Import automatique et interactif de données dans les systèmes de visualisations

Frédéric Gilbert*, David Auber*

*LaBRI, Université de Bordeaux, Talence, France
Gravité Inria Bordeaux Sud-Ouest
frederic.gilbert@labri.fr,
david.auber@labri.fr

Résumé. La première étape du processus de visualisation d'information consiste à transformer les données d'un format brut vers une structure de données utilisable par les différents composants de visualisation. Dans les applications réelles, cette première étape représente une barrière empêchant l'accès des utilisateurs novices à une riche variété de techniques de visualisation. Par exemple, il peut être techniquement impossible pour un utilisateur lambda de transformer des données arborescentes en un modèle de graphe pouvant utiliser une représentation à base de TreeMap. Une autre barrière est aussi la multitude de transformations possible des données brutes. Il faut pouvoir explorer cet ensemble de combinaisons. Basé sur nos retours d'expériences avec des utilisateurs finaux, dans cet article, nous considérons que le format brut est sous forme tabulaire. Ce format est le plus couramment utilisé et est facilement accessible par nos utilisateurs. Nous proposons une méthode novatrice permettant de générer automatiquement des graphes valués à partir de n'importe quelle table. En analysant le contenu de chaque dimension nous identifions les interconnexions entre celles-ci. Puis nous caractérisons les entités, les attributs et les relations possibles au sein des tables. Finalement, nous intégrons l'utilisateur dans le processus de transformation en lui proposant un ensemble de transformations valides.

1 Introduction

De nos jours, la collecte de données est de plus en plus aisée. Il existe un grand nombre d'outils et méthodes pour collecter et stocker facilement des données. Le plus souvent ces données sont stockées sous forme de base de données contenant des informations basiques comme des noms, des nombres, des dates, des heures... Elles sont stockées dans le but de garder des informations comme qui/quoi, quand, où, etc... Mais ces facilités de collecte et stockage ont pour conséquences de voir augmenter la quantité de grandes bases de données. Par exemple, les chercheurs sur le génome humain collectent des données dans le but de comprendre comment les gènes interagissent et les compagnies commerciales en collectent afin de mettre en évidence des comportements typiques ou atypiques de clients.

Pour répondre à ce type de questions, il existe des systèmes de visualisation comme DEVise Livny et al. (1997), Polaris Stolte et al. (2008) ou Tableau Mackinlay et al. (2007). Ces

systèmes suivent le processus d'analyse exploratoire défini par une hypothèse, puis une expérimentation et finalement une découverte. Mais avant cela, il est nécessaire de transformer les données d'un format brut vers une structure de données utilisable par les différents composants de visualisation de ces systèmes. Or cette étape représente une barrière empêchant l'accès des utilisateurs non experts à une riche variété de techniques de visualisation. Par exemple, il peut être techniquement impossible pour un utilisateur non expert de transformer des données arborescentes en un modèle de graphe pouvant utiliser une représentation à base de TreeMap Johnson et Shneiderman (1991). Mais en plus de cela, chaque système de visualisation utilise un modèle de graphe qui lui est propre donc l'utilisation des spécificités de chaque système demanderait de grandes capacités techniques aux utilisateurs non experts.

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

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

Donc en utilisant comme données d'entrée une table, nous proposons une méthode novatrice permettant de générer automatiquement des graphes valués à partir de n'importe quelle table. En analysant le contenu de chaque dimension nous identifions les interconnexions entre celles-ci. Puis nous caractérisons les entités, les attributs et les relations possibles au sein des tables. Finalement, nous intégrons l'utilisateur dans le processus en lui proposant un ensemble de transformation valide. L'utilisateur a la possibilité de modifier ces transformations avant de générer les graphes correspondants.

Cet article est structuré de la manière suivante : nous présentons tout d'abord des travaux traitant de l'import automatique de données dans les systèmes de visualisation. Puis nous décrivons notre méthode, et enfin nous concluons avec un cas d'étude.

2 Méthodes de générations de graphes

Afin de répondre aux principales attentes des utilisateurs de système de visualisation comme on peut le lire dans Smith et al. (2009), Mackinlay et al. (2007), et Slingsby et al. (2009), nous souhaitons leur fournir des visualisations simplifiées de leurs données. Pour cela nous mettons en place des méthodes de génération automatique de graphes. Nous présentons ici les principales techniques de génération de graphes. Si on étudie la littérature du domaine, trois méthodes pour générer des graphes se démarquent des autres pour en devenir les références.

La première que nous pouvons citer est Many Eyes Viegas et al. (2007). D'après l'équipe qui développe cette plateforme, il est important que plusieurs personnes puissent travailler sur les mêmes données et débattre sur les différentes visualisations générées à partir de celles-ci. Les auteurs expliquent qu'il est difficile de donner une bonne manière de visualiser un type de jeu de données. Il est possible que pour deux jeux de données, qui différent seulement d'une dimension, la manière de les visualiser soit radicalement différente. C'est pour cette raison que cette équipe a développé Many Eyes, un web service qui permet aux utilisateurs d'accéder à des jeux de données, de déposer leurs propres jeux de données et de créer des visualisations. Mais il ne s'agit là que de la première partie du processus, la seconde consistant à proposer à la discussion les visualisations générées dans le but de profiter de l'avis d'experts. Donc au lieu d'essayer de construire la "bonne" visualisation pour un jeu de données, Many Eyes préfère offrir l'opportunité d'avoir une visualisation élaborée par plusieurs experts.

La seconde méthode est celle développée par Mackinlay dans Tableau Software Mackinlay et al. (2007). Avec celle-ci, on peut facilement charger des données, les manipuler et créer des visualisations en combinant des dimensions des données. La force de cet outil est qu'il choisit à la place de l'utilisateur comment visualiser efficacement le résultat de la combinaison de certaines dimensions. Les tests combinatoires que cet outil fait sur les dimensions le rendent très intéressant. Il est capable de simplifier les données et d'y trouver des structures que les utilisateurs ne seraient pas capables de trouver par eux-même dans de très grandes tables. Mais ces tests combinatoires ne sont pas décrits.

La dernière méthode, développée par Smith dans nodeXl Software Smith et al. (2009), repose sur l'utilisation de feuilles de calcul. Les feuilles de calcul permettent, de par leur fonction première, de calculer des valeurs et bien d'autres informations en utilisant des formules logiques ou mathématiques. Ces valeurs serviront d'attribut pour les entités ou les relations. Mais cet outil permet aussi de construire un graphe à partir d'une feuille de calcul. Le graphe doit être décrit par une liste d'arêtes. La feuille de calcul doit contenir au moins deux colonnes, et chaque ligne décrit les deux entités (une dans chaque colonne) en relation. Donc cet outil fournit une méthode automatique qui ne nécessite pas l'utilisation d'un langage de programmation et qui en plus permet aux utilisateurs novices de produire une analyse visuelle de données relationnelles. Dans cette méthode, en raison de la création manuelle de la liste des arêtes, il n'est pas possible d'assurer que les utilisateurs non experts soient capables de construire un graphe valide.

Maintenant que nous avons présenté les trois grandes tendances de travail dans le domaine, nous présentons une précédente méthode que nous avons réalisée. Cette méthode Gilbert et Auber (2010) est inspirée du travail réalisé par Tableau Software. Comme les tests combinatoires qui y sont utilisés ne sont pas décrits, nous avons développé notre propre méthode combinatoire. Cette méthode vérifie l'existence des structures hiérarchiques entre les dimensions de la table et s'en sert pour la simplifier avant de générer les graphes que l'ont peut obtenir en

tenant compte de ces structures. Dans cette méthode, chaque ligne de la table décrit une entité et on cherche à savoir comment les attributs (valeurs contenues dans une ligne) hiérarchisent ces entités.

3 Notre méthode

Dans cette nouvelle méthode, nous considérons toujours que chaque cellule d'une table contient des données nominales. Ainsi, qu'elles contiennent un mot, un nombre ou une date, l'information sera traitée comme une chaîne de caractères sans distinction. Par contre, nous changeons d'approche en ce qui concerne la globalité des données contenues dans une table. Au lieu de considérer chaque ligne comme une entité décrite par des attributs, nous considérons ici que les entités sont des valeurs contenues dans les lignes. Par exemple, la table 1 contient des vols d'avions ayant une escale, donc plusieurs aéroports apparaissent dans chaque ligne de la table. Un utilisateur peut vouloir visualiser comment les aéroports sont inter-connectés. Ainsi le travail que nous réalisons ici se rapproche à la fois de celui réalisé dans Tableau Software de par l'analyse combinatoire que l'on va mener sur les dimensions de la table et également de celui réalisé dans nodeXl de par l'idée de chercher des listes d'arêtes. Toutefois, cette liste d'arêtes n'est pas clairement identifiée dans la table, et peut s'étendre sur plus de deux dimensions.

Identifiant vol	Départ	Escale	Terminus
1	Orly	Nantes	Bordeaux
2	Blagnac	Orly	Strasbourg
3	Bordeaux	Blagnac	Roissy

TAB. 1 – *Exemple d'une table regroupant des données sur des vols d'avions*

Nous allons décrire maintenant le processus que nous avons mis en place pour cette méthode. Celui-ci se décompose de la façon suivante : tout d'abord nous recherchons des intersections entre les dimensions de la table, puis proposons les intersections trouvées à l'utilisateur afin que celui-ci puisse éventuellement les modifier, et enfin une fois que l'utilisateur a validé les intersections et ses choix, nous générons les graphes entité-relation que l'utilisateur souhaite visualiser.

3.1 Recherche des intersections entre les dimensions

Afin de comparer les contenus des différentes dimensions et pour éviter d'avoir à parcourir à chaque fois toutes les valeurs d'une dimension, nous travaillons à partir des alphabets des dimensions. Nous définissons par alphabet l'ensemble des valeurs d'une dimension. Cela a pour avantage de supprimer les valeurs qui apparaissent de multiples fois dans une même dimension.
Par exemple, si les valeurs d'une dimension Dim sont : $\{A, B, D, A, E, G, F, F, B\}$. L'alphabet Σ_{Dim} associé à cette dimension sera : $\{A, B, D, E, F, G\}$.

La seconde étape consiste à déterminer s'il existe des intersections entre les dimensions. On définit qu'il existe une intersection entre deux dimensions s'il existe au moins une valeur

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

Par exemple si on souhaite déterminer s'il existe une intersection entre la dimension "Départ" et la dimension "Terminus" de la table1, il nous faut d'abord générer les alphabets de ces dimensions. On obtient ainsi : $\Sigma_{Depart} = \{Bordeaux, Blagnac, Orly\}$ et $\Sigma_{Terminus} = \{Bordeaux, Roissy, Strasbourg\}$. Puis nous devons générer un troisième alphabet à partir des deux précédents, ce qui donne :
$\Sigma_{Depart \cup Terminus} = \{Bordeaux, Blagnac, Orly, Roissy, Strasbourg\}$.
Et on peut vérifier qu'il existe bien une intersection entre ces deux dimensions puisque :
$|\Sigma_{Depart \cup Terminus}| = 5 < |\Sigma_{Depart}| + |\Sigma_{Terminus}| = 3 + 3 = 6$.

En procédant par paire de dimensions, il n'est pas évident de détecter s'il existe des intersections entre plus de deux dimensions. Par exemple, considérons trois dimensions Dim_1, Dim_2, Dim_3, il est possible qu'il y ait une intersection entre Dim_1 et Dim_2, une autre entre Dim_2 et Dim_3, mais qu'il n'y en ait pas entre Dim_1 et Dim_3. Alors afin de pouvoir stocker les informations sur les intersections, nous avons mis en place un graphe (graphe de concepts). Chaque dimension de la table est représentée dans le graphe par un sommet et à chaque fois que l'on trouve une intersection entre deux dimensions on ajoute une arête entre les deux sommets correspondant dans le graphe. Nous obtenons ainsi un graphe composé d'une ou plusieurs composantes connexes. Et nous sommes sûrs que toutes les dimensions appartenant à une même composante connexe possèdent des intersections et donc par extensions que toutes les valeurs contenues dans ces dimensions appartient à un même domaine.

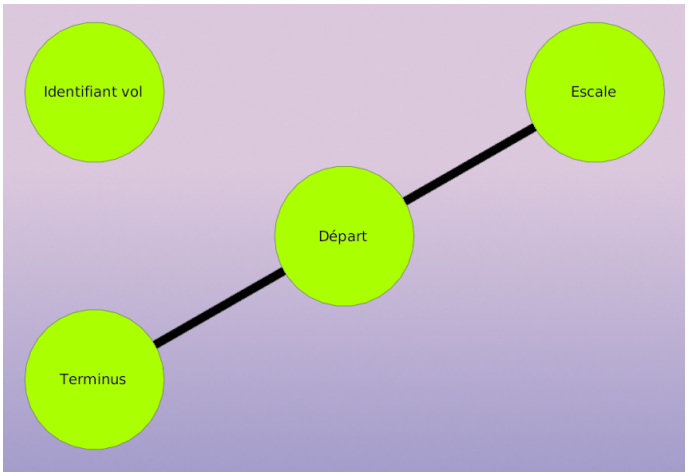

FIG. 1 – *Graphe des intersections des dimensions de la table 1*

Par exemple si nous considérons le cas de vols d'avions, chaque ligne de la table représente un vol caractérisé par les dimensions "Départ", "Terminus" et "Escale". Alors si nous trouvons des intersections entre ces dimensions, nous pouvons affirmer que ces dimensions contiennent des valeurs appartenant au même domaine, ici des aéroports. La figure 1 représente le graphe des intersections des dimensions de la table 1.

3.2 Propositions des intersections et interactions de l'utilisateur

Maintenant que les intersections entre les dimensions ont été mises en évidence, nous allons les soumettre à l'utilisateur pour qu'il puisse éventuellement les éditer. En effet, il est intéressant que l'utilisateur intervienne, car il se peut qu'en fonction de la répartition des valeurs dans les différentes dimensions de la table, des intersections n'aient pas étés détectées.

Par exemple si on considère la table 2, il n'y a une intersection qu'entre les dimensions "Départ" et "Escale". Pourtant si l'utilisateur souhaite visualiser les interconnexions entre les aéroports de la table, il faut également prendre en compte la dimension "Terminus" afin de définir le domaine "aéroport". Dans ce cas précis il est impossible de détecter l'inclusion de la dimension "Terminus" dans le domaine "aéroport".

Identifiant vol	Départ	Escale	Terminus
1	Orly	Nantes	Toulon
2	Blagnac	Orly	Strasbourg
3	Bordeaux	Blagnac	Roissy

TAB. 2 – *Exemple de table regroupant des donnés sur des vols d'avions où il n'y a pas d'intersections de la dimension "Terminus" avec les autres dimensions de la table.*

Il semble donc intéressant et indispensable pour construit des graphes les plus précis possibles, de tirer profit des connaissances de l'utilisateur. Ainsi, si deux dimensions ne possèdent pas d'intersection alors que manifestement elles désignent des entités de même type, il est important que l'utilisateur puisse spécifier cela. Dans notre exemple, les dimensions "Départ" et "Terminus" ne possèdent pas d'intersections, pourtant toutes deux renferment des valeurs d'aéroports. Il faut donc proposer à l'utilisateur d'ajouter cette information dans les ensembles d'intersections.

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

C'est en ce basant sur cette notion de limitation de la surcharge cognitive que nous avons choisi de mettre en place une interface qui permet à l'utilisateur d'éditer les intersections précédemment trouvées. Cette interface est une table dont chaque colonne représente une dimension de la table d'origine. Les lignes de cette table représentent les ensembles d'intersections. Afin

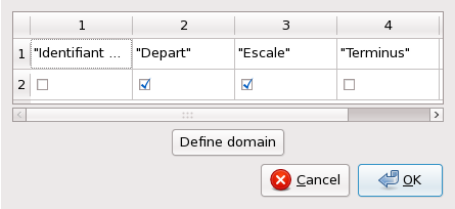

FIG. 2 – *Table permettant de visualiser les ensembles d'intersections des dimensions qui ont été trouvé lors de l'analyse de la table2. On peut voir qu'il n'y a pas d'intersection incluant la dimension "Terminus". Toutefois l'utilisateur va pouvoir corriger cela en cochant la boite correspondante.*

qu'on puisse visualiser dans une ligne l'ensemble qu'elle représente, nous avons doté chaque cellule de la table d'une boite cochable et chaque boite correspondant à une dimension qui apparait dans l'ensemble est cochée (cf figure 2).

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

Une fois que l'utilisateur a apporté toutes les modifications qu'il souhaite aux intersections, nous construisons un nouveau graphe similaire au premier que nous avons construit. Celui-ci va nous permettre de stocker les nouveaux domaines pour la suite de l'application de la méthode.

3.3 Génération des graphes entités-relation représentants les domaines

En utilisant le graphe obtenu après que l'utilisateur a apporté des modifications, nous sommes en mesure de mettre en évidence des ensembles d'entités. Ces ensembles d'entités sont identifiables grâce aux composantes connexes du graphe. En effet chaque composante connexe représente un ensemble de dimensions possédant des intersections entres elles et s'il existe un tel ensemble, alors les valeurs contenues dans ces dimensions définissent un ensemble d'entités. Donc pour chaque composante connexe de ce graphe, nous sommes capable de créer un nouveau graphe ayant comme ensemble de sommets les valeurs contenues dans les dimensions de cette composante. Maintenant, pour obtenir des graphes de types entité-relation, nous devons ajouter des arêtes à ce graphe.

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

Donc pour chaque ligne de la table nous allons ajouter une arête dans le graphe entre les valeurs des dimensions possédant des intersections. Par exemple avec la table 1 et le graphe des intersections de la figure 1, nous pouvons dire que le graphe que nous allons obtenir possédera

une arête entre les sommets "Orly" et "Nantes", une seconde entre "Orly" et "Bordeaux" ainsi qu'une troisième entre "Nantes" et "Bordeaux". En appliquant cela à chacune des lignes de la table 1, nous obtenons le graphe de la figure 3 dans lequel on retrouve les interconnexions entre les aéroports que l'utilisateur souhaitait visualiser.

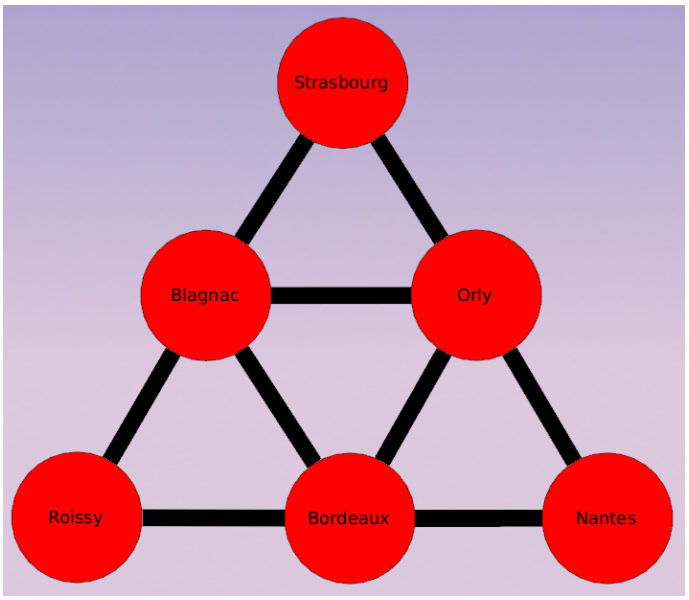

FIG. 3 – *Graphe des aéroports obtenu en tenant compte des intersections des dimensions de la table1*

3.4 Complexité

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

Nous exprimons ici les complexités en considérant que la table en entrée est composée de m dimensions et n lignes. Ainsi la construction de l'alphabet d'une dimension est en $\mathcal{O}(n \log n)$ en utilisant une structure ensembliste. Et par extension la construction de tous les alphabets est en $\mathcal{O}(m \times n \log n)$.

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

les alphabets initiaux de chaque dimension sont déjà construits, donc si on doit construire les premiers alphabets et les intersections, on obtient une complexité de $\mathcal{O}(m^2 \times n \log n)$.

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

Donc cette complexité est inférieure à celle de la construction des intersections. On peut donc dire que la complexité globale de notre méthode est de l'ordre de $\mathcal{O}(m^2 \times n \log n)$. Dans la plupart des jeux de données, le nombre de dimensions est bien inférieur au nombre de ligne, donc nous pouvons dire que notre méthode est applicable en pratique malgrès sa complexité quadratique.

4 Cas d'étude

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

	1	2	3	4	5	6	7	8
1	"Identifiant"	"Type de...	"Titre pu...	"Auteur 1"	"Auteur 2"	"Auteur 3"	"Annee"	"Titre ouvrage"
2	"inria-00472423"	"article jour...	"Animation, ...	"Archambau...	"Purchase H...	"Pinaud Bru...	2011	"IEEE Transactions on...
3	"inria-00516580"	"article conf...	"GVSR: an ...	"Pinaud Bru...	"Kuntz Pasc...	"NAN"	2010	"18th International S...
4	"inria-00514150"	"article conf...	"Difference ...	"Archambau...	"Purchase H...	"Pinaud Bru...	2010	"18th International S...
5	"hal-00520906"	"article conf...	"From Data...	"Gilbert Fred...	"Auber David"	"NAN"	2010	"Information Visualiz...
6	"hal-00495293"	"article conf...	"Living flow...	"Lambert A...	"Auber David"	"Melancon G...	2010	"Information Visualiz...
7	"hal-00495307"	"article conf...	"3D Edge Bu...	"Lambert A...	"Bourqui Ro...	"Auber David"	2010	"Information Visualiz...
8	"hal-00495279"	"article conf...	"Winding Ro...	"Lambert A...	"Bourqui Ro...	"Auber David"	2010	"Computer Graphics ...
9	"inria-00471432"	"article conf...	"The readab...	"Archambau...	"Purchase H...	"Pinaud Bru...	2010	"Eurovis 2010, 12th a...
10	"inria-00449745"	"article conf...	"PORGY : re...	"Pinaud Bru...	"Melancon G...	"NAN"	2010	"Extraction et Gestio...
11	"hal-00499430"	"article conf...	"Interactive ...	"Koenig Pier...	"Zaidi Faraz "	"Archambau...	2010	"Proceedings of Grap...
12	"halshs-00466163"	"article jour...	"Ports in mu...	"Ducruet Ce...	"Rozenblat ...	"Zaidi Faraz "	2010	"Journal of Transport ...
13	"hal-00413602"	"article conf...	"Solving the...	"Simonetto ...	"Koenig Pier...	"Zaidi Faraz "	2009	"Proceedings of IEEE ...
14	"hal-00413602"	"article conf...	"Solving the...	"Archambau...	"Gilbert Fred...	"Phan Quan...	2009	"Proceedings of IEEE ...
15	"hal-00413602"	"article conf...	"Solving the...	"Mathiaut M...	"Lambert A...	"Dubois Jon...	2009	"Proceedings of IEEE ...
16	"hal-00413602"	"article conf...	"Solving the...	"Sicre Ronan"	"Vieux Remi"	"Melancon G...	2009	"Proceedings of IEEE ...
17	"hal-00407269"	"article conf...	"Fully Auto...	"Simonetto ...	"Auber David"	"Archambau...	2009	"Computer Graphics ...
18	"hal-00407218"	"article conf...	"An Heuristi...	"Simonetto ...	"Auber David"	"NAN"	2009	"Information Visualiz...

FIG. 4 – *Extrait de la table des publications.*

La méthode que nous présentons dans cet article est implémentée à l'aide de Tulip Auber (2003). Tulip est un logiciel libre qui fournit des outils et des composants graphiques pour la visualisations de grandes masses de données.

Lors de la première étape de notre méthode, nous cherchons les intersections entre les dimensions de la table. Il est évident que notre méthode devrait pouvoir mettre en évidence des intersections entre les différentes dimensions regroupant les noms des auteurs des publications. Le résultat de la recherche des intersections est présenté dans la figure 5, et celui-ci confirme bien les intersections attendues entre les dimensions "Auteur".

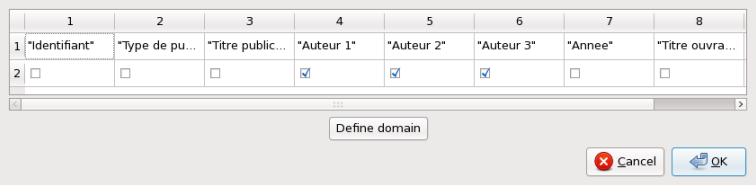

FIG. 5 – *Table permettant de visualiser les ensembles d'intersections des dimensions qui ont été trouvé lors de l'analyse de la table regroupant les publications. On peut voir il y a qu'une seule intersection incluant les dimensions "Auteur 1", "Auteur 2" et "Auteur 3".*

La seconde étape de notre méthode est la génération des graphes correspondant aux intersections. Dans notre cas, étant donné qu'une seule intersection n'a été trouvée, un seul graphe sera généré. Le graphe que nous obtenons dans le cas des auteurs est celui de la figure 6. Nous l'avons dessiné à l'aide d'un algorithme de dessin de force déjà implémenté dans Tulip.

Maintenant que nous avons importé et généré des graphes à partir de l'extraction de publication, nous pouvons entrer dans un processus d'analyse exploratoire. Ainsi si on observe le graphe obtenu, on peut voir qu'il existe des sommets particuliers qui sont des connecteurs dans le graphe. Par exemple, on peut voir dans la figure 6 que "Delest Maylis" permet de connecter deux parties du graphe. On peut donc conclure qu'elle joue un rôle important pour la cohésion de cette équipe. On peut retrouver des rôles similaires pour "Zaidi Faraz" et "Lambert Antoine". Une autre hypothèse que ce graphe nous permet de faire est que "Auber David" et "Melancon Guy" ont des positions très centrales dans cette équipe.

5 Conclusion

Nous avons présenté dans cet article une méthode générant de manière automatique des graphes à partir d'un jeu de données sous forme tabulaire. Cette méthode facilite la transformation des données afin de pouvoir les utiliser dans des systèmes de visualisation d'informations. Pour générer des graphes de type entité-relation nous tirons profit de l'expertise et des connaissances de l'utilisateur. Couplée à nos précédents travaux, cette méthode permet d'obtenir des graphes valués sur lesquels il est possible de procéder à une analyse exploratoire. Nous avons montré que la complexité de notre méthode reste applicable à de grands jeux de données. Nous avons aussi démontré au travers d'un cas d'étude l'utilité de notre méthode. En effet, en partant de données sur des publications, nous avons pu générer un graphe des auteurs qui peut s'avérer révélateur quant au rôle de certains auteurs au sein de la communauté.

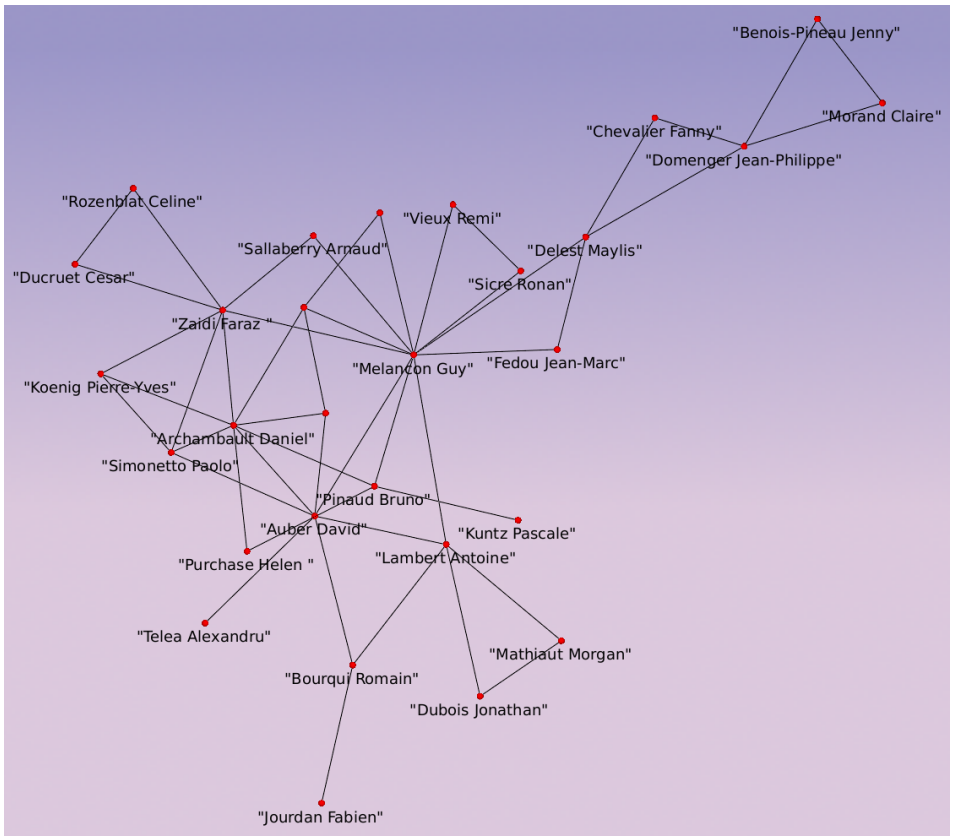

FIG. 6 – *Graphe entité-relation obtenu à partir des auteurs de la table de la figure 4.*

Références

Auber, D. (2003). Tulip : A huge graph visualisation framework. In P. Mutzel et M. Jünger (Eds.), *Graph Drawing Softwares*, Mathematics and Visualization, pp. 105–126. Springer-Verlag.

Gilbert, F. et D. Auber (2010). From Databases to Graph Visualization. In *Information Visualization 2010 14th International Conference Information Visualisation*, pp. 128–133.

Johnson, B. et B. Shneiderman (1991). Tree-maps : a space-filling approach to the visualization of hierarchical information structures. In *VIS '91 : Proceedings of the 2nd conference on Visualization '91*, Los Alamitos, CA, USA, pp. 284–291. IEEE Computer Society Press.

Livny, M., R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. MyllymÃd'ki, et K. Wenger (1997). Devise : Integrated querying and visual exploration of large datasets. In *In Proceedings of ACM SIGMOD*, pp. 301–312.

Mackinlay, J. D., P. Hanrahan, et C. Stolte (2007). Show me : Automatic presentation for visual analysis. Volume 13, pp. 1137–1144.

Slingsby, A., J. Dykes, et J. Wood (2009). Configuring hierarchical layouts to address research questions. *IEEE Trans. Vis. Comput. Graph. 15*(6), 977–984.

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

Stolte, C., D. Tang, et P. Hanrahan (2008). Polaris : a system for query, analysis, and visualization of multidimensional databases. *Commun. ACM 51*(11), 75–84.

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

Villerd, J., S. Ranwez, M. Crampes, D. Carteret, et J. M. Penalva (2007). Using concept lattices for visual navigation assistance in large databases : Application to a pantent database. In *CLA*.

Ware, C. (2000). *Information Visualization : Perception for Design*. Morgan Kaufmann Publishers.

Summary

The first step of any information visualization system is to enable end user to import their dataset into the system. However, non expert user are faced to the difficult task of choosing how their data should/could be transform to be used in these Infovis systems. In that paper we propose a novel method for automatic valuated graph generation from dataset in tabular format. Analysing each dimension of the table we highlight connections between them, we characterize entities, attributes and possible relations in tables. Finally, we include end users in the transformation process asking him to validate the set of transformations.