Séminaire d'informatique: Visualisation de l'Information Arbres Et Hiérarchies

Micheal El Betjali

DIVA
Département d'Informatique
Université de Fribourg, Suisse

Abstract

De nos jours, la taille d'une hiérarchie est de l'ordre de quelques milliers d'éléments, ce qui donne la naissance au problème suivant : comment afficher simultanément toute la hiérarchie sur un écran sans avoir besoin de grandir sa taille ? Cet article présente les résultats de différentes études qui permettent de visualiser toute une hiérarchie de taille importante dans une espace délimitée. En plus, elles permettent d'interagir avec les données et de les manipuler. Le but de cet article est de souligner comment ces approches présentent une hiérarchie et comment l'utilisateur interagit avec.

1. Introduction

Cet article fait partie du séminaire "Info visualization" proposé par le groupe DIVA de l'Université de Fribourg en Suisse. Dans la visualisation de l'information, la hiérarchie et l'arbre sont équivalents, car une hiérarchie être représentée facilement par un arbre cf. chapitre 3.

Les différentes études qui sont montionnées dans cet article, ont le même but qui est de relever le défit suivant : étant donné une place d'affichage délimitée, comment afficher tous les éléments d'un arbre de taille important sans déborder de la place d'affichage. L'affichage qui doit être le plus intuitif pour l'utilisateur, doit refléter tous les éléments et l'hiérarchie de l'arbre. Il est clair que la place d'affichage doit être utilisé efficacement.

L'interaction qui est une aspect important pour l'utilisateur, est exposée dans section 2. La section 3 explique des techniques basiques pour représenter une hiérarchie. Ensuite Les résultats des différentes études qui se basent sur ces techniques sont plus détaillés dans les sections 4-6. Finalement, la conclusion est présentée dans la section 7.

2. Interaction

Visualiser un arbre sans pouvoir interagir, apporte peu à l'utilisateur. L'interaction est un élément important, car elle permet à l'utilisateur de mieux comprendre la relation entre l'arbre et sa représentation sur l'écran. De plus elle fournit des fonctionnalités utiles pour l'utilisateur. L'interaction qui est proposée par les études présentées dans cet article, se regroupe généralement sous trois thèmes principaux.

Le premier est la manipulation de l'arbre qui permet d'ajouter, de supprimer ou de modifier un élément de l'arbre. Le second thème est la sélection d'un ou de plusieurs éléments de l'arbre selon certaine propriété propre à l'élément. La sélection peut être assurée par un ou plusieurs principes suivants :

- Brushing est la sélection d'un ou plusieurs éléments dans l'arbre selon un certain critère.
- Filtering est la sélection de plusieurs éléments selon un filtre, tous les éléments qui ont été rejetés par le filtre, ont été ôtés de la représentation de l'arbre.

Le dernier est la navigation, elle permet de visualiser l'arbre de différentes manières. Cette navigation peut être assurée par un ou plusieurs principes :

- Focus+context permet de focaliser sur un sous-arbre en gardant une vue générale de l'arbre.
- Drill down permet d'avoir seulement la vue d'un sous-arbre, il ôte de la représentation tous les éléments qui n'appartiennent pas au sous-arbre choisi.
- Drill up est l'action inverse de drill down.

3. Différentes techniques d'affichage

Une petite hiérarchie peut naturellement être représentée par un diagramme d'un arbre en 2D comme dans la figure 1, ce diagramme est le plus facile à comprendre. Cette manière de faire a une certaine limite, particulièrement quand l'hiérarchie devient de plus en plus grande. Un diagramme d'arbre peut représenter sans problème une hiérarchie ayant deux cents éléments. La faiblesse de cette technique est de ne pas utiliser efficacement la place d'affichage.

Le Treemap qui est une alternative efficace pour visualiser un arbre, a été développé par Shneiderman [2]. La figure 1 montre un simple exemple de Treemap.

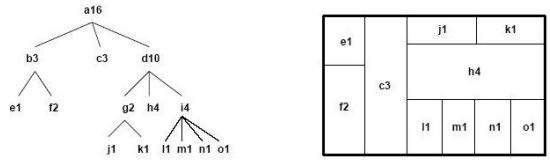


FIG. 1 – diagramme d'arbre et son Treemap

L'algorithme qui construit le Treemap, fonctionne comme suit : Un noeud qui n'a pas de fils, se figure par un rectangle. L'algorithme commence par le noeud racine qui est représenté comme un rectangle. Ensuite, il divise horizontalement le rectangle de la racine pour représenter ses files par des rectangles également. Les fils peuvent être vu comme des racines par rapport aux sous-arbres. L'algorithme recommence avec des petits fils, mais cette fois il divise chaque rectangle verticalement. L'algorithme répète cette procédure en alternant la division verticale et horizontale pour tous les éléments de l'arbre.

L'alternance de division horizontale et verticale indique le changement de niveau dans l'arbre. La taille de chaque rectangle reflète certaine propriété du noeud, par exemple le nombre de feuilles dans son sous-arbre. La figure 2 montre un Treemap qui emploie efficacement la place. Un Treemap peut présenter un arbre de 1400 noeuds. Mais il y a certaines zones où les rectangles sont tellement petits que cela rends l'interaction difficile.

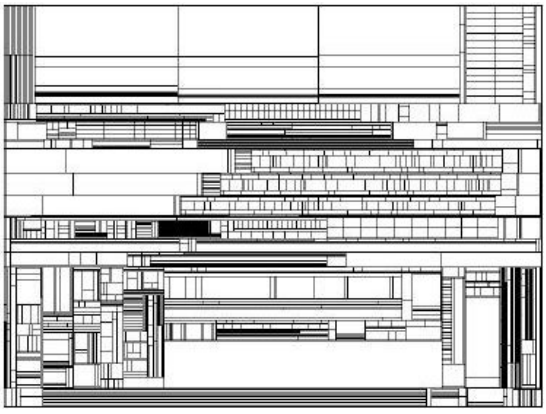


FIG. 2 – Treemap d'un système de fichiers contenant 1400 éléments

Le Treemap remplit la zone de représentation avec des rectangles pour utiliser l'espace efficacement. Il existe une autre technique de remplissage d'espace consiste à remplacer les rectangles par des couches circulaires à partir du centre. Cette technique s'appelle radial space filling(RSF) qui a donnée la naissance à plusieurs méthode pour représenter une hiérarchie.

La figure 3, qui montre les éléments de base, représente clairement une hiérarchie. Remarquons, que

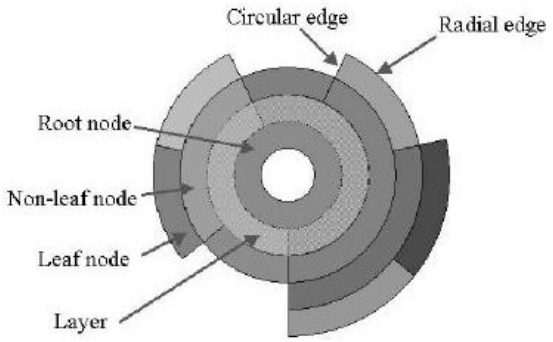


FIG. 3 – RSA : élément de base d'InterRing

plus les noeuds s'approche du centre du cercle plus ils sont proches de la racine. De plus le fils ne dépasse jamais l'angle qui est formé par le père. Cette manière montre la structure de l'arbre.

4. Treemaps

Cette section explique deux études qui se basent sur le technique de remplissage de l'espace avec des rectangles " Retangle Filling Space ". Shneidermann est un des premiers à développer RFS en utilisant le Treemaps.

4.1. Squarified Treemaps

Cette section présente une nouvelle méthode qui est basée sur l'article [3]. Le but de cette nouvelle méthode est de rendre l'interaction possible avec le lay-out de Tremap classique en éliminant les rectangles de petites tailles. Pour faire cela, elle redéfinit l'algorithme de construction de Treemap pour obtenir des rectangles proches de carrés. En plus, elle ajoute au lay-out des cadres ombrés pour regrouper des éléments et pour améliorer la perception de structure de la hiérarchie.

La figure 5 montre un Squarified Treemap qui présente le même système de fichier que la figure 2. Grâce à la nouvelle méthode, toutes les régions noires de la figure 2 disparaissent cela rend l'interaction grâce un souri avec Treemap plus facile.

L'algorithme qui construit le Squarified Treemap cf. la figure 4, est assez proche de celui de Treemap classique. Il commence par un rectangle pour afficher la racine de l'arbre. Ensuite, il divise verticalement le rectangle de racine en deux parties si ce dernier a sa largeur plus grande que sa hauteur, sinon il le divise horizontalement. Après il essaie d'insérer un par un les fils dans le première partie. En insérant le premier fils, il teste si la seconde partie garde une taux (hauteur divisée par largeur) proche de un. Si c'est le cas, il valide sa position. Sinon il recommence avec la deuxième partie pour insérer les restes des fils.

Avec ce nouvel algorithme, le lay-out perd le sens de

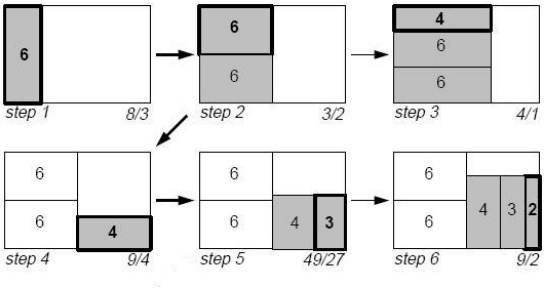
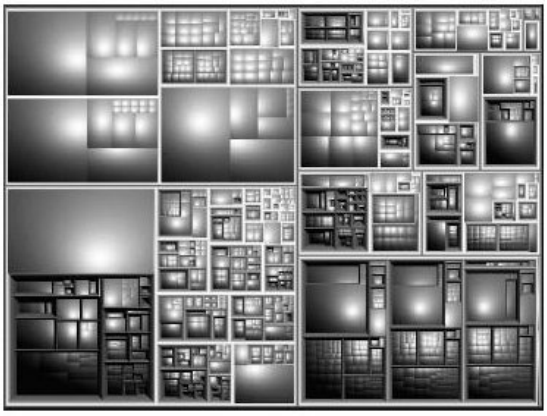


FIG. 4 – Algorithme de construction de Squarified Treemaps

hiérarchie qui est définis par Shneiderman. Pour remédier à ce problème, cette méthode utilise des cadres ombrés. Un cadre ombré représente un ou plusieurs noeuds qui ont des fils. Les rectangles et les cadres qui se trouvent à l'intérieur d'un autre cadre Y, représentent les fils de noeud correspondant à Y. L'effet de profondeur d'un rectangle, qui est donné par un cadre ombré, indique son éloignement de la racine.



(a) File system

FIG. 5 – Squaried Treemaps

Newmaps est une application qui utilise l'algorithme de Squarified Treemap pour afficher les nouvelles du monde entier ou d'un pays. Ce qui est intéressant,c'est que Newmaps montre clairement la différence entre Treemap et Squarified Treemap.

4.2. Stock Markt

Stock Markt [1], qui a la même idée que Squarified Treemap, redéfinit l'algorithme de construction de Treemap classique pour répondre au besoin de la bourse. Il est utilisé pour visualiser les bénéfices d'une hiérarchie de compagnies durant une période de temps. Il doit regrouper les compagnies par secteur. Les compagnies qui ont certaines similarités, doivent être le plus proche possible l'un de l'autre dans le lay-out.

Le nouvel algorithme altère la division verticale et horizontale à chaque insertion d'un nouveau noeud. Pour que les noeuds qui ont une relation de similarité, se trouvent assez proche de l'un de l'autre dans le lay-out. L'algorithme calcule toutes les possibilités de lay-out, ensuite il en prend le meilleure.

FIG. 6 – Marketmap

Il existe une application marketmap cf. la figure 6 qui est sur le site http ://www.smartmoney.com/marketmap/, utilise cette approche. La navigation qui est assurée par la technique de drill down et drill up, se fait par la clique droite sur le lay-out. Dans cette application, la couleur du rectangle indique la perte et le gain de la compagnie. La taille du rectangle est proportionnelle à la taille de la compagnie.

5. Radial Space Filling

Cette section expose des méthodes différentes qui se basent sur RSF.

5.1. Semi-Circular Discs

Voyons une de première approche qui se base sur RSF. Keith Andrews et Helmut Heidegger [5] ont choisi de présenter l'hiérarchie sur deux demi-cercles cf. la figure 7. Sur le premier demi cercle, leur méthode présente l'hiérarchie par des couches successives. Plus la couche est proche du centre, plus le noeud est proche de la racine. Le deuxième sert à afficher plus de détailles du noeud choisit. De cette manière, cette méthode assure le principe de Focus+context. Tout de même, elle présente un gaspillage d'espace d'affichage.

5.2. InterRing

InterRing [6] qui se bases sur RSF, est une nette amélioration de Semi-Circular Discs. InterRing améliore l'aspect interactif de lay-out en proposant de méthodes qui assurent le multifocus+context. Pour faire cela, L'interRing applique une distorsion d'un noeud qui se fait en modifiant son épaisseur ou/et angle d'ouverture cf. la figure 8.

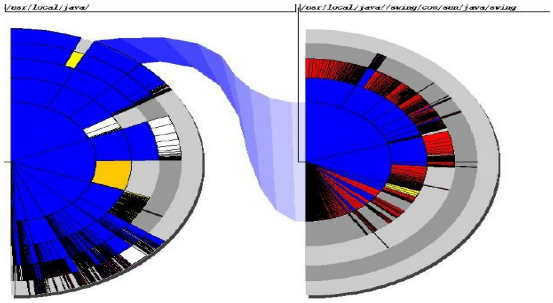


FIG. 7 – RSF :Semi-Circular Discs

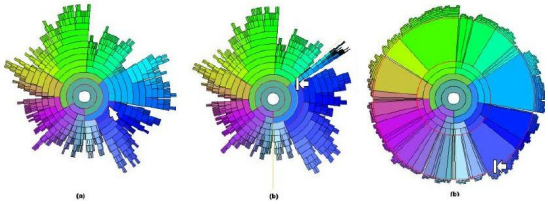


FIG. 8 – InterRing : Distorsion

L'InterRing assure le brushing par le marquage des noeuds choisit par une ligne rouge. La figure 9 montre le résultat d'une recherche des noeuds selon un certain critère. Ce critère est de sélectionner tous les noeuds qui ont au maximum six noeuds dans leurs sous-arbres. L'InterRing modifie la structure de l'arbre grâce à la sélection des noeuds et la technique de "drag and drop".

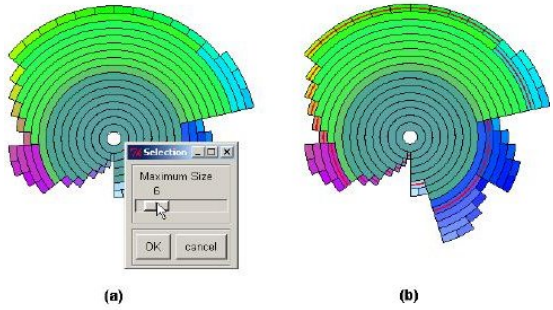


FIG. 9 – InterRing : Brushing

6. Tree

6.1. Space Tree

Le SpaceTree [7] est une méthode qui se base sur le diagramme d'arbre, a été développée pour remplacer le navigateur classique de système d'exploitation. Cette méthode utilise l'ouverture et la fermeture de la branche de l'arbre pour que le diagramme de l'arbre soit affiché à l'intérieur de la place d'affichage.

La figure 10 qui est le lay-out d'un SpaceTree, présente un exemple d'interaction avec le SpaceTree. En cliquant sur un noeud A de l'arbre son sous-arbre s'affiche, mais en cliquant sur un autre noeud B, le premier sous-arbre se referme et le sous-arbre B s'affiche. De cette manière le SpaceTree assure la navigation. Le SpaceTree marque le noeud sélectionné en changeant sa couleur et en marquant le chemin jusqu'à la racine. Donc la sélection d'un ou de plusieurs éléments se fait par un changement de couleur.

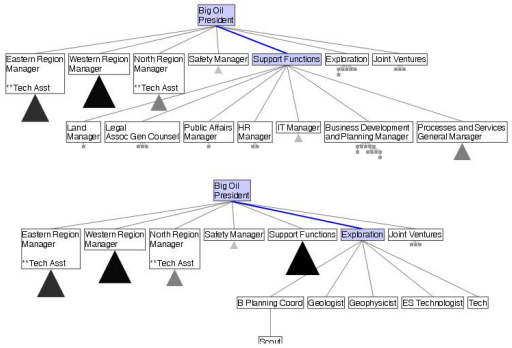


FIG. 10 – Space Tree : Animation

Le triangle qui se trouve en dessous de chaque noeud, permet d'indiquer le nombre de noeuds dans son sous-arbre avec sa taille. De plus il permet aussi d'indiquer la profondeur de sous-arbre avec une couleur. Ces triangles peuvent être changés par de petits digrammes d'arbre si le nombre de noeuds de sous-arbre est petit. En plus, le SpaceTree offre la possibilité de rendre un ou plusieurs éléments de l'arbre visible toujours dans la place d'affichage comme le montre la figure 11.

FIG. 11 – Space Tree : navigation

6.2. exploring graphs in 3D Hyperbolic space

Cette section explique la méthode qui a été développée par l'article [4] pour explorer des graphes en trois dimensions. Au lieu d'afficher des graphes directement, cette méthode construit un arbre sans cycle à partir d'un graphe. Pour faire cela, tous les éléments de graphe doivent partager une propriété hiérarchique. De plus il utilise l'animation pour améliorer l'interaction et la visualisation de l'arbre. La figure 12 montre un exemple d'application de cette méthode. Cette figure est la structure d'appels de procédure d'un programme de Fortran. Dans le lay-out, il y a des certaines arêtes qui appartiennent au graphe et n'appartient pas à l'arbre. Ces arêtes

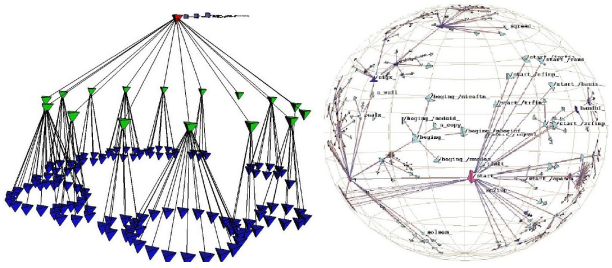


FIG. 12 – Cone Tree and the layout of graphs in 3D

sont affichées à la demande de l'utilisateur.

La construction de cette méthode est basée sur le technique de présenter l'arbre sous forme des cônes cf. la figure 12. Au lieu de placer les files d'un noeud sur le cercle qui forme la bouche de cône, elle place ces fils sur la surface de demi Sphère de leur parent. Cette surface doit aussi contenir la présentation de tous les fils et leurs sous-arbres. En sélectionnant un noeud, le noeud se place progressivement dans le centre de la sphère cf. la figure 13. Dans ce cas, tous ses fils se retrouve dans demi sphère droite et le reste de noeuds se situent dans celui de gauche. La navigation est assurée par la sélection un noeud, ensuite la rotation de la sphère autour ce noeud et changement de sa place dans la sphère.

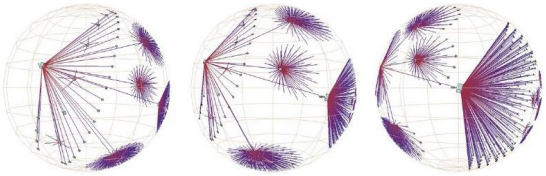


FIG. 13 – Navigation

7. Conclusion

Notre problème de départ qui est d'afficher le maximum des l'éléments d'une hiérarchie dans une place délimité. Ce défit a été relever par plusieurs études. Ces dernières ont été présentées par cet article.

Toutes ces études ont était basées sur une de techniques suivantes, la première est en utilisant un diagramme d'arbre, comme dans SpaceTree. Cette technique est le plus naturel possible. La deuxième technique qui consiste à remplir de place d'affichage par des rectangles. Le SquarifiedTreemaps qui utilise cette technique, de plus, facilite l'interaction avec son lay-out. Il permet aussi plus facilement de comparer les tailles des rectangles. Finalement, le remplissage de la place d'affichage par des radial que la méthode InterRing utilise. Donc, Il existe trois grandes familles de méthodes d'affichage, mais il n'existe aucune méthode qui utilise les trois techniques dans le même lay-out à ma connaissance.

Les approches qui sont exposées par cet article, nécessitent une période d'apprentissage pour que le utilisateur soit capable de profiter de toutes les informations dans le lay-out. Cette période peut varier d'une personne à une autre. Personnellement, je trouve que le Treemaps nécessite une longue période d'apprentissage.

8. Référence

[1] Martin Wattenberg, "Visualizing the Stock Market", SICGHI 99 Extended Abstracts Proceedings, Pittsburgh, PA, pp. 188-189, May 1999.

[2] *B. Shneiderman "Tree visualization with tree-maps : a 2d space filling approach", ACMTransactions on Graphics 1992*.

[3] *Bruls, D.M., C. Huizing, J.J. van Wijk. "Squarified Treemaps", Proceedings of EuroGraphics 2000, pp. 33-42*.

[4] *Tamara Munzner, "Exploring Large Graphs in 3-D Hyperbolic Space", IEEE Computer Graphics Applications, Vol. 18, No. 4, Jul/Aug. 1998, pp. 18-23.*

[5] *Keith Andrews, Helmut Heidegger, "Visualising and Exploring Large Hierarchies using Cascading, Semi-Circular Discs"*. 1998.

[6] *Jing Yang, Matthew Ward, and Elke Rudensteiner, Anilkumar Patro, "InterRing : a visual interface for navigating and manipulating hierarchies", Information Visualization, Vol. 2, No. 1, March 2003, pp. 16-30. (From InfoVis '02).* 2003.

[7] *Catherine Plaisant, Jesse Grosjean, and Ben Bederson, "SpaceTree : Design Evolution of a Node Link Tree Browser", Proceedings of IEEE Information Visualization 02, Boston, MA, Oct. 2002, pp. 57-64*. spacetree.