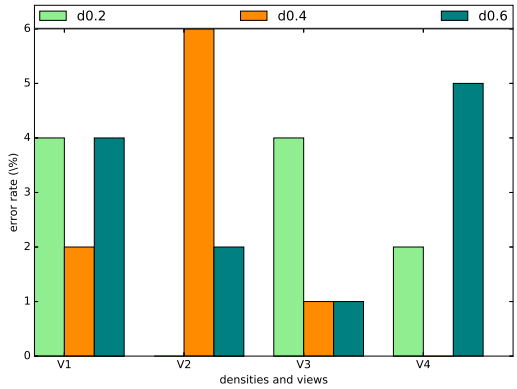
9.3 Analyse des taux d'erreur par densité pour chaque encodage et tâche



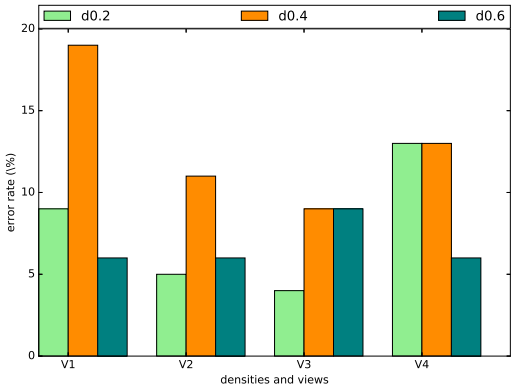
(a) taux d'erreur (%) par densité pour la tâche 1 pour chaque encodage



(b) taux d'erreur (%) par densité pour la tâche 2 pour chaque encodage



(c) taux d'erreur (%) par densité pour la tâche 3 pour chaque encodage



(d) taux d'erreur (%) par densité pour la tâche 4 pour chaque encodage

FIGURE B.10: Analyse des taux d'erreur par densité pour chaque encodage et tâche

9.4 Test statistique pour analyse des taux d'erreurs

Test de Friedman pour analyse globale

	D20	D40	D60
T1	0.3443	0.4747	0.5523
T2	**0.0224**	0.9008	0.3638
T3	0.3515	0.0542	0.4753
T4	0.1116	0.2946	0.9008

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

test pour tâche1

Encodage	D20-D40	D20-D60	D40-D60
V1	**0.0092**	0.4991	**0.0092**
V2	0.1126	0.0818	0.429
V3	**0.0015**	**0.0445**	0.0959
V4	**0.0032**	**0.0324**	0.1858

Densité	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
D20	0.4335	0.201	0.091	0.2519	0.1218	0.3098
D40	0.0921	0.421	0.1672	0.1297	0.3584	0.2223
D60	0.1104	0.1933	0.3036	0.3602	0.2389	0.3632

test pour tâche2

Encodage	D20-D40	D20-D60	D40-D60
V1	0.202	0.0646	0.2471
V2	0.1218	**0.0055**	0.0818
V3	0.2527	**0.0049**	**0.0271**
V4	0.432	**0.0491**	**0.034**

Densité	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
D20	0.3707	0.4991	0.0646	0.3707	**0.0324**	0.0646
D40	0.4991	0.4339	0.3049	0.4339	0.3049	0.2486
D60	0.2389	0.1403	**0.0491**	0.3563	0.1717	0.2816

test pour tâche3

Encodage	D20-D40	D20-D60	D40-D60
V1	0.3273	0.4977	0.3273
V2	**0.0119**	0.0794	0.1244
V3	0.1578	0.1578	0.4961
V4	0.0794	0.2046	**0.022**

Densité	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
D20	**0.0413**	0.4977	0.3273	**0.0413**	0.0794	0.3273
D40	0.1244	0.2832	0.0794	**0.0488**	**0.0119**	0.162
D60	0.3273	0.1578	0.3517	0.2832	0.2046	0.0878

test pour tâche4

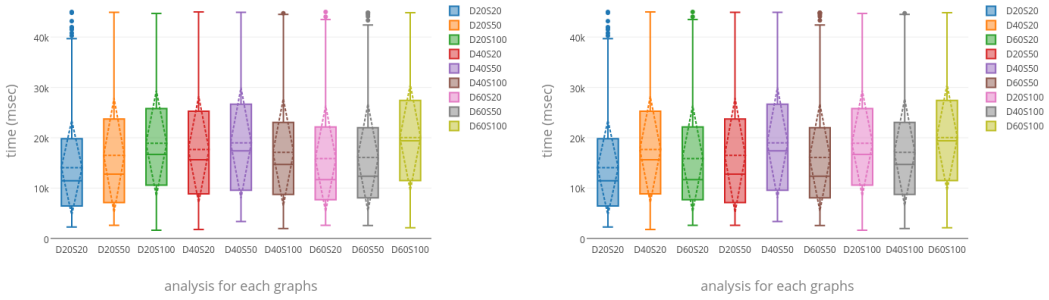
Encodage	D20-D40	D20-D60	D40-D60
V1	0.0501	0.2753	**0.0137**
V2	0.1155	0.3676	0.1934
V3	0.0964	0.0964	0.4984
V4	0.4987	0.0876	0.0876

Densité	V1-V2	V1-V3	V1-V4	V2-V3	V2-V4	V3-V4
D20	0.1753	0.0964	0.2211	0.3517	**0.0467**	**0.0215**
D40	0.0835	0.0501	0.1872	0.3943	0.309	0.2211
D60	0.4982	0.2753	0.4982	0.2753	0.4982	0.2753

10 Analyse par graphes

les plots sont dupliqués afin de faciliter la comparaison par graphes de tailles ou de densités similaires mais représentent les même données.

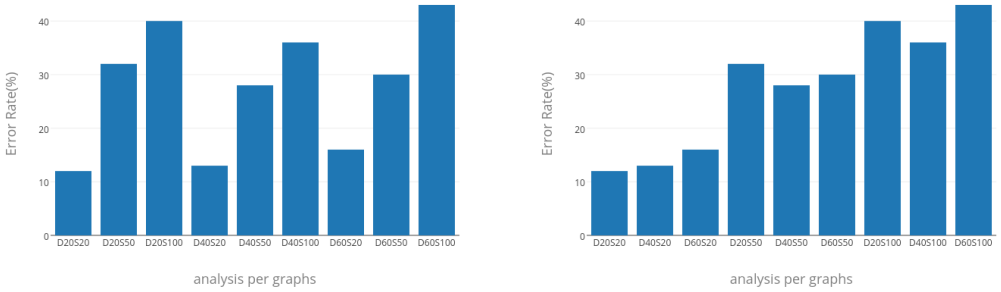
10.1 Analyse des temps de réponse par graphe



(a) Temps de réponse par graphes **(b) Temps de réponse par graphes**

FIGURE B.11: Analyse des temps de réponse par graphe

10.2 Analyse des taux d'erreur par graphe



(a) Taux d'erreur par graphes **(b) Taux d'erreur par graphes**

FIGURE B.12: Analyse des taux d'erreur par graphe

10.3 Test statistique pour analyse du temps de réponse

Test de kruskal-wallis pour analyse global

$$\boxed{< 10^{-4}}$$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

Comparaison deux à deux des 9 graphes,G1 n'apparait pas dans les colonnes parce qu'il est mis dans les colonnes et G9 n'apparait pas dans les lignes parce qu'il est comparé dans les colonnes, cela évite d'avoir une ligne et une colonne vide.
Les graphes sont numérotés par tailles et densités : G1=S20D20, G2=S20D40, G3=S20D60, G4=S40D20, etc

	G2	G3	G4	G5	G6	G7	G8	G9
G1	$< 10^{-4}$	**0.0079**	**0.0035**	$< 10^{-4}$	**0.0021**	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
G2		**0.0039**	**0.0192**	0.0589	**0.0099**	**0.0445**	0.1983	**0.0007**
G3			0.4012	$< 10^{-4}$	0.275	$< 10^{-4}$	**0.0195**	$< 10^{-4}$
G4				**0.0002**	0.3727	$< 10^{-4}$	0.0564	$< 10^{-4}$
G5					**0.0001**	0.4423	**0.0092**	**0.0417**
G6						$< 10^{-4}$	0.0511	$< 10^{-4}$
G7							**0.0038**	0.0564
G8								$< 10^{-4}$

10.4 Test statistique pour analyse des taux d'erreurs

Test de Friedman pour analyse globale

$$\boxed{< 10^{-4}}$$

Test de Mann-Whithney-Wilcoxon pour analyse approfondie

	G2	G3	G4	G5	G6	G7	G8	G9
G1	0.3713	0.0862	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
G2		0.1499	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
G3			$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$
G4				0.1522	0.3479	**0.0097**	0.0847	**0.0008**
G5					0.2623	**0.0004**	**0.0082**	$< 10^{-4}$
G6						**0.0032**	**0.0388**	**0.0002**
G7							0.1667	0.2098
G8								**0.0381**

Annexe C

Rapport statistiques : évaluation tâche complexe

1 Introduction

Données statistiques de la première évaluation de l'encodage visuel des arêtes dans les matrices d'adjacences. La convention de nommage des encodages visuels est identique à celle du manuscrit sauf pour l'encodage 1, nommé ici encodage 0. La différence d'appélation est dû au retrait d'un 5ème encodage visuel non sélectionné.

2 Résultats par évaluateurs

Tableau C.1: Résultats par évaluateur.

évaluateur	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.
0	0.031	67.626 (35.208)	18.9	135.7915
1	0.094	23.990 (9.880)	10.3375	50.148
4	0.000	38.473 (17.240)	11.973	65.0655
5	0.156	57.763 (33.767)	22.4245	150.105
6	0.250	71.784 (39.246)	18.557	149.4425
7	0.156	63.886 (25.989)	32.269	120.374
9	0.156	74.171 (31.282)	35.245	133.069
10	0.000	42.669 (20.522)	19.328	93.1695
11	0.188	53.201 (27.293)	9.68	109.931
15	0.125	47.567 (28.981)	8.288	108.228
16	0.312	81.774 (43.634)	14.679	160.888
17	0.094	45.193 (28.919)	17.741	123.642
18	0.062	57.337 (31.532)	23.872	128.099
19	0.031	59.024 (37.733)	14.2745	142.314
21	0.062	49.113 (26.067)	19.672	101.466
22	0.031	50.369 (20.116)	15.169	89.557
23	0.031	29.730 (15.413)	12.9835	72.296
24	0.094	47.307 (25.032)	18.52	91.918
25	0.125	38.705 (22.021)	9.008	76.1295
26	0.000	57.821 (26.972)	24.8715	103.796

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

3 Analyse par encodage

seuil de significativité $\alpha = 0.05$.

Tableau C.2: Résultats par encodage

Enc.	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
0	0.156	57.694 (35.999)	12.145	150.903	4.800 (0.926)	7.133 (3.686)
4	0.081	54.135 (32.750)	8.288	160.888	5.154 (0.954)	6.860 (2.684)
3	0.075	47.629 (26.037)	9.680	142.314	5.184 (0.991)	7.158 (2.983)
2	0.087	50.822 (27.967)	8.584	135.792	5.000 (1.003)	7.241 (3.103)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by View
Kruskal-Wallis chi-squared = 4.1277, df = 3, p-value = 0.248
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by View
Kruskal-Wallis chi-squared = 2.0669, df = 3, p-value = 0.5586
```

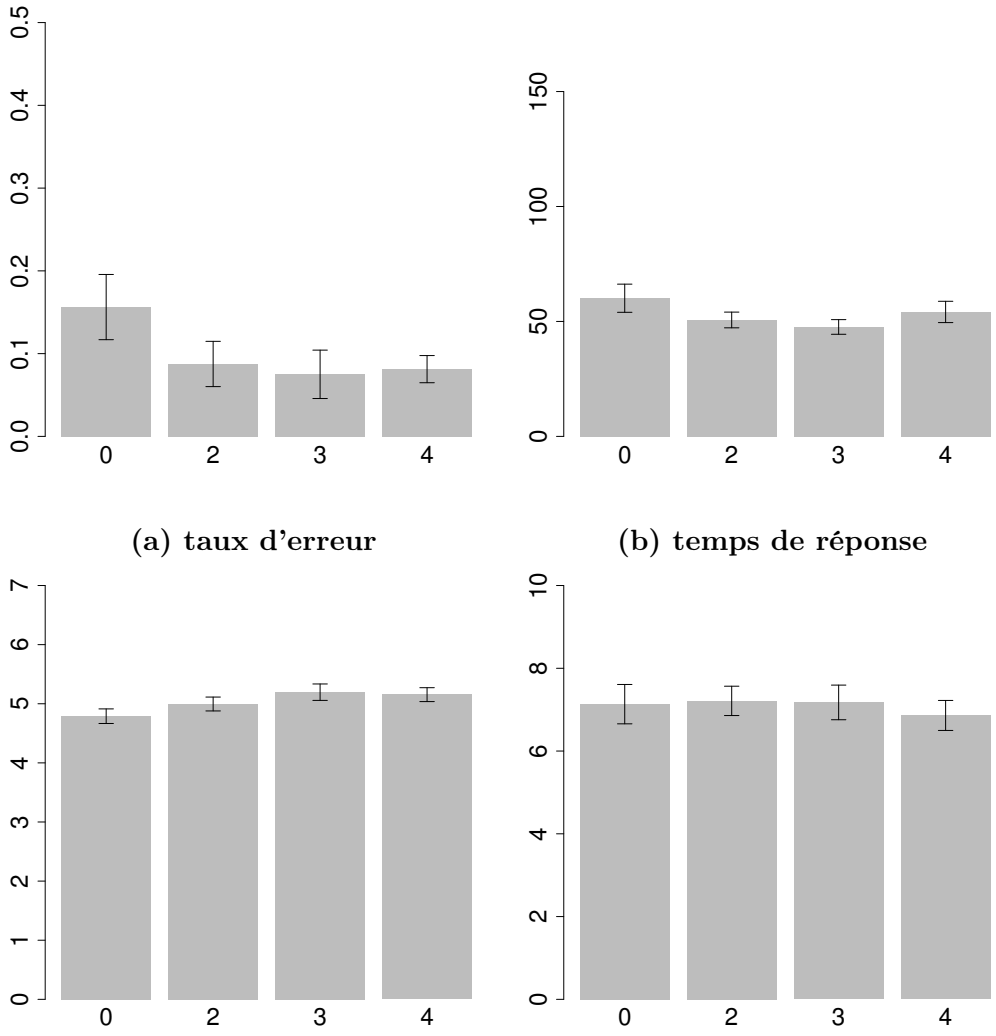
longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by View
```

```
Kruskal-Wallis chi-squared = 5.6327, df = 3, p-value = 0.1309
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by View
Kruskal-Wallis chi-squared = 0.7375, df = 3, p-value = 0.8644
```



(a) taux d'erreur **(b) temps de réponse**

(c) longueur de chemin **(d) nombre de sélection**

FIGURE C.1: Valeur moyenne des différentes mesures

4 Analyse par graphes

Seuil de significativité $\alpha = 0.05$.

Tableau C.3: Résultats par graphe

Graph	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n50k5	0.081	60.523 (25.090)	14.679	123.642	5.354 (1.060)	8.703 (3.429)
n100k10	0.037	49.029 (31.837)	12.827	160.888	5.050 (0.852)	6.244 (2.203)
n100k5	0.181	56.884 (33.291)	13.509	150.105	4.734 (1.015)	6.253 (2.549)
n50k10	0.100	43.529 (30.720)	8.288	133.069	5.004 (0.883)	7.180 (3.470)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph
Kruskal-Wallis chi-squared = 11.868, df = 3, p-value = 0.007849
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by Graph
Kruskal-Wallis chi-squared = 8.7046, df = 3, p-value = 0.03349
```

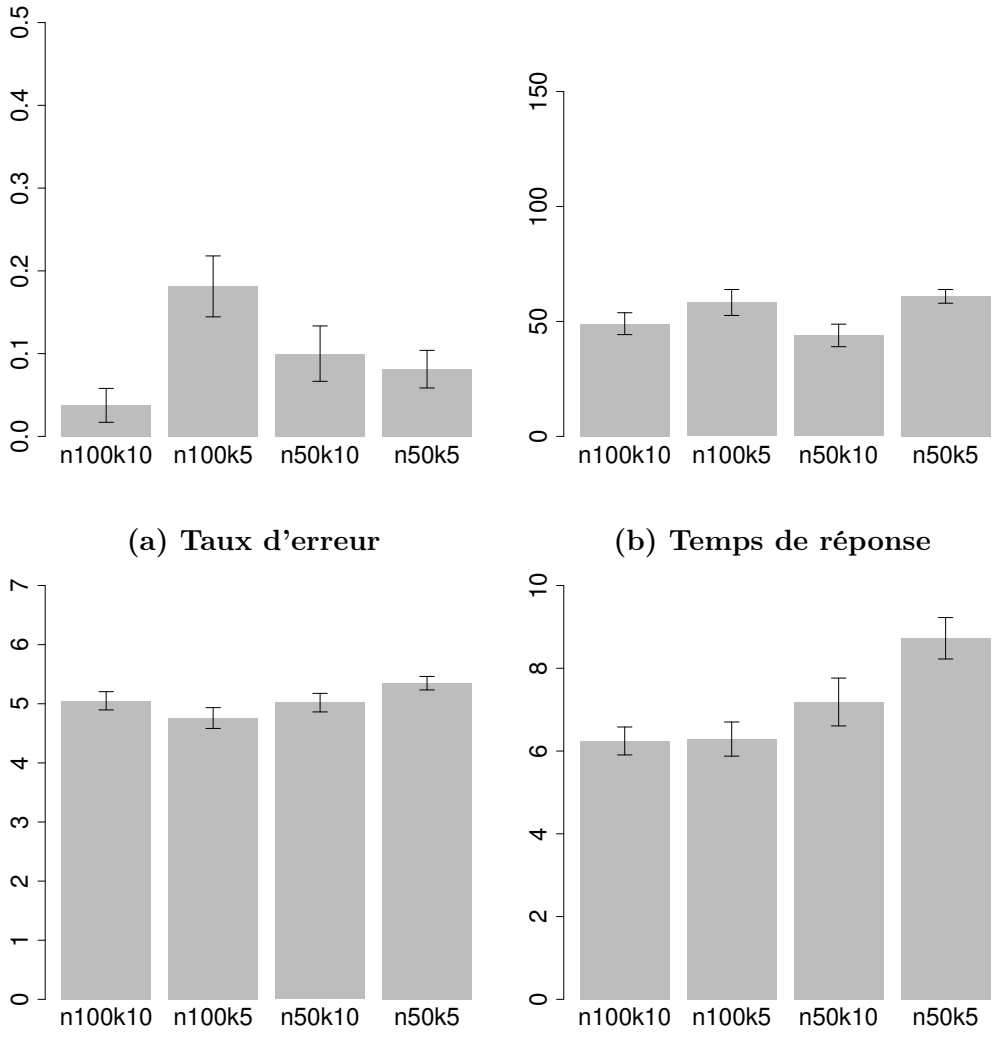
longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Graph
```

```
Kruskal-Wallis chi-squared = 10.1601, df = 3, p-value = 0.01725
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph
Kruskal-Wallis chi-squared = 15.0209, df = 3, p-value = 0.001799
```



(a) Taux d'erreur **(b) Temps de réponse**

(c) longueur de chemin **(d) nombre de sélection**

Figure C.2: Résultats par graphe

4.1 Résultats par taille de graphe

Seuil de significativité $\alpha = 0.025$.

Tableau C.4: Résultats par taille de graphe

Taille	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n100	0.181	56.884 (33.291)	13.509	150.105	4.734 (1.015)	6.253 (2.549)
n50	0.081	60.523 (25.090)	14.679	123.642	5.354 (1.060)	8.703 (3.429)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Wilcoxon rank sum test with continuity correction

data:  error by Graph_size
W = 209, p-value = 0.7908
alternative hypothesis: true location shift is not equal to 0
```

Temps de réponse

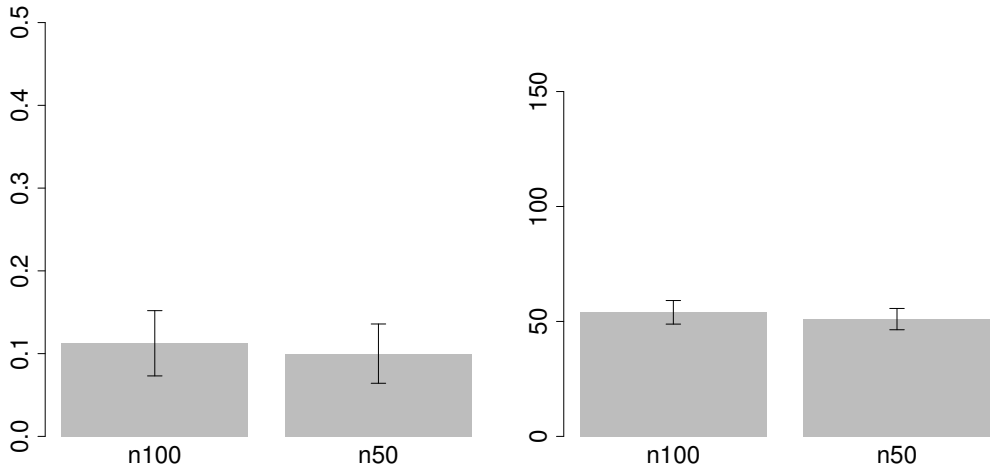
```
Wilcoxon rank sum test

data:  time by Graph_size
W = 219, p-value = 0.6205
alternative hypothesis: true location shift is not equal to 0
```

longueur de chemin

```
Wilcoxon rank sum test with continuity correction

data:  length by Graph_size
W = 128.5, p-value = 0.05419
alternative hypothesis: true location shift is not equal to 0
```

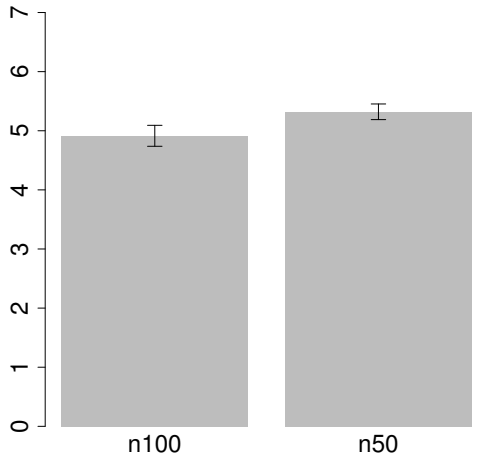
nombre de sélection

```
Wilcoxon rank sum test with continuity correction

data:  selection by Graph_size
W = 128.5, p-value = 0.05469
alternative hypothesis: true location shift is not equal to 0
```

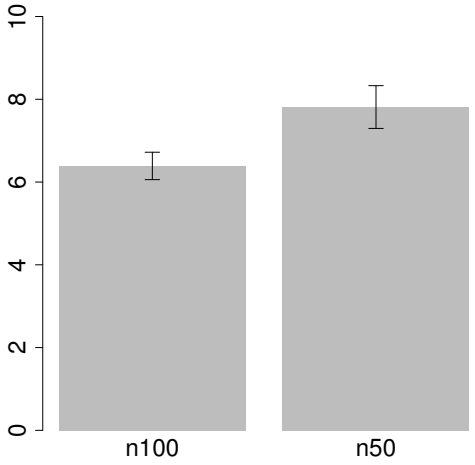


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.3: Résultats par taille de graphe

4.2 Résultats par taille de communauté

Seuil de significativité $\alpha = 0.025$.

Tableau C.5: Résultats par taille de communauté

Taille de communauté	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
k5	0.081	60.523 (25.090)	14.679	123.642	5.354 (1.060)	8.703 (3.429)
k10	0.100	43.529 (30.720)	8.288	133.069	5.004 (0.883)	7.180 (3.470)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Wilcoxon rank sum test with continuity correction

data:  error by Graph_community
W = 144.5, p-value = 0.09642
alternative hypothesis: true location shift is not equal to 0
```

Temps de réponse

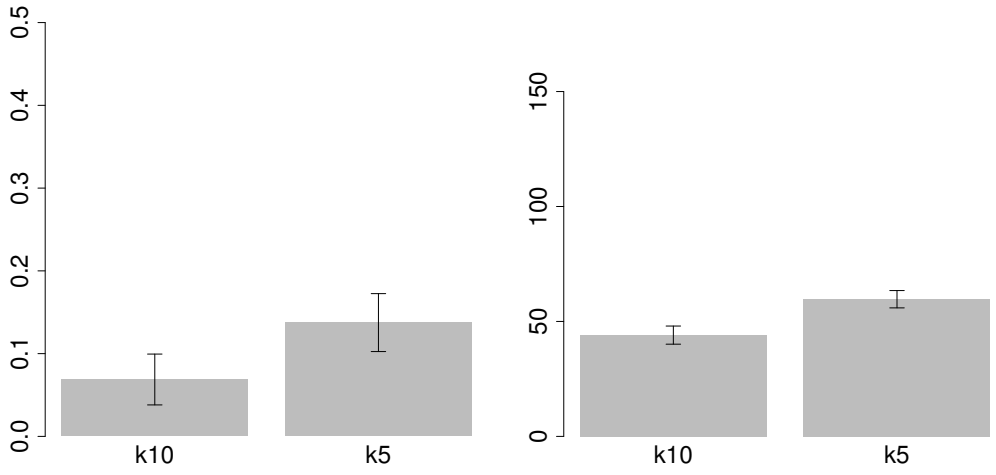
```
Wilcoxon rank sum test

data:  time by Graph_community
W = 116, p-value = 0.02272
alternative hypothesis: true location shift is not equal to 0
```

longueur de chemin

```
Wilcoxon rank sum test with continuity correction

data:  length by Graph_community
W = 185.5, p-value = 0.7041
alternative hypothesis: true location shift is not equal to 0
```

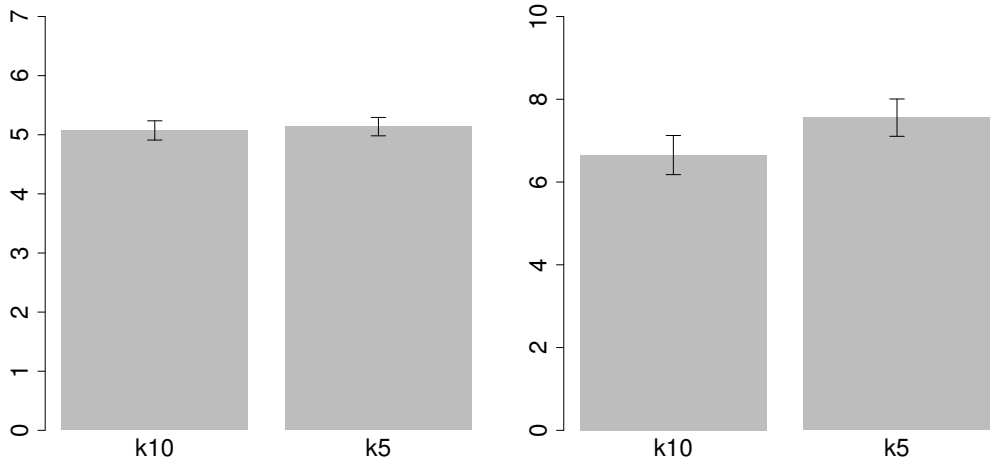
nombre de sélection

```
Wilcoxon rank sum test with continuity correction

data:  selection by Graph_community
W = 138.5, p-value = 0.09866
alternative hypothesis: true location shift is not equal to 0
```



(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin

(d) nombre de sélection

Figure C.4: Résultats par taille de communauté

5 Résultats par combinaison d'encodage et de graphe

5.1 Résultats pour le graphe n100k5 pour chaque encodage

Seuil de significativité $\alpha = 0.025$.

Tableau C.6: Résultats pour le graphe n100k5 pour chaque encodage

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n100k5	0	0.275	60.065 (37.627)	18.520	150.105	4.447 (0.985)	5.895 (2.401)
n100k5	2	0.200	51.349 (34.711)	14.274	135.792	4.632 (1.200)	6.316 (2.968)
n100k5	3	0.150	55.297 (31.100)	13.509	142.314	4.921 (1.044)	6.447 (2.608)
n100k5	4	0.100	60.629 (31.242)	20.631	132.495	4.925 (0.799)	6.350 (2.357)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by graphn100k5_view
Kruskal-Wallis chi-squared = 4.6739, df = 3, p-value = 0.1973
```

Temps de réponse

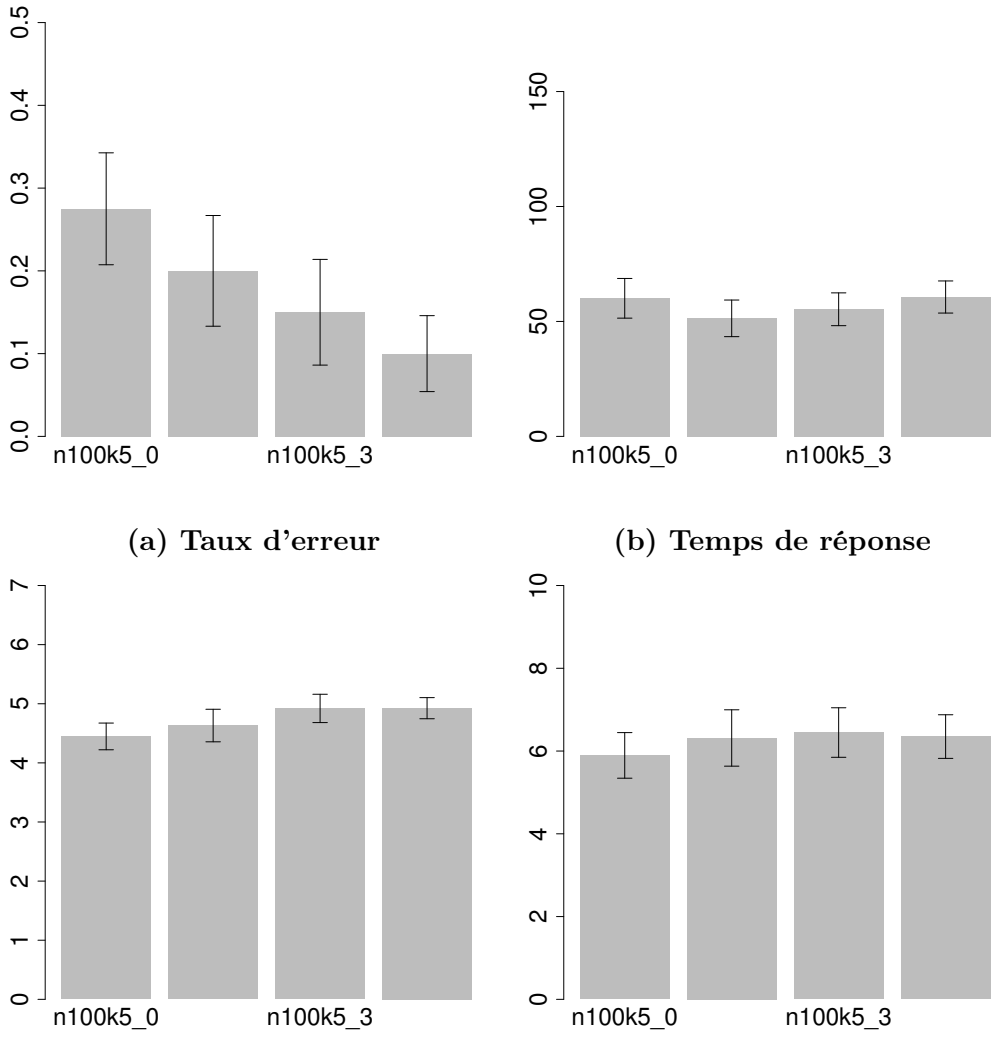
```
Kruskal-Wallis rank sum test

data:  time by graphn100k5_view
Kruskal-Wallis chi-squared = 1.6048, df = 3, p-value = 0.6583
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by graphn100k5_view
Kruskal-Wallis chi-squared = 9.5524, df = 3, p-value = 0.02278
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by graphn100k5_view
Kruskal-Wallis chi-squared = 1.197, df = 3, p-value = 0.7537
```



(a) Taux d'erreur **(b) Temps de réponse**

(c) longueur de chemin **(d) nombre de sélection**

Figure C.5: Résultats pour le graphe n100k5 pour chaque encodage

5.2 Résultats pour le graphe n100k10 pour chaque encodage

Seuil de significativité $\alpha = 0.025$.

Tableau C.7: Résultats pour le graphe n100k10 pour chaque encodage

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n100k10	0	0.025	67.079 (40.943)	20.592	150.903	5.025 (0.896)	7.450 (3.060)
n100k10	2	0.050	41.085 (16.666)	16.387	73.678	4.875 (0.825)	5.700 (1.332)
n100k10	3	0.025	37.124 (14.464)	15.815	64.972	5.225 (0.786)	5.850 (1.358)
n100k10	4	0.050	50.827 (38.588)	12.827	160.888	5.075 (0.922)	5.975 (2.262)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by graphn100k10_view
Kruskal-Wallis chi-squared = 0.7117, df = 3, p-value = 0.8704
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by graphn100k10_view
Kruskal-Wallis chi-squared = 5.3443, df = 3, p-value = 0.1483
```

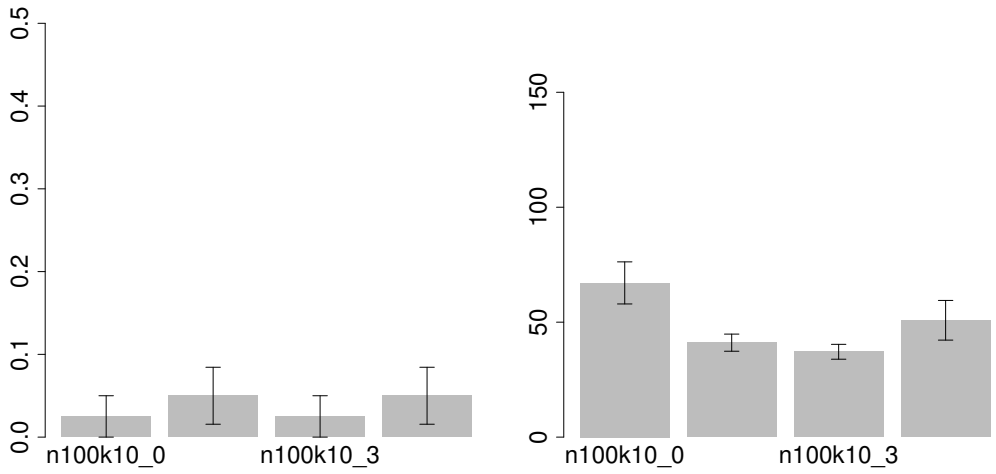
longueur de chemin

```
Kruskal-Wallis rank sum test
```

```
data:  length by graphn100k10_view
Kruskal-Wallis chi-squared = 1.9225, df = 3, p-value = 0.5887
```

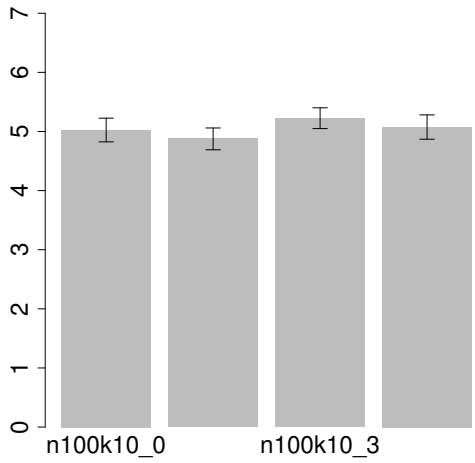
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by graphn100k10_view
Kruskal-Wallis chi-squared = 4.8087, df = 3, p-value = 0.1864
```

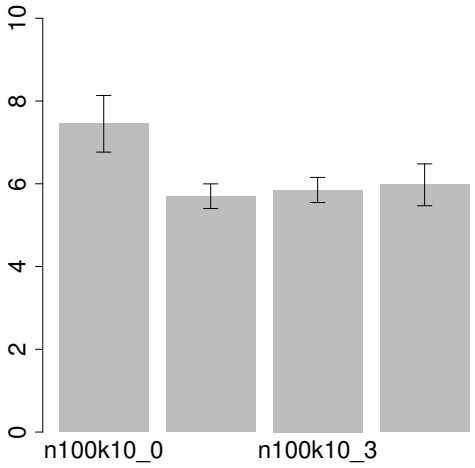


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.6: Résultats pour le graphe n100k10 pour chaque encodage

5.3 Résultats pour le graphe n50k5 pour chaque encodage

Seuil de significativité $\alpha = 0.025$.

Tableau C.8: Résultats pour le graphe n50k5 pour chaque encodage

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n50k5	0	0.150	60.529 (29.919)	22.685	116.532	5.026 (0.950)	7.816 (4.093)
n50k5	2	0.075	63.902 (21.073)	30.524	103.796	5.425 (1.115)	9.650 (3.257)
n50k5	3	0.050	55.666 (26.546)	14.679	123.642	5.475 (1.141)	8.900 (3.161)
n50k5	4	0.050	61.995 (23.419)	26.035	109.931	5.475 (1.032)	8.400 (3.152)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by graphn50k5_view
Kruskal-Wallis chi-squared = 2.4967, df = 3, p-value = 0.4759
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by graphn50k5_view
Kruskal-Wallis chi-squared = 1.9882, df = 3, p-value = 0.5749
```

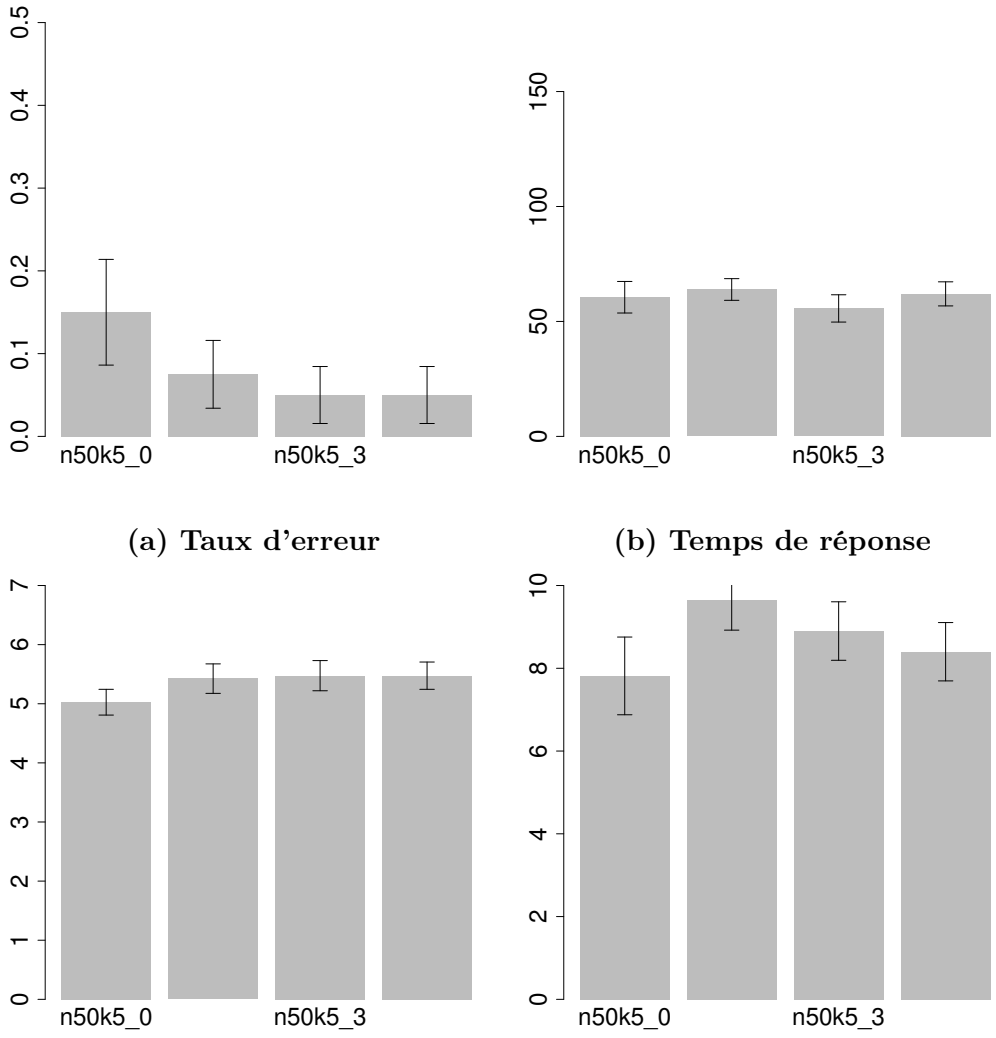
longueur de chemin

```
Kruskal-Wallis rank sum test
```

```
data:  length by graphn50k5_view
Kruskal-Wallis chi-squared = 2.0446, df = 3, p-value = 0.5632
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by graphn50k5_view
Kruskal-Wallis chi-squared = 4.947, df = 3, p-value = 0.1757
```



(a) Taux d'erreur **(b) Temps de réponse**

(c) longueur de chemin **(d) nombre de sélection**

Figure C.7: Résultats pour le graphe n50k5 pour chaque encodage

5.4 Résultats pour le graphe n50k10 pour chaque encodage

Seuil de significativité $\alpha = 0.025$.

Tableau C.9: Résultats pour le graphe n50k10 pour chaque encodage

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n50k10	0	0.175	40.832 (31.221)	12.145	122.264	4.676 (0.789)	7.382 (4.882)
n50k10	2	0.025	46.978 (32.435)	8.584	131.440	5.050 (0.705)	7.250 (3.054)
n50k10	3	0.075	42.814 (26.183)	9.680	94.558	5.100 (0.954)	7.400 (3.564)
n50k10	4	0.125	43.086 (34.594)	8.288	133.069	5.142 (1.029)	6.717 (2.395)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by graphn50k10_view
Kruskal-Wallis chi-squared = 3.2428, df = 3, p-value = 0.3557
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by graphn50k10_view
Kruskal-Wallis chi-squared = 0.7782, df = 3, p-value = 0.8547
```

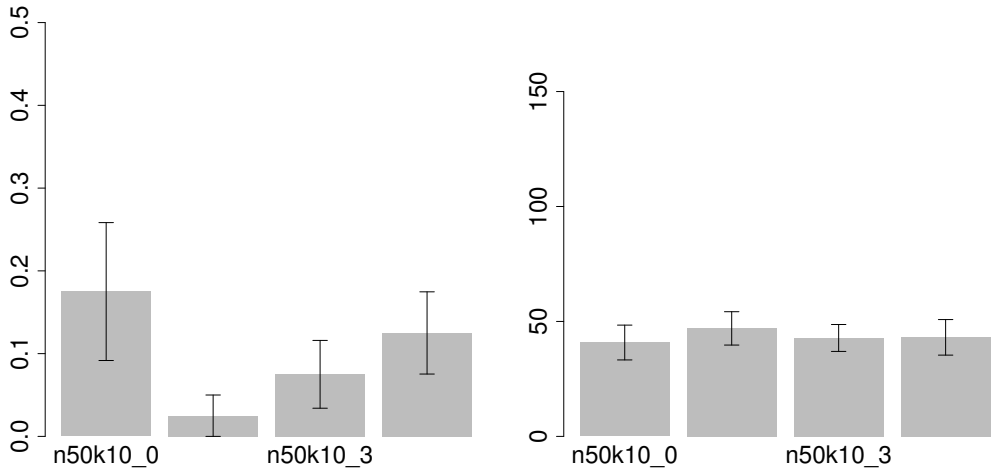
longueur de chemin

```
Kruskal-Wallis rank sum test
```

```
data:  length by graphn50k10_view
Kruskal-Wallis chi-squared = 3.4776, df = 3, p-value = 0.3237
```

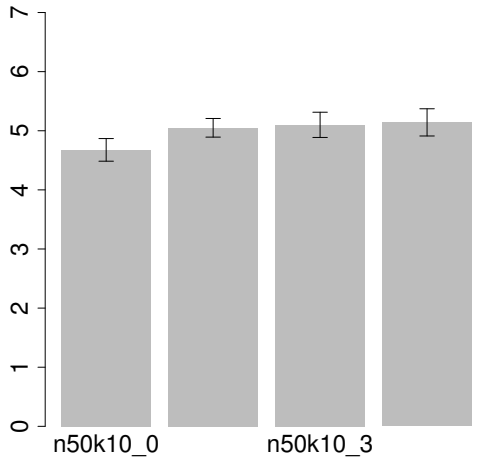
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by graphn50k10_view
Kruskal-Wallis chi-squared = 1.3062, df = 3, p-value = 0.7277
```

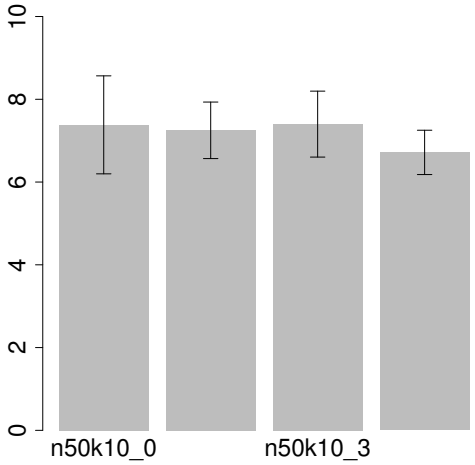


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.8: Résultats pour le graphe n50k10 pour chaque encodage

5.5 Résultats pour l'encodage 0 pour chaque graphe

Seuil de significativité $\alpha = 0.025$.

Tableau C.10: Résultats pour l'encodage 0 pour chaque graphe

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n100k5	0	0.275	60.065 (37.627)	18.520	150.105	4.447 (0.985)	5.895 (2.401)
n100k10	0	0.025	67.079 (40.943)	20.592	150.903	5.025 (0.896)	7.450 (3.060)
n50k5	0	0.150	60.529 (29.919)	22.685	116.532	5.026 (0.950)	7.816 (4.093)
n50k10	0	0.175	40.832 (31.221)	12.145	122.264	4.676 (0.789)	7.382 (4.882)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_view0
Kruskal-Wallis chi-squared = 9.8912, df = 3, p-value = 0.01951
```

Temps de réponse

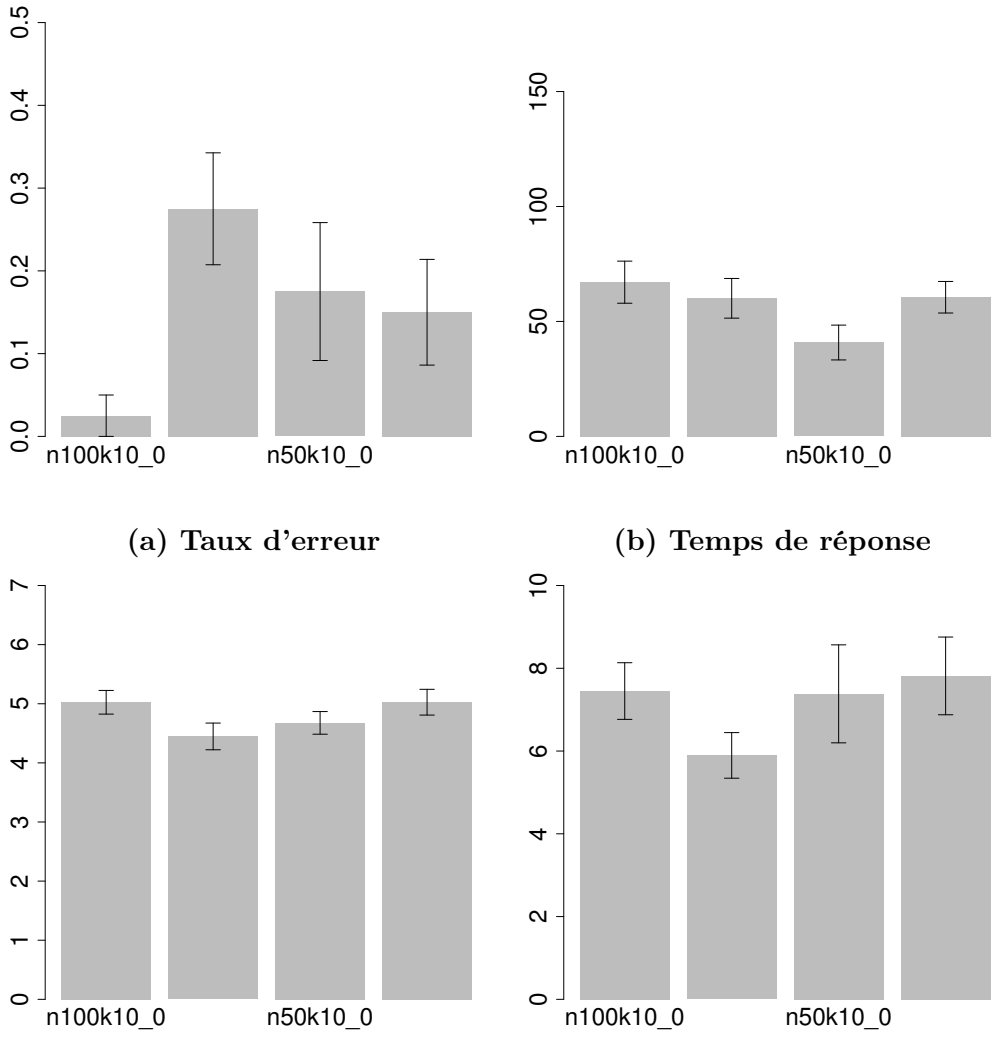
```
Kruskal-Wallis rank sum test

data:  time by Graph_view0
Kruskal-Wallis chi-squared = 7.7497, df = 3, p-value = 0.05148
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Graph_view0
Kruskal-Wallis chi-squared = 8.6644, df = 3, p-value = 0.0341
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_view0
Kruskal-Wallis chi-squared = 3.825, df = 3, p-value = 0.281
```



(a) Taux d'erreur **(b) Temps de réponse**

(c) longueur de chemin **(d) nombre de sélection**

Figure C.9: Résultats pour l'encodage 0 pour chaque graphe

5.6 Résultats pour l'encodage 2 pour chaque graphe

Seuil de significativité $\alpha = 0.025$.

Tableau C.11: Résultats pour l'encodage 2 pour chaque graphe

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n100k5	2	0.200	51.349 (34.711)	14.274	135.792	4.632 (1.200)	6.316 (2.968)
n100k10	2	0.050	41.085 (16.666)	16.387	73.678	4.875 (0.825)	5.700 (1.332)
n50k5	2	0.075	63.902 (21.073)	30.524	103.796	5.425 (1.115)	9.650 (3.257)
n50k10	2	0.025	46.978 (32.435)	8.584	131.440	5.050 (0.705)	7.250 (3.054)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_view2
Kruskal-Wallis chi-squared = 7.7675, df = 3, p-value = 0.05107
```

Temps de réponse

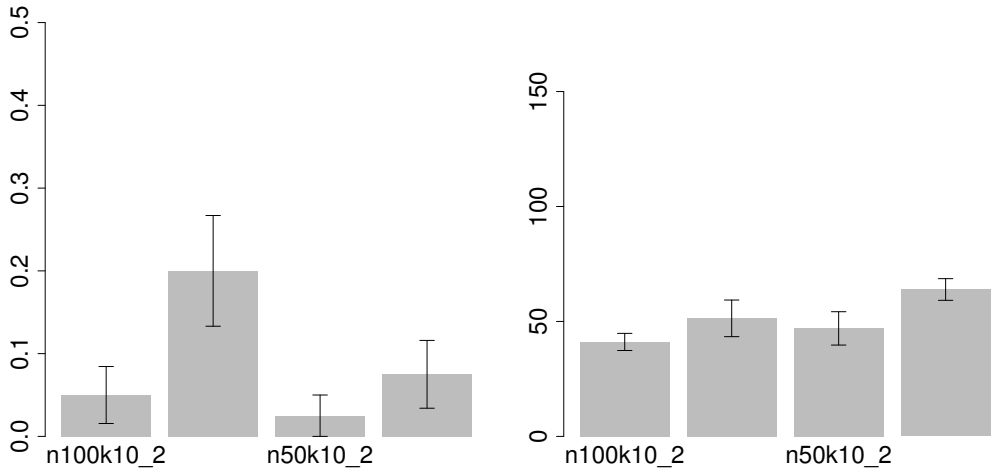
```
Kruskal-Wallis rank sum test

data:  time by Graph_view2
Kruskal-Wallis chi-squared = 10.7716, df = 3, p-value = 0.01303
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Graph_view2
Kruskal-Wallis chi-squared = 10.3746, df = 3, p-value = 0.01564
```

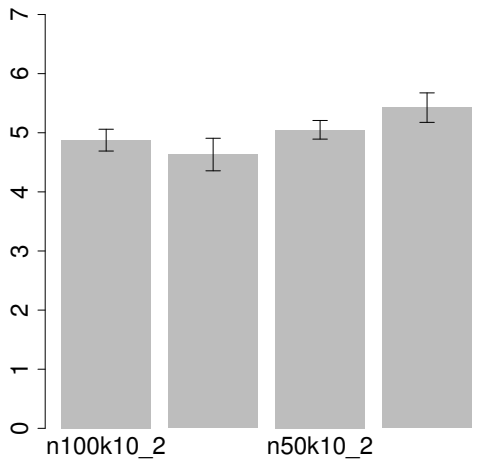
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_view2
Kruskal-Wallis chi-squared = 18.2316, df = 3, p-value = 0.000394
```

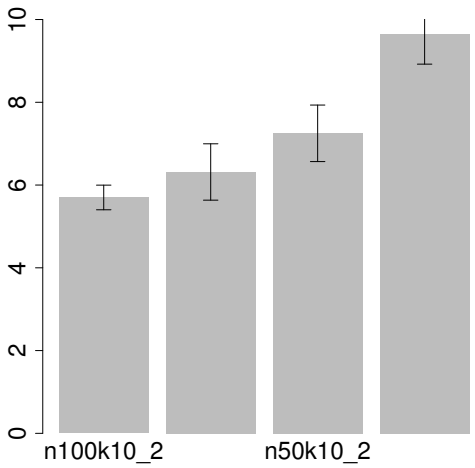


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.10: Résultats pour l'encodage 2 pour chaque graphe

5.7 Résultats pour l'encodage 3 pour chaque graphe

Seuil de significativité $\alpha = 0.025$.

Tableau C.12: Résultats pour l'encodage 3 pour chaque graphe

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n100k5	3	0.150	55.297 (31.100)	13.509	142.314	4.921 (1.044)	6.447 (2.608)
n100k10	3	0.025	37.124 (14.464)	15.815	64.972	5.225 (0.786)	5.850 (1.358)
n50k5	3	0.050	55.666 (26.546)	14.679	123.642	5.475 (1.141)	8.900 (3.161)
n50k10	3	0.075	42.814 (26.183)	9.680	94.558	5.100 (0.954)	7.400 (3.564)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_view3
Kruskal-Wallis chi-squared = 3.8058, df = 3, p-value = 0.2832
```

Temps de réponse

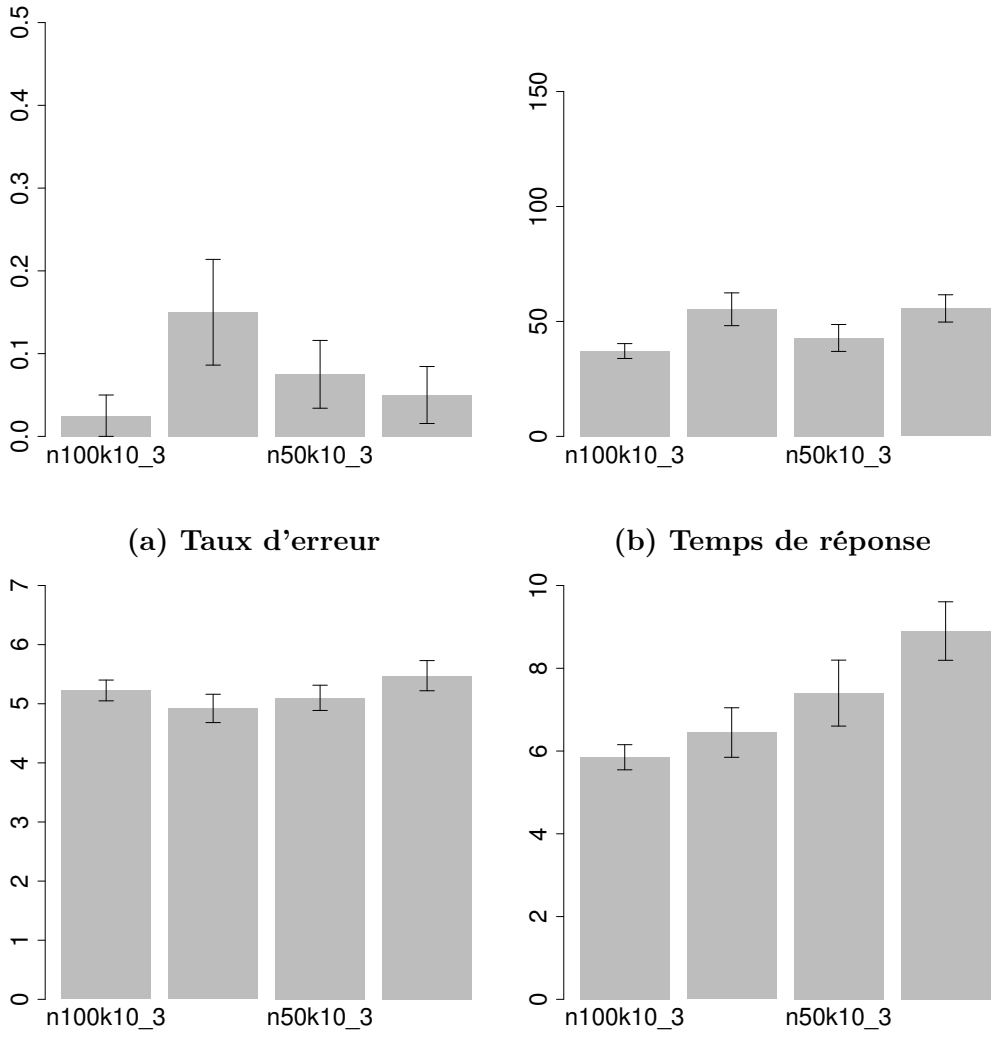
```
Kruskal-Wallis rank sum test

data:  time by Graph_view3
Kruskal-Wallis chi-squared = 6.9936, df = 3, p-value = 0.0721
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Graph_view3
Kruskal-Wallis chi-squared = 3.9799, df = 3, p-value = 0.2636
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_view3
Kruskal-Wallis chi-squared = 11.9027, df = 3, p-value = 0.007724
```



(a) Taux d'erreur **(b) Temps de réponse**

(c) longueur de chemin **(d) nombre de sélection**

Figure C.11: Résultats pour l'encodage 3 pour chaque graphe

5.8 Résultats pour l'encodage 4 pour chaque graphe

Seuil de significativité $\alpha = 0.025$.

Tableau C.13: Résultats pour l'encodage 4 pour chaque graphe

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n100k5	4	0.100	60.629 (31.242)	20.631	132.495	4.925 (0.799)	6.350 (2.357)
n100k10	4	0.050	50.827 (38.588)	12.827	160.888	5.075 (0.922)	5.975 (2.262)
n50k5	4	0.050	61.995 (23.419)	26.035	109.931	5.475 (1.032)	8.400 (3.152)
n50k10	4	0.125	43.086 (34.594)	8.288	133.069	5.142 (1.029)	6.717 (2.395)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_view4
Kruskal-Wallis chi-squared = 2.4489, df = 3, p-value = 0.4846
```

Temps de réponse

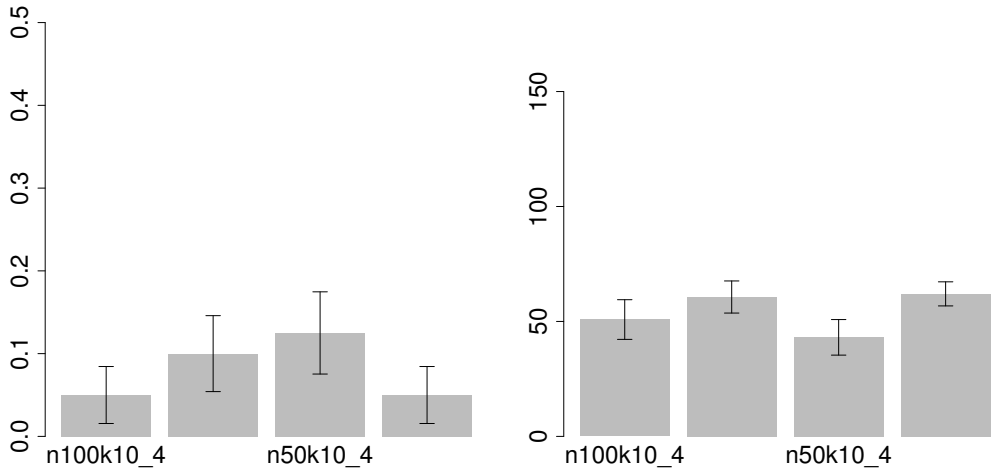
```
Kruskal-Wallis rank sum test

data:  time by Graph_view4
Kruskal-Wallis chi-squared = 8.8996, df = 3, p-value = 0.03066
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Graph_view4
Kruskal-Wallis chi-squared = 3.2597, df = 3, p-value = 0.3533
```

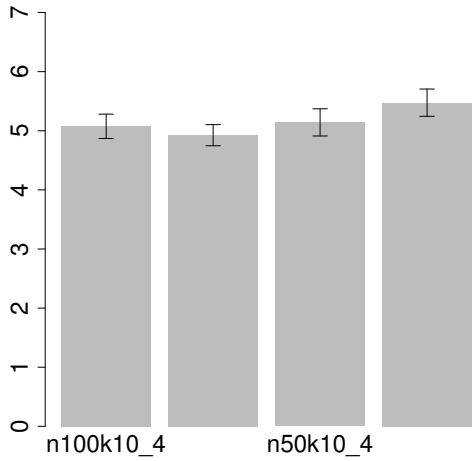
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_view4
Kruskal-Wallis chi-squared = 9.8546, df = 3, p-value = 0.01984
```

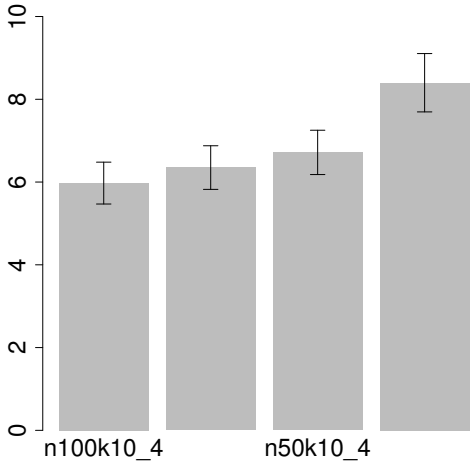


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.12: Résultats pour l'encodage 4 pour chaque graphe

5.9 Résultats pour chaque encodage et pour chaque taille de graphe

Seuil de significativité $\alpha = 0.025$.

Tableau C.14: Résultats pour chaque encodage et pour chaque taille de graphe

Taille de graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
n100	0	0.275	60.065 (37.627)	18.520	150.105	4.447 (0.985)	5.895 (2.401)
n100	2	0.200	51.349 (34.711)	14.274	135.792	4.632 (1.200)	6.316 (2.968)
n100	3	0.025	37.124 (14.464)	15.815	64.972	5.225 (0.786)	5.850 (1.358)
n100	4	0.100	60.629 (31.242)	20.631	132.495	4.925 (0.799)	6.350 (2.357)
n50	0	0.150	60.529 (29.919)	22.685	116.532	5.026 (0.950)	7.816 (4.093)
n50	2	0.075	63.902 (21.073)	30.524	103.796	5.425 (1.115)	9.650 (3.257)
n50	3	0.050	55.666 (26.546)	14.679	123.642	5.475 (1.141)	8.900 (3.161)
n50	4	0.050	61.995 (23.419)	26.035	109.931	5.475 (1.032)	8.400 (3.152)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_size_view
Kruskal-Wallis chi-squared = 6.44, df = 7, p-value = 0.4894
```

Temps de réponse

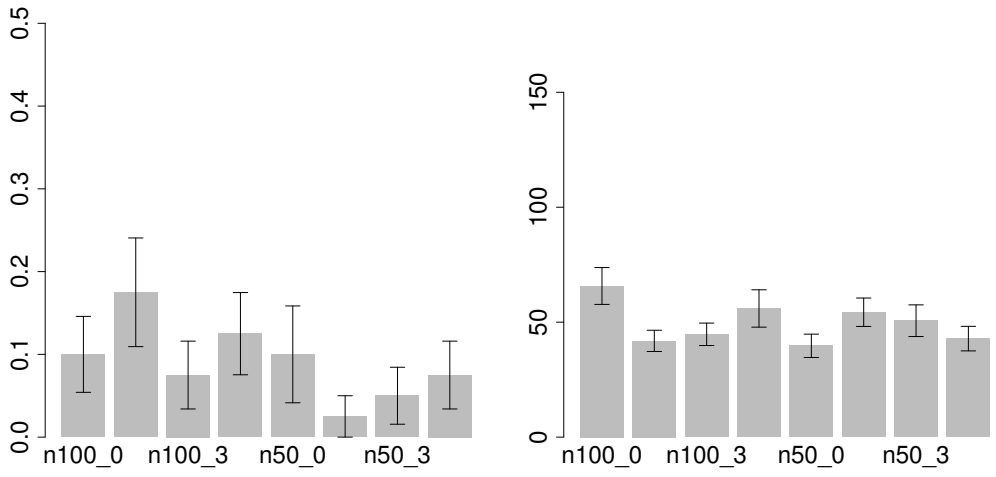
```
Kruskal-Wallis rank sum test

data:  time by Graph_size_view
Kruskal-Wallis chi-squared = 10.1669, df = 7, p-value = 0.1793
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Graph_size_view
Kruskal-Wallis chi-squared = 8.981, df = 7, p-value = 0.254
```

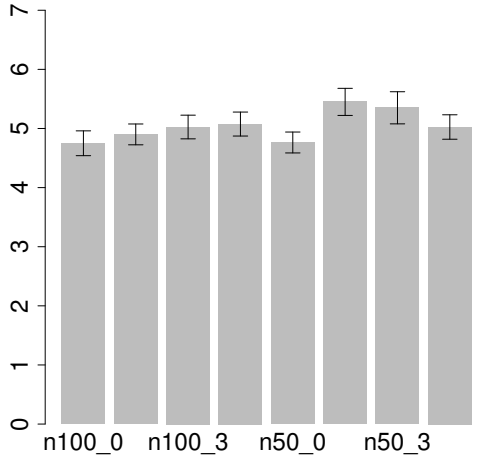
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_size_view
Kruskal-Wallis chi-squared = 15.972, df = 7, p-value = 0.02537
```

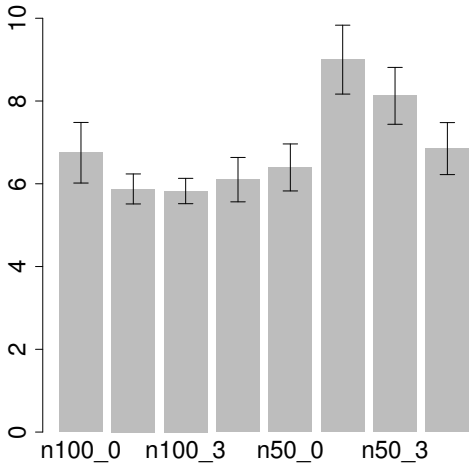


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.13: Résultats pour chaque encodage et pour chaque taille de graphe

5.10 Résultats pour chaque encodage et pour chaque taille de communauté

Seuil de significativité $\alpha = 0.025$.

Tableau C.15: Résultats pour chaque encodage et pour chaque taille de communauté

graphe	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
k5	0	0.275	60.065 (37.627)	18.520	150.105	4.447 (0.985)	5.895 (2.401)
k5	2	0.200	51.349 (34.711)	14.274	135.792	4.632 (1.200)	6.316 (2.968)
k5	3	0.050	55.666 (26.546)	14.679	123.642	5.475 (1.141)	8.900 (3.161)
k5	4	0.050	61.995 (23.419)	26.035	109.931	5.475 (1.032)	8.400 (3.152)
k10	0	0.175	40.832 (31.221)	12.145	122.264	4.676 (0.789)	7.382 (4.882)
k10	2	0.050	41.085 (16.666)	16.387	73.678	4.875 (0.825)	5.700 (1.332)
k10	3	0.025	37.124 (14.464)	15.815	64.972	5.225 (0.786)	5.850 (1.358)
k10	4	0.125	43.086 (34.594)	8.288	133.069	5.142 (1.029)	6.717 (2.395)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_comm_view
Kruskal-Wallis chi-squared = 6.5648, df = 7, p-value = 0.4756
```

Temps de réponse

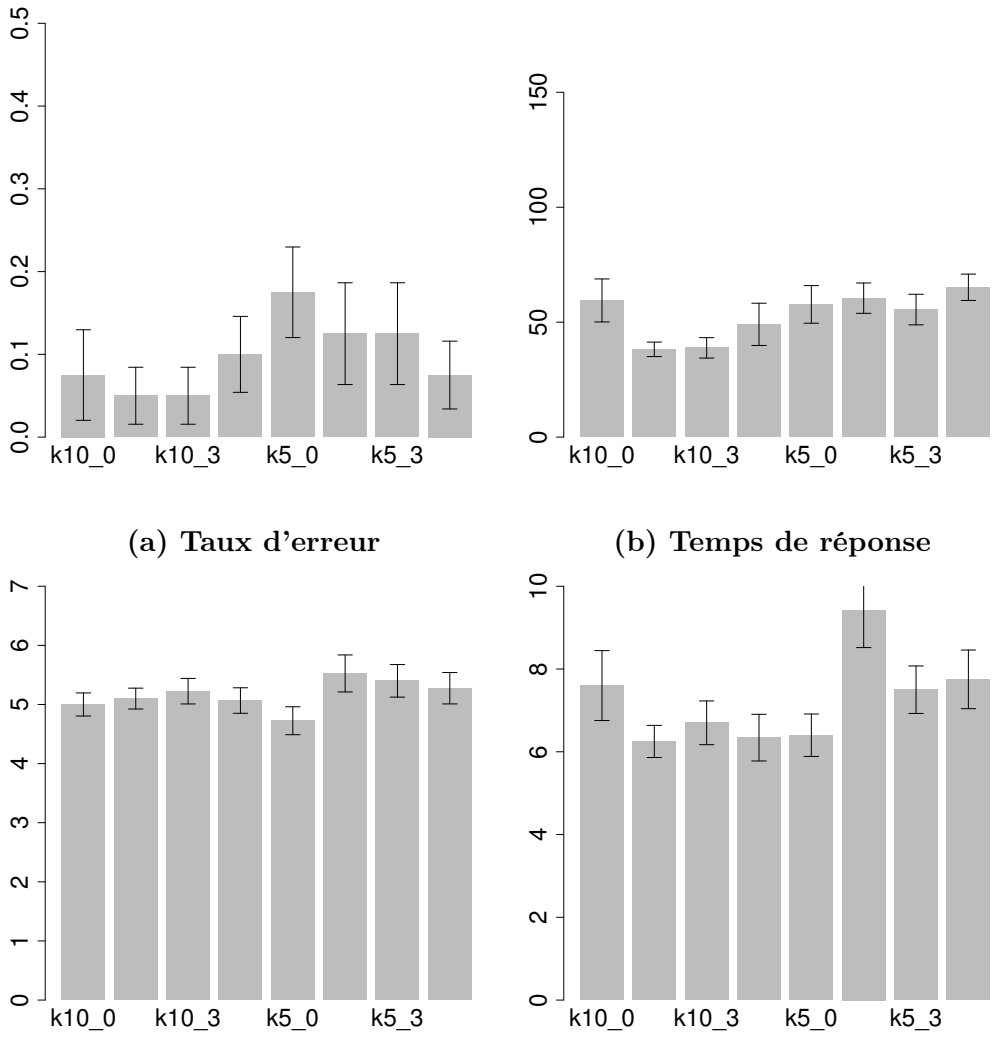
```
Kruskal-Wallis rank sum test

data:  time by Graph_comm_view
Kruskal-Wallis chi-squared = 17.1208, df = 7, p-value = 0.01663
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Graph_comm_view
Kruskal-Wallis chi-squared = 7.6987, df = 7, p-value = 0.3599
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_comm_view
Kruskal-Wallis chi-squared = 11.4692, df = 7, p-value = 0.1194
```



(a) Taux d'erreur

(b) Temps de réponse

(c) longueur de chemin

(d) nombre de sélection

Figure C.14: Résultats pour chaque encodage et pour chaque taille de communauté

6 Résultats par densité

— Densité 1 (0.04) : graphe n100k10

— Densité 2 (0.07) : graphe n50k5 et n50k10

— Denstité 3 (0.1) : graphe n100k5

6.1 Résultats pour chaque densité

Seuil de significativité $\alpha = 0.05$.

Tableau C.16: Résultats pour chaque densité

Densité	taux d'erreur	temps de réponse	temps de time	temps de time	longueur de chemin	nombre de sélection
dens. 3	0.100	43.529 (30.720)	8.288	133.069	5.004 (0.883)	7.180 (3.470)
dens. 1	0.059	54.643 (21.296)	24.847	149.442	5.190 (0.705)	7.429 (2.180)
dens. 2	0.181	56.884 (33.291)	13.509	150.105	4.734 (1.015)	6.253 (2.549)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Diff
Kruskal-Wallis chi-squared = 6.3399, df = 2, p-value = 0.04201
```

Temps de réponse

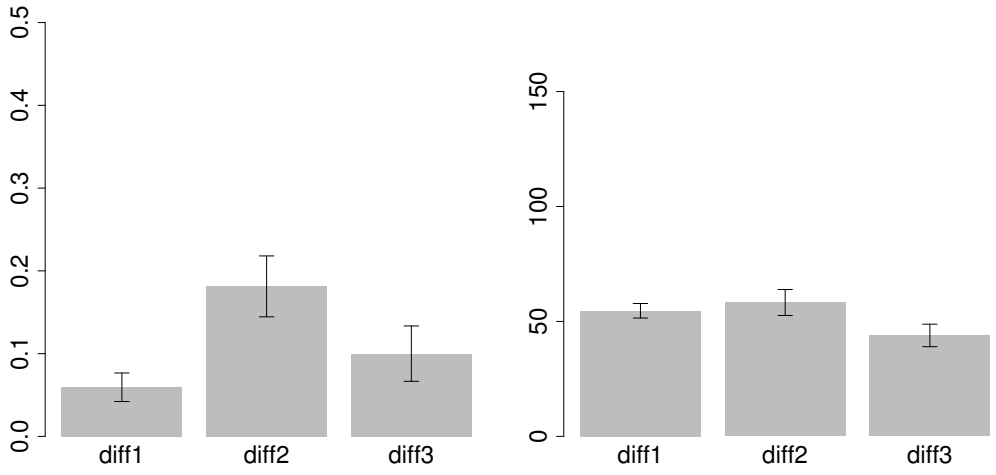
```
Kruskal-Wallis rank sum test

data:  time by Diff
Kruskal-Wallis chi-squared = 4.8593, df = 2, p-value = 0.08807
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Diff
Kruskal-Wallis chi-squared = 7.5744, df = 2, p-value = 0.02266
```

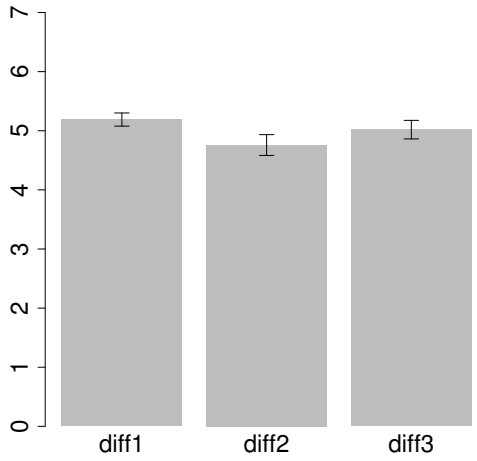
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Diff
Kruskal-Wallis chi-squared = 4.6455, df = 2, p-value = 0.098
```

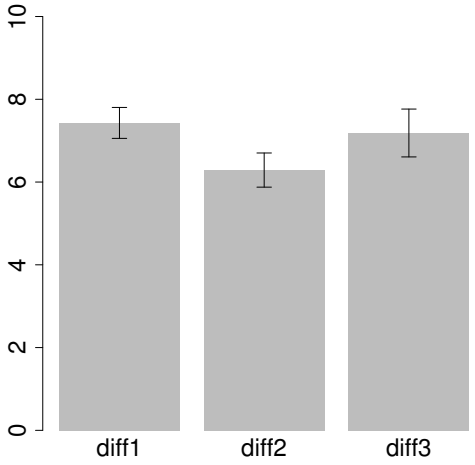


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.15: Résultats pour chaque densité

6.2 Résultats pour chaque encodage et par densité du graphe

— densité 1 (0.04) : graphe n100k10

— densité 2 (0.07) : graphe n50k5 et n50k10

— densité 3 (0.1) : graphe n100k5

Seuil de significativité $\alpha = 0.025$.

Tableau C.17: Résultats pour chaque encodage et par densité du graphe

densité	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
dens. 1	0	0.087	65.332 (30.082)	26.203	149.442	5.037 (0.708)	7.650 (2.788)
dens. 1	2	0.062	52.055 (13.393)	25.150	69.134	5.117 (0.690)	7.667 (2.189)
dens. 1	3	0.037	47.029 (14.191)	24.847	80.238	5.362 (0.678)	7.400 (1.885)
dens. 1	4	0.050	54.156 (20.282)	28.052	98.049	5.242 (0.753)	7.000 (1.826)
dens. 2	0	0.275	60.065 (37.627)	18.520	150.105	4.447 (0.985)	5.895 (2.401)
dens. 2	2	0.200	51.349 (34.711)	14.274	135.792	4.632 (1.200)	6.316 (2.968)
dens. 2	3	0.150	55.297 (31.100)	13.509	142.314	4.921 (1.044)	6.447 (2.608)
dens. 2	4	0.100	60.629 (31.242)	20.631	132.495	4.925 (0.799)	6.350 (2.357)
dens. 3	0	0.175	40.832 (31.221)	12.145	122.264	4.676 (0.789)	7.382 (4.882)
dens. 3	2	0.025	46.978 (32.435)	8.584	131.440	5.050 (0.705)	7.250 (3.054)
dens. 3	3	0.075	42.814 (26.183)	9.680	94.558	5.100 (0.954)	7.400 (3.564)
dens. 3	4	0.125	43.086 (34.594)	8.288	133.069	5.142 (1.029)	6.717 (2.395)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_comm_difficulty
```

```
Kruskal-Wallis chi-squared = 17.9672, df = 11, p-value = 0.08234
```

Temps de réponse

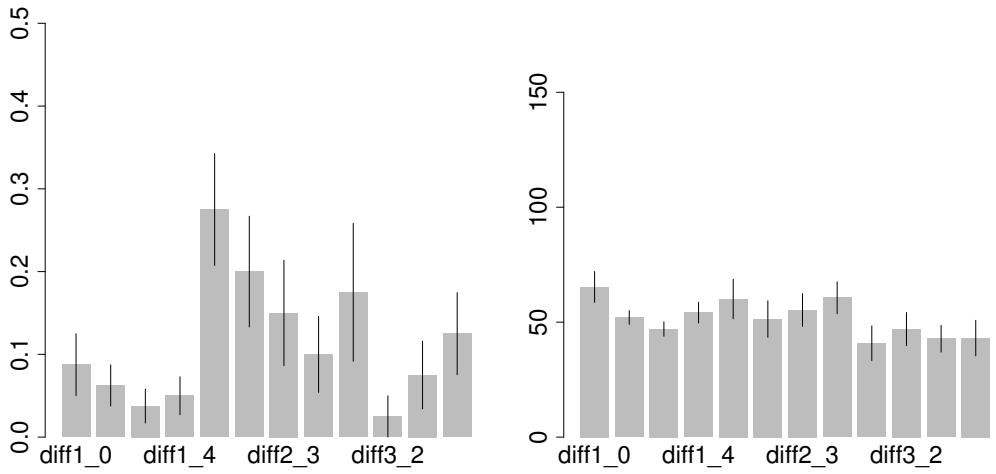
```
Kruskal-Wallis rank sum test

data:  time by Graph_comm_difficulty
Kruskal-Wallis chi-squared = 21.756, df = 11, p-value = 0.02633
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by Graph_comm_difficulty
Kruskal-Wallis chi-squared = 34.5005, df = 11, p-value = 0.0002994
```

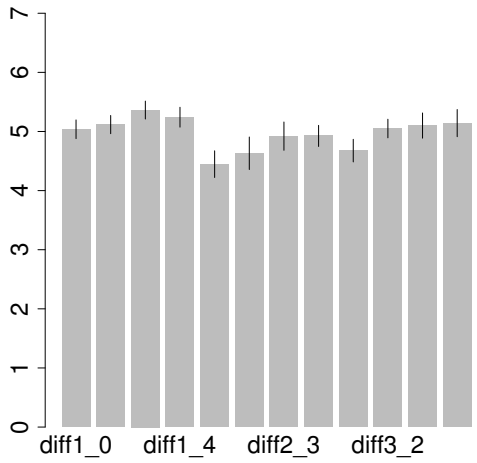
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_comm_difficulty
Kruskal-Wallis chi-squared = 18.4865, df = 11, p-value = 0.07096
```

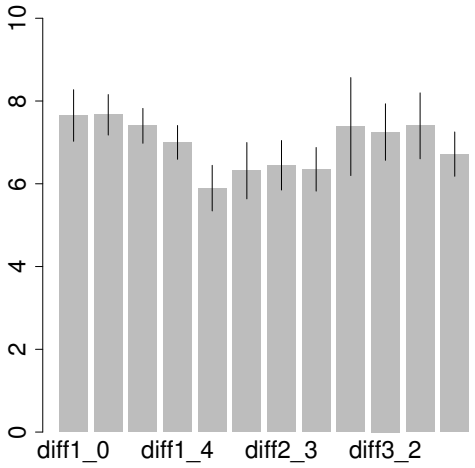


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.16: Résultats pour chaque encodage et par densité du graphe

6.3 Résultats pour l'encodage 0 pour chaque densité

— densité 1 (0.04) : graphe n100k10

— densité 2 (0.07) : graphe n50k5 et n50k10

— densité 3 (0.1) : graphe n100k5

Seuil de significativité $\alpha = 0.025$.

Tableau C.18: Résultats pour l'encodage 0 pour chaque densité

densité	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
dens. 1	0	0.087	65.332 (30.082)	26.203	149.442	5.037 (0.708)	7.650 (2.788)
dens. 2	0	0.275	60.065 (37.627)	18.520	150.105	4.447 (0.985)	5.895 (2.401)
dens. 3	0	0.175	40.832 (31.221)	12.145	122.264	4.676 (0.789)	7.382 (4.882)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_v0_difficulty
Kruskal-Wallis chi-squared = 4.7029, df = 2, p-value = 0.09523
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by Graph_v0_difficulty
Kruskal-Wallis chi-squared = 8.0772, df = 2, p-value = 0.01762
```

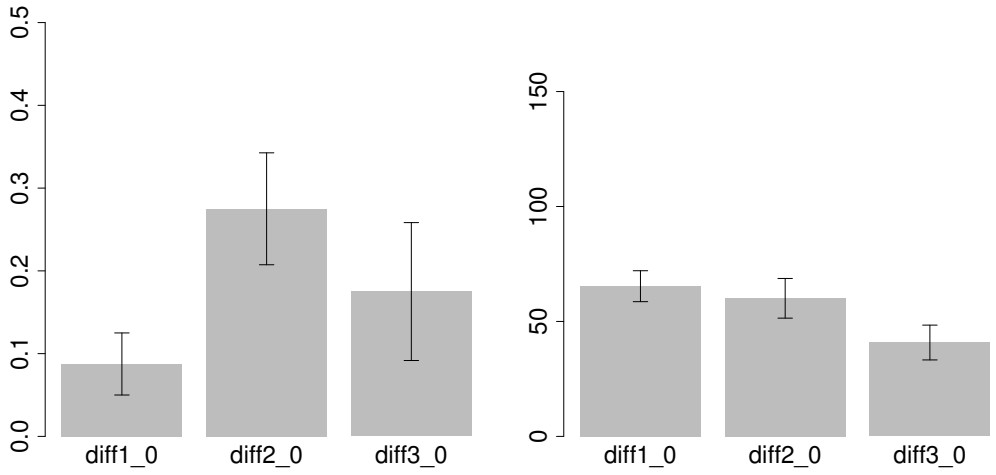
longueur de chemin

```
Kruskal-Wallis rank sum test
```

```
data:  length by Graph_v0_difficulty
Kruskal-Wallis chi-squared = 9.824, df = 2, p-value = 0.007358
```

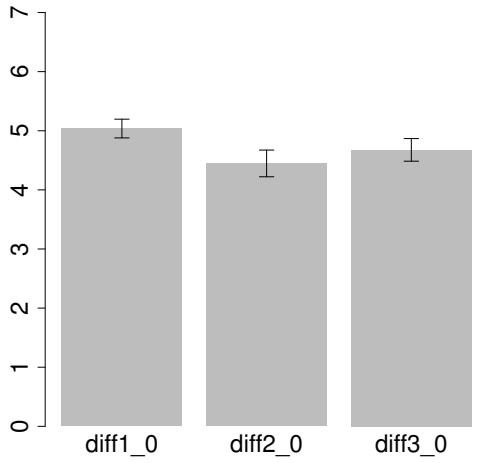
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_v0_difficulty
Kruskal-Wallis chi-squared = 4.2082, df = 2, p-value = 0.122
```

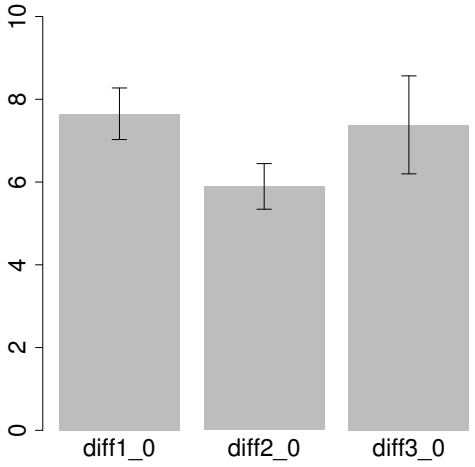


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.17: Résultats pour l'encodage 0 pour chaque densité

6.4 Résultats pour l'encodage 2 pour chaque densité

— densité 1 (0.04) : graphe n100k10

— densité 2 (0.07) : graphe n50k5 et n50k10

— densité 3 (0.1) : graphe n100k5

Seuil de significativité $\alpha = 0.025$.

Tableau C.19: Résultats pour l'encodage 2 pour chaque densité

densité	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
dens. 1	2	0.062	52.055 (13.393)	25.150	69.134	5.117 (0.690)	7.667 (2.189)
dens. 2	2	0.200	51.349 (34.711)	14.274	135.792	4.632 (1.200)	6.316 (2.968)
dens. 3	2	0.025	46.978 (32.435)	8.584	131.440	5.050 (0.705)	7.250 (3.054)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_v2_difficulty
Kruskal-Wallis chi-squared = 6.087, df = 2, p-value = 0.04767
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by Graph_v2_difficulty
Kruskal-Wallis chi-squared = 3.4937, df = 2, p-value = 0.1743
```

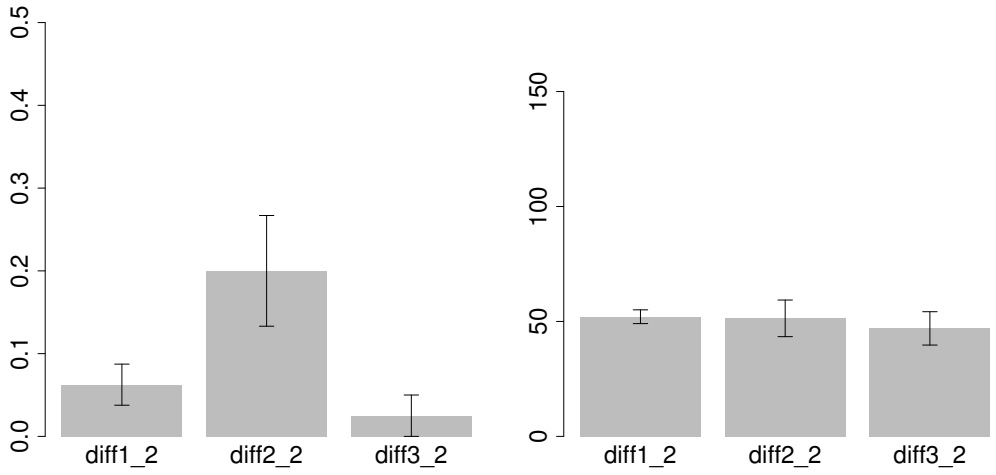
longueur de chemin

```
Kruskal-Wallis rank sum test
```

```
data:  length by Graph_v2_difficulty
Kruskal-Wallis chi-squared = 9.1764, df = 2, p-value = 0.01017
```

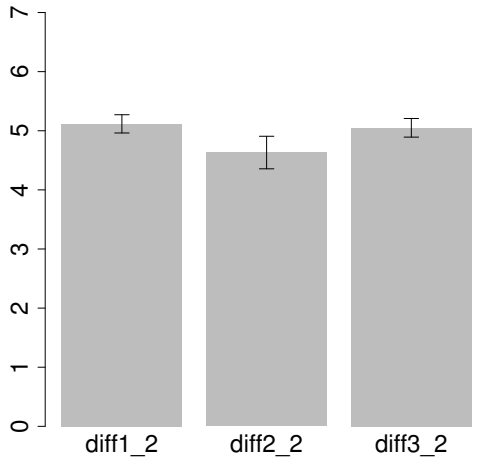
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_v2_difficulty
Kruskal-Wallis chi-squared = 5.307, df = 2, p-value = 0.07041
```

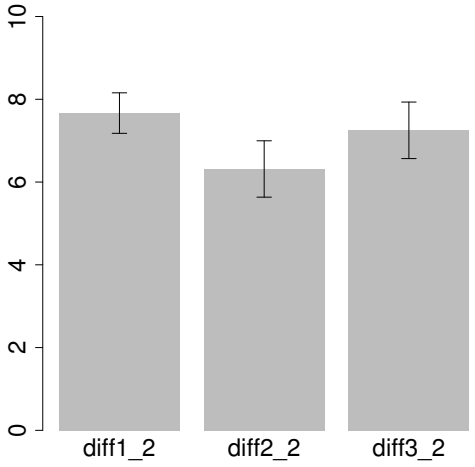


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.18: Résultats pour l'encodage 2 pour chaque densité

6.5 Résultats pour l'encodage 3 pour chaque densité

— densité 1 (0.04) : graphe n100k10

— densité 2 (0.07) : graphe n50k5 et n50k10

— densité 3 (0.1) : graphe n100k5

Seuil de significativité $\alpha = 0.025$.

Tableau C.20: Résultats pour l'encodage 3 pour chaque densité

densité	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
dens. 1	3	0.037	47.029 (14.191)	24.847	80.238	5.362 (0.678)	7.400 (1.885)
dens. 2	3	0.150	55.297 (31.100)	13.509	142.314	4.921 (1.044)	6.447 (2.608)
dens. 3	3	0.075	42.814 (26.183)	9.680	94.558	5.100 (0.954)	7.400 (3.564)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_v3_difficulty
Kruskal-Wallis chi-squared = 1.3419, df = 2, p-value = 0.5112
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by Graph_v3_difficulty
Kruskal-Wallis chi-squared = 1.9407, df = 2, p-value = 0.379
```

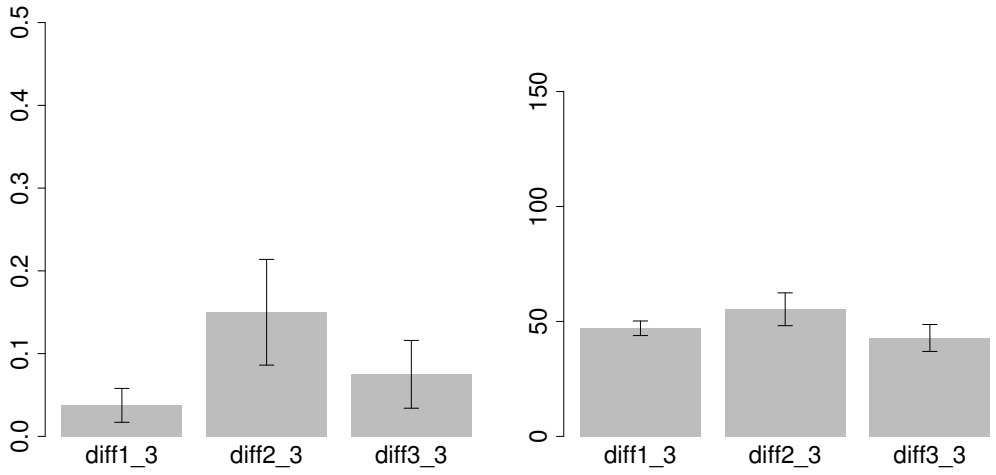
longueur de chemin

```
Kruskal-Wallis rank sum test
```

```
data:  length by Graph_v3_difficulty
Kruskal-Wallis chi-squared = 4.1542, df = 2, p-value = 0.1253
```

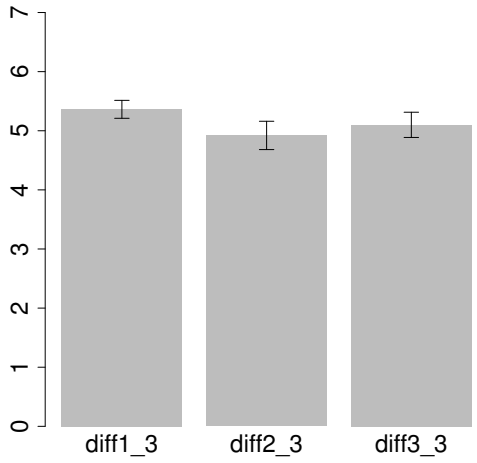
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_v3_difficulty
Kruskal-Wallis chi-squared = 2.6536, df = 2, p-value = 0.2653
```

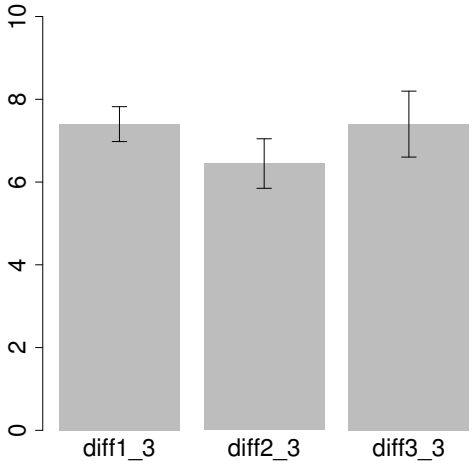


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.19: Résultats pour l'encodage 3 pour chaque densité

6.6 Résultats pour l'encodage 4 pour chaque densité

— densité 1 (0.04) : graphe n100k10

— densité 2 (0.07) : graphe n50k5 et n50k10

— densité 3 (0.1) : graphe n100k5

Seuil de significativité $\alpha = 0.025$.

Tableau C.21: Résultats pour l'encodage 4 pour chaque densité

densité	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
dens. 1	4	0.050	54.156 (20.282)	28.052	98.049	5.242 (0.753)	7.000 (1.826)
dens. 2	4	0.100	60.629 (31.242)	20.631	132.495	4.925 (0.799)	6.350 (2.357)
dens. 3	4	0.125	43.086 (34.594)	8.288	133.069	5.142 (1.029)	6.717 (2.395)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by Graph_v4_difficulty
Kruskal-Wallis chi-squared = 0.5365, df = 2, p-value = 0.7647
```

Temps de réponse

```
Kruskal-Wallis rank sum test

data:  time by Graph_v4_difficulty
Kruskal-Wallis chi-squared = 6.5374, df = 2, p-value = 0.03806
```

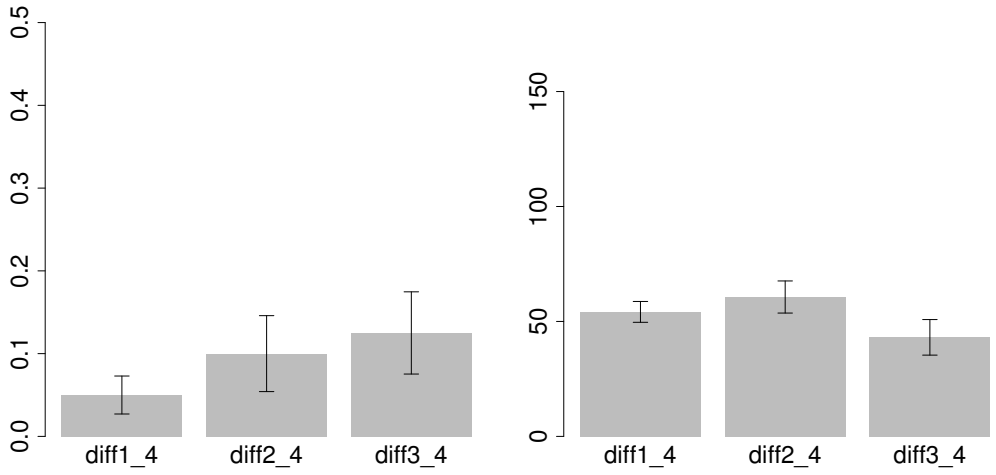
longueur de chemin

```
Kruskal-Wallis rank sum test
```

```
data:  length by Graph_v4_difficulty
Kruskal-Wallis chi-squared = 2.0511, df = 2, p-value = 0.3586
```

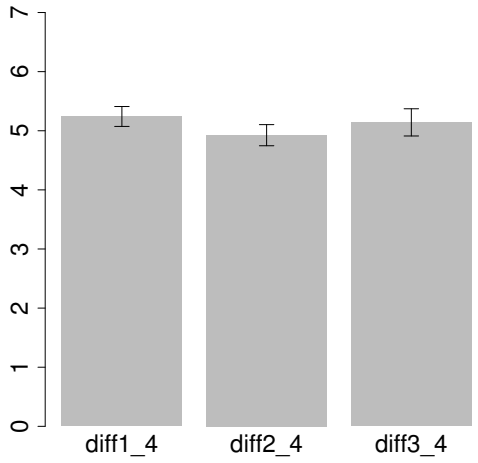
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by Graph_v4_difficulty
Kruskal-Wallis chi-squared = 1.9564, df = 2, p-value = 0.376
```

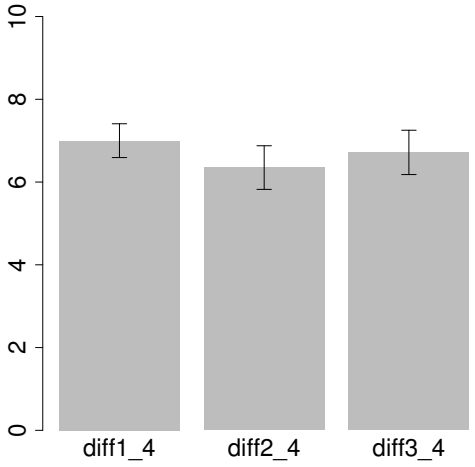


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.20: Résultats pour l'encodage 4 pour chaque densité

7 Résultats pour chaque densité et chaque encodage

7.1 Résultats pour chaque encodage pour la densité 0.04

— densité 1 (0.04) : graphe n100k10

— densité 2 (0.07) : graphe n50k5 et n50k10

— densité 3 (0.1) : graphe n100k5

Seuil de significativité $\alpha = 0.025$.

Tableau C.22: Résultats pour chaque encodage pour la densité 0.04

densité	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
dens. 1	0	0.087	65.332 (30.082)	26.203	149.442	5.037 (0.708)	7.650 (2.788)
dens. 1	2	0.062	52.055 (13.393)	25.150	69.134	5.117 (0.690)	7.667 (2.189)
dens. 1	3	0.037	47.029 (14.191)	24.847	80.238	5.362 (0.678)	7.400 (1.885)
dens. 1	4	0.050	54.156 (20.282)	28.052	98.049	5.242 (0.753)	7.000 (1.826)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by view_dens1
Kruskal-Wallis chi-squared = 0.9888, df = 3, p-value = 0.804
```

Temps de réponse

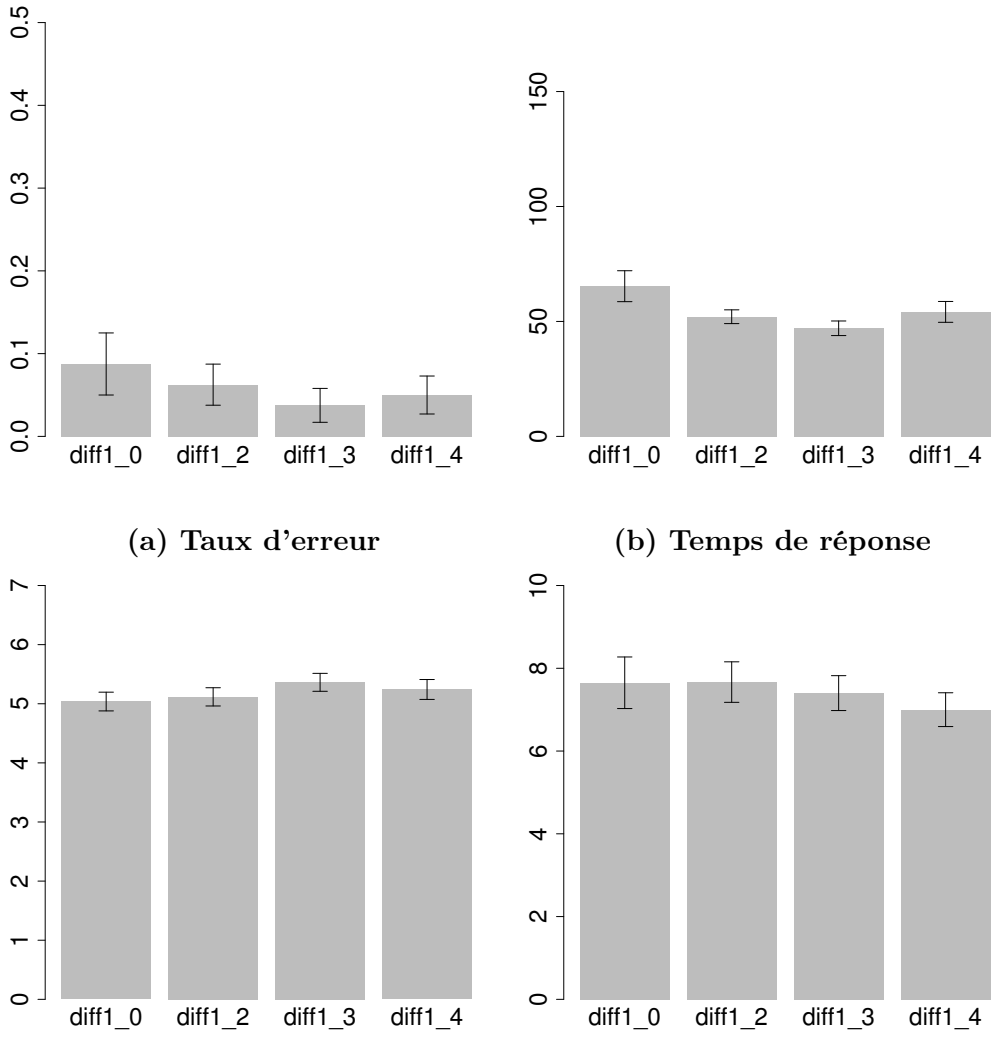
```
Kruskal-Wallis rank sum test

data:  time by view_dens1
Kruskal-Wallis chi-squared = 5.2354, df = 3, p-value = 0.1554
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by view_diff1
Kruskal-Wallis chi-squared = 2.0848, df = 3, p-value = 0.555
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by view_diff1
Kruskal-Wallis chi-squared = 1.0264, df = 3, p-value = 0.7949
```



(a) Taux d'erreur

(b) Temps de réponse

(c) longueur de chemin

(d) nombre de sélection

Figure C.21: Résultats pour chaque encodage pour la densité 0.04

7.2 Résultats pour chaque encodage pour la densité 0.07

— densité 1 (0.04) : graphe n100k10

— densité 2 (0.07) : graphe n50k5 et n50k10

— densité 3 (0.1) : graphe n100k5

Seuil de significativité $\alpha = 0.025$.

Tableau C.23: Résultats pour chaque encodage pour la densité 0.07

densité	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
dens. 2	0	0.275	60.065 (37.627)	18.520	150.105	4.447 (0.985)	5.895 (2.401)
dens. 2	2	0.200	51.349 (34.711)	14.274	135.792	4.632 (1.200)	6.316 (2.968)
dens. 2	3	0.150	55.297 (31.100)	13.509	142.314	4.921 (1.044)	6.447 (2.608)
dens. 2	4	0.100	60.629 (31.242)	20.631	132.495	4.925 (0.799)	6.350 (2.357)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by view_dens2
Kruskal-Wallis chi-squared = 4.6739, df = 3, p-value = 0.1973
```

Temps de réponse

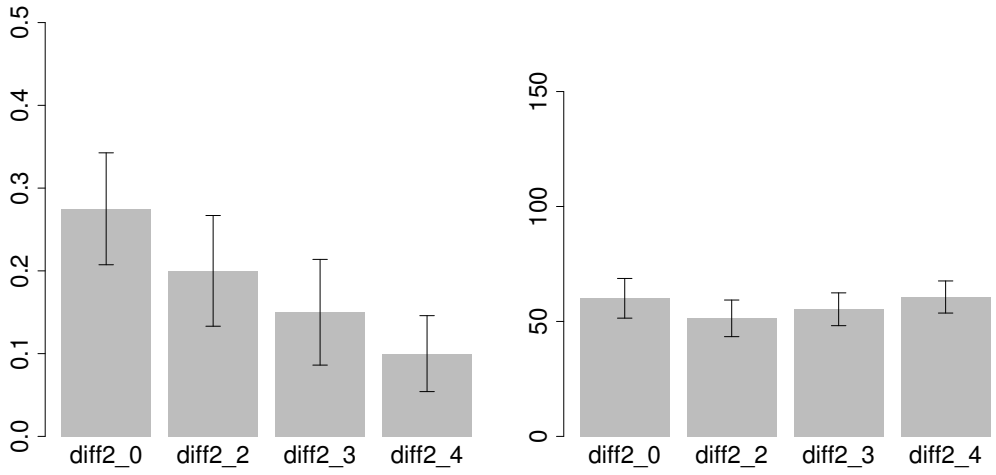
```
Kruskal-Wallis rank sum test

data:  time by view_dens2
Kruskal-Wallis chi-squared = 1.6048, df = 3, p-value = 0.6583
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by view_diff2
Kruskal-Wallis chi-squared = 9.5524, df = 3, p-value = 0.02278
```

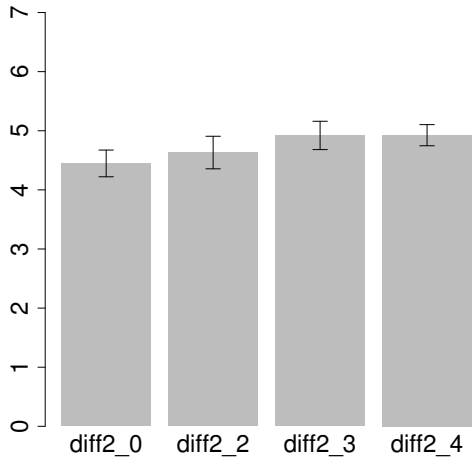
nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by view_diff2
Kruskal-Wallis chi-squared = 1.197, df = 3, p-value = 0.7537
```

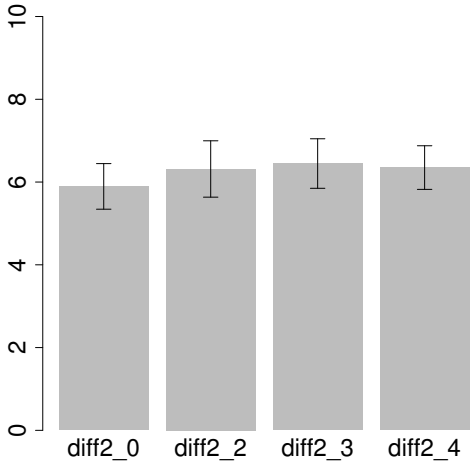


(a) Taux d'erreur

(b) Temps de réponse



(c) longueur de chemin



(d) nombre de sélection

Figure C.22: Résultats pour chaque encodage pour la densité 0.07

7.3 Résultats pour chaque encodage pour la densité 0.1

— densité 1 (0.04) : graphe n100k10

— densité 2 (0.07) : graphe n50k5 et n50k10

— densité 3 (0.1) : graphe n100k5

Seuil de significativité $\alpha = 0.025$.

Tableau C.24: Résultats pour chaque encodage pour la densité 0.1

densité	encodage	taux d'erreur	temps de réponse	temps de réponse min.	temps de réponse max.	longueur de chemin	nombre de sélection
dens. 3	0	0.175	40.832 (31.221)	12.145	122.264	4.676 (0.789)	7.382 (4.882)
dens. 3	2	0.025	46.978 (32.435)	8.584	131.440	5.050 (0.705)	7.250 (3.054)
dens. 3	3	0.075	42.814 (26.183)	9.680	94.558	5.100 (0.954)	7.400 (3.564)
dens. 3	4	0.125	43.086 (34.594)	8.288	133.069	5.142 (1.029)	6.717 (2.395)

Les valeurs indiquées sont des moyennes avec l'écart-type entre parenthèse (sauf pour le taux d'erreur et les valeurs min. et max.)

Taux d'erreur

```
Kruskal-Wallis rank sum test

data:  error by view_dens3
Kruskal-Wallis chi-squared = 3.2428, df = 3, p-value = 0.3557
```

Temps de réponse

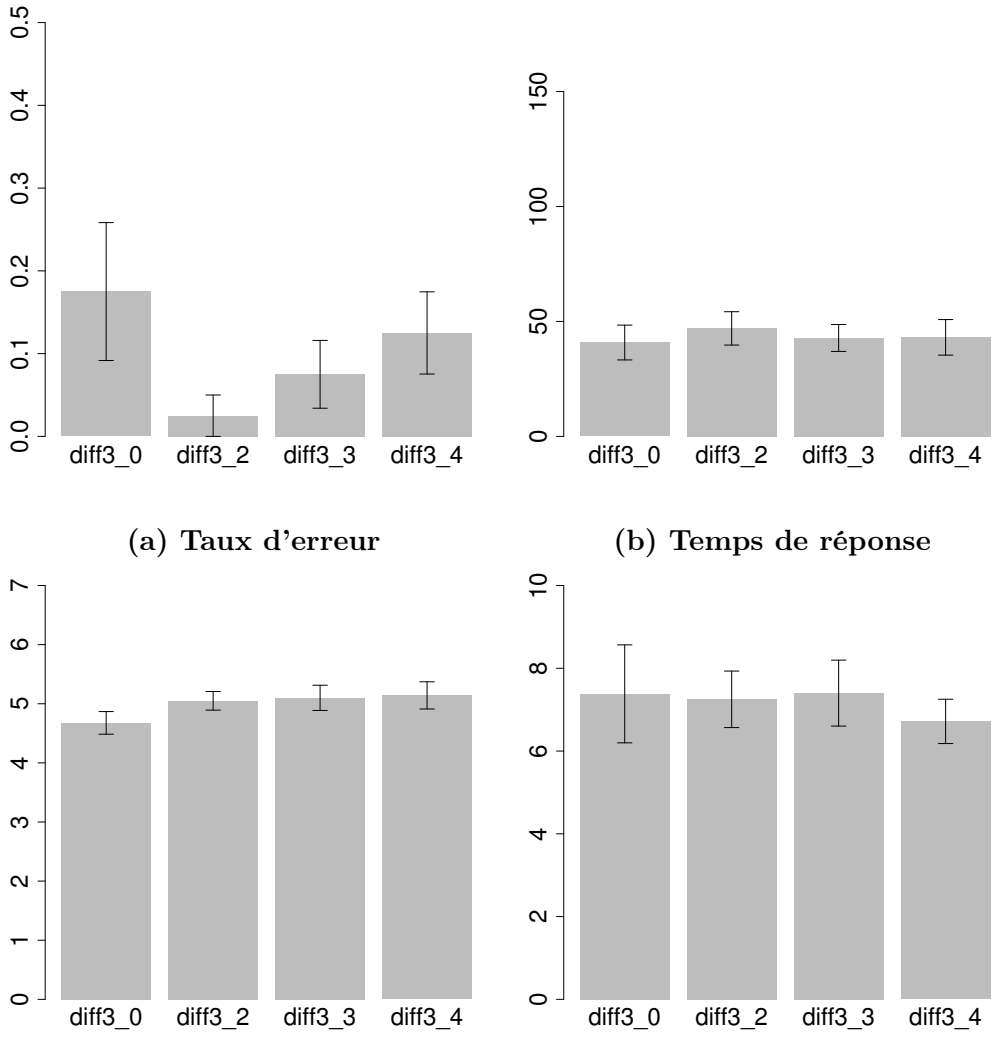
```
Kruskal-Wallis rank sum test

data:  time by view_dens3
Kruskal-Wallis chi-squared = 0.7782, df = 3, p-value = 0.8547
```

longueur de chemin

```
Kruskal-Wallis rank sum test

data:  length by view_diff3
Kruskal-Wallis chi-squared = 3.4776, df = 3, p-value = 0.3237
```

nombre de sélection

```
Kruskal-Wallis rank sum test

data:  selection by view_diff3
Kruskal-Wallis chi-squared = 1.3062, df = 3, p-value = 0.7277
```



(a) Taux d'erreur **(b) Temps de réponse**

(c) longueur de chemin **(d) nombre de sélection**

Figure C.23: Résultats pour chaque encodage pour la densité 0.1

Bibliographie

[1] Abello, J. et Korn, J., 2002. MGV : a system for visualizing massive multidigraphs. *IEEE Transactions on Visualization and Computer Graphics*, 8(1) :21–38. doi :10.1109/2945.981849.

[2] Abello, J. et van Ham, F., 2004. Matrix Zoom : A Visual Interface to Semi-External Graphs. Dans *IEEE Symposium on Information Visualization, 2004. INFOVIS 2004*, pages 183–190. doi :10.1109/INFVIS.2004.46.

[3] Abello, J., van Ham, F. et Krishnan, N., 2006. ASK-GraphView : A Large Scale Graph Visualization System. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :669–676. doi :10.1109/TVCG.2006.120.

[4] Abello, James, Kobourov, Stephen G. et Yusufov, Roman, 2004. Visualizing Large Graphs with Compound-fisheye Views and Treemaps. Dans *Proceedings of the 12th International Conference on Graph Drawing*, GD'04, pages 431–441. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-24528-6 978-3-540-24528-5. doi :10.1007/978-3-540-31843-9_44.
URL `http://dx.doi.org/10.1007/978-3-540-31843-9_44`

[5] Alemasoom, Haleh, Samavati, Faramarz F., Brosz, John et Layzell, David, 2014. Interactive Visualization of Energy System. Dans *2014 International Conference on Cyberworlds*. Institute of Electrical and Electronics Engineers (IEEE). doi :10.1109/cw.2014.39.
URL `https://doi.org/10.1109%2Fcw.2014.39`

[6] Alper, Basak, Bach, Benjamin, Henry Riche, Nathalie, Isenberg, Tobias et Fekete, Jean-Daniel, 2013. Weighted Graph Comparison Techniques for Brain Connectivity Analysis. Dans *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 483–492. ACM, New York, NY, USA. ISBN 978-1-4503-1899-0. doi :10.1145/2470654.2470724.
URL `http://doi.acm.org/10.1145/2470654.2470724`

[7] Alpern, Bowen et Carter, Larry, 1991. The Hyperbox. Dans *Proceedings of the 2Nd Conference on Visualization '91*, VIS '91, pages 133–139.

IEEE Computer Society Press, Los Alamitos, CA, USA. ISBN 0-8186-2245-8.
URL `http://dl.acm.org/citation.cfm?id=949607.949629`

[8] Andrienko, G. et Andrienko, N., 2004. Parallel coordinates for exploring properties of subsets. Dans *Second International Conference on Coordinated and Multiple Views in Exploratory Visualization, 2004. Proceedings*, pages 93–104. doi :10.1109/CMV.2004.1319530.

[9] Archambault, D., Munzner, T. et Auber, D., 2007. TopoLayout : Multilevel Graph Layout by Topological Features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2) :305–317. doi :10.1109/TVCG.2007.46.

[10] Archambault, D., Munzner, T. et Auber, D., 2008. GrouseFlocks : Steerable Exploration of Graph Hierarchy Space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4) :900–913. doi :10.1109/TVCG.2008.34.

[11] Archambault, Daniel, Munzner, Tamara et Auber, David, 2007. Grouse : Feature-Based, Steerable Graph Hierarchy Exploration. Dans K. Museth, T. Moeller et A. Ynnerman, rédacteurs, *Eurographics/ IEEE-VGTC Symposium on Visualization*. The Eurographics Association. ISBN 978-3-905673-45-6. doi :10.2312/VisSym/EuroVis07/067-074.

[12] Artero, A. O., De Oliveira, M. C. F. et Levkowitz, H., 2004. Uncovering Clusters in Crowded Parallel Coordinates Visualizations. Dans *IEEE Symposium on Information Visualization, 2004. INFOVIS 2004*, pages 81–88. doi :10.1109/INFVIS.2004.68.

[13] Auber, D., Huet, C., Lambert, A., Renoust, B., Sallaberry, A. et Saulnier, A., 2013. GosperMap : Using a Gosper Curve for Laying Out Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 19(11) :1820–1832. doi :10.1109/TVCG.2013.91.

[14] Auber, David, 2002. *Outils de visualisation de larges structures de données*. Thèse de doctorat, Bordeaux 1.

[15] Bae, J. et Watson, B., 2011. Developing and Evaluating Quilts for the Depiction of Large Layered Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(12) :2268–2275. doi :10.1109/TVCG.2011.187.

[16] Balzer, Michael et Deussen, Oliver, 2007. Level-of-detail visualization of clustered graph layouts. Dans *Visualization, 2007. APVIS'07. 2007 6th International Asia-Pacific Symposium on*, pages 133–140. IEEE.

[17] Balzer, Michael, Deussen, Oliver et Lewerentz, Claus, 2005. Voronoi treemaps for the visualization of software metrics. Dans *Proceedings of the 2005 ACM symposium on Software visualization*, pages 165–172. ACM.

[18] Battista, Di, Eades, Peter, Tollis, Ioannis G et Tamassia, Roberto, 1999. *Graph drawing : algorithms for the visualization of graphs*.

[19] Bendix, F., Kosara, R. et Hauser, H., 2005. Parallel sets : visual analysis of categorical data. Dans *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, pages 133–140. doi :10.1109/INFVIS.2005.1532139.

[20] Bertin, Jacques, 1977. *La graphique et le traitement graphique de l'information*. 91 (084.21) BER.

[21] Bertini, Enrico et Santucci, Giuseppe, 2006. Give chance a chance : modeling density to enhance scatter plot quality through random data sampling. *Information Visualization*, 5(2) :95–110.

[22] Bikakis, Nikos, Liagouris, John, Krommyda, Maria, Papastefanatos, George et Sellis, Timos K., 2016. graphVizdb : A Scalable Platform for Interactive Large Graph Visualization. *CoRR*, abs/1602.06401.

[23] Blondel, V. D., Guillaume, J.-L., Lambiotte, R. et Lefebvre, E., 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 10 :10008. doi :10.1088/1742-5468/2008/10/P10008.

[24] Bruls, Mark, Huizing, Kees et Van Wijk, Jarke J., 2000. Squarified treemaps. Dans *Data Visualization 2000*, pages 33–42. Springer.

[25] Chan, Winnie Wing-Yi, 2006. A survey on multivariate data visualization. *Department of Computer Science and Engineering. Hong Kong University of Science and Technology*, 8(6) :1–29.

[26] Chang, Fay, Dean, Jeffrey, Ghemawat, Sanjay, Hsieh, Wilson C., Wallach, Deborah A., Burrows, Mike, Chandra, Tushar, Fikes, Andrew et Gruber, Robert E., 2008. Bigtable : A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2) :4 :1–4 :26. doi :10.1145/1365815.1365816.
URL `http://doi.acm.org/10.1145/1365815.1365816`

[27] Childs, Hank et Miller, Mark, 2006. Beyond meat grinders : An analysis framework addressing the scale and complexity of large data sets. *Simulation Series*, 38(1) :181.

[28] Claessen, J. H. T. et Van Wijk, Jarke J., 2011. Flexible Linked Axes for Multivariate Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12) :2310–2316. doi :10.1109/TVCG.2011.201.

[29] Collberg, Christian, Kobourov, Stephen, Nagra, Jasvir, Pitts, Jacob et Wampler, Kevin, 2003. A system for graph-based visualization of the evolution of software. Dans *Proceedings of the 2003 ACM symposium on Software visualization*, pages 77–ff. ACM.

[30] Cox, Michael et Ellsworth, David, 1997. Application-controlled Demand Paging for Out-of-core Visualization. Dans *Proceedings of the 8th Conference on Visualization '97*, VIS '97, pages 235–ff. IEEE Computer Society Press, Los Alamitos, CA, USA. ISBN 1-58113-011-2.
URL `http://dl.acm.org/citation.cfm?id=266989.267068`

[31] Dwyer, T., Marriott, K., Schreiber, F., Stuckey, P., Woodward, M. et Wybrow, M., 2008. Exploration of Networks using overview+detail with Constraint-based cooperative layout. *IEEE Transactions on Visualization and Computer Graphics*, 14(6) :1293–1300. doi : 10.1109/TVCG.2008.130.

[32] Dwyer, Tim, Marriott, Kim et Wybrow, Michael, 2008. Topology Preserving Constrained Graph Layout. Dans Ioannis G. Tollis et Maurizio Patrignani, rédacteurs, *Graph Drawing*, numéro 5417 dans Lecture Notes in Computer Science, pages 230–241. Springer Berlin Heidelberg.
URL `http://link.springer.com/chapter/10.1007/978-3-642-00219-9_22`

[33] Eades, Peter et Feng, Qing-Wen, 1996. Multilevel visualization of clustered graphs. Dans Stephen North, rédacteur, *Graph Drawing*, numéro 1190 dans Lecture Notes in Computer Science, pages 101–112. Springer Berlin Heidelberg.
URL `http://link.springer.com/chapter/10.1007/3-540-62495-3_41`

[34] Eldawy, A., Mokbel, M. F., Alharthi, S., Alzaidy, A., Tarek, K. et Ghani, S., 2015. SHAHED : A MapReduce-based system for querying and visualizing spatio-temporal satellite data. Dans *2015 IEEE 31st International Conference on Data Engineering*, pages 1585–1596. doi :10.1109/ICDE.2015.7113427.

[35] Eldawy, Ahmed et Mokbel, Mohamed F., 2013. A Demonstration of SpatialHadoop : An Efficient Mapreduce Framework for Spatial Data. *Proc. VLDB Endow.*, 6(12) :1230–1233. doi :10.14778/2536274.2536283.
URL `http://dx.doi.org/10.14778/2536274.2536283`

[36] Elmqvist, N., Do, Thanh-Nghi, Goodell, H., Henry, N. et Fekete, J., 2008. ZAME : Interactive Large-Scale Graph Visualization. Dans *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, pages 215–222. doi :10.1109/PACIFICVIS.2008.4475479.

[37] Elmqvist, N., Dragicevic, P. et Fekete, J., 2008. Rolling the Dice : Multidimensional Visual Exploration using Scatterplot Matrix Navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6) :1539–1148. doi :10.1109/TVCG.2008.153.

[38] Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg et Xu, Xiaowei, 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press.

[39] Fekete, J, Wang, D, Dang, Niem, Aris, Aleks et Plaisant, Catherine, 2003. Interactive poster : Overlaying graph links on treemaps. Dans *Proceedings of the IEEE Symposium on Information Visualization Conference Compendium (InfoVis' 03)*, pages 82–83. Citeseer.

[40] Fisher, Danyel, Popov, Igor, Drucker, Steven et schraefel, m.c., 2012. Trust Me, I'M Partially Right : Incremental Visualization Lets Analysts Explore Large Datasets Faster. Dans *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1673–1682. ACM, New York, NY, USA. ISBN 978-1-4503-1015-4. doi : 10.1145/2207676.2208294.
URL `http://doi.acm.org/10.1145/2207676.2208294`

[41] Foundation, Apache Software, 2011. Welcome to Apache™ Hadoop®!
URL `hadoop.apache.org`

[42] Fua, Ying-Huey, Ward, M. O. et Rundensteiner, E. A., 1999. Hierarchical parallel coordinates for exploration of large datasets. Dans *Visualization '99. Proceedings*, pages 43–508. doi :10.1109/VISUAL.1999.809866.

[43] Fua, Ying-Huey, Ward, Matthew O. et Rundensteiner, Elke A., 1999. Navigating Hierarchies with Structure-Based Brushes. Dans *Proceedings of the 1999 IEEE Symposium on Information Visualization*, INFOVIS '99, pages 58–. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-0431-0.
URL `http://dl.acm.org/citation.cfm?id=857189.857662`

[44] Gajer, Pawel et Kobourov, Stephen G., 2000. GRIP : Graph dRawing with Intelligent Placement. Dans Joe Marks, rédacteur, *Graph Drawing*, numéro 1984 dans Lecture Notes in Computer Science, pages 222–228. Springer Berlin Heidelberg. ISBN 978-3-540-41554-1 978-3-540-44541-8.

URL http://link.springer.com/chapter/10.1007/3-540-44541-2_21

[45] GANNETTS, Henry, 1880. General Summary Showing the Rank of States by Ratios.

[46] GANSNER, Emden, KOREN, Yehuda et NORTH, Stephen, 2004. Topological Fisheye Views for Visualizing Large Graphs. Dans *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pages 175–182. IEEE Computer Society, Washington, DC, USA. ISBN 0-7803-8779-3. doi :10.1109/INFOVIS.2004.66.
URL http://dx.doi.org/10.1109/INFOVIS.2004.66

[47] GANSNER, Emden R. et NORTH, Stephen C., 2000. An open graph visualization system and its applications to software engineering. *Software : Practice and Experience*, 30(11) :1203–1233. doi :10.1002/1097-024X(200009)30:11<1203::AID-SPE338>3.0.CO;2-N.
URL http://onlinelibrary.wiley.com/doi/10.1002/1097-024X(200009)30:11<1203::AID-SPE338>3.0.CO;2-N/abstract

[48] GAREY, Michael, 1979. *Computers and intractability : a guide to the theory of Np-completeness*. W.H. Freeman, San Francisco. ISBN 0-7167-1045-5.

[49] GAREY, Michael R et JOHNSON, David S, 1983. Crossing number is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3) :312–316.

[50] GHONIEM, M., FEKETE, J. et CASTAGLIOLA, P., 2004. A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. Dans *IEEE Symposium on Information Visualization, 2004. INFOVIS 2004*, pages 17–24. doi :10.1109/INFVIS.2004.1.

[51] GHONIEM, Mohammad, FEKETE, Jean-Daniel et CASTAGLIOLA, Philippe, 2005. On the Readability of Graphs Using Node-Link and Matrix-Based Representations : Controlled Experiment and Statistical Analysis. *Information Visualization*, 4(2) :114–135.

[52] GRINSTEIN, Georges, TRUTSCHL, Marjan et CVEK, Urška, 2002. High-Dimensional Visualizations.

[53] HACHUL, Stefan et JÜNGER, Michael, 2005. Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm. Dans János Pach, rédacteur, *Graph Drawing : 12th International Symposium, GD 2004, New York, NY, USA, September 29-October 2, 2004, Revised Selected Papers*, pages 285–295. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-31843-9.
URL http://dx.doi.org/10.1007/978-3-540-31843-9_29

[54] Ham, F. van, 2003. Using multilevel call matrices in large software projects. Dans *IEEE Symposium on Information Visualization, 2003. INFOVIS 2003*, pages 227–232. doi :10.1109/INFVIS.2003.1249030.

[55] Harel, David et Koren, Yehuda, 2002. Graph Drawing by High-Dimensional Embedding. Dans Michael T. Goodrich et Stephen G. Kobourov, rédacteurs, *Graph Drawing*, numéro 2528 dans Lecture Notes in Computer Science, pages 207–219. Springer Berlin Heidelberg. ISBN 978-3-540-00158-4 978-3-540-36151-0.
URL `http://link.springer.com/chapter/10.1007/3-540-36151-0_20`

[56] Heer, J. et Robertson, G., 2007. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1240–1247. doi :10.1109/TVCG.2007.70539.

[57] Heinrich, Julian et Weiskopf, Daniel, 2013. State of the Art of Parallel Coordinates. Dans M. Sbert et L. Szirmay-Kalos, rédacteurs, *Eurographics 2013 - State of the Art Reports*. The Eurographics Association. doi : 10.2312/conf/EG2013/stars/095-116.

[58] Henry, Nathalie, Bezerianos, Anastasia et Fekete, Jean-Daniel, 2008. Improving the Readability of Clustered Social Networks using Node Dulication. *IEEE Transactions on Visualization and Computer Graphics*, 14 :1317–1324.

[59] Henry, Nathalie et Fekete, Jean-Daniel, 2006. MatrixExplorer : a Dual-Representation System to Explore Social Networks. Dans A. C. M. Press, rédacteur, *IHM 2006*, Proceedings of the 18th international conference on Association Francophone d'Interaction Homme-Machine, pages 67 – 74. AFIHM, Montréal, Canada.

[60] Henry, Nathalie et Fekete, Jean-Daniel, 2007. MatLink : Enhanced Matrix Visualization for Analyzing Social Networks. Dans Cécilia Baranauskas, Philippe Palanque, Julio Abascal et Simone Diniz Junqueira Barbosa, rédacteurs, *Human-Computer Interaction - INTERACT 2007*, tome 4663 de *Lecture Notes in Computer Science*, pages 288–302. Springer, Rio de Janeiro, Brésil. doi :10.1007/978-3-540-74800-7_24.

[61] Henry, Nathalie, Fekete, Jean-Daniel et McGuffin, Michael J., 2007. NodeTrix : A Hybrid Visualization of Social Networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6) :1302–1309. doi :10.1109/TVCG.2007.70582.

[62] Herman, I, 2000. Graph visualization and navigation in information visualization : A survey. IEEE Trans Vis Comput Graph. *Visualization*

and Computer Graphics, IEEE Transactions on, 6(1) :24 – 43. doi : 10.1109/2945.841119.

[63] Hinneburg, Alexander, Hinneburg, Er et Keim, Daniel A., 1998. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. pages 58–65. AAAI Press.

[64] Hlawatsch, M., Burch, M. et Weiskopf, D., 2014. Visual Adjacency Lists for Dynamic Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 20(11) :1590–1603. doi :10.1109/TVCG.2014.2322594.

[65] Hoffman, Patrick et Grinstein, Georges, 1997. *Visualizations for high dimensional data mining-table visualizations*. Citeseer.

[66] Holliman, Nick et Watson, Paul, 2015. Scalable Real-Time Visualization Using the Cloud. *IEEE Cloud Computing*, 2(6) :90–96.

[67] Holten, D., 2006. Hierarchical Edge Bundles : Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :741–748. doi :10.1109/TVCG.2006.147.

[68] Holten, Danny et van Wijk, Jarke J., 2009. A User Study on Visualizing Directed Edges in Graphs. Dans *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2299–2308. ACM, New York, NY, USA. ISBN 978-1-60558-246-7. doi :10.1145/1518701.1519054.
URL `http://doi.acm.org/10.1145/1518701.1519054`

[69] Huang, Mao Lin et Eades, Peter, 1998. A fully animated interactive system for clustering and navigating huge graphs. Dans *International Symposium on Graph Drawing*, pages 374–383. Springer.

[70] Im, J. F., Villegas, F. G. et McGuffin, M. J., 2013. VisReduce : Fast and responsive incremental information visualization of large datasets. Dans *2013 IEEE International Conference on Big Data*, pages 25–32. doi :10.1109/BigData.2013.6691710.

[71] Inselberg, A et Dimsdale, B, 1990. Parallel coordinates : a tool for visualizing multi-dimensional geometry. *San Francisco CA*, pages 361–375.

[72] Inselberg, Alfred, 1985. The plane with parallel coordinates. *The Visual Computer*, 1(2) :69–91.

[73] Jacobs, Adam, 2009. The pathologies of big data. *Communications of the ACM*, 52(8) :36–44.

[74] Johansson, J., Ljung, P., Jern, M. et Cooper, M., 2005. Revealing structure within clustered parallel coordinates displays. Dans *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, pages 125–132. doi :10.1109/INFVIS.2005.1532138.

[75] Johnson, B. et Shneiderman, B., 1991. Tree-maps : a space-filling approach to the visualization of hierarchical information structures. Dans *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, pages 284–291. doi :10.1109/VISUAL.1991.175815.

[76] Jonker, David, Langevin, Scott, Giesbrecht, David, Crouch, Michael et Kronenfeld, Nathan, 2016. Graph mapping : Multi-scale community visualization of massive graph data. *Information Visualization*, page 1473871616661195.

[77] Jugel, Uwe, Jerzak, Zbigniew, Hackenbroich, Gregor et Markl, Volker, 2014. Faster Visual Analytics through Pixel-Perfect Aggregation - p1705-jugel.pdf. Dans *Proceedings of the 40th International Conference on Very Large Data Bases*, tome 7, pages 1705 – 1708. Hangzhou, China.
URL `http://www.vldb.org/pvldb/vol7/p1705-jugel.pdf`

[78] Jugel, Uwe, Jerzak, Zbigniew, Hackenbroich, Gregor et Markl, Volker, 2014. M4 : a visualization-oriented time series data aggregation. *Proceedings of the VLDB Endowment*, 7(10) :797–808.

[79] Jugel, Uwe, Jerzak, Zbigniew, Hackenbroich, Gregor et Markl, Volker, 2015. VDDA : automatic visualization-driven data aggregation in relational databases. *The VLDB Journal*, 25(1) :53–77. doi : 10.1007/s00778-015-0396-z.
URL `http://link.springer.com/article/10.1007/s00778-015-0396-z`

[80] Juvan, Martin et Mohar, Bojan, 1992. Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics*, 36(2) :153–168. doi :10.1016/0166-218X(92)90229-4.
URL `http://www.sciencedirect.com/science/article/pii/0166218X92902294`

[81] Keller, René, Eckert, Claudia M. et Clarkson, P. John, 2006. Matrices or Node-link Diagrams : Which Visual Representation is Better for Visualising Connectivity Models ? *Information Visualization*, 5(1) :62–76. doi :10.1057/palgrave.ivs.9500116.

[82] Kirkpatrick, Scott, Jr, D. Gelatt et Vecchi, Mario P., 1983. Optimization by Simmulated Annealing. *Science*, 220(4598) :671–680.

[83] Koren, Yehuda, Carmel, L. et Harel, D., 2002. ACE : a fast multiscale eigenvectors computation for drawing huge graphs. Dans *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002*, pages 137–144. doi :10.1109/INFVIS.2002.1173159.

[84] Koren, Yehuda et Harel, David, 2002. A Multi-scale Algorithm for the Linear Arrangement Problem. Dans Gerhard Goos, Juris Hartmanis, Jan van Leeuwen et Luděk Kučera, rédacteurs, *Graph-Theoretic Concepts in Computer Science : 28th International Workshop, WG 2002 Český Krumlov, Czech Republic, June 13–15, 2002 Revised Papers*, pages 296–309. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-36379-8.
URL `http://dx.doi.org/10.1007/3-540-36379-3_26`

[85] Kornaropoulos, Evgenios M. et Tollis, Ioannis G., 2013. DAGView : An Approach for Visualizing Large Graphs. Dans Walter Didimo et Maurizio Patrignani, rédacteurs, *Graph Drawing : 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers*, pages 499–510. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-36763-2.
URL `http://dx.doi.org/10.1007/978-3-642-36763-2_44`

[86] Kosara, R., Bendix, F. et Hauser, H., 2006. Parallel Sets : interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4) :558–568. doi :10.1109/TVCG.2006.76.

[87] Kosara, R., Jankun-Kelly, T. J. et Chlan, E., 2007. *IEEE InfoVis 2007 Contest : InfoVis goes to the movies.*

[88] Lauther, Ulrich, 2006. Multipole-Based Force Approximation Revisited – A Simple but Fast Implementation Using a Dynamized Enclosing-Circle-Enhanced k-d-Tree. Dans Michael Kaufmann et Dorothea Wagner, rédacteurs, *Graph Drawing*, numéro 4372 dans Lecture Notes in Computer Science, pages 20–29. Springer Berlin Heidelberg.
URL `http://link.springer.com/chapter/10.1007/978-3-540-70904-6_4`

[89] Li, Jing, Martens, Jean-Bernard et van Wijk, Jarke J., 2008. Judging correlation from scatterplots and parallel coordinate plots. *Information Visualization*. doi :10.1057/palgrave.ivs.9500179.
URL `http://www.palgrave-journals.com/doifinder/10.1057/palgrave.ivs.9500179`

[90] Lind, M., Johansson, J. et Cooper, M., 2009. Many-to-Many Relational Parallel Coordinates Displays. Dans *Information Visualisation, 2009 13th International Conference*, pages 25–31. doi :10.1109/IV.2009.43.

[91] Lins, L., Klosowski, J. T. et Scheidegger, C., 2013. Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12) :2456–2465. doi :10.1109/TVCG.2013.179.

[92] Liu, Zhicheng, Jiang, Biye et Heer, Jeffrey, 2013. imMens : Real-time Visual Querying of Big Data. *Computer Graphics Forum*, 32(3pt4) :421–430. doi :10.1111/cgf.12129.
URL `http://onlinelibrary.wiley.com/doi/10.1111/cgf.12129/abstract`

[93] MacQueen, James, 1967. Some methods for classification and analysis of multivariate observations. Dans *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, tome 1, pages 281–297. Oakland, CA, USA.

[94] McCallum, Andrew, Nigam, Kamal et Ungar, Lyle H, 2000. Efficient clustering of high-dimensional data sets with application to reference matching. Dans *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM.

[95] McDonnell, K. T. et Mueller, K., 2008. Illustrative Parallel Coordinates. *Computer Graphics Forum*, 27(3) :1031–1038. doi :10.1111/j.1467-8659.2008.01239.x.
URL `http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2008.01239.x/abstract`

[96] Minard, Charles-Joseph, 1862. *Des Tableaux graphiques et des cartes figuratives, par M. Minard,...* Thunot.

[97] Muelder, C. et Ma, K. L., 2008. Rapid Graph Layout Using Space Filling Curves. *IEEE Transactions on Visualization and Computer Graphics*, 14(6) :1301–1308. doi :10.1109/TVCG.2008.158.

[98] Mueller, C., Martin, B. et Lumsdaine, A., 2007. A comparison of vertex ordering algorithms for large graph visualization. Dans *2007 6th International Asia-Pacific Symposium on Visualization*, pages 141–148. doi :10.1109/APVIS.2007.329289.

[99] Nakagawa, Shinichi, 2004. A farewell to Bonferroni : the problems of low statistical power and publication bias. *Behavioral Ecology*, 15(6) :1044–1045. doi :10.1093/beheco/arh107.

URL http://beheco.oxfordjournals.org/content/15/6/1044.short

[100] Neumann, Petra, Schlechtweg, Dr. Stefan et Carpendale, Sheelagh, 2005. ArcTrees : Visualizing Relations in Hierarchical Data. Dans Ken Brodlie, David Duke et Ken Joy, rédacteurs, *EUROVIS 2005 : Eurographics / IEEE VGTC Symposium on Visualization*. The Eurographics Association. ISBN 3-905673-19-3. doi :10.2312/VisSym/EuroVis05/053-060.

[101] Novotny, M. et Hauser, H., 2006. Outlier-Preserving Focus+Context Visualization in Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :893–900. doi :10.1109/TVCG.2006.170.

[102] Palmas, G., Bachynskyi, M., Oulasvirta, A., Seidel, H.P. et Weinkauf, T., 2014. An Edge-Bundling Layout for Interactive Parallel Coordinates. Dans *2014 IEEE Pacific Visualization Symposium (PacificVis)*, pages 57–64. doi :10.1109/PacificVis.2014.40.

[103] Perin, C., Dragicevic, P. et Fekete, J. D., 2014. Revisiting Bertin Matrices : New Interactions for Crafting Tabular Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 20(12) :2082–2091. doi :10.1109/TVCG.2014.2346279.

[104] Perrot, Alexandre, Bourqui, Romain, Hanusse, Nicolas, Lalanne, Frédéric et Auber, David, 2015. Large interactive visualization of density functions on big data infrastructure. Dans *Large Data Analysis and Visualization (LDAV), 2015 IEEE 5th Symposium on*, pages 99–106. IEEE.

[105] Petit, Jordi, 2003. Experiments on the Minimum Linear Arrangement Problem. *J. Exp. Algorithmics*, 8. doi :10.1145/996546.996554.
URL http://doi.acm.org/10.1145/996546.996554

[106] Press, Gil, 2013. A very short history of big data. *Forbes Tech Magazine, May*, 9.

[107] Purchase, Helen, 1997. Which aesthetic has the greatest effect on human understanding ? Dans Giuseppe DiBattista, rédacteur, *Graph Drawing : 5th International Symposium, GD '97 Rome, Italy, September 18–20, 1997 Proceedings*, pages 248–261. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-69674-2.
URL http://dx.doi.org/10.1007/3-540-63938-1_67

[108] Purchase, Helen C, 1998. The effects of graph layout. Dans *Computer Human Interaction Conference, 1998. Proceedings. 1998 Australasian*, pages 80–86. IEEE.

[109] Purchase, Helen C., 2012. *Experimental Human-Computer Interaction - A Practical Guide with Visual Examples*. Cambridge University Press. ISBN 978-0-521-27954-3.

[110] Purchase, Helen C., Cohen, Robert F. et James, Murray I, 1997. An experimental study of the basis for graph drawing algorithms. *Journal of Experimental Algorithmics (JEA)*, 2 :4.

[111] Rafiei, Davood, 2005. Effectively visualizing large networks through sampling. Dans *Visualization, 2005. VIS 05. IEEE*, pages 375–382. IEEE.

[112] Raitner, Marcus, 2005. Visual Navigation of Compound Graphs. Dans János Pach, rédacteur, *Graph Drawing : 12th International Symposium, GD 2004, New York, NY, USA, September 29-October 2, 2004, Revised Selected Papers*, pages 403–413. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-31843-9.
URL http://dx.doi.org/10.1007/978-3-540-31843-9_41

[113] Rider, Fremont, 1944. scholar and the future of the research library.

[114] Riehmann, P., Hanfler, M. et Froehlich, B., 2005. Interactive Sankey diagrams. Dans *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*, pages 233–240. doi :10.1109/INFVIS.2005.1532152.

[115] Rodriguez-Tello, Eduardo, Hao, Jin-Kao et Torres-Jimenez, Jose, 2008. An effective two-stage simulated annealing algorithm for the minimum linear arrangement problem. *Computers & Operations Research*, 35(10) :3331–3346. doi :10.1016/j.cor.2007.03.001.
URL http://www.sciencedirect.com/science/article/pii/S0305054807000676

[116] Rubel, O., Prabhat, Wu, Kesheng, Childs, H., Meredith, J., Geddes, C.G.R., Cormier-Michel, E., Ahern, S., Weber, G.H., Messmer, P., Hagen, H., Hamann, B. et Bethel, E.W., 2008. High performance multivariate visual data exploration for extremely large data. Dans *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–12. doi : 10.1109/SC.2008.5214436.

[117] Sagiroglu, Seref et Sinanc, Duygu, 2013. Big data : A review. Dans *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, pages 42–47. IEEE.

[118] Samet, Hanan, 1984. The Quadtree and Related Hierarchical Data Structures. *ACM Comput. Surv.*, 16(2) :187–260. doi :10.1145/356924.356930.
URL http://doi.acm.org/10.1145/356924.356930

[119] Sankey, HR, 1898. Introductory note on the thermal efficiency of steam-engines. Dans *Minutes of Proceedings of The Institution of Civil Engineers*, pages 278–283.

[120] Sansen, J., Bourqui, R., Pinaud, B. et Purchase, H., 2015. Edge Visual Encodings in Matrix-Based Diagrams. Dans *2015 19th International Conference on Information Visualisation (iV)*, pages 62–67. doi :10.1109/iV.2015.22.

[121] Sansen, J., Lalanne, F., Auber, D. et Bourqui, R., 2015. Adjasankey : Visualization of Huge Hierarchical Weighted and Directed Graphs. Dans *2015 19th International Conference on Information Visualisation (iV)*, pages 211–216. doi :10.1109/iV.2015.46.

[122] Santos, Selan dos et Brodlie, Ken, 2004. Gaining understanding of multivariate and multidimensional data through visualization. *Computers & Graphics*, 28(3) :311 – 325. doi : http://dx.doi.org/10.1016/j.cag.2004.03.013.
URL `http://www.sciencedirect.com/science/article/pii/S0097849304000251`

[123] Schaffer, Doug, Zuo, Zhengping, Greenberg, Saul, Bartram, Lyn, Dill, John, Dubs, Shelli et Roseman, Mark, 1996. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3(2) :162–188.

[124] Selassie, D., Heller, B. et Heer, J., 2011. Divided Edge Bundling for Directional Network Data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12) :2354–2363. doi :10.1109/TVCG.2011.190.

[125] Shen, Zeqian et Ma, Kwan-Liu, 2007. Path Visualization for Adjacency Matrices. Dans K. Museth, T. Moeller et A. Ynnerman, rédacteurs, *Eurographics/ IEEE-VGTC Symposium on Visualization*. The Eurographics Association. ISBN 978-3-905673-45-6. doi :10.2312/VisSym/EuroVis07/083-090.

[126] Shneiderman, Ben, 1992. Tree Visualization with Tree-maps : 2-d Space-filling Approach. *ACM Trans. Graph.*, 11(1) :92–99. doi :10.1145/102377.115768.
URL `http://doi.acm.org/10.1145/102377.115768`

[127] Six, Janet M. et Tollis, Ioannis G., 1999. A Framework for Circular Drawings of Networks. Dans Jan Kratochvíyl, rédacteur, *Graph Drawing*, numéro 1731 dans Lecture Notes in Computer Science, pages 107–116. Springer Berlin Heidelberg. ISBN 978-3-540-66904-3 978-3-540-46648-2.

URL http://link.springer.com/chapter/10.1007/3-540-46648-7_11

[128] Sugiyama, K. et Misue, K., 1991. Visualization of structural information : automatic drawing of compound digraphs. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(4) :876–892. doi :10.1109/21.108304.

[129] Tominski, Christian, Abello, James et Schumann, Heidrun, 2004. Axes-based visualizations with radial layouts. Dans *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1242–1247. ACM.

[130] Van Dongen, Stijn Marinus, 2000. *Graph clustering by flow simulation.* Thèse de doctorat, University of Utrecht.

[131] Van Ham, Frank et Van Wijk, Jarke J., 2004. Interactive visualization of small world graphs. Dans *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 199–206. IEEE.

[132] Van Long, Tran et Linsen, Lars, 2009. MultiClusterTree : Interactive Visual Exploration of Hierarchical Clusters in Multidimensional Multivariate Data. *Computer Graphics Forum*, 28(3) :823–830. doi : 10.1111/j.1467-8659.2009.01468.x.
URL http://dx.doi.org/10.1111/j.1467-8659.2009.01468.x

[133] Van Wijk, Jarke J. et Liere, R. van, 1993. HyperSlice. Dans *Visualization, 1993. Visualization '93, Proceedings., IEEE Conference on*, pages 119–125. doi :10.1109/VISUAL.1993.398859.

[134] Van Wijk, Jarke J et Van de Wetering, Huub, 1999. Cushion treemaps : Visualization of hierarchical information. Dans *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on*, pages 73–78. IEEE.

[135] Varley, Ian Thomas, Aziz, Adnan, Aziz, Co-supervisors Adnan et Miranker, Daniel, 2009. No relation : The mixed blessings of non-relational databases.

[136] Vehlow, C., Beck, F., Auwärter, P. et Weiskopf, D., 2015. Visualizing the Evolution of Communities in Dynamic Graphs. *Computer Graphics Forum*, 34(1) :277–288. doi :10.1111/cgf.12512.
URL http://onlinelibrary.wiley.com/doi/10.1111/cgf.12512/abstract

[137] Ward, Matthew, Grinstein, Georges et Keim, Daniel, 2010. Interactive Data Visualization : Foundations. *Techniques, and Applications, AK Peters, Ltd., Natick, MA*.

[138] Ware, Colin, 2012. *Information visualization : perception for design*. Elsevier.

[139] Ware, Colin, Purchase, Helen, Colpoys, Linda et McGill, Matthew, 2002. Cognitive Measurements of Graph Aesthetics. *Information Visualization*, 1(2) :103–110. doi :10.1057/palgrave.ivs.9500013.
URL `http://dx.doi.org/10.1057/palgrave.ivs.9500013`

[140] Watson, Benjamin, Brink, David, Stallmann, Matthias, Devarajan, Ravi, Rakow, Matthew, Rhyne, Theresa-Marie et Patel, Himesh, 2007. Visualizing very large layered graphs with quilts. Dans *IEEE Information Visualization Conference Poster*, tome 3.

[141] Wattenberg, Martin, 2002. Arc diagrams : Visualizing structure in strings. Dans *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 110–116. IEEE.

[142] Wattenberg, Martin, 2006. Visual exploration of multivariate graphs. Dans *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 811–819. ACM.

[143] Wegman, Edward J. et Luo, Qiang, 1997. High Dimensional Clustering Using Parallel Coordinates and the Grand Tour. *COMPUTING SCIENCE AND STATISTICS*, 28 :361–368.

[144] Wilkinson, Leland et Friendly, Michael, 2009. The History of the Cluster Heat Map. *The American Statistician*, 63(2) :179–184. doi : 10.1198/tas.2009.0033.
URL `http://amstat.tandfonline.com/doi/abs/10.1198/tas.2009.0033`

[145] Wong, Pak Chung, Crabb, Andrew H et Bergeron, R Daniel, 1996. Dual multiresolution HyperSlice for multivariate data visualization. Dans *infovis*, tome 96, page 74. Citeseer.

[146] Wongsuphasawat, K. et Gotz, D., 2012. Exploring Flow, Factors, and Outcomes of Temporal Event Sequences with the Outflow Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12) :2659–2668. doi :10.1109/TVCG.2012.225.

[147] Zaharia, Matei, Chowdhury, Mosharaf, Franklin, Michael J., Shenker, Scott et Stoica, Ion, 2010. Spark : Cluster Computing with Working Sets. Dans *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10. USENIX Association, Berkeley, CA, USA.
URL `http://dl.acm.org/citation.cfm?id=1863103.1863113`

[148] Zakai, Alon, 2011. Emscripten : An LLVM-to-JavaScript Compiler. Dans *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion*, OOPSLA '11, pages 301–312. ACM, New York, NY, USA. ISBN 978-1-4503-0942-4. doi :10.1145/2048147.2048224.
URL `http://doi.acm.org/10.1145/2048147.2048224`

[149] Zhang, Tian, Ramakrishnan, Raghu et Livny, Miron, 1996. BIRCH : An Efficient Data Clustering Method for Very Large Databases. Dans *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114. ACM, New York, NY, USA. ISBN 0-89791-794-4. doi :10.1145/233269.233324.
URL `http://doi.acm.org/10.1145/233269.233324`

[150] Zhao, Shengdong, McGuffin, M. J. et Chignell, M. H., 2005. Elastic hierarchies : combining treemaps and node-link diagrams. Dans *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 57–64. doi :10.1109/INFVIS.2005.1532129.

[151] Zhou, Hong, Cui, Weiwei, Qu, Huamin, Wu, Yingcai, Yuan, Xiaoru et Zhuo, Wei, 2009. Splatting the lines in parallel coordinates. Dans *Computer Graphics Forum*, tome 28, pages 759–766. Wiley Online Library.

[152] Zhou, Hong, Yuan, Xiaoru, Qu, Huamin, Cui, Weiwei et Chen, Baoquan, 2008. Visual clustering in parallel coordinates. *Computer Graphics Forum*, 27 :1047–1054.

[153] Zongker, Douglas E et Salesin, David H, 2003. On creating animated presentations. Dans *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 298–308. Eurographics Association.