Mise à jour de l'état de l'art sur les techniques de visualisation pour l'analyse visuelle de phénomènes dynamiques

Réalisation numéro 4.3.4 du projet RNTL OADymPPaC

Jean-Daniel Fekete Mohammad Ghoniem

École des Mines de Nantes

04 avril 2003

1 Introduction

L'amélioration extraordinaire des supports informatiques au cours des dernières décennies a permis le stockage de quantités énormes d'informations. Le traitement de ces informations a donné naissance à plusieurs disciplines dont la gestion des bases de données et le data-mining. Ainsi, à l'aide de calculs mathématiques puissants et de métriques ad hoc peut-on, dans une certaine mesure, déterminer des corrélations entre diverses données. Seulement, sans connaissance préalable du domaine et sans la bonne intuition, il est souvent très difficile d'explorer une masse énorme d'information à l'aide de ces outils traditionnels.

Or, le système perceptif de l'être humain, notamment les yeux, possède des potentiels très peu exploités jusqu'à nos jours dans les applications scientifiques et dans les interfaces logicielles. Partant de ce constat, le domaine de l'IHM (interaction homme machine) a connu un intérêt accru et un essor important visant à adapter l'outil informatique aux capacités motrices et sensorielles de l'homme. Idéalement, la machine deviendrait alors un prolongement des capacités humaines et non une contrainte avec laquelle l'utilisateur doit composer tant bien que mal.

Les travaux en visualisation s'inscrivent dans cette logique visant à mieux exploiter le potentiel de notre système de perception. Dans le domaine de la visualisation, on distingue essentiellement deux branches : la visualisation scientifique et la visualisation d'informations à proprement parler. La première s'intéresse à la représentation d'objets réels (molécules, organes etc.) possédant une représentation intrinsèque. La seconde porte sur des entités conceptuelles ne possédant pas de représentation propre. Le choix de la représentation incombe donc totalement à l'utilisateur et dépend de la nature des données.

Ainsi de nombreux travaux portent-ils sur des données à une, deux ou trois dimensions et plus généralement sur des données multidimensionnelles. D'autres portent sur des données structurées sous forme d'arbres ou de graphes ou, peu ou pas structurées, sous la forme d'un nuage de points. D'autres encore portent sur des données séquentielles et des données temps réel. Par ailleurs, des techniques d'interaction ont été proposées pour faciliter la navigation dans les données et garder un difficile compromis entre une vue globale des données et l'accès au détail de chaque nœud d'information. Pour faire face à l'énorme volume de données disponibles, on a recours à des techniques d'agrégation se contentant de représenter des entités de plus haut niveau et fournissant sur demande une vue plus détaillée. Enfin, l'animation des transitions entre diverses vues des mêmes données permet à l'utilisateur de garder ses repères mentaux et de suivre les transformations de son espace d'exploration.

Parallèlement, dans le domaine de la résolution de problèmes avec contraintes, une multitude de travaux sont faits pour résoudre diverses classes de problèmes connus et/ou réputés difficiles. Diverses heuristiques sont proposées pour trouver une solution dans un temps humainement acceptable. Des améliorations d'algorithmes connus sont testées. La résolution de tels problèmes génère théoriquement des espaces de recherche de taille exponentielle par rapport au nombre de variables manipulées.

L'efficacité des algorithmes de résolution se mesure donc à leur capacité à élaguer les parties infructueuses de l'espace de recherche pour converger rapidement vers la ou les solutions du problème. On associe donc à la résolution de tout problème avec contraintes diverses structures de données comme l'arbre de choix, l'arbre de valuation et le graphe de contraintes

et son graphe dual. Jusqu'à présent, les plateformes de résolution de problèmes avec contraintes sont peu ou pas outillées pour visualiser de telles structures, les utilisateurs devant souvent se contenter de sorties textuelles.

Il semblait donc naturel d'appliquer les travaux faits dans le domaine de la visualisation d'information au domaine de la résolution de problèmes avec contraintes, ce dernier offrant des défis intéressants de par la taille et la nature des données manipulées.

Avant de passer en revue les techniques de visualisation d'informations susceptibles de nous intéresser, nous ferons un premier point dans la partie suivante sur les objets et concepts de la programmation par contraintes susceptibles d'être visualisés.

2 Besoins en visualisation de la programmation par contraintes

2.1 Les arbres

La résolution d'un problème avec contraintes consiste à tester toutes les valeurs de toutes les variables jusqu'à trouver une ou plusieurs solutions si elles existent. On distingue dans ce processus trois phases principales :

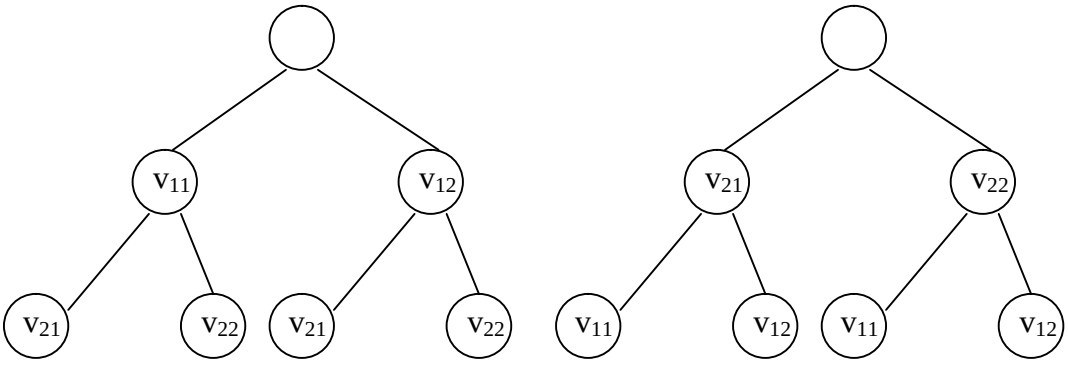
- la définition des variables et des contraintes qui les lient
- l'élimination de valeurs impossibles, il s'agit du *narrowing*
- une série d'essais et échecs des valeurs restantes, il s'agit du *labeling*.

Divers projets comme Oz et Discipl ont permis de mieux cerner les besoins de la communauté de programmation avec contraintes. Nous nous basons sur les conclusions de Discipl pour définir les structures d'information et les concepts pertinents dans la compréhension, l'analyse et le débogage d'un programme avec contraintes.

2.1.1 L'arbre de valuation

Nous appelons arbre de valuation l'arbre correspondant à toutes les affectations possibles de valeurs des variables. Il correspond au produit cartésien D_1 x D_2 et représente l'espace totale de recherche. Pour m variables à n valeurs par domaine, et pour un ordre donné sur les variables, l'arbre de valuation comporte n^m feuilles c'est-à-dire n^m affectations. Nous avons donc pour un problème donné !m arbres possibles comportant chacun m^n feuilles.

Soit un problème à deux variables V_1 et V_2 ayant pour domaine $D_1=\{v_{11}, v_{12}\}$ et $D_2=\{v_{21}, v_{22}\}$. Sur ce petit exemple, nous obtenons donc les deux arbres suivants :



Notons que dans les problèmes de taille réelle, on peut manipuler plusieurs centaines de variables avec autant de valeurs dans leurs domaines respectifs. Il est donc évident que, pour de tels problèmes, il est impossible de visualiser l'intégralité de l'arbre de valuation tel quel. Il faudra éventuellement faire recours à des techniques d'agrégation et de proposer le détail d'une région de l'arbre sur demande uniquement.

2.1.2 L'arbre de choix

En pratique, les utilisateurs et les concepteurs d'heuristiques s'intéressent davantage aux choix réellement effectués pendant la phase de labeling. Cela correspond à un arbre de plus petite taille appelé arbre de choix. On pourra le représenter intégralement dans une certaine

mesure mais l'application de techniques d'agrégation sera probablement nécessaire pour des algorithmes de résolution peu adaptés ou des problèmes particulièrement difficiles.

2.2 Les graphes

2.2.1 Le graphe variables/contraintes et son dual

On peut associer à un problème avec contraintes un graphe dont les sommets représentent les variables du problème. Un arc reliant deux sommets représente alors une contrainte liant les variables correspondantes. La connectivité du graphe donnerait des indications intéressantes sur la séparabilité du problème en sous-problème plus simple et révèlerait ainsi les influences croisées ou l'absence d'influence entre diverses parties du problème.

On peut également tracer le graphe dual dont les sommets représentent les contraintes et les arcs les variables communes.

Une multitude de travaux dans le domaine du dessin de graphes pourront être appliquées selon les propriétés du graphe (planarité, connectivité, orientation, présence ou absence de cycles, topographie etc.).

Par ailleurs, comme pour les arbres, la taille des problèmes manipulés rend la visualisation des graphes complets relativement difficile. C'est pourquoi il est utile voire nécessaire de construire des vues de plus haut niveau par l'agrégation de certaines parties du graphe complet. L'utilisateur aura ainsi le choix de visualiser des objets abstraits dans un premier temps puis, au besoin, afficher le détail d'une partie ou de la totalité du graphe.

2.2.2 Le graphe attributé

On peut également véhiculer davantage de sens en paramétrant les propriétés géométriques du graphe selon les données du problème. Par exemple, l'épaisseur des arcs et leur couleur peut quantifier le nombre de fois où une contrainte est mise en jeu.

La couleur peut aussi servir à identifier les contraintes ou les variables. Il faut tenir compte des limites du système perceptif dans le choix et la distribution des couleurs. Le plus pratique est probablement de permettre à l'utilisateur d'affecter les couleurs manuellement aux objets qui l'intéressent. Cela n'exclut pas une affectation automatique qu'il faudra définir plus précisément.

De même, la taille des nœuds, leurs formes, leurs couleurs pourra véhiculer une information pertinente comme la taille des domaines à un instant donné, le degré du sommet c'est-à-dire le nombre d'arcs qui en sont issus etc.

La distance entre les sommets peut également traduire le couplage existant entre les variables du problème : les variables (les sommets) sont plus proches si elles partagent un nombre important de contraintes ; inversement, elles sont distantes si peu de contraintes les lient.

2.2.3 L'animation du graphe attributé

L'animation du graphe attributé permet d'intégrer la dimension temps. On peut ainsi observer l'évolution du graphe au cours du temps par le rapprochement ou l'éloignement de certaines parties, par l'évolution des propriétés géométriques des sommets et des arcs (taille, épaisseur etc.) et par l'évolution de la coloration tout au long de la résolution du problème.

2.3 Les domaines des variables

On s'intéresse également aux domaines des variables à tout instant et à leur évolution au cours du temps. Une technique proposée par Carro et Hermenegildo [9] consiste à dessiner une matrice avec, en ligne, les variables du problème et, en colonne, les valeurs de leurs domaines respectifs. La couleur d'une cellule peut alors indiquer si la valeur est en attente d'être testée, conduit à un échec, conduit à une solution ou est en cours d'essai. Nous illustrerons cette technique ainsi que d'autres techniques candidates pour cette tâche ultérieurement.

2.4 Des visualisations ad-hoc

Dans certains domaines d'application spécifique, il existe des représentations graphiques fréquemment utilisées. Par exemple, on proposera un diagramme de Gantt pour représenter un problème d'allocation de ressources ou d'ordonnancement. D'autres types de visualisations ad-hoc manipulant des objets sémantiques de haut niveau devront probablement être proposés et validés en concertation avec les spécialistes de la programmation avec contraintes.

D'autre part, on peut afficher plusieurs indicateurs intéressants comme le nombre d'inférences par seconde, la profondeur maximale explorée ou autres. Il faudra préciser ce point en fonction des besoins exprimés en PaC.

2.5 L'animation d'algorithmes

L'animation d'algorithmes vise à comprendre le comportement d'une heuristique ou d'un algorithme vis-à-vis d'un problème ou d'une classe particulière de problèmes. Elle sert par conséquent au pistage de bogues dans les algorithmes et à la détection d'anomalies difficile à déceler sans représentation graphique appropriée. L'utilisateur doit être en mesure de contrôler l'animation, l'enregistrer, la démarrer, la suspendre, la poursuivre, la ralentir, l'accélérer, l'arrêter et la rejouer.

3 Représentation de graphes

Le dessin de graphes est un domaine ancien à cheval sur plusieurs disciplines étant donnée l'ubiquité des graphes dans toutes sortes de données. Comme le soulignent Di Battista et al. dans [11], diverses techniques ont été proposées et un certain consensus s'est dégagé à propos de quelques conventions de dessin et règles d'esthétique communément admises. Dans cette partie, nous allons les passer en revue et nous dresserons l'inventaire des approches adoptées dans le domaine du dessin de graphes. Nous donnerons un aperçu des techniques émergeantes dans le domaine du dessin de graphes en 3D et conclurons enfin par la représentation matricielle des graphes.

3.1 Structure et Définitions

Les graphes servent à modéliser des structures relationnelles comportant un ensemble d'entités et des relations liant ces entités entre elles. Plus formellement, un graphe G=(V,E) consiste en un ensemble fini de sommets V et un ensemble d'arêtes E (des couples (u,v)).

On dit que les extrémités u et v d'une arête e=(u,v) sont adjacentes et que l'arête e est incidente à u et v. Les voisins de v sont les sommets qui lui sont adjacents et le degré de v désigne le nombre de ses voisins.

Un graphe est dit orienté lorsqu'un sens est imposé à ses arêtes c'est-à-dire lorsqu'un ordre est établi sur les sommets des arêtes. Une arête orientée sera désignée par « arc ». Un sommet est une source lorsqu'il ne comporte que des arcs sortants. Inversement, il est qualifié de puits lorsqu'il ne comporte que des arcs entrants.

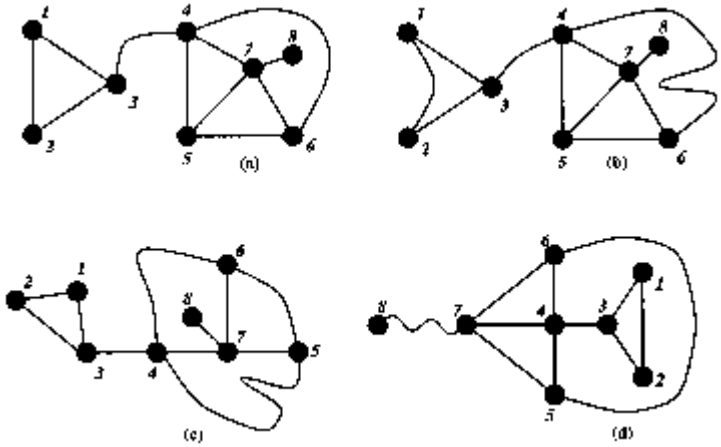
Un chemin (orienté) dans un graphe (orienté) G=(V, E)est une séquence de sommets distincts (v_1, …, v_k) où toutes les arêtes (v_i, v_{i+1}) pour i < k appartiennent à E. Un chemin (orienté) est un cycle (orienté) si l'arête (v_k, v_1) appartient à E. Un graphe orienté est acyclique s'il ne comporte pas de cycle. On désigne souvent les graphes orientés acycliques par l'acronyme DAG (de l'anglais *direct acyclic graph*).

Dans un graphe orienté, une arête (u, v) est dite transitive s'il existe un chemin reliant u et v et ne contenant pas l'arête (u, v). La fermeture transitive G' du graphe orienté G est un graphe où toutes les paires (u, v) reliées par un chemin sont aussi reliées par une arête (u, v). En général, les arêtes transitives surchargent le graphe inutilement. C'est pourquoi on préfère souvent dessiner la forme réduite d'un graphe orienté c'est-à-dire le graphe dépourvu d'arêtes transitives.

Le graphe G'=(V', E') est un sous-graphe de G=(V, E) si V' et E' sont des sous-ensembles de V et E respectivement.

Un graphe comportant n sommets peut être représenté par une matrice d'adjacence nxn où les lignes et les colonnes représentent les sommets et où le coefficient A_{ij} vaut 1 si les sommets i et j sont reliés et zéro sinon.

Il faut noter qu'un graphe est une structure de données indépendante de ses représentations. En effet, on peut représenter le même graphe de diverses manières comme le montre la figure suivante [11, pp. 2-30].



Une représentation de graphe est dite planaire si s'il n'y a pas d'intersection entre deux arêtes distinctes. Un graphe est planaire s'il admet une représentation planaire. L'un des intérêts des graphes planaires est leur lisibilité relative.

Un graphe est connexe s'il existe un chemin reliant toute paire de sommets u et v. Dans un graphe G, on désigne par *cutvertex* tout sommet dont le retrait déconnecte le graphe. Un graphe ne comportant pas de *cutvertex* est biconnexe. La composante connexe d'un graphe G est le sous-graphe connexe maximal de G. La localisation de composantes connexes dans un graphe est primordiale dans les algorithmes d'agrégation. [22]

3.2 Conventions de dessin

3.2.1 Polyline drawing

Selon cette convention, les arcs du graphe sont représentés par une ligne polygonale permettant une certaine flexibilité dans le dessin. Néanmoins, au-delà de deux ou trois segments par arc, la lisibilité du dessin est compromise.

Les « straight-line drawing » et « orthogonal drawing » sont deux variantes de cette convention. La première, très classique, consiste à représenter les arcs par des segments droits. La seconde consiste à représenter les arcs par une suite de segments horizontaux ou verticaux alternativement. Elle est beaucoup utilisée dans l'industrie des semi-conducteurs pour schématiser les circuits intégrés.

3.2.2 Grid drawing

Le dessin se fait sur une grille orthogonale de manière à ce que les sommets, points d'intersection et segments aient des coordonnées entières.

3.2.3 Planar drawing

Dessin de graphes où les arcs ne se coupent pas. Il ne s'applique qu'aux graphes planaires par définition.

3.2.4 Upward/Downward drawing

Les arcs des graphes acycliques sont représentés par des courbes décroissantes (croissante respectivement) dans le sens vertical.

3.3 Règles d'esthétique

- Minimiser le nombre d'intersections
- Minimiser la surface occupée par le dessin
- Minimiser la longueur totale des arcs
- Minimiser la longueur maximale d'un arc
- Minimiser la variance de la longueur des arcs
- Minimiser le nombre total d'inflexions des arcs
- Minimiser le nombre maximal d'inflexions par arc
- Minimiser la variance du nombre d'inflexions des arcs
- Maximiser l'angle minimal entre deux arcs issus du même sommet
- Minimiser le rapport entre le segment le plus long et le segment le plus court
- Afficher des symétries.

3.4 Diverses approches

3.4.1 L'approche par topologie – forme – métrique

Les dessins orthogonaux sont très fréquents dans les applications réelles e.g. les modèles entité – relation, les diagrammes de dataflow employés dans les systèmes de bases de données sont souvent représentés par des dessins orthogonaux. L'approche de dessin par topologie – forme – métrique proposée par [34] a été conçue pour construire des dessins sur grille orthogonale et permettent de satisfaire un grand nombre de contraintes et de règles d'esthétique.

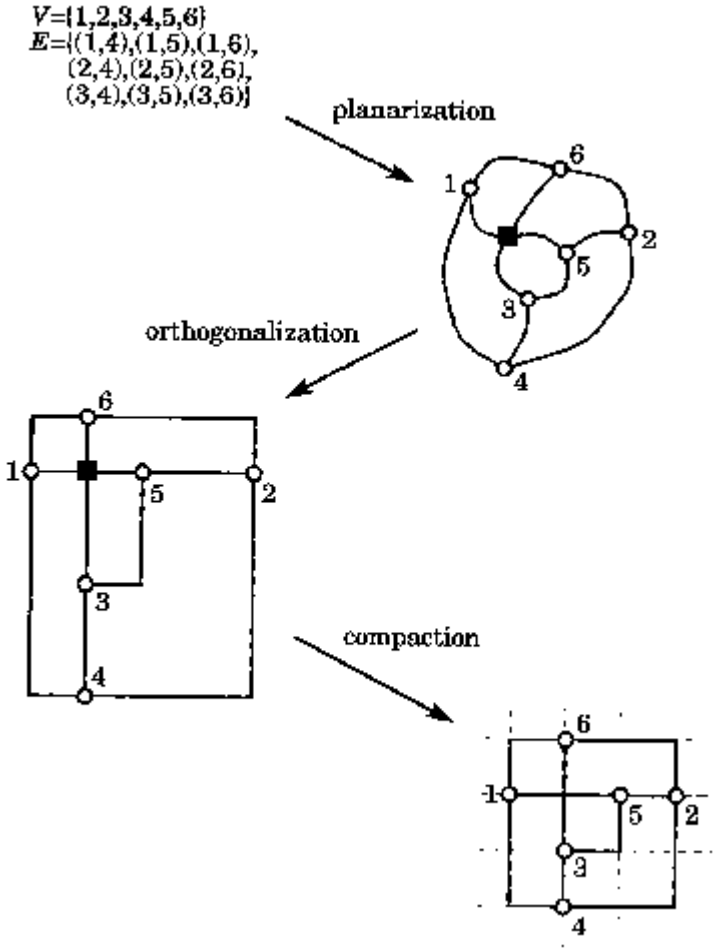
Les dessins ainsi produits possèdent trois propriétés fondamentales :

- La topologie : deux dessins orthogonaux possèdent la même topologie si l'on peut passer de l'un à l'autre par une série de déformations continues préservant l'orientation des faces i.e. conservant l'ordre des sommets.
- La forme : deux dessins orthogonaux ont la même forme s'ils ont la même topologie et que l'on peut passer de l'un à l'autre en modifiant la longueur des segments verticaux ou horizontaux composant les arcs.
- La métrique : deux dessins orthogonaux ont la même métrique s'ils sont congruents à une translation ou une rotation près.

La réalisation de tels dessins comporte trois étapes :

- L'aplanissement de la topologie en réduisant le nombre d'intersections autant que possible. Cela implique notamment l'insertion de sommets factices permettant de mettre à plat la structure. Ce sujet a été développé dans la littérature dans [11, p.20].
- L'orthogonalisation du dessin. Les sommets ne possèdent pas de coordonnées mais une liste d'angles correspondant à la succession de segments orthogonaux constituant un arc.

- Le compactage du dessin de manière à éliminer les sommets factices et réduire la surface occupée.

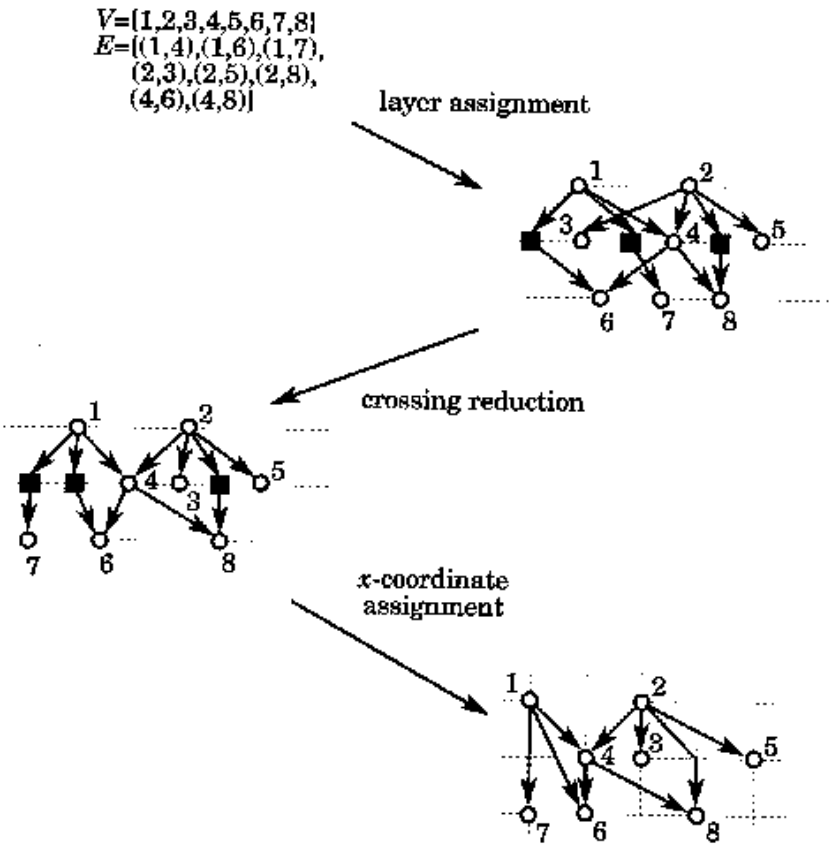


On notera que toute stratégie détermine un ordre sur les conventions d'esthétique vues précédemment. Par exemple, la topologie aura un impact sur le nombre de segments alors que le duo forme et topologie aura une incidence sur la surface du dessin etc.

3.4.2 L'approche hiérarchique

Cette approche [11, p. 22] s'applique facilement aux DAG (graphes acycliques orientés) et peut être étendue aux graphes orientés de manière plus générale. Elle comporte 3 étapes :

- Le classement des nœuds par niveau de manière à ce que l'origine d'un arc soit au-dessus de son extrémité.
- La réduction des intersections consistant à ordonner les nœuds au sein d'un même niveau et choisir la configuration minimisant le nombre d'intersections
- L'assignation d'une abscisse aux sommets préservant l'ordre précédemment établi

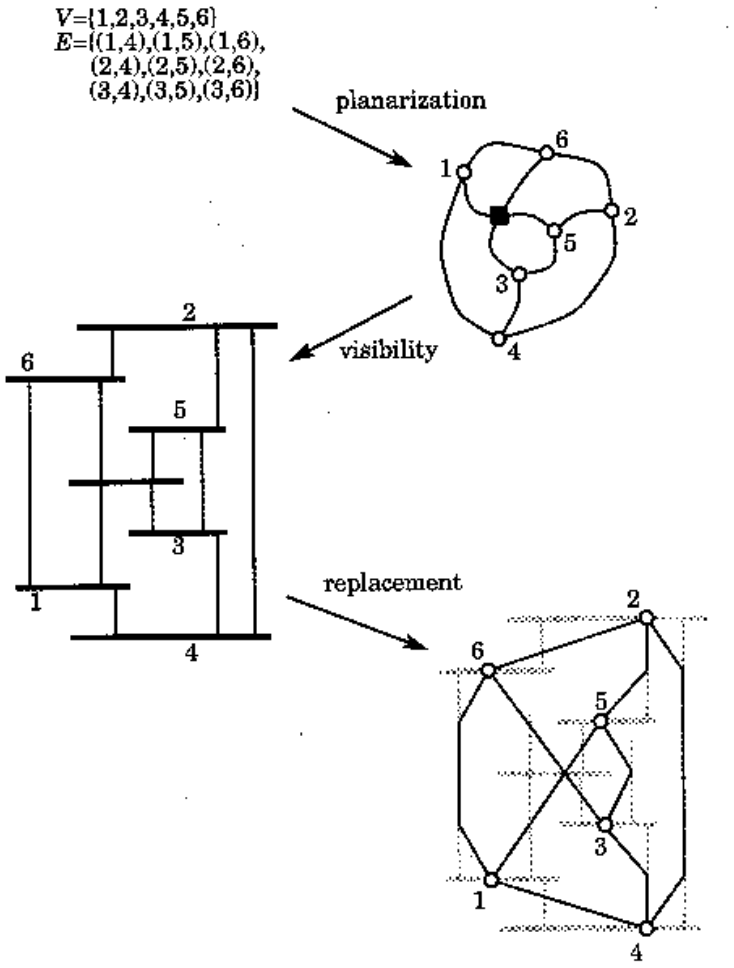


Cette approche est généralisable aux graphes orientés comportant des cycles en inversant provisoirement l'orientation d'un sous-ensemble des arêtes du graphe, puis de la rétablir après le placement des sommets. Elle peut aussi être généralisée aux graphes non-orientés en introduisant artificiellement une orientation acyclique.

3.4.3 L'approche par visibilité

Cette technique générique suit la convention du « polyline drawing » et se décompose en trois étapes :

- L'aplanissement du graphe selon l'une des nombreuses techniques proposées dans la littérature.
- La construction du diagramme de visibilité où chaque sommet est représenté par un segment horizontal et chaque arc par un segment vertical. Ce diagramme sert de squelette pour la représentation finale.
- La phase de substitution où les segments horizontaux sont remplacés par des points correspondant aux sommets et les segments verticaux par des lignes polygonales.

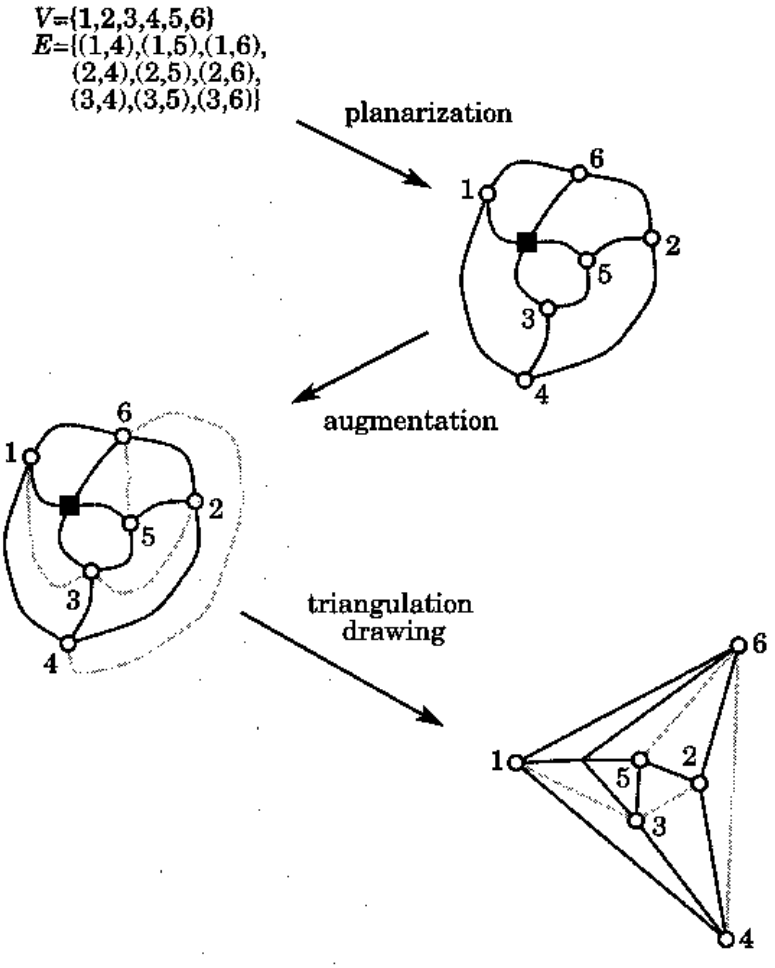


Diverses contraintes esthétiques sont observées au cours de ce processus selon la stratégie adoptée.

3.4.4 L'approche par augmentation

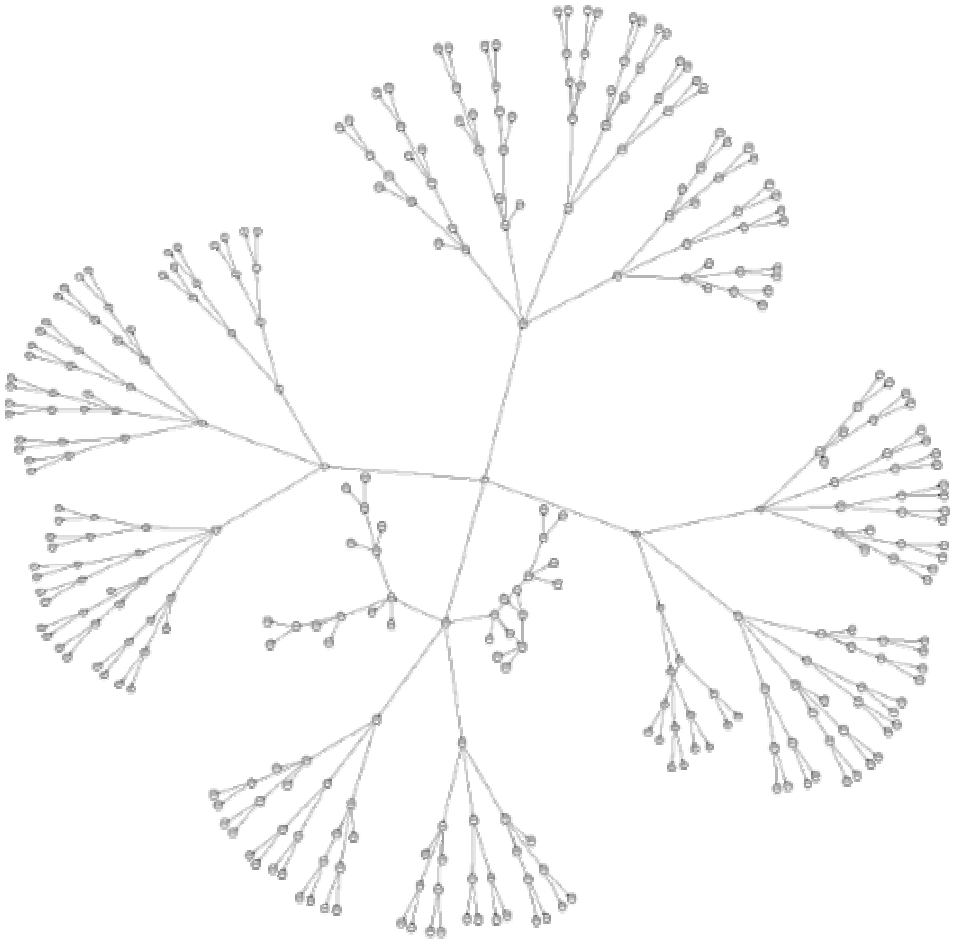
Cette approche est encore plus générique que la précédente et suit également la convention de dessin par lignes polygonales. Elle consiste essentiellement à ajouter au graphe des arcs et des sommets renforçant sa structure et donc ses propriétés graphiques. Elle comporte trois étapes :

- L'aplanissement du graphe (cf. supra)
- L'addition d'un ensemble bien choisi d'arcs et de sommets permettant d'obtenir un graphe planaire maximal (graphe dont toutes les faces sont bordées par trois arcs).
- La triangulation du graphe puis l'omission des sommets et arcs factices.



3.4.5 L'approche par champs de forces

Les algorithmes de champs de forces permettent de dessiner des graphes non-orientés selon la convention du « straight line drawing » i.e. les arcs sont représentés par des segments de droite. Il s'agit schématiquement de simuler un modèle de particules reliées par des ressorts, le but étant de trouver une configuration réalisant un minimum local de la fonction énergie du système. Des modèles de forces plus sophistiqués permettent d'imposer des contraintes spécifiques à un sous-ensemble de sommets.

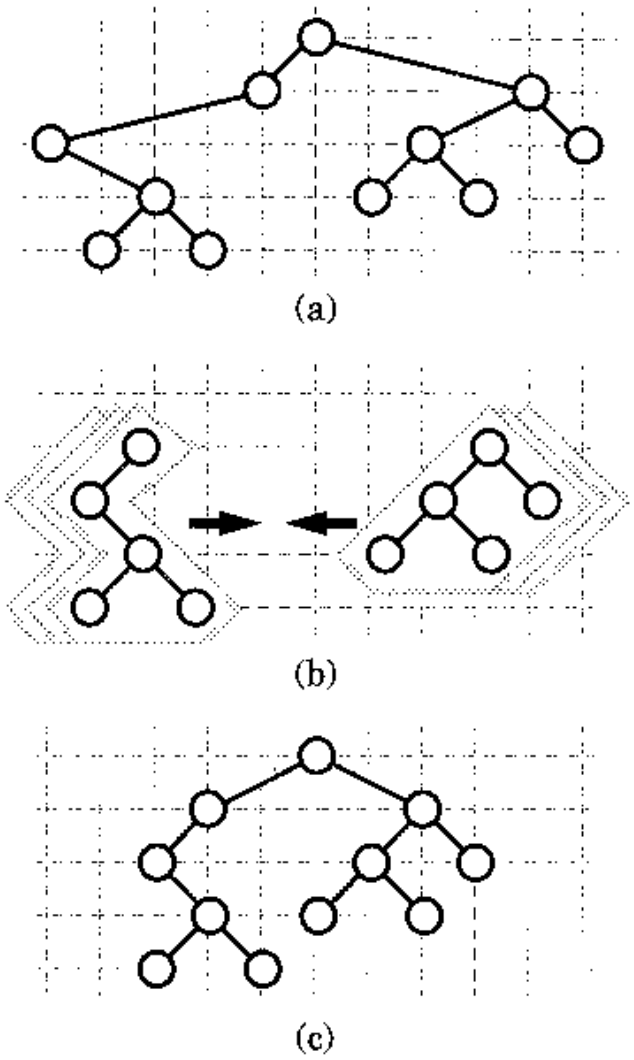


Sur la figure ci-dessus, on constate l'effet de forces d'attraction et répulsion entre certains nœuds.

3.4.6 L'approche par subdivision

Techniquement très populaire dans le domaine du dessin de graphes [11], elle comporte trois phases principales :

- Diviser le graphe en sous-graphes
- Dessiner récursivement les sous-graphes
- Dessiner le graphe de départ en collant de manière appropriée ses sous-parties.



Cette technique s'applique bien aux graphes dont on peut extraire facilement des sous-graphes comme les arbres (cf. la figure ci-dessus).

3.5 La complexité des algorithmes de dessin

La plupart des algorithmes de placement de graphes ont une complexité telle qu'ils ne sont pas adaptés à la représentation de très grands graphes. Ci-dessous, nous donnons une idée de la complexité des grandes classes d'algorithmes d'après [33, 8 pp. 354-358]:

Classe de graphes	Problème	Complexité temporelle
Graphes généraux	Minimisation des intersections	NP-difficile
Graphes généraux	Détermination du sous-graphe planaire maximal	NP-difficile
Graphes généraux	Test de planarité	O(n)
Graphes orientés généraux	Test de planarité en upward drawing	$O(n^2)$
Graphes orientés généraux à source unique	Test de planarité en upward drawing	O(n)
Graphes planaires	Dessin en segment de droite à longueur constante	NP-difficile

Gaphes planaires	Dessin en segment de droite avec une résolution angulaire maximale	NP-difficile
Graphes planaires	Dessin planaire en segment de droite sur une grille en une surface en $O(n^2)$ et une résolution angulaire en $O(1/n^2)$	O(n)
Graphes planaires	Dessin planaire en polylignes avec une surface en $O(n^2)$ et une résolution angulaire en O(1/d)	O(n)
Graphes planaires de degré 3	Dessin orthogonal minimisant le nombre de segment par arête avec une surface en $O(n^2)$	$O(n^5 \log n)$
Gaphes planaires de degré 4	Dessin orthogonal avec une surface en $O(n^2)$ et un nombre de segments par arrête en O(n)	O(n)

3.6 Les représentations en 3D

Un certain nombre de travaux récents se sont intéressés au dessin de graphes en trois dimensions et représentent de manière générale des extensions des algorithmes connus en deux dimensions. Des exemples de visualisation en 3D sont donnés dans la partie traitant des frameworks existants de visualisations de graphes.

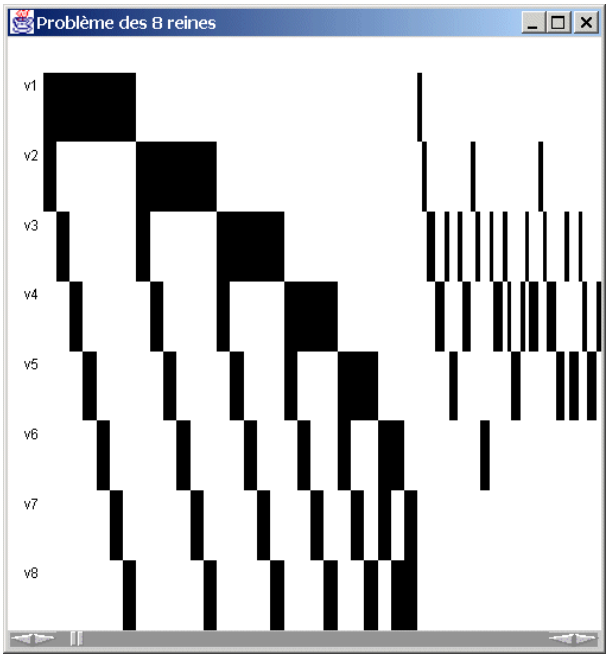
3.7 La représentation matricielle des graphes et des réseaux

On représente souvent un graphe par la matrice de connectivité qui lui est associé. Les sommets du graphe sont disposés en lignes et en colonnes. A chaque fois qu'une arête relie deux sommets V_i et V_j, le coefficient m_{ij} vaut un et sinon il vaut zéro. Bien entendu, lorsque les arêtes sont pondérées, le coefficient m_{ij} peut avoir pour valeur le poids en question.

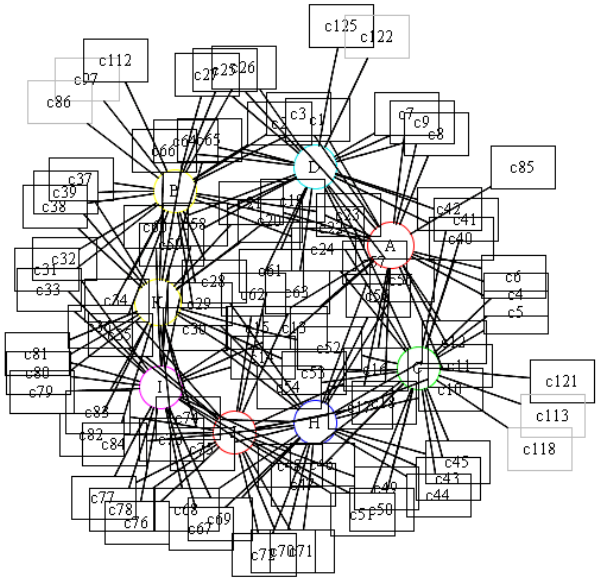
Curieusement cette représentation a rarement été exploitée pour la visualisation des graphes alors qu'elle offre un potentiel intéressant. Becker et al. figurent parmi les rares auteurs ayant mis en œuvre une visualisation matricielle des graphes, dans le système *SeeNet*, destiné à la surveillance de réseaux de télécommunication. Toutefois, aucune étude approfondie n'a été consacrée aux avantages et inconvénients de ce type de visualisations.

L'avantage indéniable de la visualisation matricielle des graphes réside en premier lieu dans le fait que tous les liens sans exception sont représentés et ce, sans le moindre chevauchement ni croisement. Or, ce sont précisément ces chevauchements qui rendent illisible, et donc inexploitable, la représentation classique en nœud – lien dès que la densité ou, subsidiairement, la taille du graphe devient importante.

En ce qui nous concerne, nous avons mis en œuvre cette idée pour la représentation de graphes contraintes-variables issus de la programmation par contraintes (voir la figure suivante). [17] Des travaux complémentaires sur la représentation des graphes d'explications nous semblent prometteurs.



Le graphe contraintes-variables du problème des 8 reines sous forme de matrice d'adjacence.



Le graphe contraintes-variables du problème des 8 reines sous forme de diagramme nœuds-liens.

3.8 La dimension temporelle

Les représentations que nous avons vues jusque-là sont statiques et ne tiennent donc pas compte de l'écoulement du temps. Mais, dans la perspective de débogage dynamique et de pilotage de programmes avec contraintes, nous devons envisager l'impact du temps sur les représentations passées en revue précédemment en ce qui concerne les graphes et celles qui vont suivre pour ce qui est des arbres et des domaines des variables.

Le temps peut apporter plusieurs sortes de changements. D'abord, la structure de données peut être modifiée par l'ajout ou le retrait d'un ou plusieurs nœuds. Il convient alors d'animer de telles modifications de manière à préserver l'image mentale de l'utilisateur.

Par ailleurs, l'écoulement du temps peut en lui-même être un indicateur intéressant. Dans ce cas, on pourra utiliser un code couleur approprié pour refléter le temps écoulé depuis un certain type d'événements. Par exemple, les sommets ou les arêtes d'un graphe ayant subi une modification récente peuvent prendre une couleur vive puis, au fil du temps, devenir de plus en plus sombres. De cette manière, on pourrait par exemple déceler des zones plus actives que d'autres dans le graphe et en tirer les enseignements qui s'imposent. Nous avons expérimenté ces idées avec succès sur la représentation matricielle précédente. [17]

Enfin, à partir de certaines représentations 2D, on peut construire des vues 3D où le temps constitue la troisième dimension.

En somme, le temps peut intervenir en tant que dimension à part entière de la représentation ou de manière masquée en influençant certains attributs géométriques ou chromatiques de la représentation.

3.9 Les techniques d'agrégation

La visualisation de grands graphes représente un grand défi car les techniques proposées pour les graphes de petite ou moyenne taille sont souvent peu efficaces à cause de l'enchevêtrement des arcs et surtout de la complexité rédhibitoire des algorithmes mis en œuvre. La densité de la visualisation compromet toute interaction avec le graphe et met à mal la navigation. [17, 22]

Face à ce problème de passage à l'échelle, une solution consisterait à réduire le nombre d'éléments affichés tout en s'efforçant de véhiculer aussi fidèlement que possible l'information que renferme le graphe complet.

En matière d'agrégation, la littérature [17] distingue essentiellement deux approches, l'agrégation structurelle ou naturelle et l'agrégation sémantique. La première se sert de la structure même des données pour opérer des regroupements. De fait, elle a l'avantage de préserver la structure du graphe d'origine et facilite donc l'orientation au sein du graphe. La seconde se base sur le contenu des données pour faire des associations. Elle est nécessairement spécifique à un domaine d'application particulier et peut être combinée avec les techniques d'agrégation structurelle.

Il faut noter que l'agrégation permet de réaliser des opérations de recherche et de filtrage dans la mesure où les parties sans grand intérêt sont agrégées alors que les parties importantes sont mises en valeur et affichées de manière détaillée.

La plupart des algorithmes d'agrégation cherchent un compromis entre le nombre d'agrégats et le nombre de nœuds par agrégat. En effet, un nombre réduit d'agrégat facilite le dessin et la navigation mais véhicule une plus faible quantité d'information. La technique la plus courante consiste représenter les agrégats par des glyphes et les traiter comme des super-nœuds faisant partie d'un graphe de plus haut niveau.

Lorsque l'agrégation est réalisée par l'application du même processus d'agrégation aux regroupements issus d'itérations précédentes, il s'agit alors d'agrégation hiérarchique. Le résultat peut être une structure d'inclusion que l'on peut parcourir de manière arborescente. Toutes ces approches impliquent la découverte préalable d'agrégats, puis l'affichage du graphe d'agrégats. Parallèlement, une approche par champs de forces donne de bons résultats après quelques itérations.

Plusieurs métriques peuvent être appliquées pour agréger un graphe. A ce sujet, on peut se référer à la littérature [12, 17, 22] et notamment la fonction de degré d'intérêt proposée par [13] permettant de masquer, d'ombrer et de regrouper des nœuds selon que l'on veut leur accorder de l'attention ou non. (cf. l'illustration donnée dans la partie sur les arbres).

Très souvent la construction d'agrégats se base sur l'extraction de composantes connexes des graphes. Cette tâche étant très coûteuse, elle ne peut plus s'appliquer aux très grands graphes. L'affichage d'un arbre recouvrant du graphe en question est alors plus judicieux dans la mesure où les algorithmes de placement d'arbres sont souvent plus performants que les algorithmes de placement de graphes.

3.10 Les techniques de navigation et d'interaction

La navigation et l'interaction sont essentielles dans la visualisation d'information et notamment pour les problèmes de visualisation de très grands graphes. Ci-dessous nous passons en revue les grandes techniques d'interaction mises en oeuvre dans le domaine de la visualisation d'information.

3.10.1 Zoom & Pan

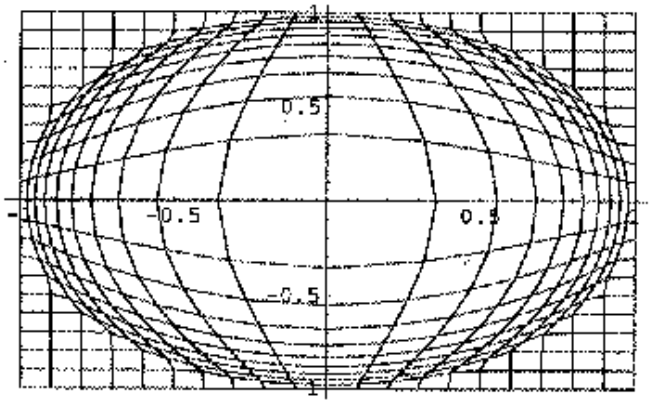
Les techniques de zoom & pan sont très utiles pour la visualisation de grands graphes. Le zoom est particulièrement bien adapté à la visualisation du fait de la simplicité des formes affichées (des lignes et des formes géométriques simples). Il y a principalement deux sortes de zoom : le zoom géométrique qui consiste simplement à grossir une partie de la vue et le zoom sémantique qui implique l'enrichissement de la partie grossie par un supplément de détails préalablement invisibles. La difficulté provient donc de la définition du niveau de détails approprié et ce en faisant appel aux techniques d'agrégation vues précédemment.

Pendant le zoom, il est très fréquent qu'un réajustement de la trajectoire vers la zoné ciblée soit nécesaire, c'est ce que l'on qualifie de « pan » dans la terminologie anglaise. Ainsi en pratique, depuis une vue d'ensemble, une série de zooms et d'ajustements sont effectués jusqu'à l'obtention de la vue détaillée souhaitée. Les problèmes soulevés par les techniques de zoom et pan sont abordés plus en détails en dans [13, 15]

3.10.2 Focus + Contexte

Un problème connu du zoom est la perte du contexte entourant la zone grossie. Plusieurs techniques ont été proposées pour pallier ce problème. Ces techniques sont plus généralement qualifiées de techniques de focus + contexte. Elles complètent les techniques de zoom + pan vues dans la partie précédente.

Les techniques de distorsion par fisheye sont très populaires et possèdent des implémentations plus ou moins sophistiquées. De manière générale, le graphe est plaqué sur le plan et un point focal est défini. Une fonction de déformation est alors appliquée au voisinage du point focal. Selon la fonction de distorsion choisie, la déformation sera plus ou moins importante.

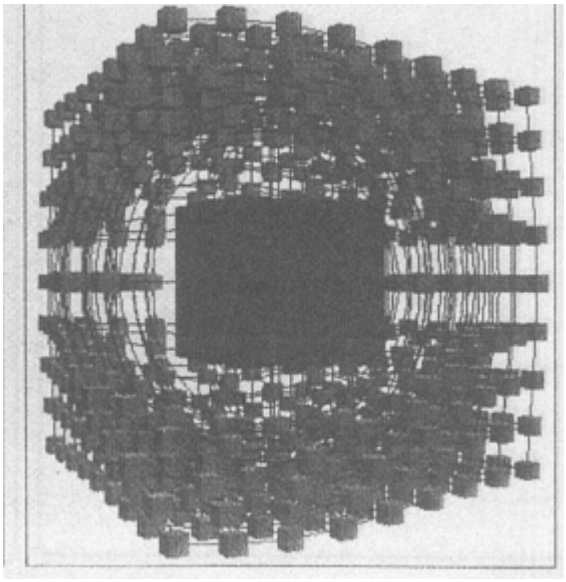


Pour plus d'exemples, se référer à [5]

Le principal obstacle à l'utilisation de ces techniques vient du fait que les plateformes graphiques standard n'offrent pas le support nécessaire à la transformation des lignes droites

en des courbes complexes selon la fonction de déformation appliquée. Il se peut néanmoins que les prochains travaux sur le standard SVG facilitent de telles manipulations. Actuellement seule la position des sommets est correctement déformée alors que les arcs sont toujours approchés par des segments droits. Par conséquent, des problèmes de croisement entre les arcs peuvent être constatés.

Ces techniques 2D ont été étendues à la 3D dans divers travaux comme [4, 11, 29].



D'autres techniques ont porté sur le zoom multifocal [4, 11] c'est-à-dire avec plusieurs points focaux simultanés.

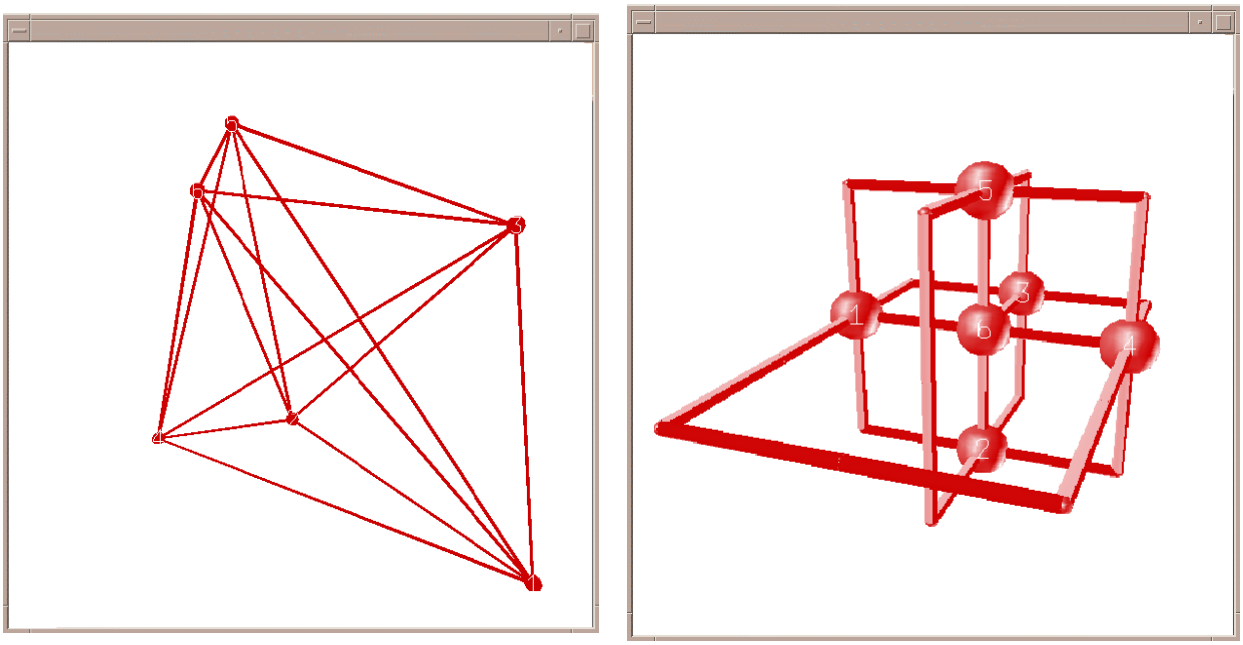
3.10.3 L'exploration incrémentale

Cette technique relativement récente porte sur les problèmes beaucoup trop importants pour être visualisés en entier. Seule une partie du graphe est alors affichée et le reste est affiché au gré de l'exploration. Ces techniques trouvent un champ d'application « naturel » dans la visualisation du *world wide web*. Pour plus de détails, se reporter à [11]

3.11 Les frameworks de visualisation de graphes

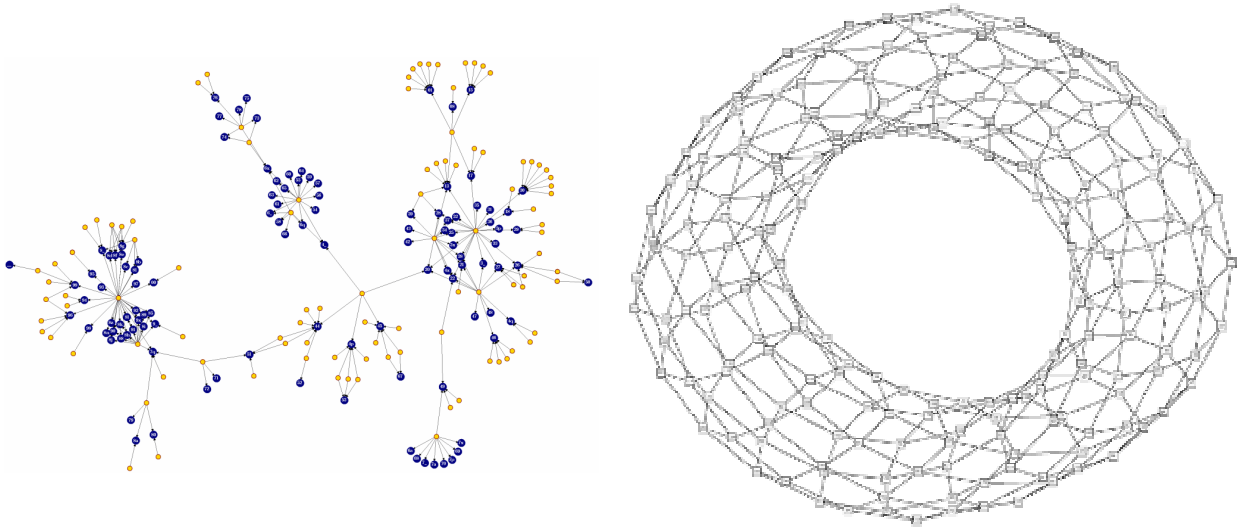
3.11.1 3D Cube

Développé à l'université de Rome III, ce framework propose des visualisations en 3D selon plusieurs algorithmes proposés dans la littérature.



Voir http://www.dia.uniroma3.it/~patrigna/3dcube/

3.11.2 Aisee



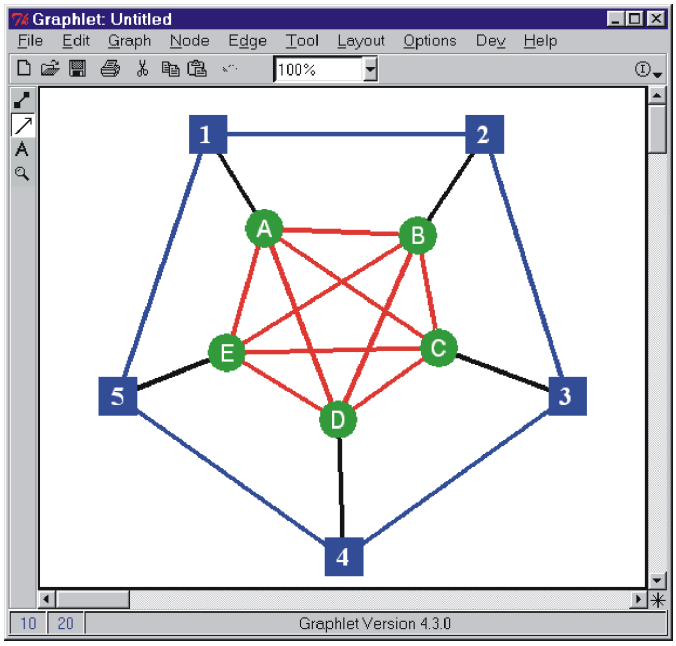
Ce framework offre une quinzaine d'algorithmes de dessin de graphes pour des structures de taille moyenne. Il implémente des techniques de zoom, navigation et animation et fait appel à GDL, un langage de description de graphes.

Voir http://www.absint.de/aisee/

3.11.3 GraphEd & Graphlet

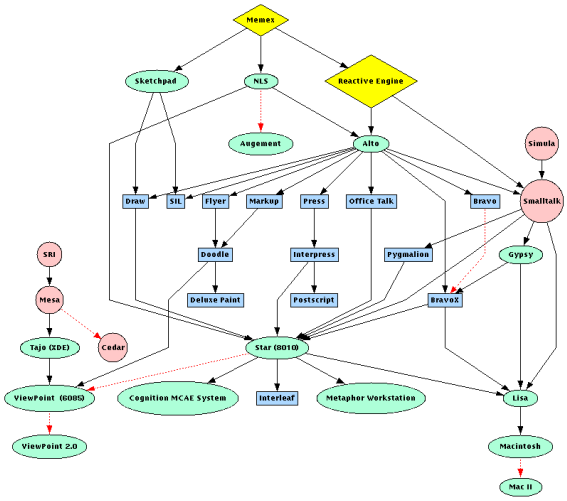
GraphEd et son successeur Graphlet ont été développés par l'université de Passau en Allemagne. Ils proposent un cadre d'édition de graphes doté d'un grand nombre d'algorithme de placement pour les graphes orientés et non-orientés. Ils sont disponibles sur internet aux adresses :

http://www.infosun.fmi.uni-passau.de/GraphEd/
http://www.infosun.fmi.uni-passau.de/Graphlet/



3.11.4 Da Vinci

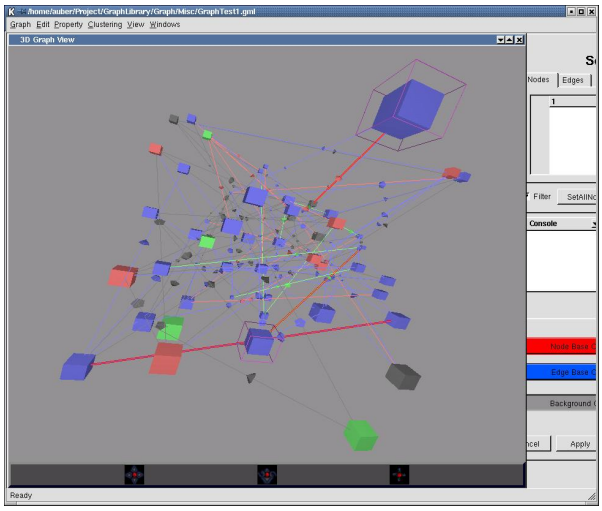
Ce framework se spécialise dans le dessin de graphes orientés. Il est développé à l'université de Brême.

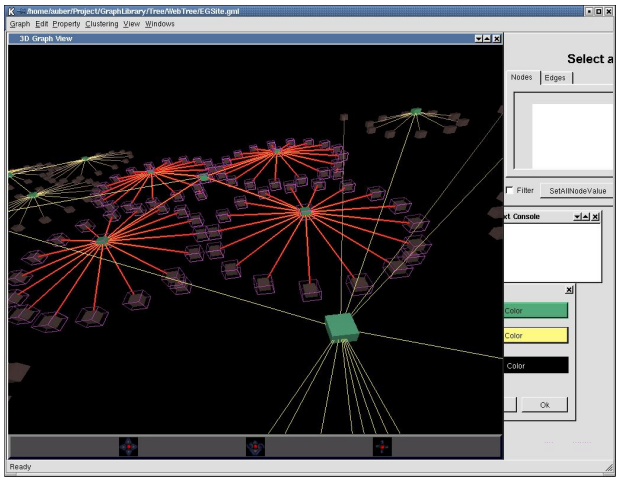


http://www.informatik.uni-bremen.de/agbkb/forschung/daVinci/daVinci.html

3.11.5 Tulip

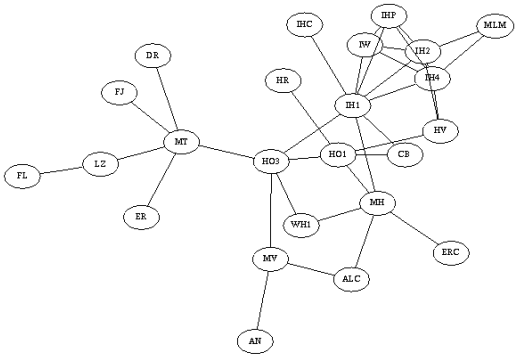
Ce framework a été réalisé au LABRI à l'université de Bordeaux. Il est développé en C++ et OpenGL et permet de visualiser de grands graphes, de l'ordre de 500 mille nœuds. Voir http://www.tulip-software.org/

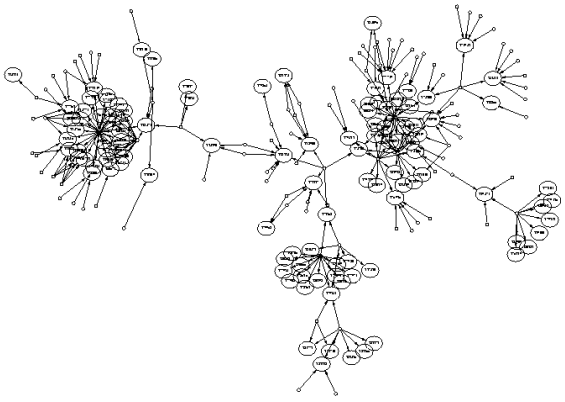




Le dessin de graphes étant un thème de recherche à part entière, nous avons passé en revue les grandes lignes qui ont retenu notre attention. Pour un complément d'information, nous recommandons au lecteur de se référer à la littérature abondante dans ce domaine et notamment l'excellente synthèse disponible dans [11]

3.11.6 Graphviz





C'est un package opensource développé par AT&T. Nous nous en sommes servis pour nos premières expérimentations de visualisation du graphe de contraintes.

Voir http://www.research.att.com/sw/tools/graphviz/

4 Représentation d'arbres

4.1 Structure et Définitions

Les arbres constituent une classe particulière de graphes. Plus précisément, un arbre est un graphe acyclique connexe c'est-à-dire qu'il existe un chemin reliant toute paire de nœuds de l'arbre. Un arbre enraciné est un arbre où l'on distingue un sommet particulier appelé racine de l'arbre. Chaque nœud possède exactement un parent exception faite de la racine de l'arbre qui ne possède pas de parents. Lorsque aucune racine n'est désignée, on parle d'arbres libres. Nous nous intéresserons ici aux arbres enracinés. A sa convenance, le lecteur se reportera à [11, p. 55] pour la représentation des arbres libres

Un arbre peut être considéré comme un graphe orienté où les arcs divergent de la racine. Etant donné l'arc (u, v), on dira que u est le parent de v et que v est le fils de u. Un nœud ne possédant pas de fils est une feuille de l'arbre.

Un arbre ordonné est la donnée d'un arbre enraciné et d'un ordre sur les nœuds fils de tout nœud de l'arbre. Un arbre où chaque nœud possède au plus deux fils est un arbre binaire. La plupart du temps les arbres binaires sont ordonnés. On parle alors de fils droit et de fils gauche.

Soit un nœud v de l'arbre, le sous-arbre enraciné en v n'est autre que le sous-graphe induit par les sommets se trouvant sur un chemin issu de v. Dans un arbre binaire, si le sommet v possède deux fils, alors le sous-arbre enraciné dans le fils droit (le fils gauche respectivement) sera qualifié de sous-arbre droit (sous-arbre gauche respectivement) de v.

La profondeur d'un nœud v est le nombre de nœuds sur le chemin reliant v à la racine de l'arbre. La hauteur d'un arbre est la profondeur maximale de ses nœuds.

4.2 Les Techniques en nœud – lien

4.2.1 Les diagrammes en nœud – lien

La technique classique du "node-link diagram" a été utilisée par les biologistes depuis le 18ème siècle. Elle consiste à représenter les nœuds de l'arbre par une figure géométrique (ronds, carrés ou autre) puis de représenter la relation de filiation par un trait ou une flèche reliant chaque nœud à son unique parent. Traditionnellement, la racine de l'arbre est placée en haut de la représentation comme sur la figure ci-dessous.

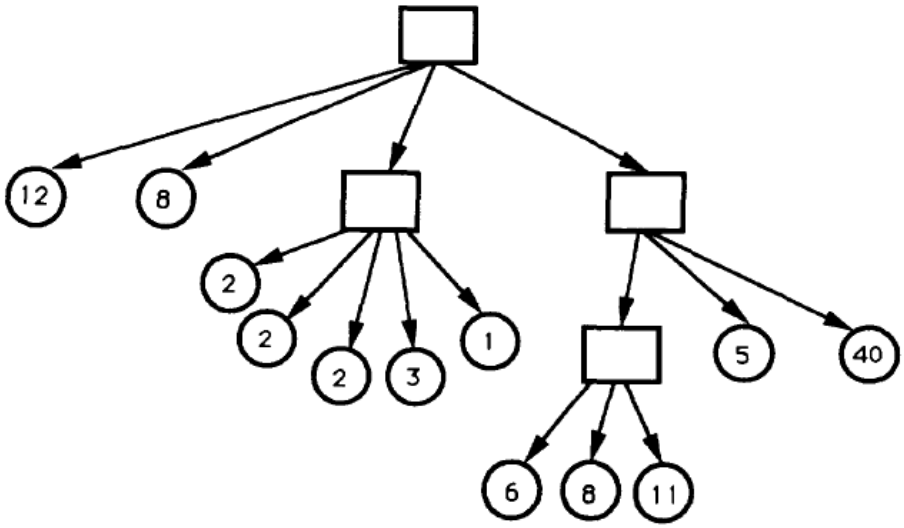


Fig. 1. Typical 3-level tree structure with numbers indicating size of each leaf node.

Note : Le diagramme précédent est en réalité une forme élaborée de la technique du nœud – lien dans la mesure où l'on distingue les nœuds structurels représentés par des rectangles et les feuilles de l'arbre représentées par des cercles. La version primitive de cette technique représente tous les nœuds par le même symbole graphique, par exemple avec des ronds partout.

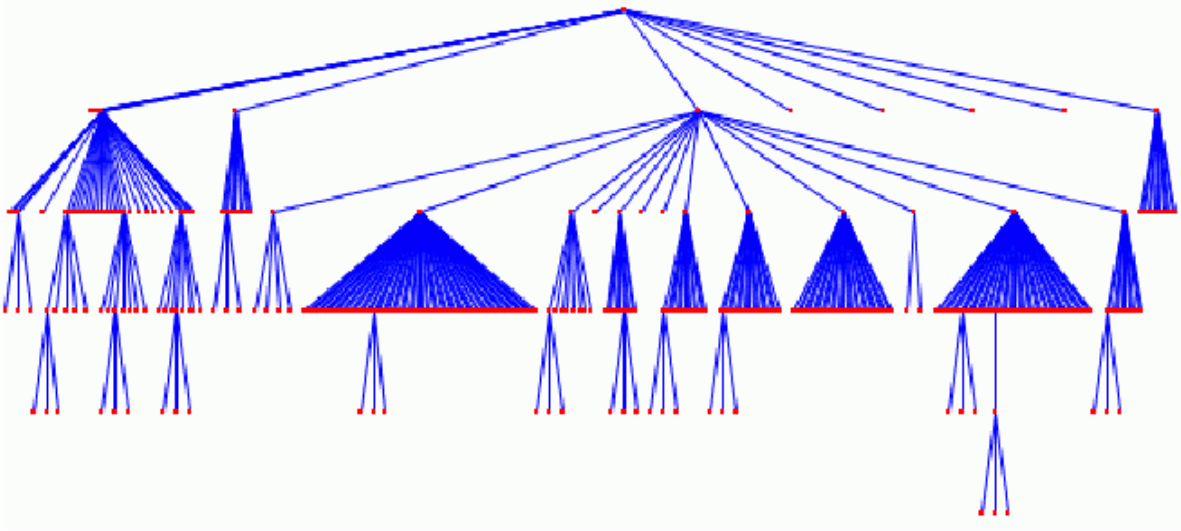
L'avantage principal de la représentation en nœud – lien est la facilité de se repérer dans l'arborescence et la mise en évidence des liens de parenté entre les nœuds. Il est possible d'enrichir cette représentation en inscrivant une information textuelle minime dans les nœuds ou à côté, et en les colorant pour traduire d'autres caractéristiques non-structurelles de nos données. Toutefois, on note que, de façon générale, la largeur d'un tel arbre croît exponentiellement par rapport à sa profondeur et que, par conséquent, toute vue globale de données massives se réduit inévitablement à une ligne [4, p. 149] .

Cette technique ainsi que ses variantes exposées dans la section suivante concernent les arbres enracinés et maintiennent une notion d'ordre sur la profondeur des nœuds. Les nœuds sont placés par couches ou niveaux. Dans une présentation rectangulaire, la racine est placée en haut et les nœuds fils sont placés sous leur parent. Dans une représentation polaire, la racine est placée au centre et les nœuds fils sont placés sur des cercles concentriques qui s'éloignent du centre la profondeur croissant.

4.2.2 Diverses Améliorations

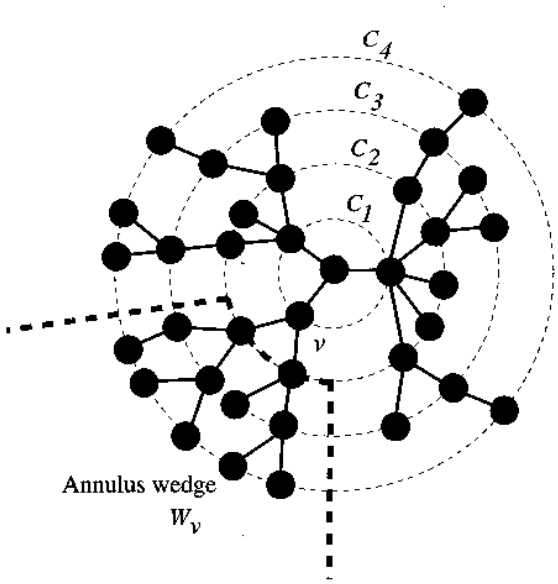
4.2.2.1 La variante de Reingold & Tilford

Plusieurs travaux effectués dans le domaine du dessin de graphes ont permis des améliorations de cette technique. Par exemple, la technique de Reingold Tilford [17, 28] permet de répondre à des contraintes d'esthétique que l'on s'impose dans le domaine du « graph drawing » et donne lieu à des représentations semblables à la figure suivante.



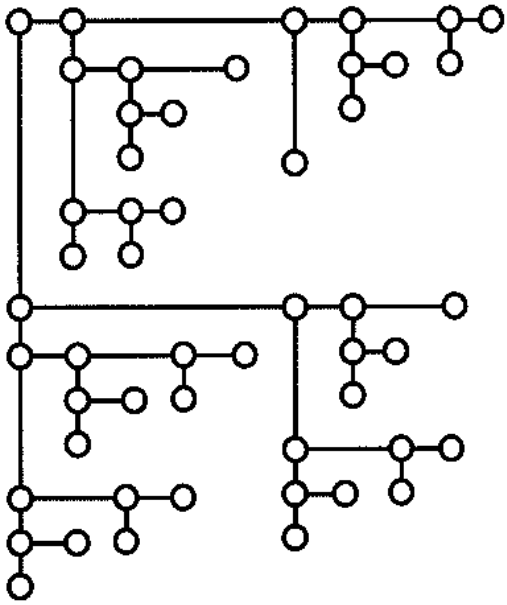
4.2.2.2 Diverses représentations des arbres enracinés

4.2.2.2.1 La représentation radiale ou circulaire



Cette technique consiste essentiellement à placer les nœuds fils sur des cercles concentriques dont le centre est la racine de l'arbre. Les nœuds d'une fratrie se partagent le secteur angulaire affecté à leur parent de manière équitable ou proportionnellement à l'importance du sous-arbre dont ils sont l'origine, cette dernière option conduisant à des intersections indésirables. [11, 24]

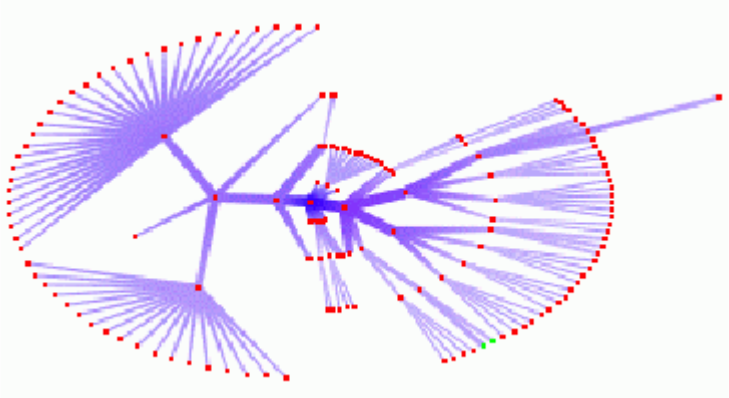
4.2.2.2.2 La représentation en HV



Cette technique [11, 17] de représentation en « Horizontale – Verticale » s'applique aux arbres binaires et consiste à placer récursivement un nœud fils sur l'horizontal de son parent et le second nœud fils à sa verticale.

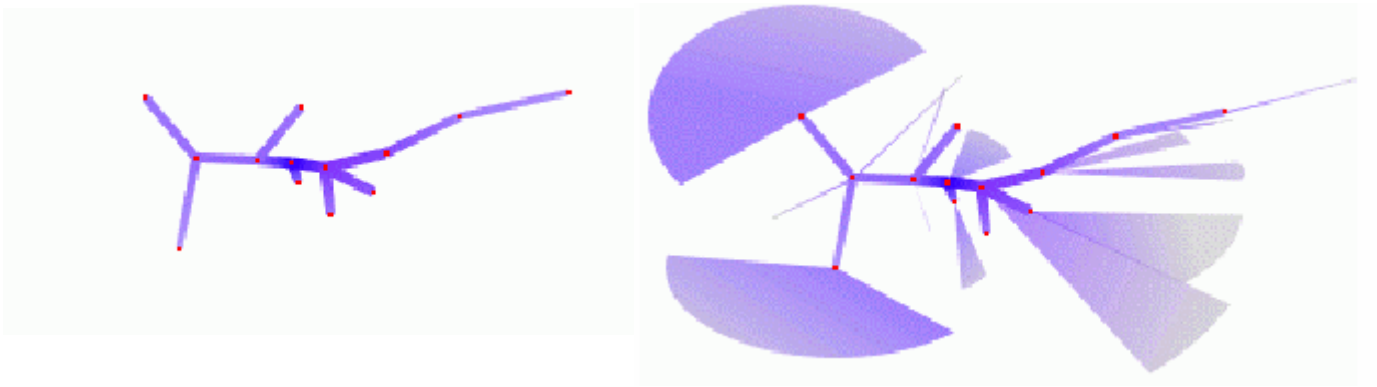
4.2.2.3 L'application des nombres de Strahler

Plus récemment, l'application des nombres de Strahler [17, 24] au dessin d'arbres permet de comparer visuellement les sous-arbres selon leur importance et facilite l'orientation lors d'un changement de focus.



Sur la figure ci-dessus, les nombres de Strahler sont appliqués à une représentation radiale.

Il est possible d'améliorer ces techniques davantage en leur associant des techniques de filtrage et d'agrégation (« clustering ») pour tenter de faire face à la croissance exponentielle des arborescences et des techniques d'interaction de type focus+contexte telles que le fisheye.



Sur les deux figures ci-dessus, les techniques de masquage, ombrage et agrégation sont combinées avec les nombres de Strahler.

4.3 Techniques de remplissage spatial

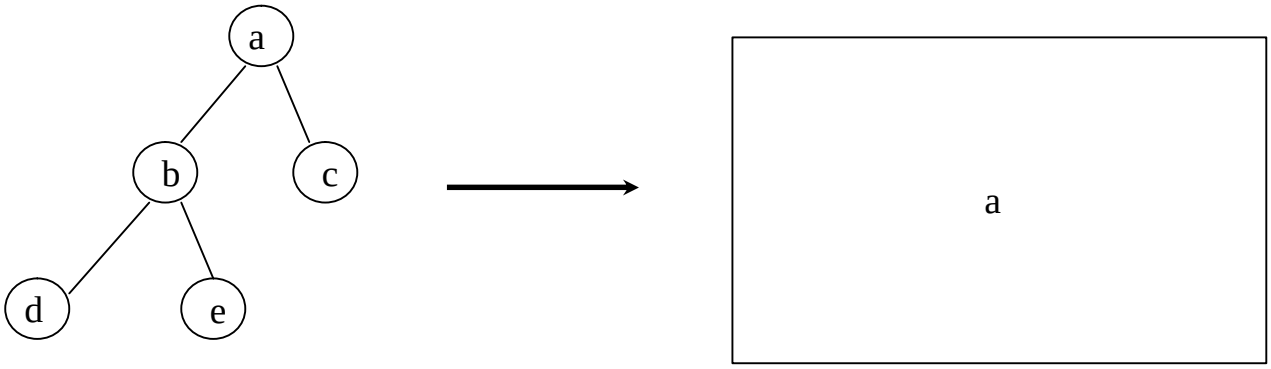
4.3.1 2D par remplissage d'espace

4.3.1.1 Les Treemaps

L'algorithme du treemap, introduit par Brian Johnson et Ben Shneiderman [21] en 1991, permet de représenter une arborescence par space-filling (remplissage d'espace). Cette technique a l'avantage d'utiliser tout l'espace d'affichage pour construire une représentation des données. Par essence, l'algorithme du treemap propose une vue globale de l'information tout en ayant la possibilité de zoomer pour mieux voir les détails.

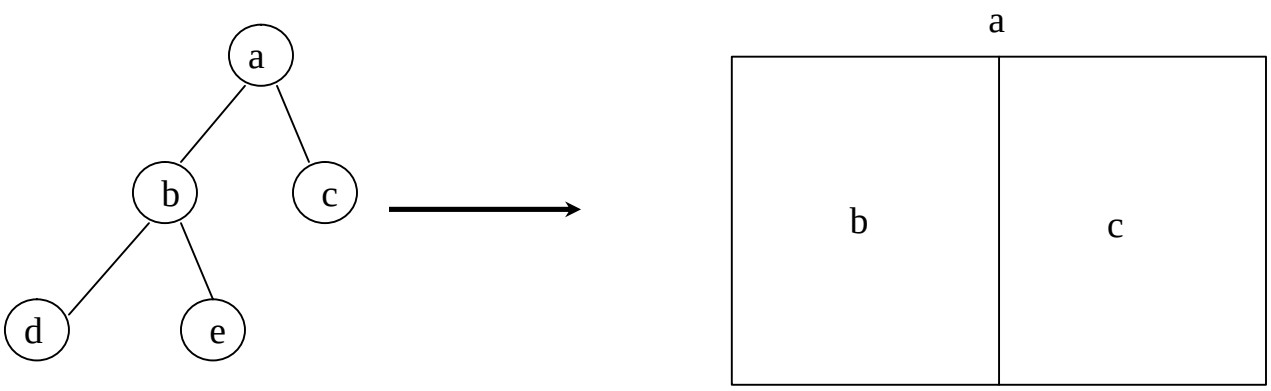
Concrètement, chaque nœud de l'arbre est représenté par un rectangle. Mais le lien de parenté se traduit par l'inclusion du rectangle représentatif d'un nœud dans celui de son ascendant. La taille des rectangles dépend du poids affecté à chaque nœud de la hiérarchie. L'habillage du rectangle que ce soit une simple couleur ou un remplissage plus élaboré permet de refléter d'autres caractéristiques des données. De surcroît, quand le rectangle représentatif d'un nœud est suffisamment grand, une étiquette peut y être inscrite.

Ci-dessous, nous illustrons la construction d'un treemap sur un petit arbre matérialisé par un "node-link diagram" où tous les nœuds d'un même niveau ont le même poids.



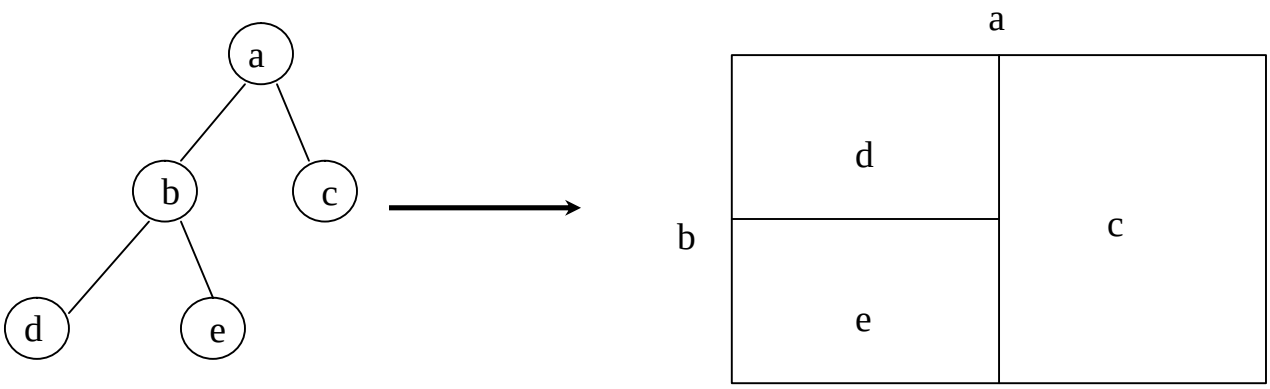
Etape 1

Le noeud "a" est la racine de l'arbre : on lui affecte la totalité de la surface d'affichage.



Etape 2

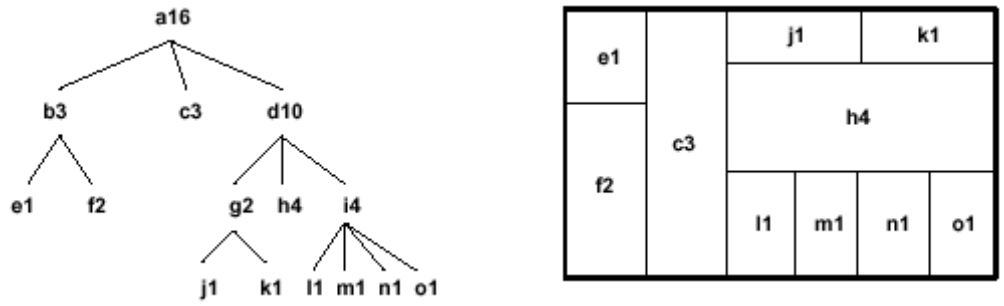
Le nœud "a" possède deux fils "b" et "c" : on répartit la surface de "a" entre ses enfants b et c.



Etape 3

Le nœud "b" possède deux fils d et e : on répartit la surface de "b "entre ses enfants "d" et "e".

Cette représentation permet de repérer rapidement les nœuds les plus importants de l'arborescence comme on peut le voir sur l'exemple ci-dessous.



A gauche, une hiérarchie représentée par un node-link diagram. A droite, la même arborescence représentée par un treemap [4]. On voit clairement sur le treemap, que les nœuds h4, c3 et f2 ont un poids nettement plus important que les autres nœuds de la hiérarchie.

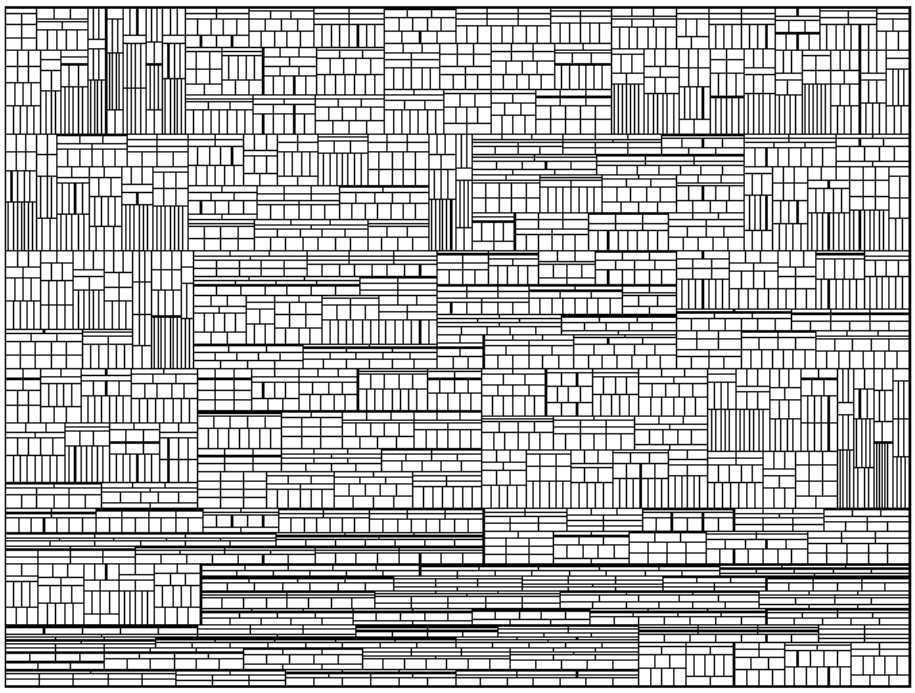
Contrairement aux techniques en nœud – lien, on perd la notion d'ordre sur la profondeur et, par la même occasion, le repérage dans la structure de données devient difficile. Néanmoins, la densité d'information est accrue du fait que l'on utilise la totalité de l'espace d'affichage. Mais cela n'est pas sans poser des problèmes à cause d'explosion exponentielle des arbres de manière générale.

4.3.1.2 Variantes des Treemaps

L'algorithme du treemap dans sa version primitive présente quelques inconvénients que certains travaux ont tenté de résoudre avec plus ou moins de succès [2, 30, 35, 36]. Nous allons voir dans les sections suivantes les principales améliorations proposées.

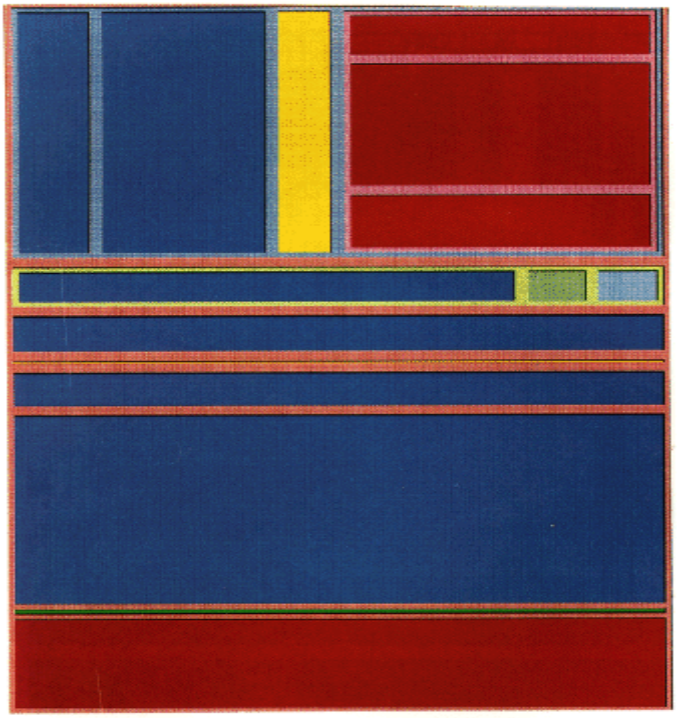
4.3.1.2.1 Nested Treemaps

On constate sur la figure précédente que les feuilles de l'arbre masquent les nœuds intermédiaires. On constate de surcroît que, dans une hiérarchie profonde comportant un grand nombre de nœuds et assez bien équilibrée, il devient difficile de se repérer comme dans l'exemple ci-dessous.



Cette figure représente la hiérarchie des employés d'une université. Ce treemap comporte 6 niveaux de profondeurs et 3060 feuilles. La hiérarchie qui sous-tend cette représentation est extrêmement difficile à déceler, tout comme l'est la position d'un nœud quelconque par rapport à la totalité de la structure.

Pour remédier à ce problème, Brian Johnson et Ben Shneiderman [21] proposent une amélioration consistant à dessiner une marge entre un nœud et ses enfants. En d'autres termes, plutôt que de partager le rectangle de leur parent, les nœuds d'une fratrie se partagent un rectangle inscrit dans celui de leur parent. Ainsi, devient-il possible dans une certaine mesure de mettre en évidence les niveaux intermédiaires de la hiérarchie comme on peut le voir sur la représentation suivante.



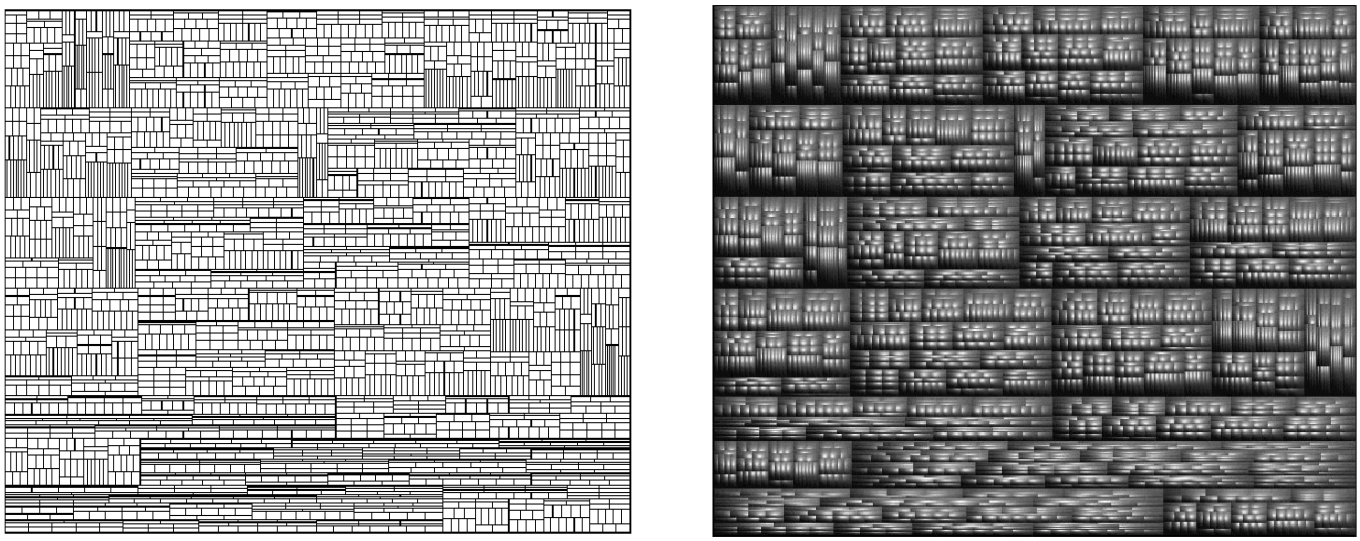
Sur le treemap ci-dessus, les nœuds partageant le même parent sont entourés d'une bordure de même couleur correspondant à une caractéristique de leur parent.

Cette technique est intéressante pour distinguer 4 ou 5 niveaux de profondeur. Au-delà de cette limite, elle présente l'inconvénient non négligeable de dilapider l'espace d'affichage. Par conséquent, on doit limiter cette amélioration aux quelques premiers niveaux de la hiérarchie dans la vue globale puis l'appliquer au fur et à mesure des zooms dans les vues détaillées.

4.3.1.2.2 Cushion Treemaps

Outre la dilapidation de l'espace d'affichage occasionnée par la technique du nesting, Jarke J. van Wijk et Huub van de Wetering [36] soulignent que pour les hiérarchies profondes la technique du nesting requiert un effort de concentration important de la part de l'utilisateur. C'est pourquoi ils proposent la variante des Cushion Treemaps basée sur l'ombrage (shading).

Cette amélioration consiste à ombrer la surface des rectangles selon un modèle géométrique 2-D relativement simple. Le résultat est assez convaincant comme on peut le voir sur la figure ci-dessous.

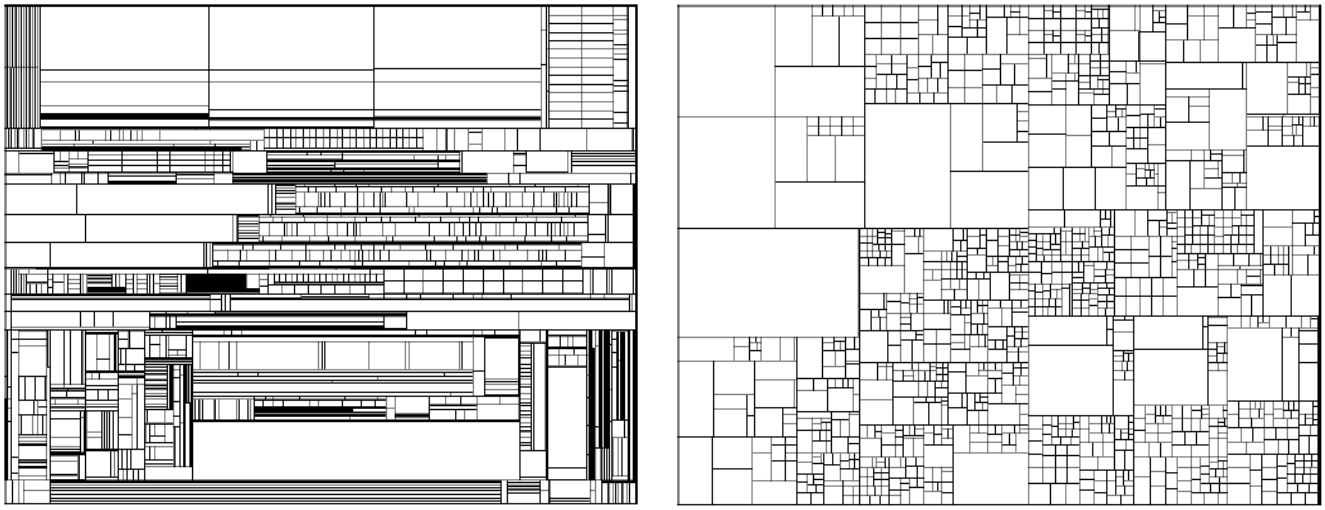


Les deux treemaps ci-dessus représentent une hiérarchie à 6 niveaux comportant 3060 feuilles. Sur la figure de droite, la technique du cushion treemap permet de mieux détacher les différents niveaux de la structure. Il faut cependant faire preuve de vigilance quant aux éventuelles illusions d'optique qui peuvent naître dans ce genre de visualisations.

4.3.1.2.3 Squarified Treemaps

Nous avons vu que l'algorithme du treemap se base sur l'attribution d'un critère de poids aux nœuds de l'arborescence. Or le poids des nœuds prend souvent des valeurs très variables. Dans ce cas, les nœuds de faible importance sont souvent écrasés par les nœuds importants de leur fratrie. Leur représentation se réduit alors à une bande rectangulaire fine dont il est difficile d'apprécier la surface et qu'il est difficile de manipuler à l'aide de la souris.

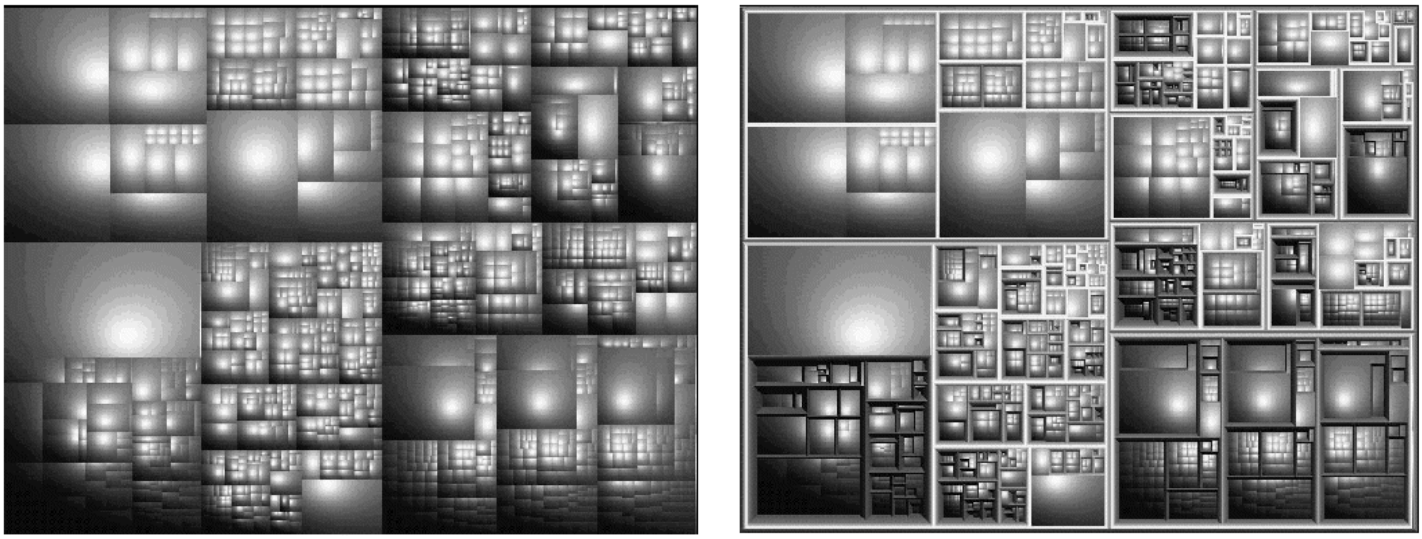
Pour résoudre ce problème, Mark Bruls, Kees Huizing, et Jarke J. van Wijk [2] proposent une technique de squarification qui consiste à représenter les nœuds par des rectangles aussi carrés que possible c'est-à-dire des rectangles dont le rapport longueur/largeur se rapproche de 1 autant que possible. L'application de cet algorithme à un système de fichiers donne le résultat suivant :



Les deux treemaps ci-dessus représentent un système de fichiers. Sur la figure de droite, les nœuds ont été « squarifiés ». La hiérarchie en est plus visible et surtout il devient assez aisé de comparer visuellement le poids des nœuds entre eux.

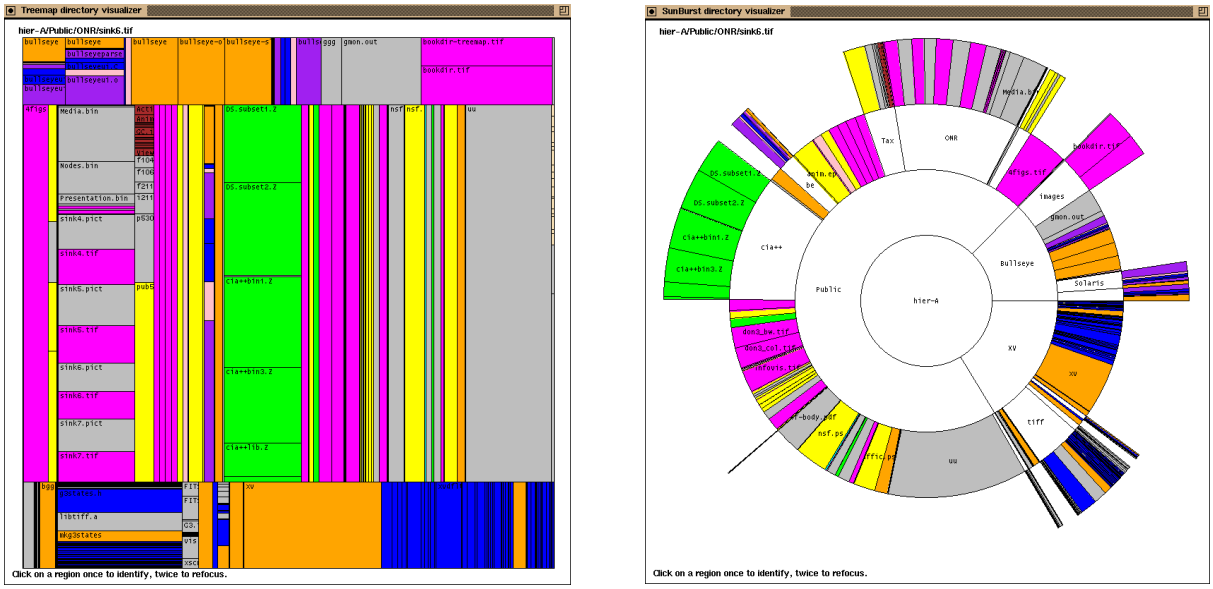
Toutefois, cette amélioration a la malheureuse conséquence de rompre l'ordre des données et ce faisant elle compromet la détection de motifs visuels qui apparaîtraient sur le treemap classique. Par conséquent, la squarification prend tout son sens pour des données dont l'ordre n'a pas d'importance pour l'utilisateur contrairement au poids des nœuds.

Dans le même article, les auteurs proposent une combinaison des améliorations vues précédemment à savoir le shading et le nesting combinés à la squarification. Ils obtiennent les résultats suivants :



Sur la gauche, la représentation combine squarification et shading. Sur la droite, on applique la technique du shading aux nœuds ainsi qu'aux marges du nesting mettant en évidence d'une façon encore plus nette la structure de l'arborescence.

4.3.1.3 Le Sunburst

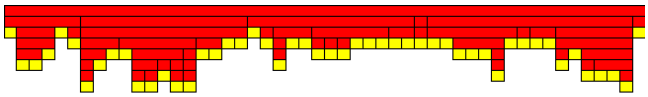


La technique du sunburst (à droite) met en évidence les différences de profondeur au sein de la hiérarchie, différences occultées sur le treemap correspondant. [32]

Il s'agit d'une représentation polaire basée sur un principe comparable à celui des *treemaps*. La racine est représentée par un disque central. Les nœuds qui en descendent se partagent l'angle au centre proportionnellement à leur poids respectif. Chaque niveau de la hiérarchie est représenté par un anneau entourant le niveau précédent. La figure ci-dessus représente à gauche un *treemap* et à droite le *sunburst* équivalent.

4.3.1.4 Les arbres à glaçons

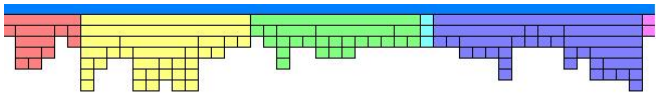
La représentation en arbre à glaçons, ou *icicle tree* dans la littérature anglaise, est le pendant cartésien de la technique du *sunburst*. Elle possède la bonne propriété de mettre en évidence les variations de profondeur au sein de la hiérarchie au même titre que les *sunbursts*.



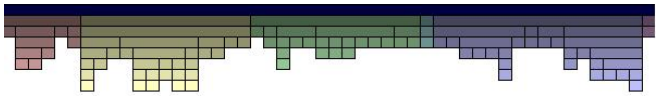
Un arbre à glaçons avec les feuilles en jaune et les nœuds internes en rouge.

Un arbre à glaçons coloré en niveaux de gris en fonction de la profondeur.

Un arbre à glaçon avec un gradient de couleurs appliqué de haut en bas.



Un arbre à glaçon avec un gradient de couleurs appliqué sur les branches.



La combinaison d'un gradient d'intensité et d'un gradient de couleurs.

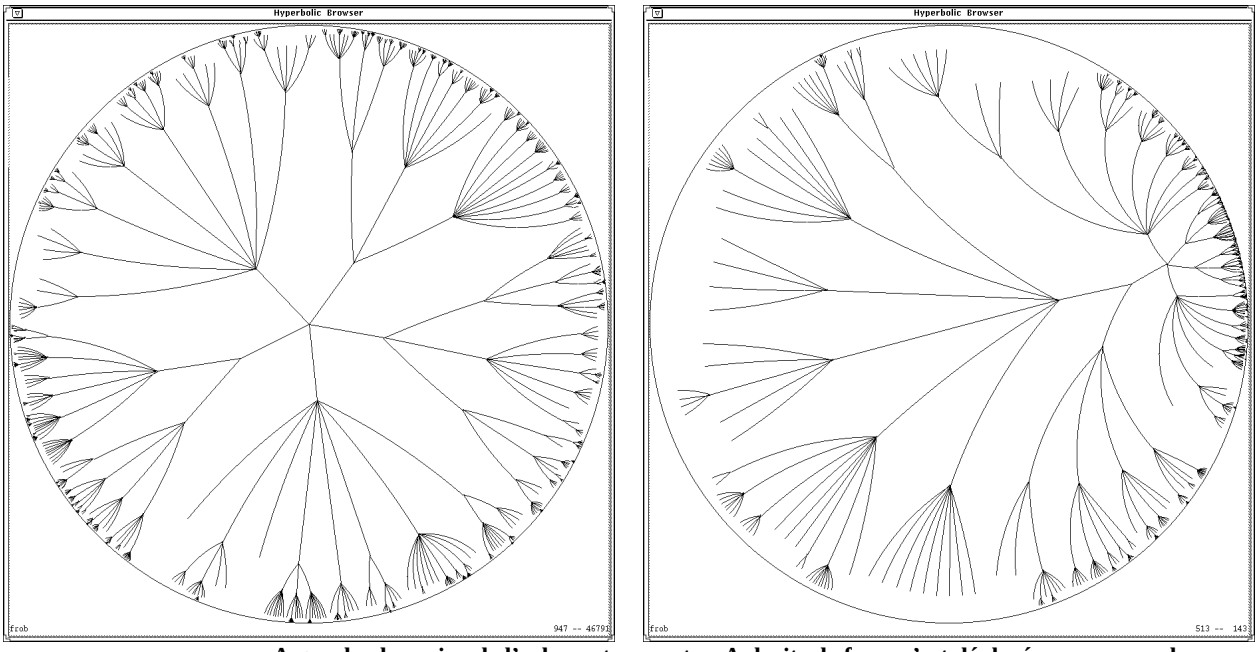
Divers schémas de coloration peuvent être appliqués interactivement aux nœuds de l'arborescence, soit pour traduire des données topologiques comme sur les figures ci-dessus, soit pour traduire des attributs quantitatifs, catégoriels ou même séquentiels associés aux nœuds.

De même, diverses pondérations peuvent être appliquées aux nœuds interactivement que ce soit par rapport à des attributs topologiques ou à des données utilisateur associés au nœuds. Sur les figures ci-dessus, le poids des nœuds correspond au nombre de feuilles du sous-arbre qui en émane. On peut également envisager de répartir l'espace équitablement entre les nœuds d'une même fratrie pour donner plus d'importance aux nœuds de plus haut niveau.

Bien entendu, toutes ces considérations peuvent s'appliquer aux diverses techniques par remplissage d'espace que nous avons abordées précédemment.

4.3.2 Techniques 2D par déformation de l'espace : Les arbres hyperboliques

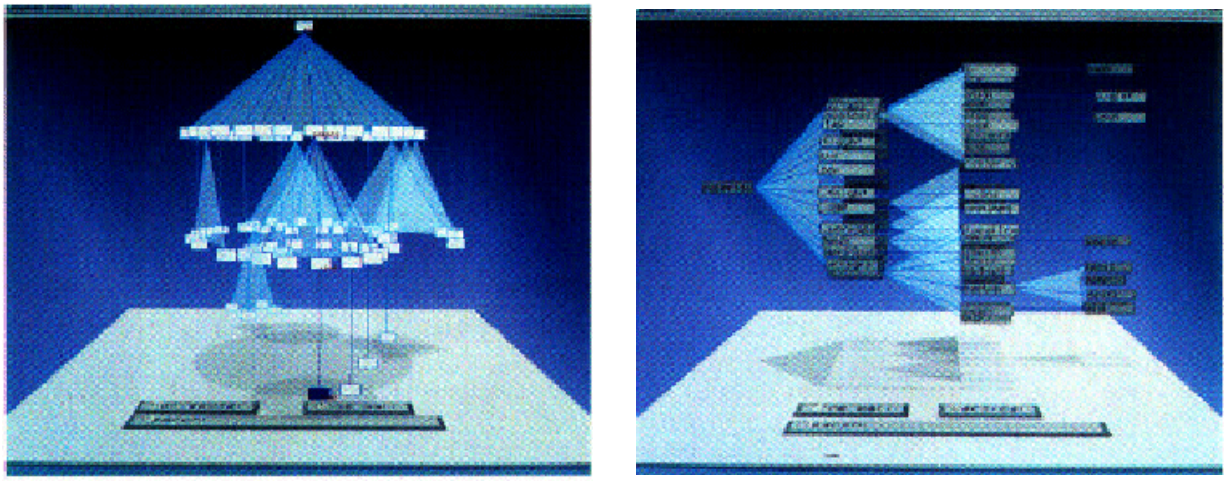
Cette technique a été développée par Xerox [23]. Elle consiste à représenter la structure arborescente dans un plan hyperbolique ramené à un disque. Etat donné que dans un plan hyperbolique la circonférence d'un cercle croît exponentiellement par rapport à son rayon, l'espace est exponentiellement plus important quand on s'éloigne du centre du cercle. Ainsi une hiérarchie croissant exponentiellement peut-elle être représentée de manière uniforme.



A gauche, la racine de l'arbre est au centre. A droite, le focus s'est déplacé vers un nœud périphérique désormais positionné au centre et reléguant le reste de la hiérarchie en périphérie.

4.4 Techniques 3D

4.4.1 Les arbres coniques



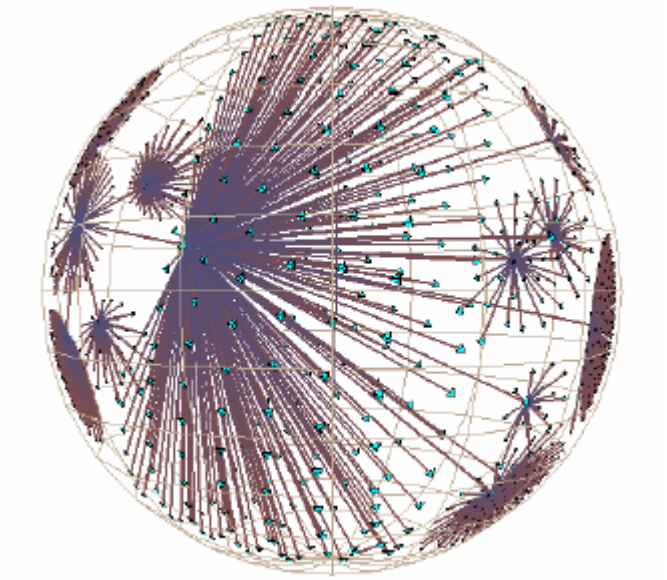
Cette technique émanant de Xerox PARC [29] consiste à placer le nœud parent au sommet d'un cône circulaire droit et de positionner les nœuds fils sur la circonférence de la base et

d'itérer ce processus à tous les niveaux de la hiérarchie. Les cônes sont semi-transparents pour réduire les problèmes d'occultation propres aux techniques de visualisations 3D. La largeur des cônes et leur hauteur dépend de la taille de l'arborescence. Cette technique pose néanmoins des problèmes analogues à la technique du node-link en ce qui concerne l'explosion du nombre de nœuds.

Pour y remédier, des techniques d'animation et de navigation 3D sont proposées. La projection d'ombres permet de donner une idée de la densité des branches de l'arbre et la lumière permet de mieux distinguer les branches du premier plan créant ainsi un effet de fisheye. La possibilité d'élaguer des branches de l'arbre et donc de se concentrer sur les parties restantes tente avec peu de succès de pallier le problème de la croissance exponentielle du nombre de nœuds. Le principal intérêt de cette technique réside dans les possibilités d'interaction et de navigation autour de la structure. D'après les auteurs de cette technique, elle atteint ses limites autour de mille nœuds, chiffre confirmé en [8].

Une première évaluation conduite par Cockburn et McKenzie [8] fait état d'une efficacité moindre des « cone trees » par rapport à un explorateur de fichiers classique (à la windows) pour des tâches de recherche de chemin mais que les utilisateurs estimaient qu'un entraînement améliorerait leurs performances. Globalement, les cone trees ne semblent pas avoir convaincu quant à leur utilisabilité. [7, 8]

4.4.2 Les arbres en espace 3D hyperbolique



Cette technique est l'extension 3D de la représentation 2D en géométrie hyperbolique. Elle a été proposée par T. Munzner [25] de l'université de Stanford. Comme pour toutes les techniques 3D, l'implémentation de techniques de navigation et d'interaction sont cruciales pour pallier les problèmes d'occlusion.

4.5 La dimension temporelle

Comme nous l'avons déjà dit pour les graphes, le temps peut constituer l'une des dimensions de la représentation ou avoir un impact sur les attributs géométriques (taille des nœuds,

épaisseur des arêtes) ou chromatiques (couleur des nœuds et des arêtes) de la représentation graphique, auquel cas l'animation des transitions s'impose.

4.6 Les techniques d'agrégation

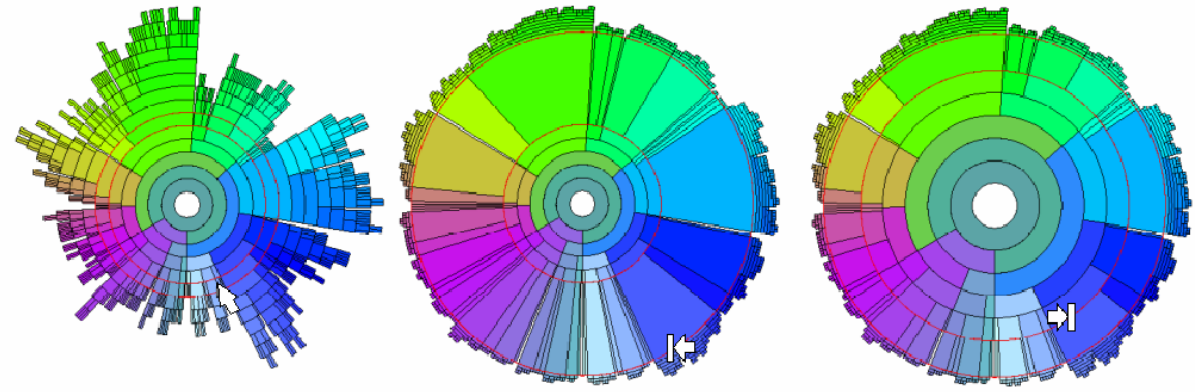
L'essentiel a été dit dans la partie consacrée aux graphes. Néanmoins, il faut noter que les arbres étant par nature des graphes connexes. Tout sous-arbre de l'arbre complet est une composante connexe que l'on peut agréger ou développer selon qu'on veut avoir une vue de haut niveau ou une vue détaillée de la région où se situe le sous-arbre en question. Il s'agit donc d'agrégation structurelle. Ceci est valable aussi bien pour les techniques en nœud – lien que pour celles en remplissage d'espace que la représentation soit en 2D ou en 3D.

Il serait intéressant de voir quelles métriques permettraient de faire des agrégations sémantiques au sein de l'arbre. Bien évidemment, ceci n'aura de sens que dans le cadre d'un domaine d'application précis en l'occurrence la programmation avec contraintes.

4.7 La navigation et l'interaction

De même, peu de choses peuvent être ajoutées par rapport à la partie analogue sur les graphes. On peut néanmoins noter que l'agrégation peut se faire de manière interactive c'est-à-dire que moyennant un clic l'utilisateur peut agréger ou développer certaines régions de l'arbre.

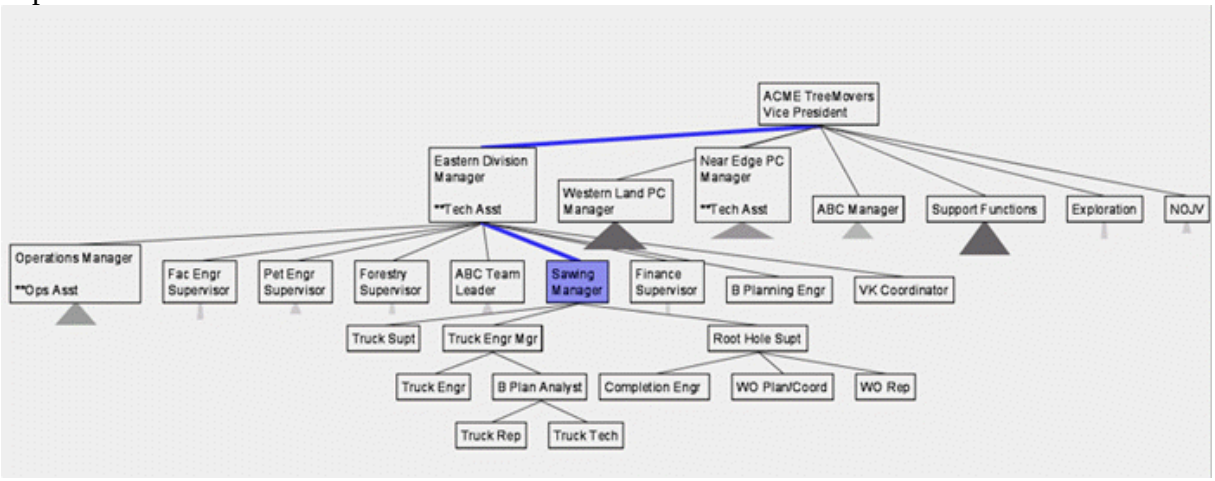
Des systèmes comme interRing [37] permettent à l'utilisateur d'hypertrophier certaines parties de l'arbre de différentes manières. On peut par exemple accorder interactivement plus d'importance à une branche, comme on peut augmenter la surface d'affichage de certains niveaux de l'arbres : un niveau en particulier, tous les niveaux inférieurs à une profondeur donnée ou tous les niveaux au-delà d'une certaine profondeur etc.



L'utilisateur sélectionne un nœud : toutes les niveaux inférieurs de l'arborescence sont hypertrophiés visuellement.

D'autres systèmes comme *SpaceTree* [26] proposent un *fisheye* sémantique sur des arborescences en nœud – lien. Initialement, la racine de l'arborescence et ses fils sont affichés entièrement sous la forme d'un organigramme hiérarchique. Les sous-arbres issus des nœuds de premier niveau sont représentés par des icônes triangulaires dont la hauteur est proportionnelle à la profondeur du sous-arbre correspondant, et dont la largeur traduit la largeur du sous-arbre. Ces triangles sont colorés en niveaux de gris selon le nombre de nœuds qu'ils comportent.

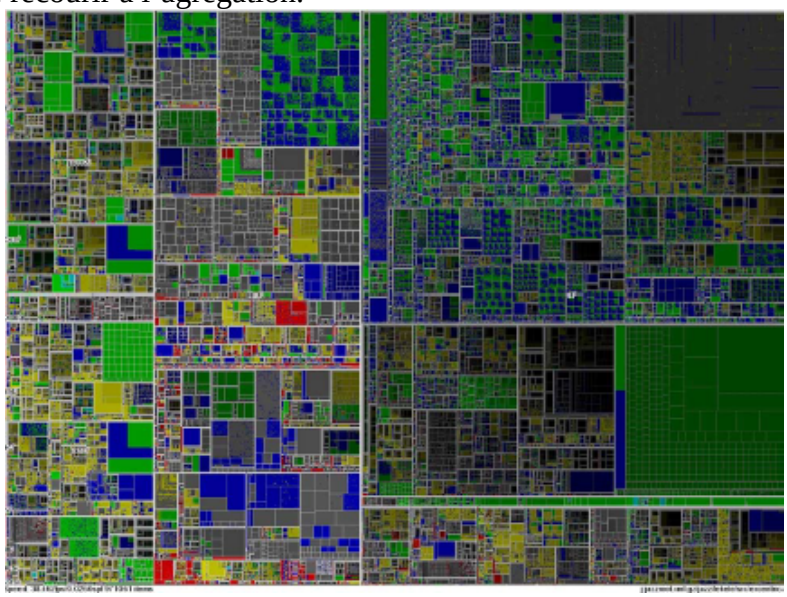
Les auteurs partent en effet du constat que lorsque les nœuds sont trop petits pour être affichés en détail (labels illisibles etc.), il n'est pas utile de faire un zoom géométrique continu. Lorsque l'utilisateur sélectionne un sous-arbre, celui-ci est déployé de manière à pouvoir explorer son contenu.



En substance, lorsque l'utilisateur exprime son intérêt vis-à-vis de certains nœuds de l'arborescence, ceux-ci sont déployés.

Card et Nation constatent le décalage croissant entre la quantité importante d'information que l'on souhaite visualiser et la petitesse de l'espace d'affichage disponible. Ils présentent dans [5] les *degree-of-interest trees*, un système représentant les hiérarchies sous forme de nœuds – liens et qui combine trois catégories de zoom pour déterminer la partie la plus pertinente de la hiérarchie à afficher dans l'espace disponible. Ils utilisent à la fois un zoom géométrique, sémantique et une notion de degré d'intérêt si bien que les nœuds situés à l'extérieur de la zone focale sont agrégés tandis que les nœuds sous le focus sont décomposés en leurs parties constitutives.

Dans [13], Fekete et Plaisant ont montré que le matériel dont nous disposons actuellement sur le marché permet d'afficher jusqu'à un million d'items à l'aide de cartes graphiques accélérées sur un écran de résolution 1600x1200. Des optimisations particulières permettent d'interagir avec cette masse de données de manière fluide. Au-delà de cette limite, il est indispensable de recourir à l'agrégation.



Un treemap affichant une arborescence de 970 mille fichiers.

5 Représentations des domaines des variables

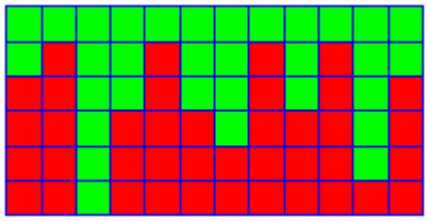
5.1 Représentation matricielle

Dans le cadre du projet Discipl, une représentation matricielle [9] a été proposée pour afficher le domaine des variables. On se place dans le cadre de domaines discrets. Le domaine d'une variable est représenté par une ligne ou une colonne dont les cellules sont colorées en rouge lorsque la valeur est retirée du domaine et en vert sinon. (cf. la figure ci-dessous à gauche) Une variante affecte une couleur de plus en plus sombre selon l'ancienneté du retrait. (cf. la figure ci-dessous à droite)

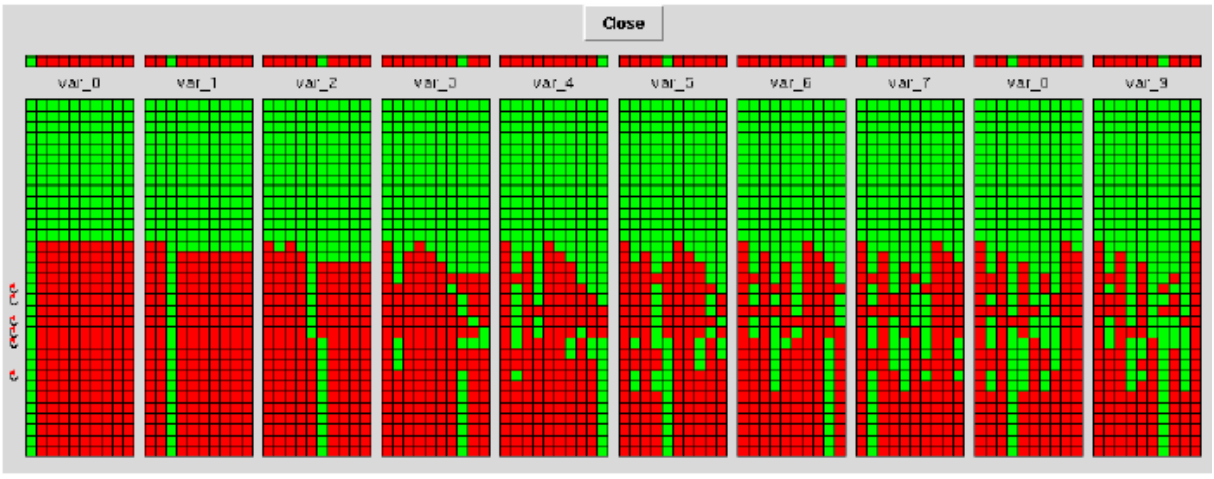
Vue instantanée du domaine fini d'une variable
Les cellules rouges correspondent aux valeurs retirées.

Vue instantanée du domaine d'une variable
La couleur traduit l'ancienneté du retrait

En juxtaposant les états successifs des domaines des variable, nous pouvons visualiser l'historique de la variable comme sur la figure suivante.



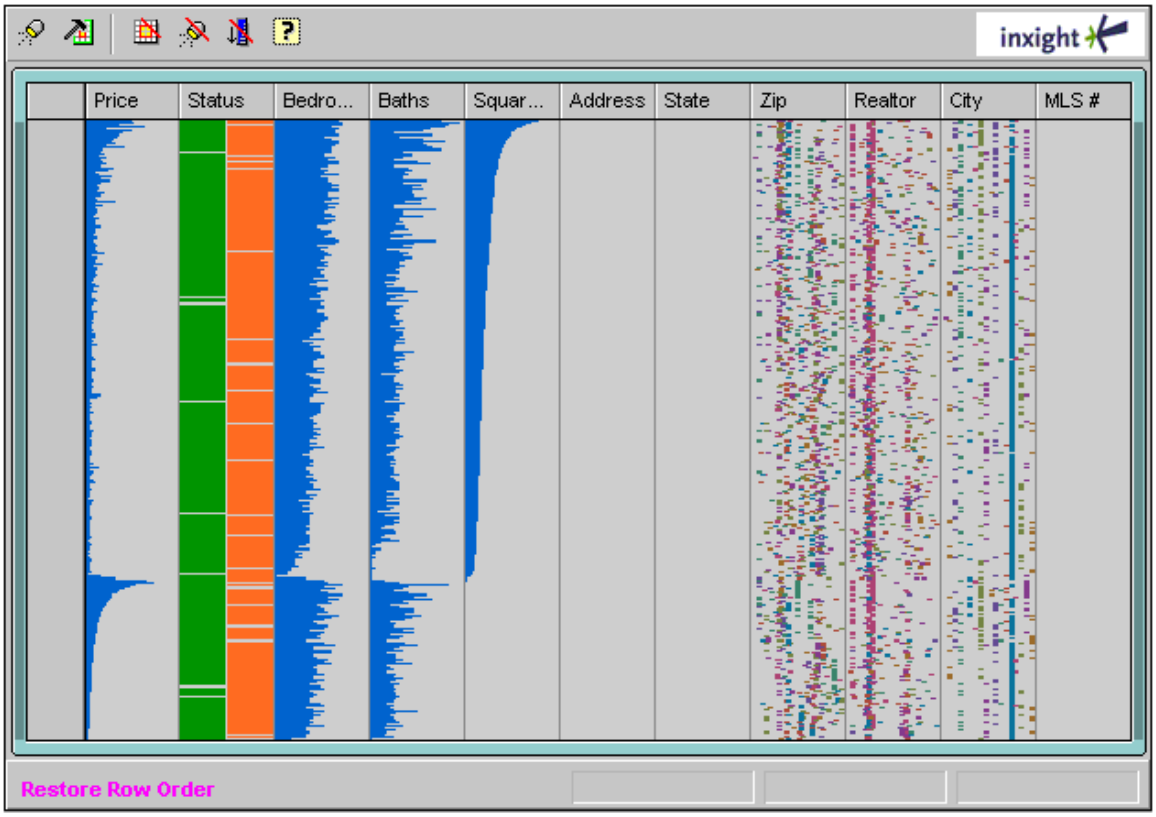
Historique du domaine d'une variable



Historique de plusieurs variables

De même, on peut juxtaposer les historiques de plusieurs variables pour aboutir à une vue d'ensemble comme sur la figure ci-dessus. Par ailleurs, on peut partir de cette représentation matricielle et construire une représentation 3D par la juxtaposition de l'historique de chacune des variables.

5.2 Les Tables Lens



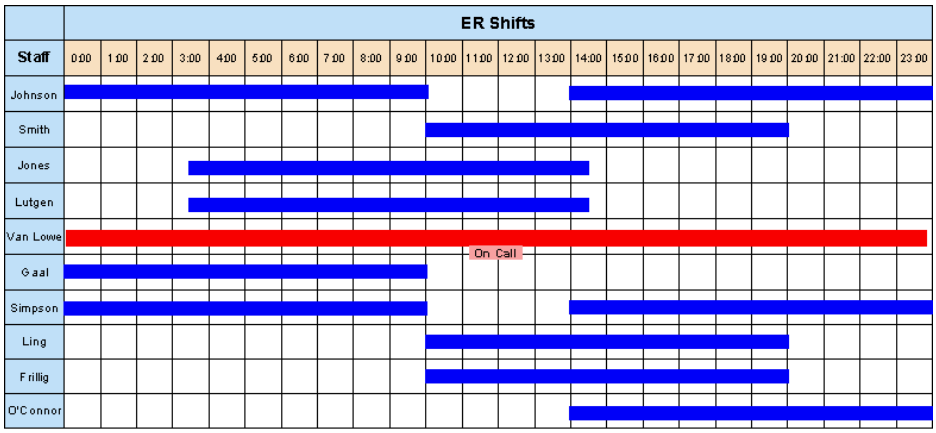
Ce type de représentation [4, 20] permet de visualiser des données multidimensionnelles représentant chaque dimension par une colonne. On peut alors déceler des corrélations entre les diverses dimensions et l'impact qu'elles peuvent avoir les unes sur les autres. Pour ce qui nous concerne, il peut être intéressant de représenter les différentes variables côte à côte de manière à voir l'évolution de leur domaine au cours de l'exécution. Bien entendu, on ne pourra visualiser qu'une vingtaine de colonnes au plus simultanément. Le temps est représenté verticalement sur ce type de figure.

6 Représentations ad-hoc

Dans cette partie, nous n'avons pas la prétention de répertorier toutes les techniques qui existent car il y en a pratiquement autant que de domaines d'application spécifiques. Néanmoins, nous citons deux représentations répandues : les diagrammes de Gantt et de Pert.

6.1 Le diagramme de Gantt

Ce diagramme est souvent utilisé dans les problèmes d'ordonnancement ou d'allocation de ressources.



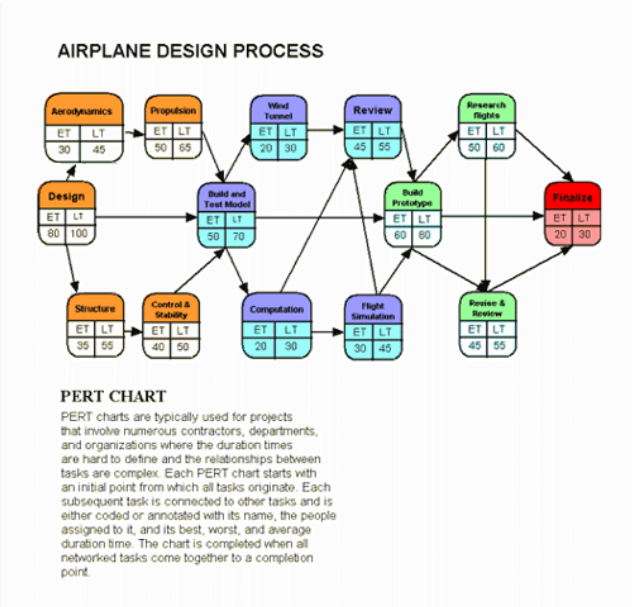
GANTT CHART
Gantt charts are often used to schedule individuals in the health care profession and in the military.

(source : http://www.smartdraw.com/)

Il est souvent utilisé pour la planification d'opérations de production notamment l'affectation de tâches à des machines.

6.2 Le diagramme de Pert

Ce diagramme sert pour les projets impliquant plusieurs acteurs avec des durées difficiles à définir et des relations complexes entre les tâches.



PERT CHART
PERT charts are typically used for projects that involve numerous contractors, departments, and organizations where the duration times are hard to define and the relationships between tasks are complex. Each PERT chart starts with an initial point from which all tasks originate. Each subsequent task is connected to other tasks and is either coded or annotated with its name, the people assigned to it, and its best, worst, and average duration time. The chart is completed when all networked tasks come together to a completion point.

7 Références

1. Barlow, T. & Nevil, P, A Comparison of 2-D Visualizations of Hierarchies, *Proceedings of IEEE Symposium on Information Visualization 2001* (Infovis'01), 22-23 October 2001, San Diego, CA, USA. IEEE Computer Society pp. 131-138.
2. Becker, R. A., Eick, S. and Miller, E. O., 1995. *Visualizing Network Data. IEEE Transactions on Visualization and Computer Graphics*, 1 (1 MARCH), 16–28.
3. Bruls, Huizing & van Wijk, Eindhoven, Squarified Treemaps, *Eurographics/IEEE TVCG 2000 Symposium*, University of Technology, Dept. of Mathematics and Computer Science, The Netherlands, 2000.
4. Card, Mackinlay & Shneiderman, Readings in Information Visualization, Morgan Kaufmann 1999.
5. Card, S. K., Nation, D. Degree-of-interest trees: a component of an attention-reactive user interface. *Advanced Visual Interfaces* (AVI 2002); 2002 May 22-24; Trento; Italy.
6. Carpendale, Cowperthwaite and Fracchia, Distortion Viewing Techniques for 3-dimensonal Data, INFOVIS '96.
7. Carrière and Kazman, Interacting with Huge Hierarchies: Beyond Cone Trees (1995)
8. A Cockburn and B McKenzie. An Evaluation of Cone Trees *People and Computers XIV: British Computer Society Conference on Human Computer Interaction 2000*, pages 425-436. Springer-Verlag
9. Carro and Hermenegildo, Some Design Issues in the Visualization of Constraint Logic Program Execution, AGP'98, Spain.
10. Di Battista, Eades, Algorithms for Drawing Graphs: an Annotated Bibliography, Computational Geometry: Theory and Applications, 4 (5), 1994, 235-282.
11. Di Battista, Eades, Tamassia and Tollis, Graph Drawing Algorithms For the Visualization of Graphs, Prentice-Hall, 1999.
12. Eades, Feng and Nagamochi, Drawing Clustered Graphs on an Orthogonal Grid, Journal of Graph Algorithms and Applications, *1999.*
13. Fekete, J.-D. and Plaisant, C., Interactive Information Visualization of a Million Items. *Proceedings of IEEE Symposium on Information Visualization 2002* (InfoVis 2002), IEEE Press, pages 117-124, Boston, USA, Octobre 2002.
14. Furnas and Bederson, Space-Scale Diagrams: Understanding Multiscale Interfaces, Human Factors in Computing Systems, *CHI'95 Conference Proceedings*, ACM Press, pp. 234-241.
15. Furnas and Zhang, MuSE: A Multi-Scale Editor, *Proceedings of the UIST'98 Symposium*, ACM Press, 1998.
16. Garg, Goodrich and Tamassia, Planar Upward Tree Drawings with Optimal Area, *Internat. J. Comput. Geom. Appl.*, 6(3):333-356, 1996.
17. Ghoniem, M. and Fekete, J.-D., 2002. Visualisation de graphes de co-activité par matrices d'adjacence. In Proceedings of IHM 2002, International Conference Proceedings Series, ACM 2002, Poitiers, France, 279–282.
18. Herman, Melançon and Scott Marshall, Graph Visualization and Navigation in Information Visualization : a Survey, IEEE 2000.
19. Herman, Delest and Melançon, Tree visualization and navigation clues for information visualization, INS-R9806 Marcg 31, 1998.
20. Inxight Corp., http://www.inxight.com/demos/tl/mls.html.
21. Johnson & Shneiderman, Treemaps : A Space-Filling Approach to the Visualization of Hierarchical information Structures, *Proceedings of IEEE Visualization 91*, IEEE Computer Soc. Press, Los Alamitos, California, October 1991, pp. 189-194.

22. Kimelman, Leban, Roth and Zernik, Reduction of Visual Complexity in Dynamic Graphs, DIMACS Graph Drawing 94.
23. Lamping and Rao, Laying Out and Visualizing Large Trees Using a Hyperbolic Space. Proceedings of UIST'94, pp. 13-14.
24. Melançon and Herman, Circular drawings of rooted trees, INS-R9817 December 31, 1998.
25. Munzner, H3: Laying out Large Directed Graphs in 3D Hyperbolic Space, *Proceedings of 1997 IEEE Symposium on Information Visualization* (InfoViz'97), IEEE CS Press, pp. 2-10, 1997.
26. Plaisant, C., Grosjean, J., Bederson, B. B.. SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation, *Proceedings 2002 IEEE Symposium on Information Visualization* (Infovis 2002), IEEE Computer Soc. Press, Boston, Massachusetts, 28-29 October 2002, pp. 57-64.
27. Rao & Card, The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information, *Proceedings of CHI'94*, 1994.
28. Reingold and Tilford, Tidier Drawing of Trees, *IEEE Transactions on Software Engineering*, Vol SE-7, No. 2, pp. 223-228, 1981.
29. Robertson, Mackinlay, and Card, Cone Trees, Animated 3D Visualizations of Hierarchical Information, *Proceedings of CHI'91*, pp. 189-194.
30. Schaffer, Zuo, Greenberg, Bartram, Dill, Dubs and Roseman, Navigating Hierarchically Clustered Networks through Fisheye and Full-zoom Methods, *ACM Transactions on Computer-Human Interaction*, Vol. 3, No. 2, pp. 162-188, 1996.
31. Shneiderman, Tree Visualization with Tree-maps: A 2-D space-filling approach. *ACM Transactions on Graphics*, January 1992.
32. Stasko, J., Catrambone, R., Guzdial, M. and McDonald, K.. An evaluation of space-filling information visualizations for depicting hierarchical structures. *International Journal of Human-Computer Studies*, 53(5): 663–694, November 1998.
33. Tamassia, Graph Drawing, CRC Handbook of Discrete and Computational Geometry, 1997.
34. Tamassia, On Embedding a Graph in the Grid with the Minimum Number of Bends, SIAM J. Comput., 16, no. 3, 421-444, 1987.
35. Turo & Johnson, Improving the Visualization of Hierarchies with Treemaps: Design Issues and Experimentation, *3rd IEEE Conference on Visualization*, 1992.
36. Van Wijk & Van de Wetering, Cushion Treemaps – visualization of hierarchical information, *Proceedings 1999 IEEE Symposium on Information Visualization* (InfoVis'99), pp. 73-78, October 1999.
37. Yang, J., Ward, M. O. and Rundensteiner, E. A., InterRing: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures, *Proceedings 2002 IEEE Symposium on Information Visualization* (Infovis 2002), IEEE Computer Soc. Press, Boston, Massachusetts, 28-29 October 2002, pp. 77-84.