Cours: VISUALISATION DE DONNEES

Master MIV, 2022/2023

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

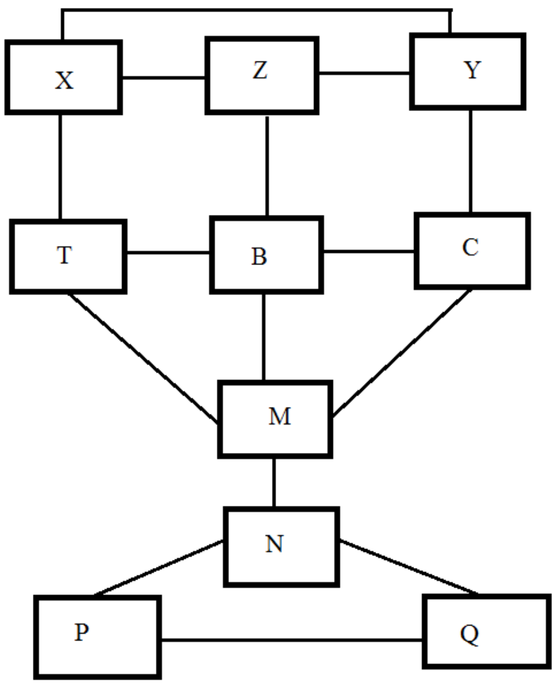
Sommaire

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.1 Intérêts et applications

Graphe G=(V,E)
Où V est l'ensemble des nœuds (vertex)
E est l'ensemble des arcs (Edges)

X	Z Y T
Z	X Y B
Y	X C Z
T	B M X
....	....



Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.1 Intérêts et applications

Le besoin de visualisation de graphes a été ressenti suite à la demande des différents secteurs: militaire, transport, communication.

La génération automatique de graphes est devenu crucial surtout pour de large graphes (plus de quelques milliers de nœuds).

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.1 Intérêts et applications

Soit le réseau social suivant [1]:

Farid-Aadil, Latif-Aadil, Farid-Latif, Carol-Andre, Carol-Fernando, Carol-Diane, Andre-Diane, Farid-Izdihar, Andre-Fernando, Izdihar-Mawsil, Andre-Beverly, Jane-Farid, Fernando-Diane, Fernando-Garth, Fernando-Heather, Diane-Beverly, Diane-Garth, Diane-Ed, Beverly-Garth, Beverly-Ed, Garth-Ed, Garth-Heather, Jane-Aadil, Heather-Jane, Mawsil-Latifg.

[1] https://math.nist.gov/mcsd/Seminars/2012/2012-05-07-Hu-presentation.pdf

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

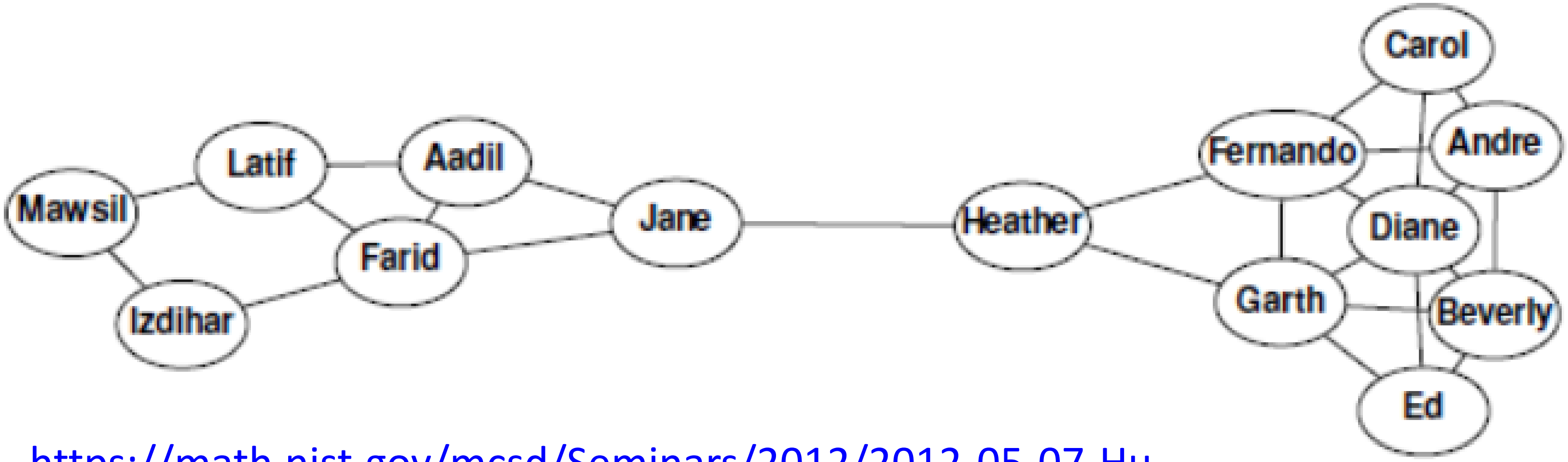
4.1 Intérêts et applications

On ne peut pas inférer de structure dans ce réseau.

Après visualisation, on peut constater que Jane et Heather sont les personnes qui connecte les deux réseaux.

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.1 Intérêts et applications



https://math.nist.gov/mcsd/Seminars/2012/2012-05-07-Hu-presentation.pdf

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.1 Intérêts et applications

Applications:

- Tournois
- Organigrammes
- Généalogie
- Interactions biologiques (gènes, protéines)
- Réseaux informatiques, routiers, aériens, communications, Réseaux sociaux
- Simulation et modélisation
- Conception de circuits intégrés
- ………….

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.1 Intérêts et applications

Applications:

Réseaux hydrographiques

Racines des arbres

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.1 Intérêts et applications

Applications:

Réseaux transport

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.1 Intérêts et applications

Intérêts

On peut citer quelques uns:

● Visualiser la relation entre individus/objets (e.g., réseaux sociaux)

● Visualiser l'échange d'information (e.g., traffic aérien)

● Visualiser le fonctionnement d'algorithmes (e.g., réseaux de neurones, chaînes de Markov, réseaux bayésiens…)

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.1 Objectif

La visualisation d'un graphe avec peu de nœuds et de liens est simple: on peut choisir le positionnement des éléments à visualiser.

L'augmentation des tailles de graphes étudiés aujourd'hui conduit à rechercher des méthodes efficaces de visualisation.

On cherche à passer, pour un grand nombre de nœuds, d'un placement aléatoire initial et monochrome à une visualisation colorée et aérée

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.2 Attributs du graphes

Un graphe est défini par un ensemble V de nœuds et un ensemble E d'arêtes, les paramètres de la modélisation seront les attributs possibles pour les éléments de chacune de ces catégories.

Ainsi, pour les nœuds, les attributs sont:

- position
- taille
- couleur
- forme (circulaire ou non)

Pour les arêtes:

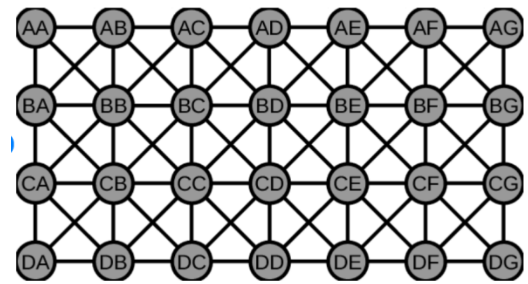
- forme (courbes, lignes droites ou brisées)
- position (chevauchements, écart angulaire)
- épaisseur du trait
- couleur

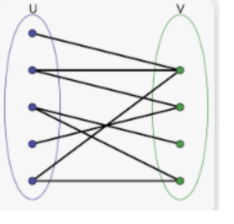
Chapitre 4: *VISUALISATION DE GRAPHES ET ARBRES*

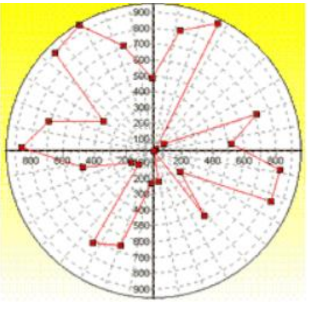
4.2 Méthodes de Visualisation de graphes

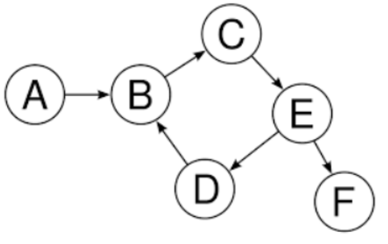
4.2.3 Typologie des graphes

- Homogènes
    - Grilles
- Hierarchiques
    - Arbres, Forêts
    - Graphes bipartis
- Cycliques
- Polaires









Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

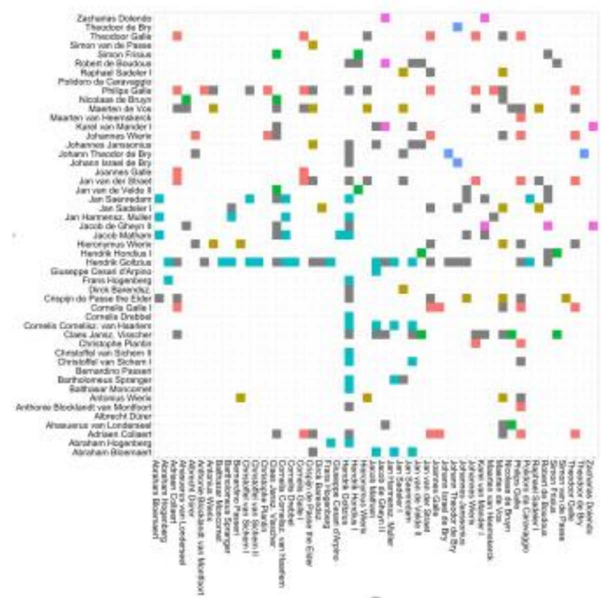
4.2 Méthodes de Visualisation de graphes

4.2.4 Représentation des graphes

- Matrice d'adjacence (a_{ij} est le nombre d'arêtes liant le sommet *i* au sommet *j*)

- Liste d'adjacence (est la liste des voisins de chaque sommet)

- Matrice d'incidence(-1 si l'arc x_j sort du sommet v_i, 1 si l'arc x_j entre dans le sommet v_i, 0 sinon)



Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.5 Difficultés majeures

- Gros graphes
    - Clustering
    - Approches hiérarchiques
- Dimension : essentiellement 2D
    - Réduction de dimension
- Layout
    - arêtes droites/courbes
    - méthodes mathématiques

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

G={V,E} graphe non orienté , |V| nombre de nœuds, |E| nombre d'arrêtes.
I ←→ j signifie que les nœuds i et j sont reliés.

L'objet de visualisation (laying out) d'un graphe est d'assigner une position à chaque nœud telle que le dessin obtenu donne une visualisation esthétique de la connexion entre nœuds.

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

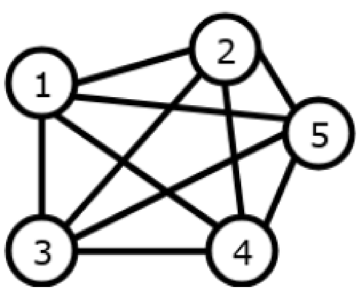
4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

a) Premiers algorithmes

Planarité d'un graphe et qualité de la visualisation

Un graphe est dit planaire s'il existe une représentation dans le plan telle qu'aucune arête n'en croise une autre. Un graphe est dit non planaire si *toutes* ses représentations dans le plan comportent *au moins un* croisement d'arêtes.



Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

a) Premiers algorithmes

Planarité d'un graphe et qualité de la visualisation

Très étudiée en théorie des graphes durant le 20e siècle pour son intérêt mathématique, est intervenue dans la partie « visualisation des graphes » à la fin des années 1970.

Les chercheurs ont eu l'intuition qu'elle pourrait avoir un lien avec la qualité d'une représentation, ou du moins la faculté des humains à l'interpréter. L'idée est la suivante : une représentation planaire permet de faire des images jolies et *immédiatement* compréhensibles.

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

a) Premiers algorithmes

Planarité d'un graphe et qualité de la visualisation

Des travaux ultérieurs ont effectivement montré qu'il y avait des corrélations significatives entre les erreurs de compréhension et la quantité de croisements d'arêtes. De même, les brisures de lignes sont également associées à des erreurs.

En somme, il est naturel de rechercher des représentations d'un graphe où l'on minimise :

- les croisements d'arêtes
- les lignes brisées

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

a) Premiers algorithmes

Algorithme de W. Tutte

Tutte, William Thomas. « How to draw a graph. » Proceedings of the London Mathematical Society 3.1 (1963): 743-767.

La discipline de la représentation de graphes naît véritablement en 1963 avec la publication d'un article fondateur, « How to draw a graph » .

Son auteur, **William Tutte**, est un mathématicien anglais brillant, qui était aux côtés d'Alan Turing pour déchiffrer les codes allemands à Bletchley Park.

Il introduit un algorithme barycentrique pour construire une représentation d'un graphe, capable de produire des dessins avec des lignes droites (qui se chevauchent éventuellement).

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).
a) Premiers algorithmes
algorithme de W. Tutte

L'algorithme prend en entrée un graphe G=(V,E)G=(V,E).

La sortie de l'algorithme est une représentation avec les lignes droites du dit graphe, appelé **p** (et on note **p(u)** la position de **u** dans **p**).

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).
a) Premiers algorithmes
algorithme de W. Tutte

1. on choisit un sous ensemble de sommets, A, de V
2. on choisit une position p(A) pour chaque sommet $a \in A$.
3. pour chaque sommet $u \in V - A$, on a :

$$p(u) = \frac{1}{deg(u)} \sum_{v \in N(u)} p(v) \quad \text{avec } N(u) \text{ l'ensemble des voisins de } u.$$

Ce qui donne un système de $2n$ équations :

$$\begin{cases} x(u) & = & \frac{1}{deg(u)} \sum_{v \in N(u)} x(v) \\ y(u) & = & \frac{1}{deg(u)} \sum_{v \in N(u)} y(v) \end{cases}$$

deg(u) est le nombre de connections du noeud **u**.

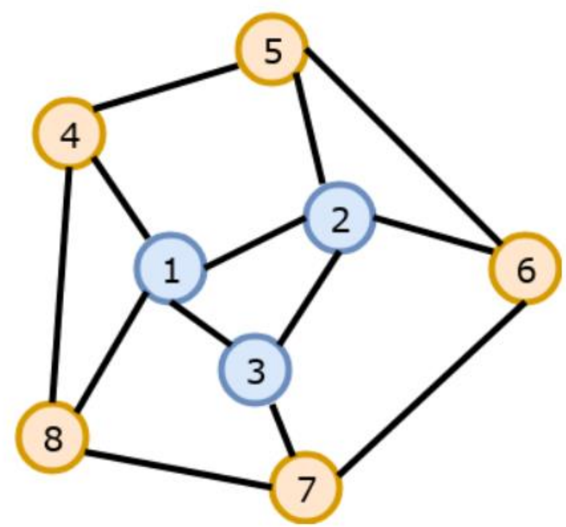
4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

a)Premiers algorithmes

Algorithme de W. Tutte

Les sommets 4 à 8 sont placés au début, on cherche en fonction de ceux-ci les positions des sommets 1 à 3.



1. Initialement, on choisit $A = 4, 5, 6, 7, 8$.
2. On choisit x_i et y_i pour $i \in 4, 5, 6, 7, 8$.
3. On doit ensuite trouver $x_1, y_1, x_2, y_2, x_3, y_3$ tels que :

$$\begin{cases} x_1 & = & \frac{1}{4}(x_2 + x_3 + x'_4 + x'_8) \\ x_2 & = & \frac{1}{4}(x_1 + x_3 + x'_5 + x'_6) \\ x_3 & = & \frac{1}{3}(x_1 + x_2 + x'_7) \end{cases}$$

$$\begin{cases} y_1 & = & \frac{1}{4}(y_2 + y_3 + y'_4 + y'_8) \\ y_2 & = & \frac{1}{4}(y_1 + y_3 + y'_5 + y'_6) \\ y_3 & = & \frac{1}{3}(y_1 + y_2 + y'_7) \end{cases}$$

Avec x'_i et y'_i les valeurs choisies à l'étape précédente.

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

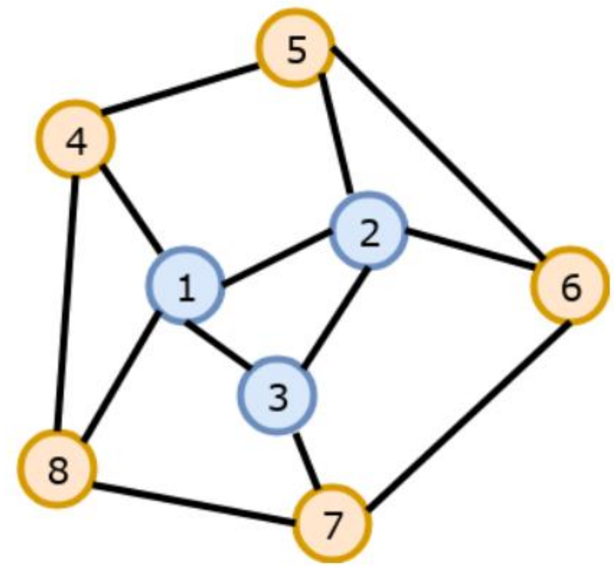
a) Premiers algorithmes

Algorithme de W. Tutte

$$\begin{cases} 4x_1 - x_2 - x_3 & = & x'_4 + x'_8 & = & c_1 \\ -x_1 + 4x_2 - x_3 & = & x'_5 + x'_6 & = & c_2 \\ -x_1 - x_2 + 3x_3 & = & x'_5 & = & c_3 \end{cases}$$

$$\begin{cases} 4y_1 - y_2 - y_3 & = & y'_4 + y'_8 & = & d_1 \\ -y_1 + 4y_2 - y_3 & = & y'_5 + y'_6 & = & d_2 \\ -y_1 - y_2 + 3y_3 & = & y'_5 & = & d_3 \end{cases}$$

Avec $c_1, c_2, c_3, d_1, d_2, d_3$ qui sont des constantes.



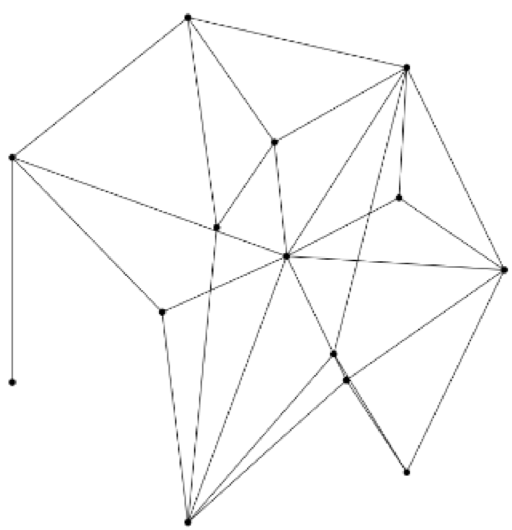
Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

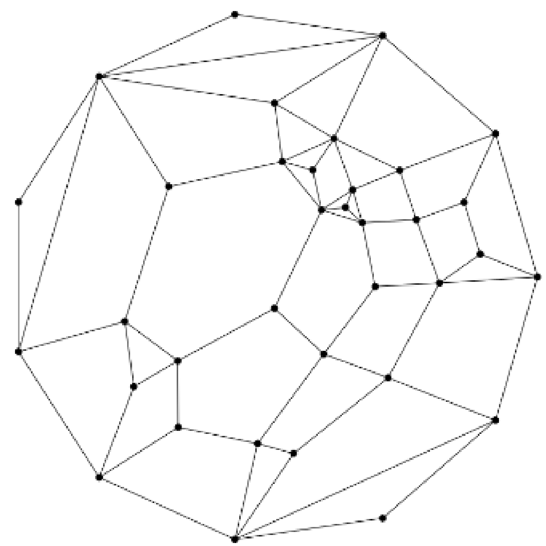
4.2.6 Algorithmes de visualisation de Graphes (networks).

a)Premiers algorithmes

Algorithme de W. Tutte



Graphe non planaire

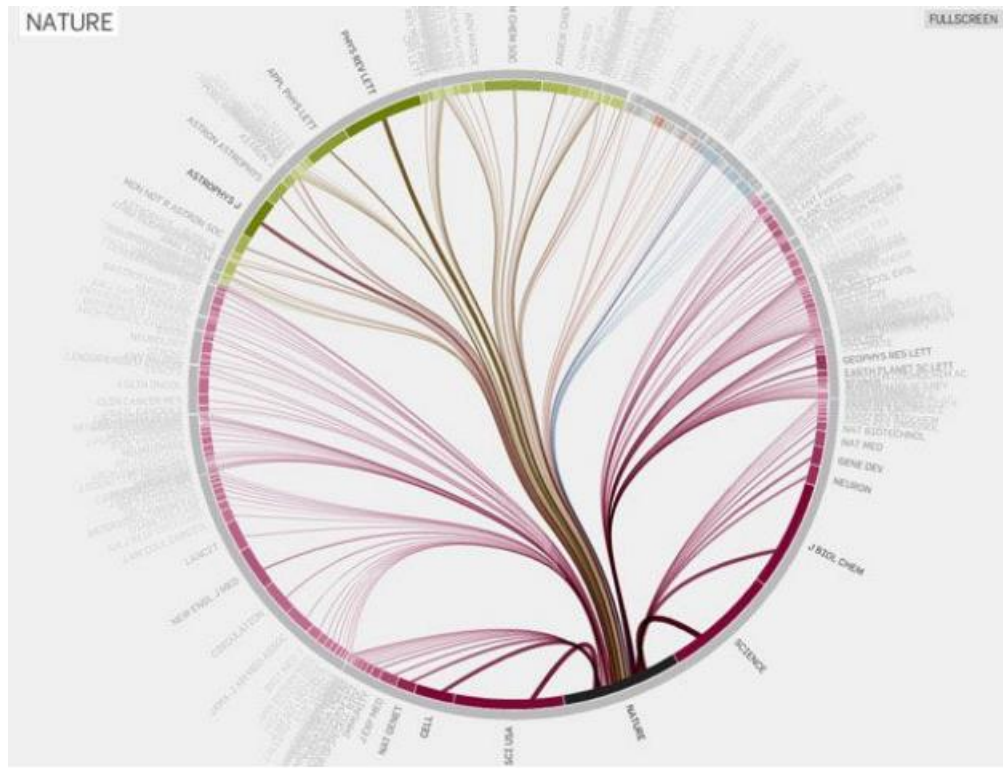


Graphe planaire

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents algorithmes

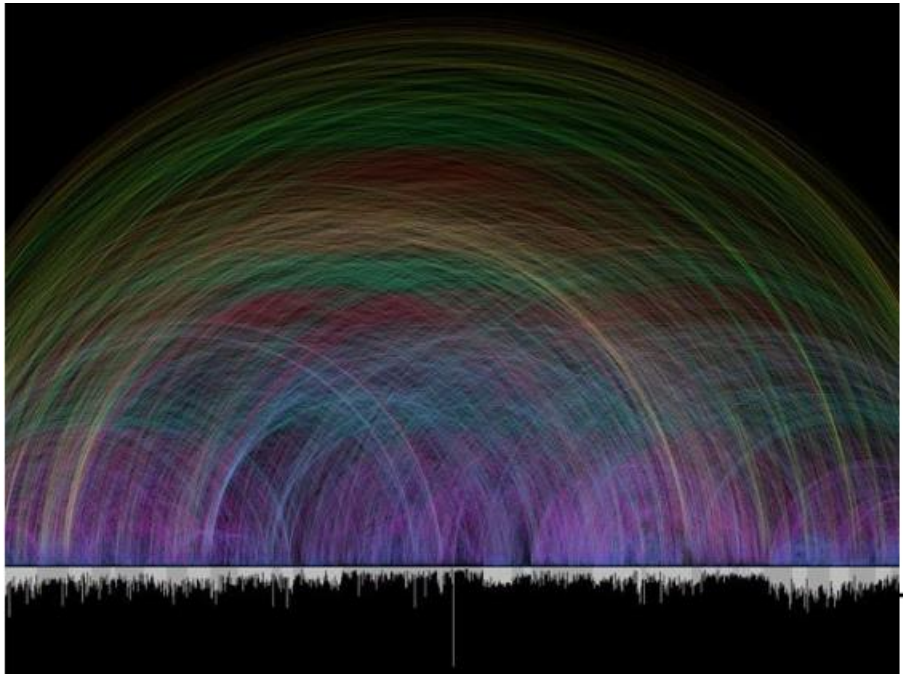
Chord diagrams



4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents algorithmes

Arc diagrams

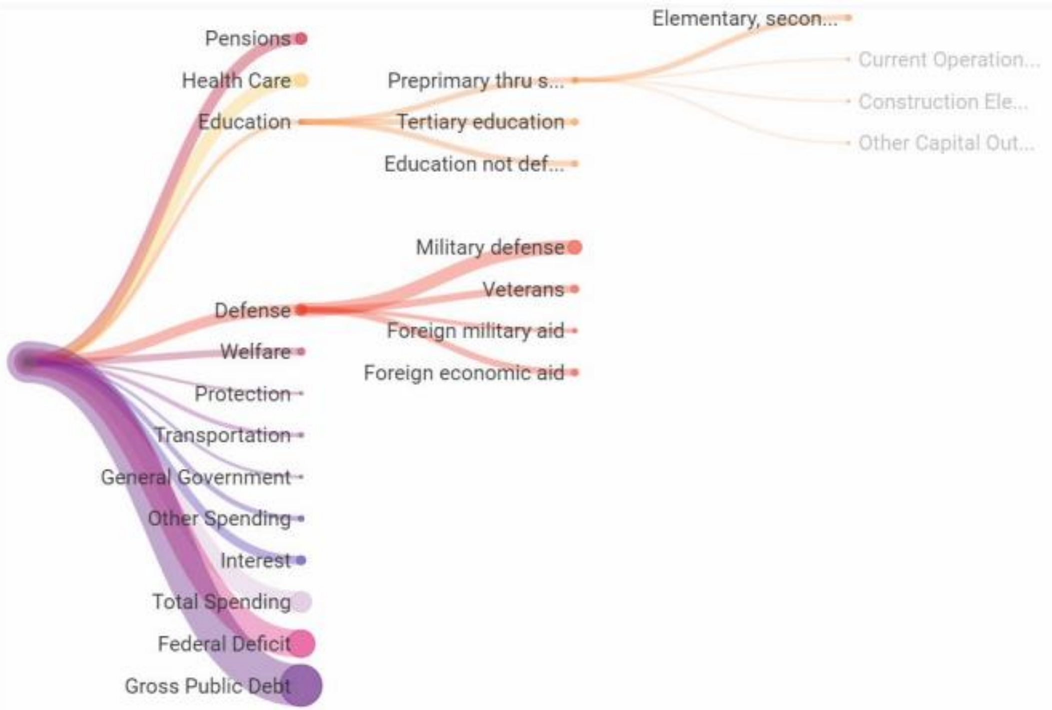


Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents algorithmes

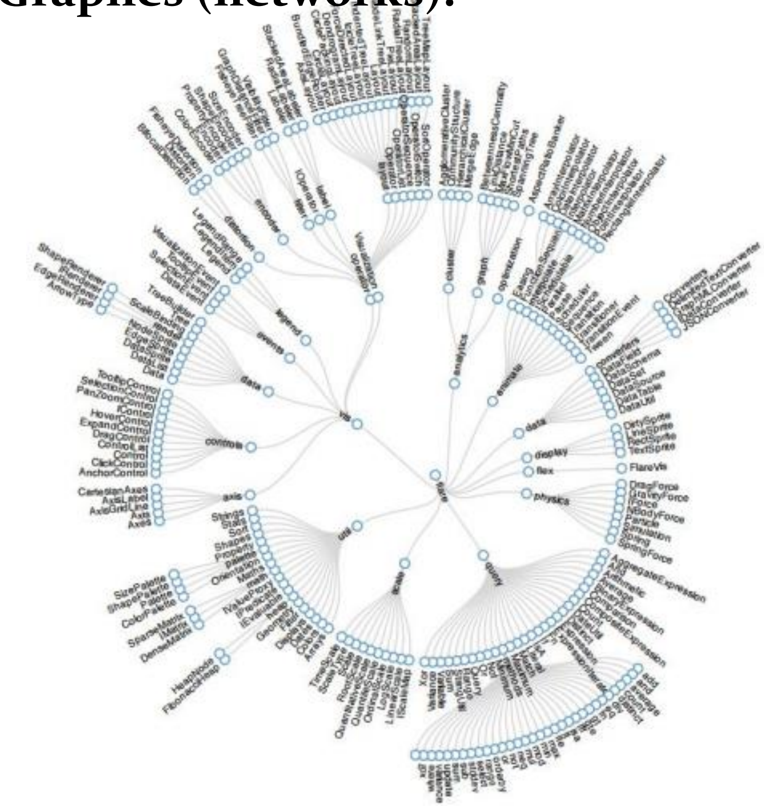
Weighted Tree



4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents algorithmes

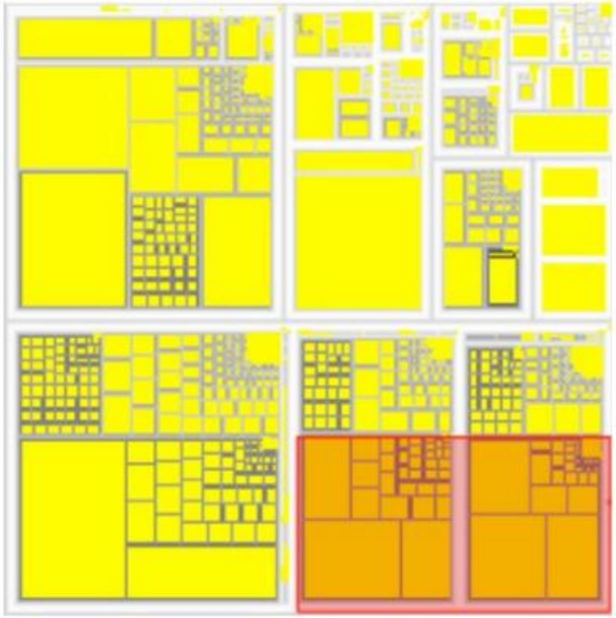
Radial Node Link



4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents algorithmes

TreeMap



Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents algorithmes

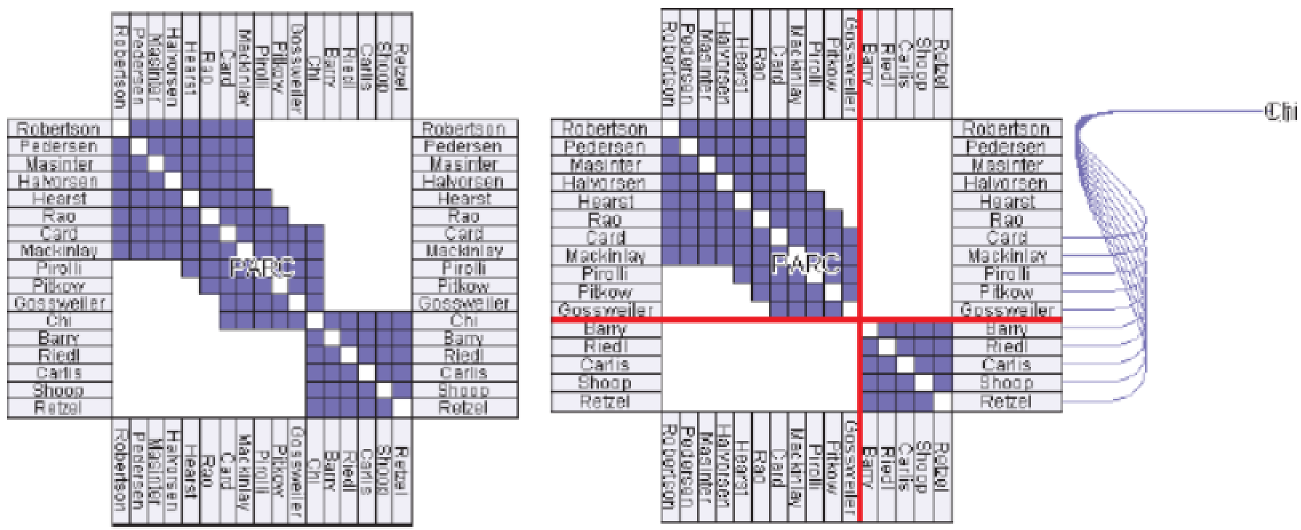
N. Henry, J. -D. Fekete and M. J. McGuffin, "**NodeTrix**: a Hybrid Visualization of Social Networks," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1302-1309, Nov.-Dec. 2007, doi: 10.1109/TVCG.2007.70582.

NodeTrix est une représentation hybride des réseaux basée sur le diagramme nœud-lien où les communautés peuvent être représentées sous forme de matrices. Les relations intracommunautaires utilisent la représentation de la matrice d'adjacence tandis que les relations intercommunautaires utilisent des liens normaux.

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents al



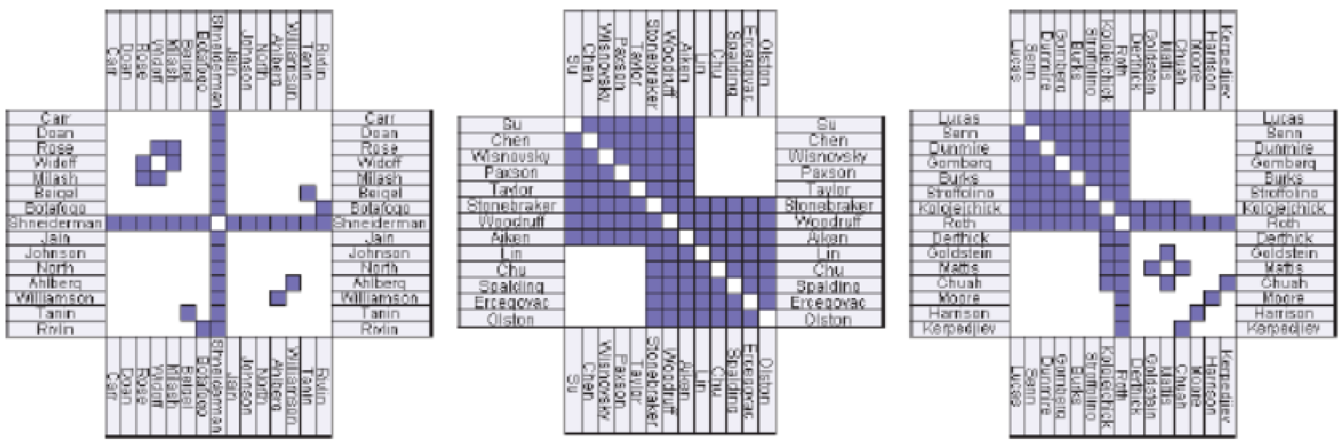
(a) PARC Community (b) Ed Chi's influence

Moving a node in and out of a matrix. In the second case, red lines indicate that the matrix is disconnected in two groups (upper left and lower right). Ed Chi is the bridge between these two groups.

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents algorithmes



(a) Cross pattern (b) Block pattern (c) Intermediate pattern

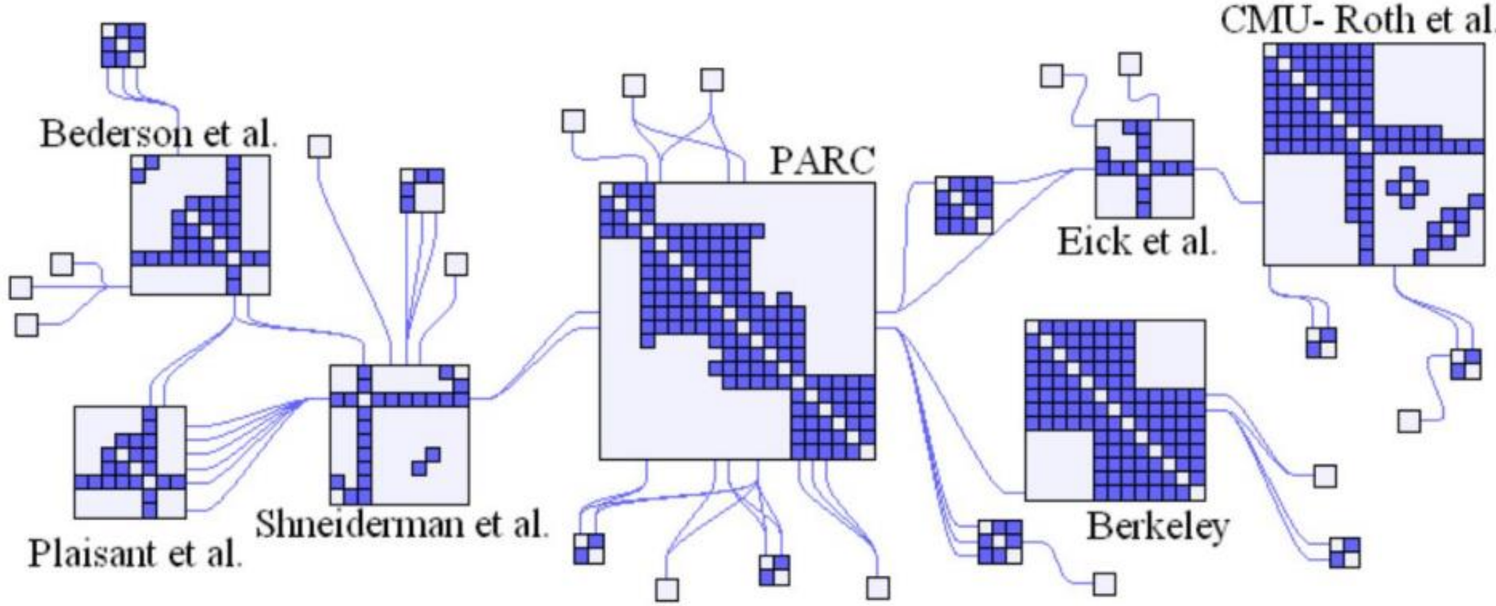
Three collaboration patterns: (a) Shneiderman and his collaborators, (b) Researchers at Berkeley, (c) Roth and his collaborators at CMU.

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Récents algorithmes

NodeTrix:

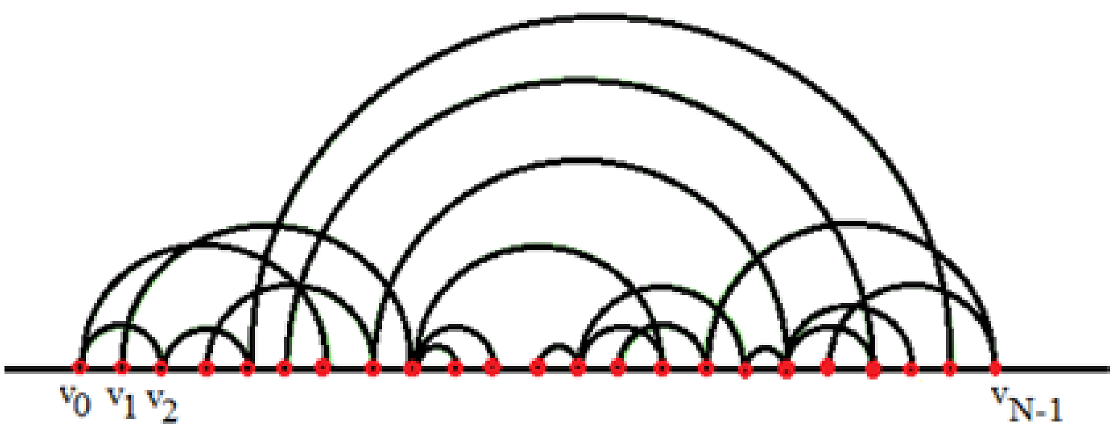


Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

Exemple d'algorithme: Arc Diagrams



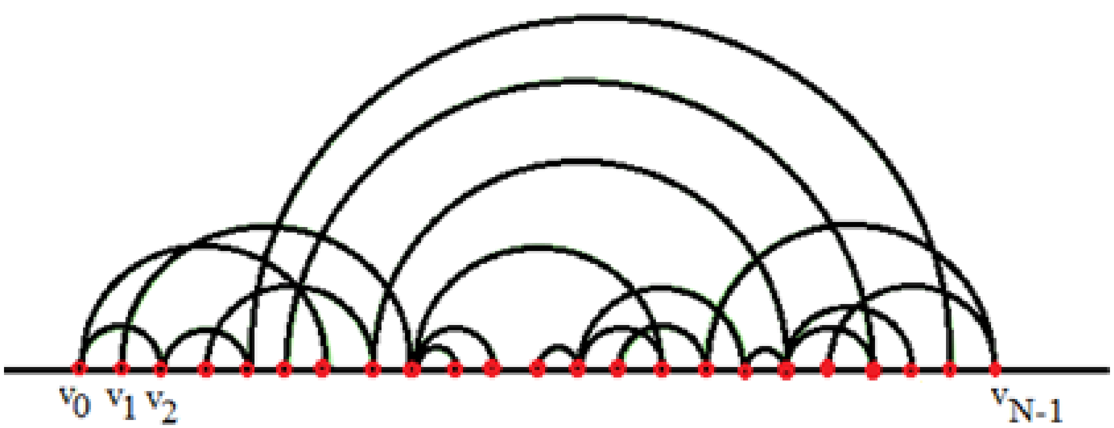
4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

Exemple d'algorithme: Arc Diagrams

Le but de cet algorithme est de construire un diagramme en arcs à partir d'un graphe G à N nœuds.

Un diagramme en arcs consiste à placer les nœuds (v_0, v_1, v_2, …, v_{N-1}) de G sur un axe et remplacer chaque arête par un arc les joignant comme indiqué par la figure 1 ci-dessous.



4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Exemple d'algorithme: Arc Diagrams

Plus la distance séparant deux nœuds connectés sur l'axe augmente, plus les arcs seront éloignés de l'axe, ce qui fait augmenter l'espace de visualisation. L'objectif est donc de trouver le placement des nœuds sur l'axe qui réduit les longueurs des arcs à dessiner.
Pour ce faire, il faudra **rapprocher** chaque nœud de ses voisins ce qui revient à minimiser la distance de chaque nœud du barycentre de ses voisins.

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).
Exemple d'algorithme: Arc Diagrams

Une mesure $d(v_k)$ est associée à v_k indiquant sa distance au barycentre de ses voisins (càd des nœuds auxquels il est connecté dans G). En supposant que les nœuds sont rangés dans un tableau T_i, $d(v_k)$ est calculé comme suit où v_k est le nœud considéré, nb est le nombre de voisins **y compris $\mathbf{v_k}$** et i_j est l'indice du voisin numéro j dans T_i.

$$d(v_k) = \frac{1}{nb} \sum_{j=1}^{j=nb} i_j$$

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).
Exemple d'algorithme: Arc Diagrams

Soit T_0 le tableau contenant initialement les nœuds (v_0, v_1, v_2, ..., v_{N-1}) disposés dans cet ordre sur l'axe. T_0 sera transformée en T_1, puis T_1 en T_2 et ainsi de suite. La position d'un nœud v_k dans le tableau T_i ($i>1$) dépend de la position de ses voisins dans T_{i-1}.

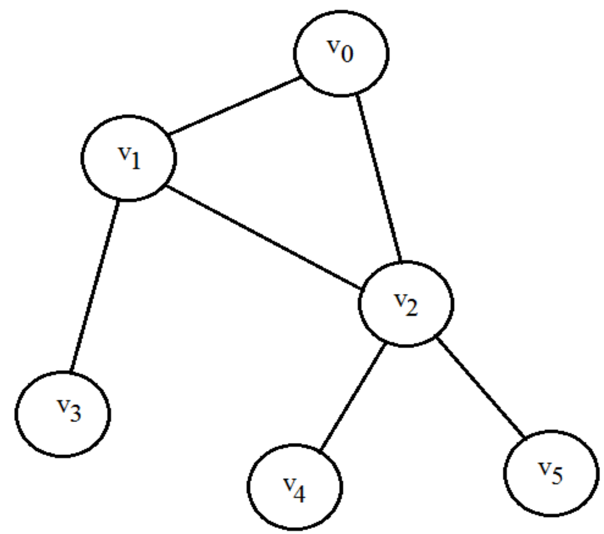


Figure. Un graphe de 6 nœuds

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Exemple d'algorithme: Arc Diagrams

Soit G un graphe à 6 nœuds comme indiqué par la figure 2. Evaluez pour chaque v_k la distance $d(v_k)$ sachant que

$T_0 =$

v_0	v_2	v_4	v_5	v_3	v_1

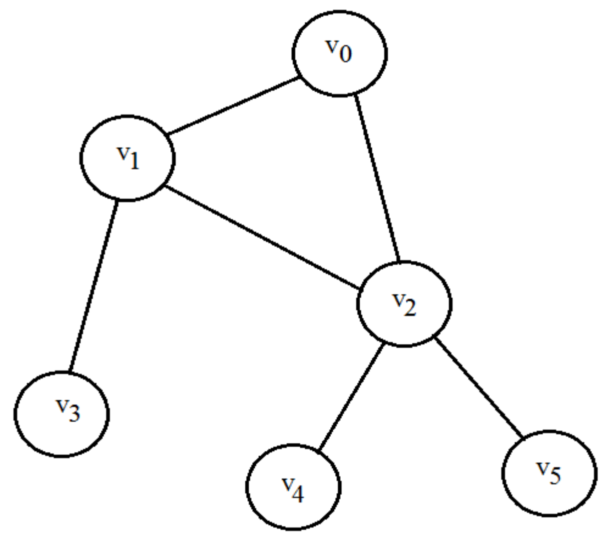


Figure. Un graphe de 6 nœuds

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks).

Exemple d'algorithme: Arc Diagrams

Si T_0 est transformé en T_1 tel que $T_1 =$

v_4	v_0	v_5	v_2	v_1	v_3

puis T_1 est transformé en T_2 tel que T_2=

v_4	v_2	v_5	v_0	v_1	v_3

puis T_2 est transformé en T_3 tel que T_3 =

v_4	v_5	v_2	v_0	v_1	v_3

Que peut-on déduire sur les distances $d(v_k)$ dans les tableaux T_i . Que peut-on dire de cette transformation ?

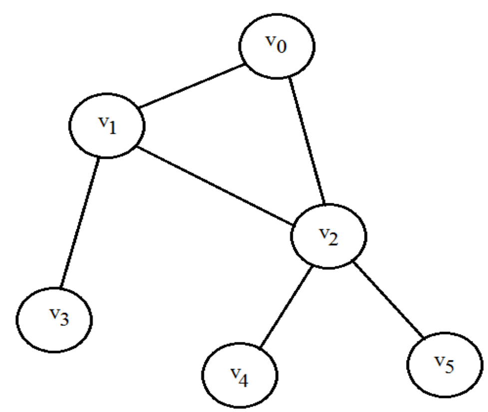


Figure. Un graphe de 6 nœuds

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

Mathématiques des visualisations

Graphe fondé sur les forces (Force-directed graph)

Système masse-resort (1984)
Le principe de ces algorithmes est de simuler un modèle physique masse-ressort. Les sommets du graphe sont représentés par des objets physiques qui sont reliés par un ressort si les sommets du graphe correspondants sont reliés par une arête.

4.2 Méthodes de Visualisation de graphes

4.2.6 Algorithmes de visualisation de Graphes (networks). Exemple d'algorithme

Arc Diagrams: Visualizing Structure in Strings

Martin Wattenberg
IBM Research
One Rogers Street
Cambridge MA 02142

4.2 Méthodes de Visualisation de graphes

Mathématiques des visualisations

Algorithme

Répéter n fois :
Pour chaque sommet v :
Calculer les forces d'attraction appliquées sur le sommet v par ces voisins
Calculer les forces de répulsion appliquées sur le sommet v par tous les autres Pour chaque sommet v :
Déplacer le sommet v en fonction de ces forces

Pour un nœud v,
$ftotale(v)$ = $fattr(v)$ + $frep(v)$

4.2 Méthodes de Visualisation de graphes

Spring-electrical Model

Proposé par Peter Eades en 1984.

Pour dessiner un graphique, il remplace les sommets par des anneaux d'acier et chaque arrête par un ressort pour former un système mécanique. Les sommets sont placés dans une disposition initiale ce qui permet aux forces des ressorts sur les anneaux de déplacer le système à un état énergétique minimal"

4.2 Méthodes de Visualisation de graphes

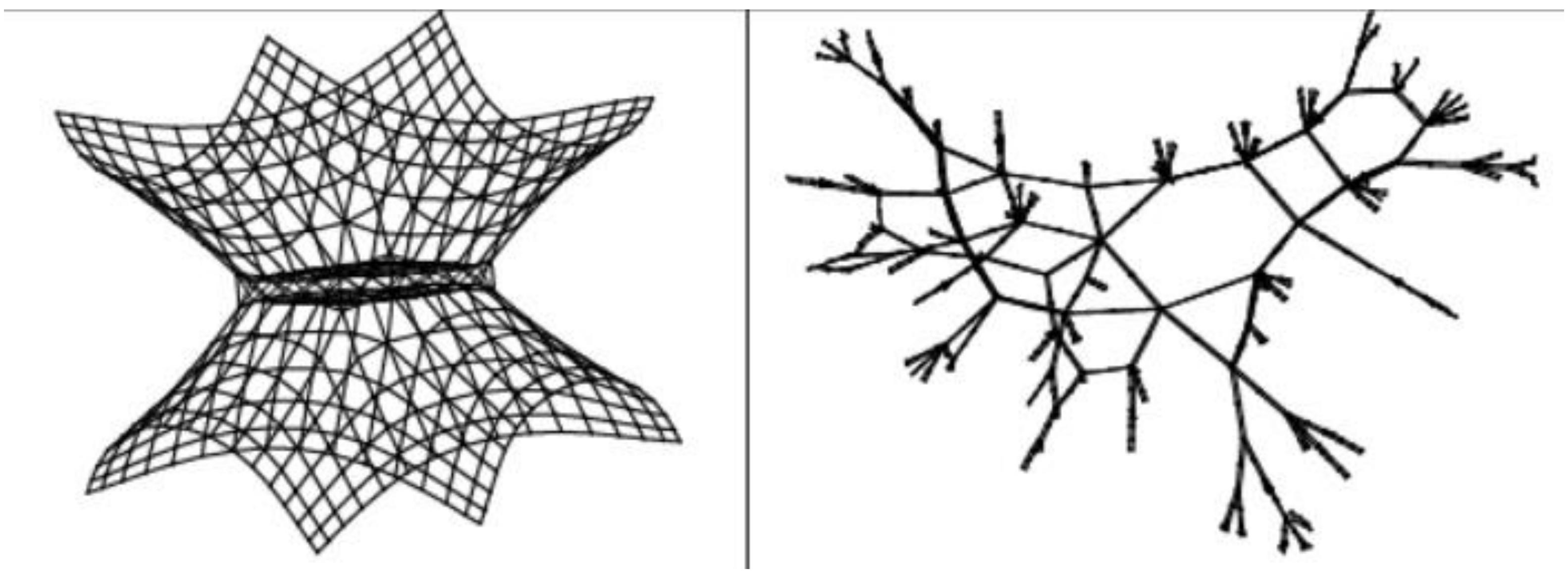
Spring-electrical Model

Algorithm 1 ForceDirectedAlgorithm(G, x, tol, K)

input: graph $G = \{V, E\}$, initial positions x, tolerance tol, and nominal edge length K
set $step$ = initial step length
repeat
    $x^0 = x$
    for $(i \in V)$ {
        $f = 0$ // f is a 2/3D vector
        for $(j \leftrightarrow i,\ j \in V)$ $f \leftarrow f + F_a(i, j)$ // attractive force, see equation (1)
        for $(j \neq i,\ j \in V)$ $f \leftarrow f + F_r(i, j)$ // repulsive force, see equation (2)
        $x_i \leftarrow x_i + step * (f/||f||)$ // update position of vertex i
    }
until $(||x - x^0|| < tol * K)$
return x

4.2 Méthodes de Visualisation de graphes

Spring-electrical Model

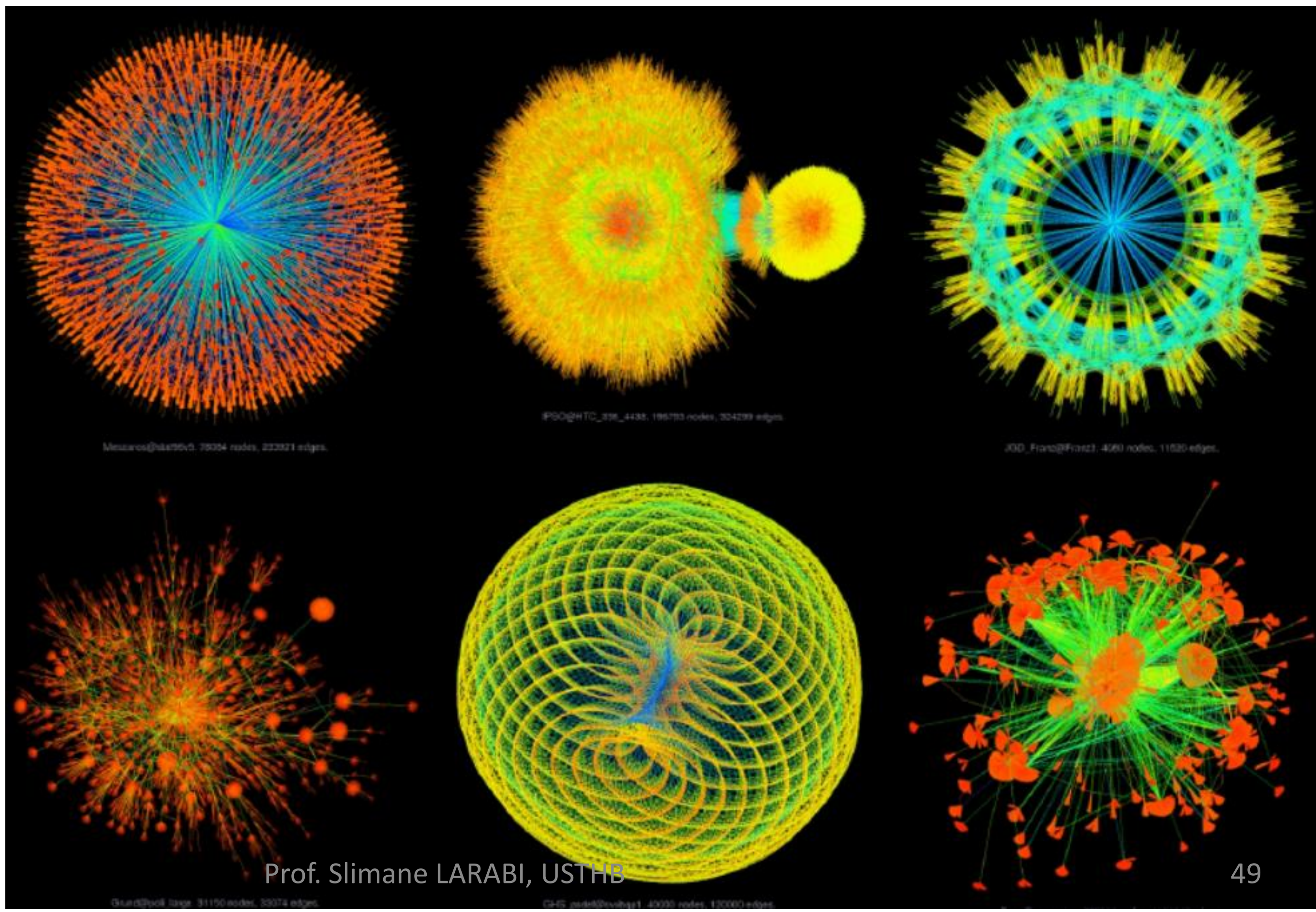


Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation de graphes

Spring-electrical Model

Exemple de dessin de larges graphes



4.3 Méthodes de Visualisation d'arbres

Space-Filling Methods

Occupent entièrement l'espace de visualisation.
Les approches existantes, l'affichage des niveaux hiérarchiques est soit **rectangulaire** ou **radiale**.

- Treemap: est la plus répandue pour l'affichage rectangulaire.

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

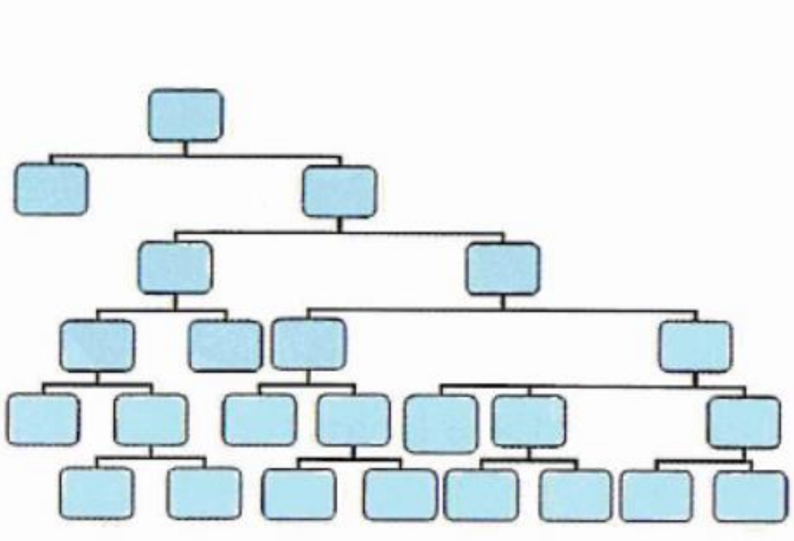
4.3 Méthodes de Visualisation d'arbres

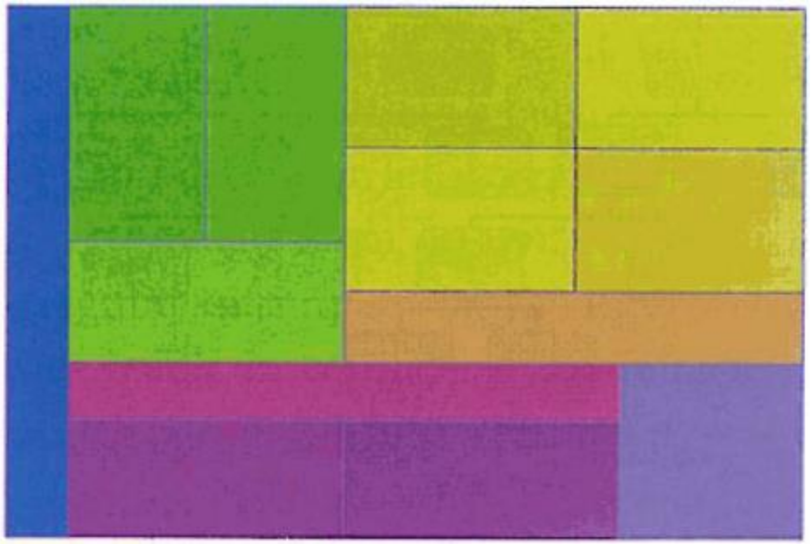
Space-Filling Methods

Pour un affichage basique de la treemap, un rectangle est divisé récursivement en parties alternativement en horizontal et vertical en se basant sur la valeur cumulée des nœuds des sous arbres à un niveau donné

4.3 Méthodes de Visualisation d'arbres

Space-Filling Methods



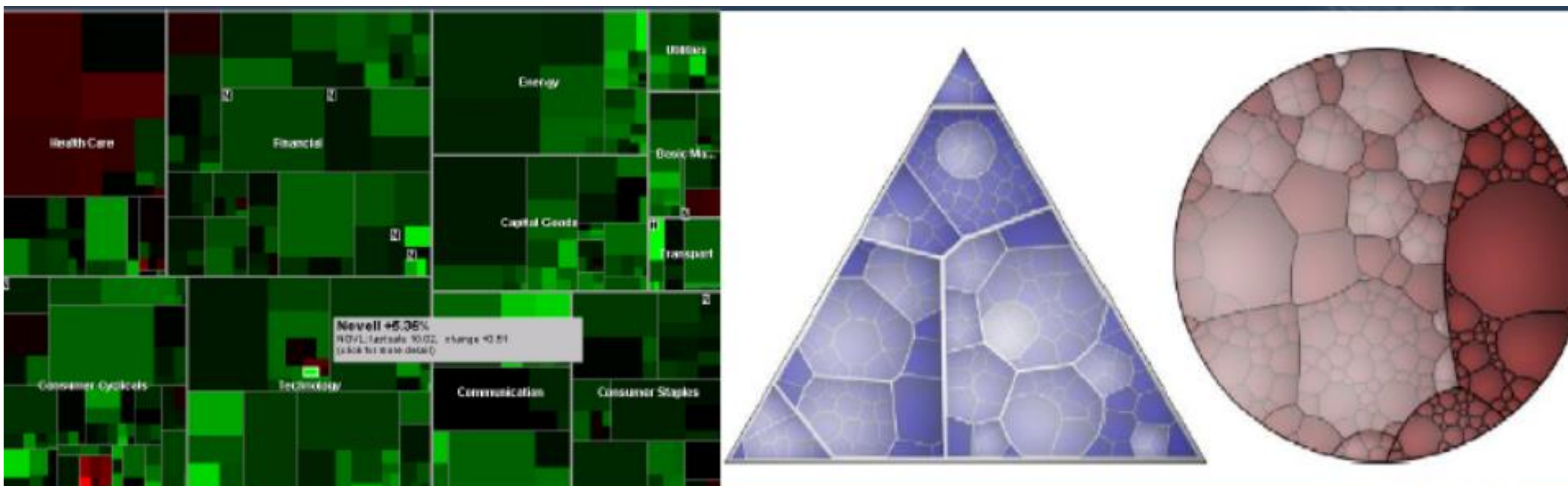


4.3 Méthodes de Visualisation d'arbres

Space-Filling Methods

Treemaps

Encodage de la hiérarchie à l'aide d'une enceinte spatiale
Technique de remplissage.



4.3 Méthodes de Visualisation d'arbres

Space-Filling Methods

Pseudo code pour visualiser une treemap.

```
Main Program
Begin
        Width = width of rectangle
        Height = height of rectangle
        Node = root node of the tree
        Origin = position of rectangle, e.g. , [0, 0]
        Orientation = direction of cuts, alternating between horizontal and vertical
        Treemap(Node, Orientation, Origin, Width, Height)
End
```

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.2 Méthodes de Visualisation d'arbres

Space-Filling Methods

Treemap(node n, orientation o, position orig, hsize **W,** vsize h)
if n is a terminal node
    draw-rectangle(orig, w, h)
    return
for each child of **n** (child-i),
    -get number of terminal nodes in subtree
    -sum up number of terminal nodes
    -compute percentage of terminal nodes in n from each subtree (percent-i)
    -if orientation is horizontal
        -for each subtree
        -compute offset of origin based on origin and width (offset-i)
        -treemap(chiıd-i, vertical, orig + offset-i, w* percent-i, h)

4.3 Méthodes de Visualisation d'arbres

Space-Filling Methods

else
-for each subtree
-compute offset of origin based on origin and height (offset-i)
-treemap(chiıd-i, horizontal, orig + off set-i, **W,** h * percent-i)
End: Treemap

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.3 Méthodes de Visualisation d'arbres

Space-Filling Methods

- Méthodes de remlissage radial de l'espace de visualisation

Connues sous la notation “sunburst displays”.

La racine de la structure hiérarchique est au centre de l'espace d'affichage.
Utilise des anneaux imbriqués pour représenter la hiérarchie.
Chaque anneau est divisé en fonction du nombre de noeuds du niveau associé.

Le nombre de noeuds du niveau donné détermine l'espace alloué sur l'anneau.

4.3 Méthodes de Visualisation d'arbres

Space-Filling Methods

Main Program
Begin
Start = start angle for a node (initially 0)
End = end angle for a node (initially 360)
Origin = position of center of sunburst, e.g., [0,0]
Level = current level of hierarchy (initially 0)
Width = thickness of each radial band - based on max depth and display size
Sunburst(Node, Start, End, Level)
End

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.3 Méthodes de Visualisation d'arbres

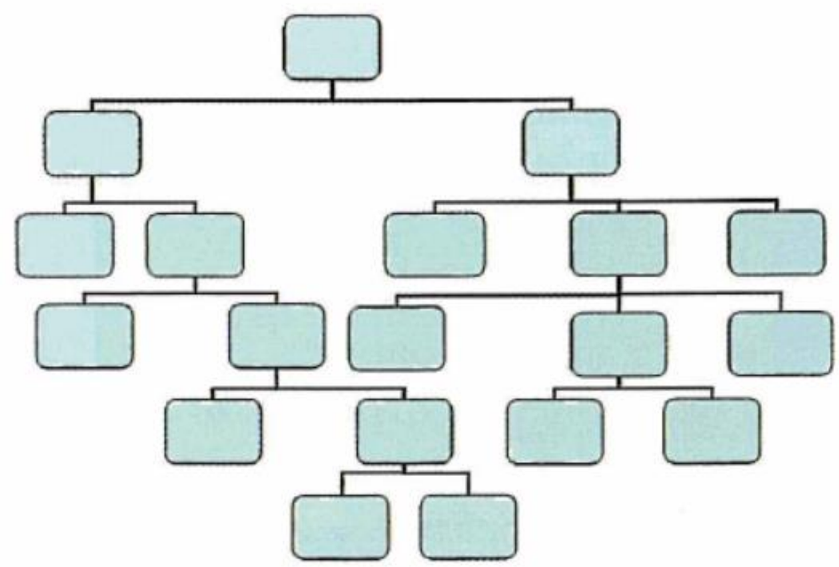
Space-Filling Methods

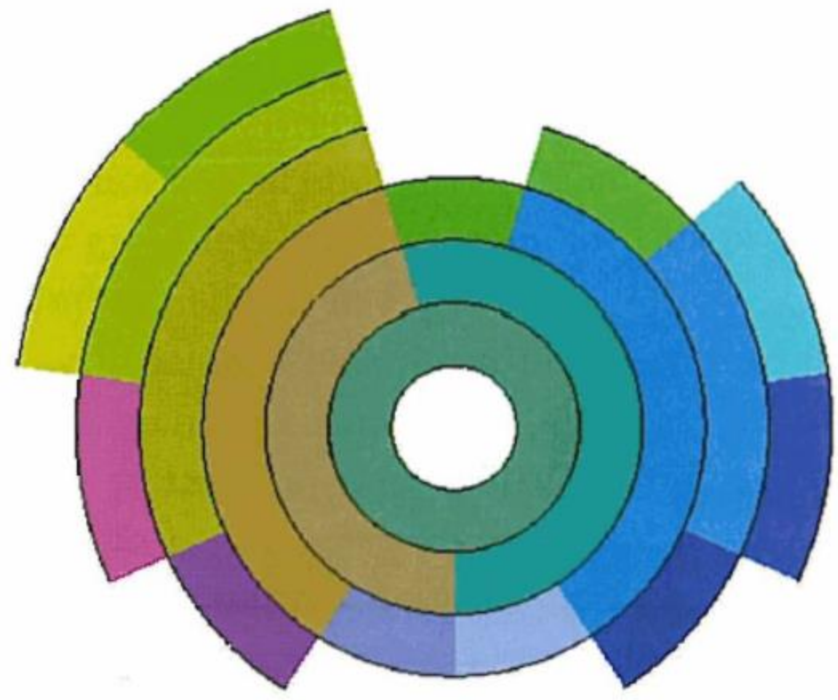
```
Sunburst (node n, angle st, angle en, level 1)
if n is a terminal node
        draw-radial-section(Origin, st, en, l * width, (1+i) * width)
        return
for each child of n (child-i),
        get number of terminal nodes in subtree
        sum up number of terminal nodes
        compute percentage of terminal nodes in n from each subtree
        compute start/end angle based on size of subtrees, order, and angle range
        Sunburst (child-i, st-i, en-i, 1+i)
End: Sunburst
```

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.3 Méthodes de Visualisation d'arbres

Space-Filling Methods





Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.3 Méthodes de Visualisation d'arbres

No **Space-Filling** Methods

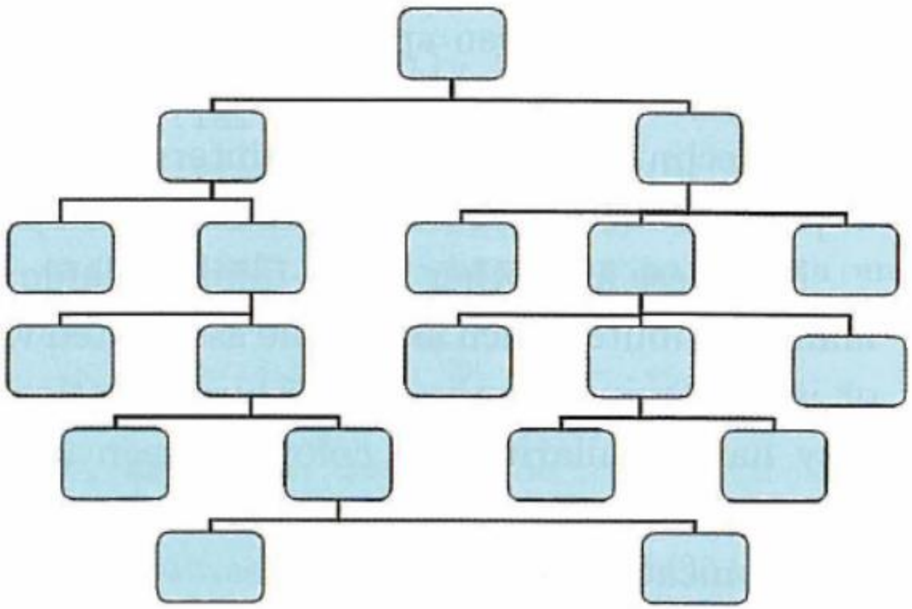
node-link diagram est la représentation la plus utilisée.

Quelques règles à vérifier:

- Minimiser croisement des lignes
- Minimiser le surface du dessin
- Minimiser la longueur des arêtes
- Minimiser le nombre des angles et courbures

4.3 Méthodes de Visualisation d'arbres

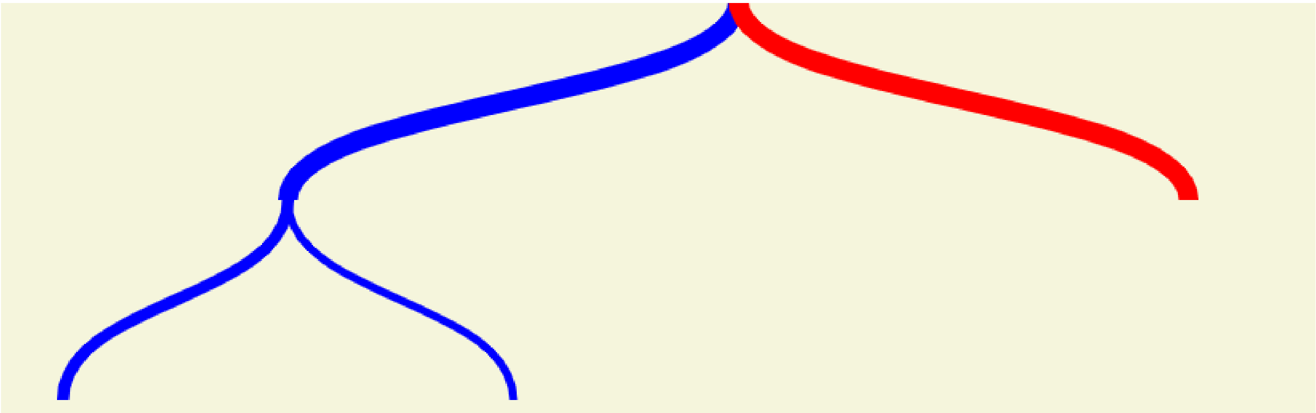
No **Space-Filling** Methods



Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.4 Liabrairies

D3js

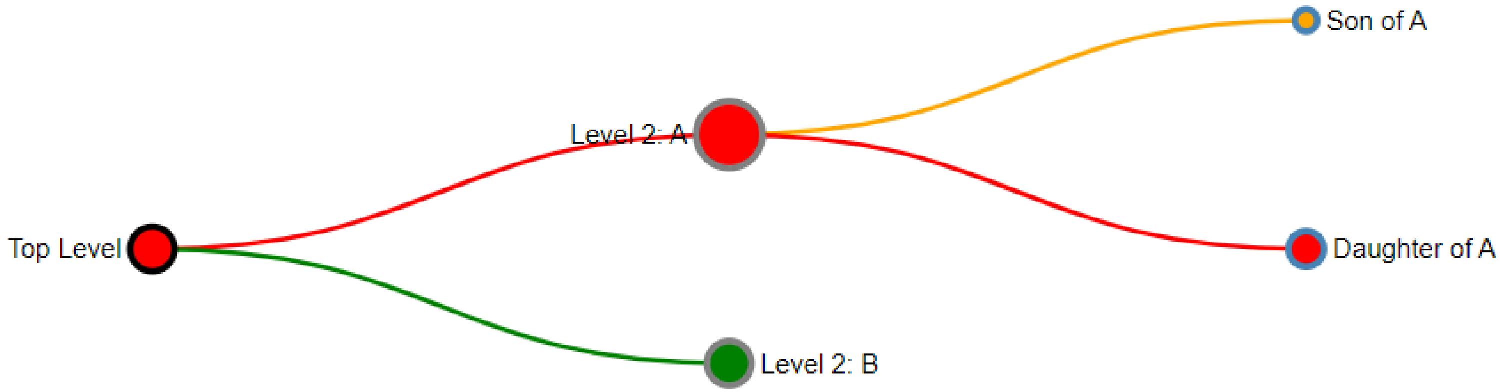


Voir demo/tp01.html

4.4 Liabrairies

D3js

```
var tree = d3.layout.tree().size([height, width]);
```



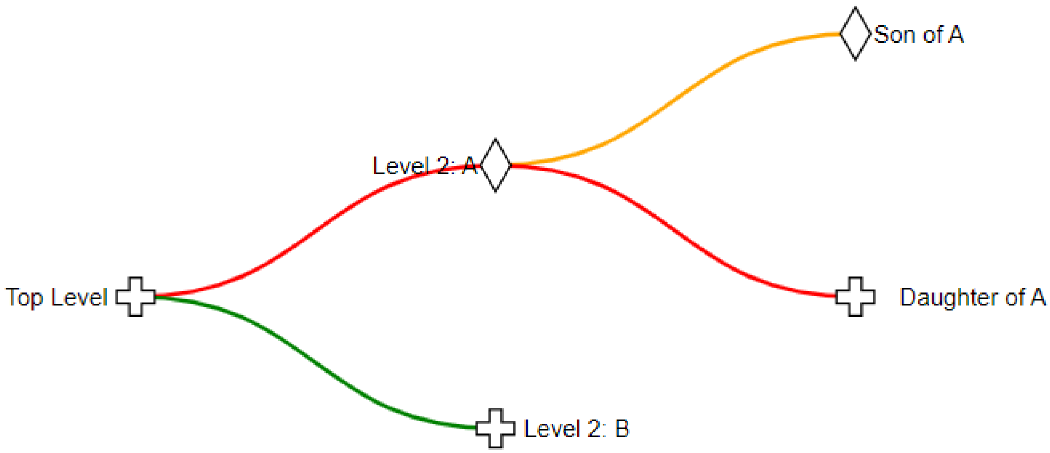
Voir demo/tp02.html

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.4 Liabrairies

D3js

```
var tree = d3.layout.tree().size([height, width]);
```



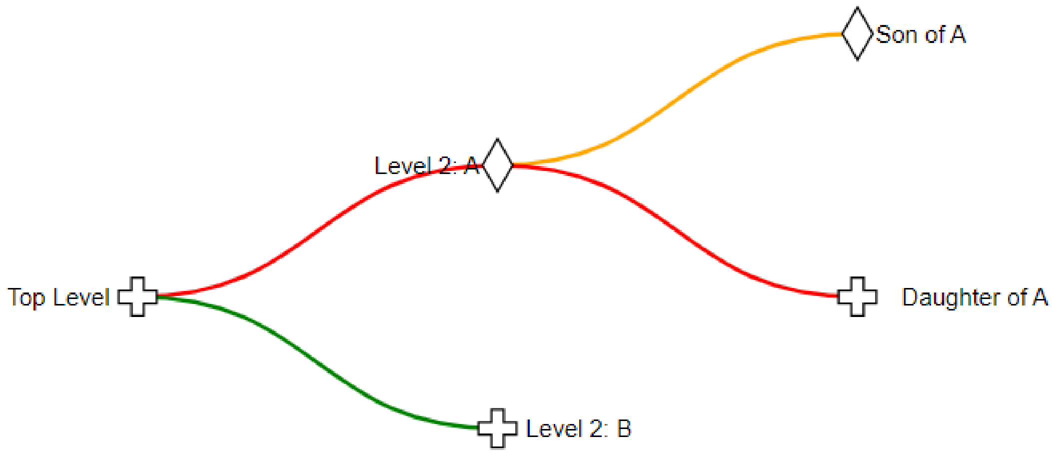
Voir demo/tp03.html

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.4 Liabrairies

D3js

```
var tree = d3.layout.tree().size([height, width]);
```

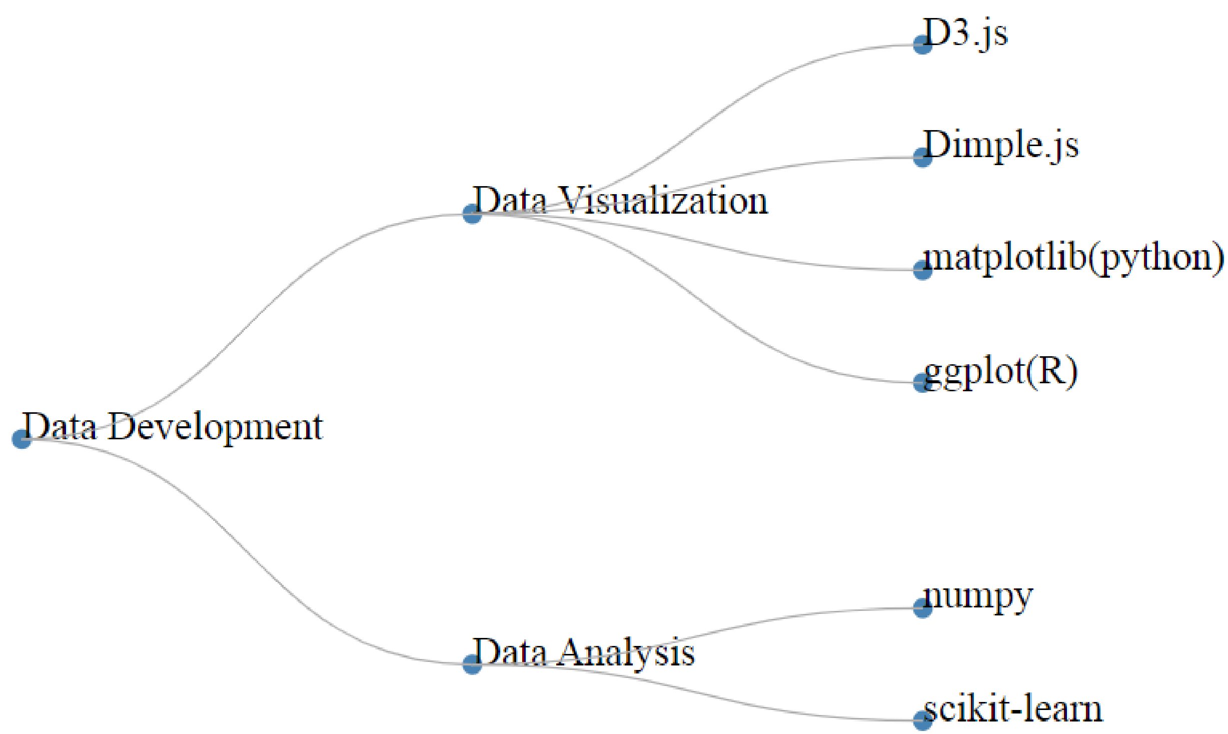


Voir demo/tp03.html

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.4 Liabrairies

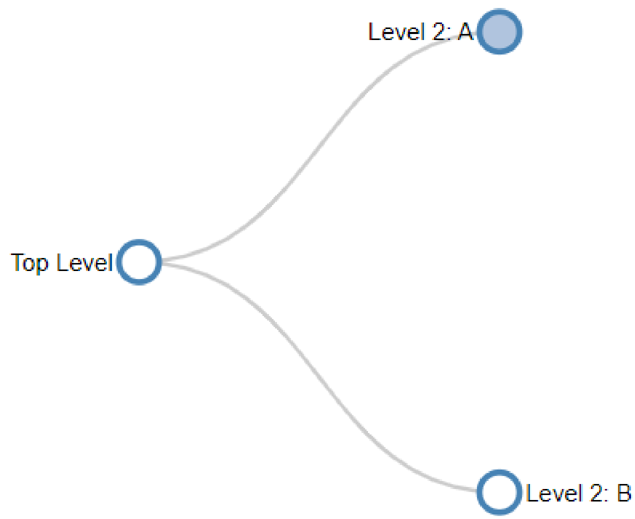
D3js



Voir demo/tp04.html

4.4 Liabrairies

D3js

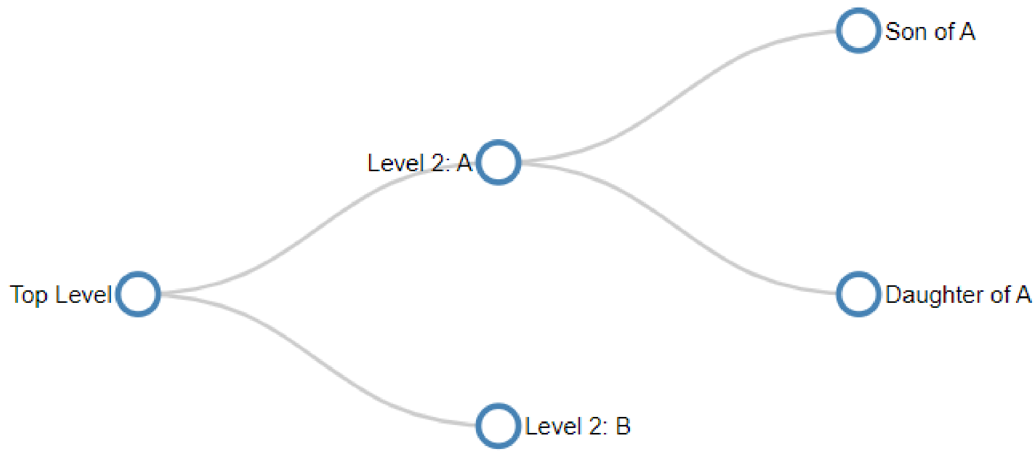


Voir demo/tp05.html

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.4 Liabrairies

D3js

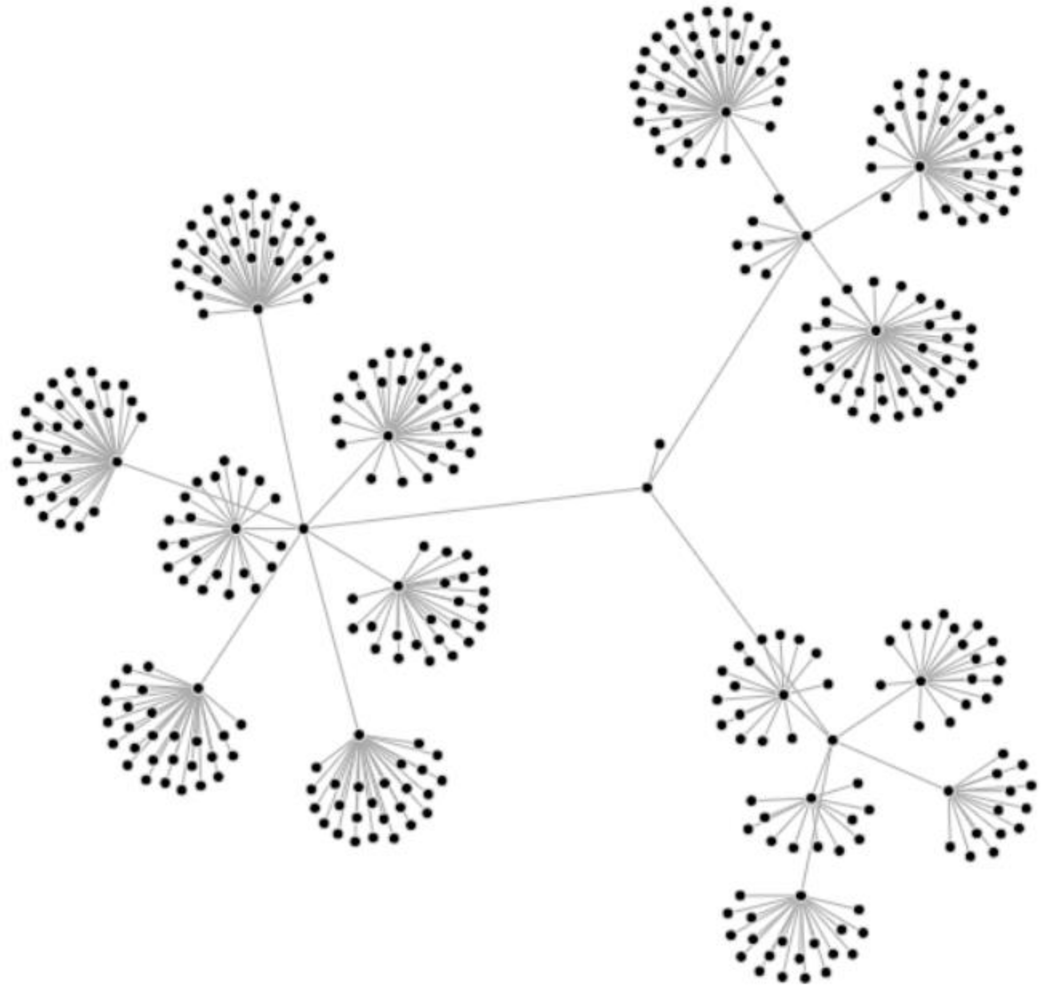


Voir demo/tp05.html

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.4 Liabrairies

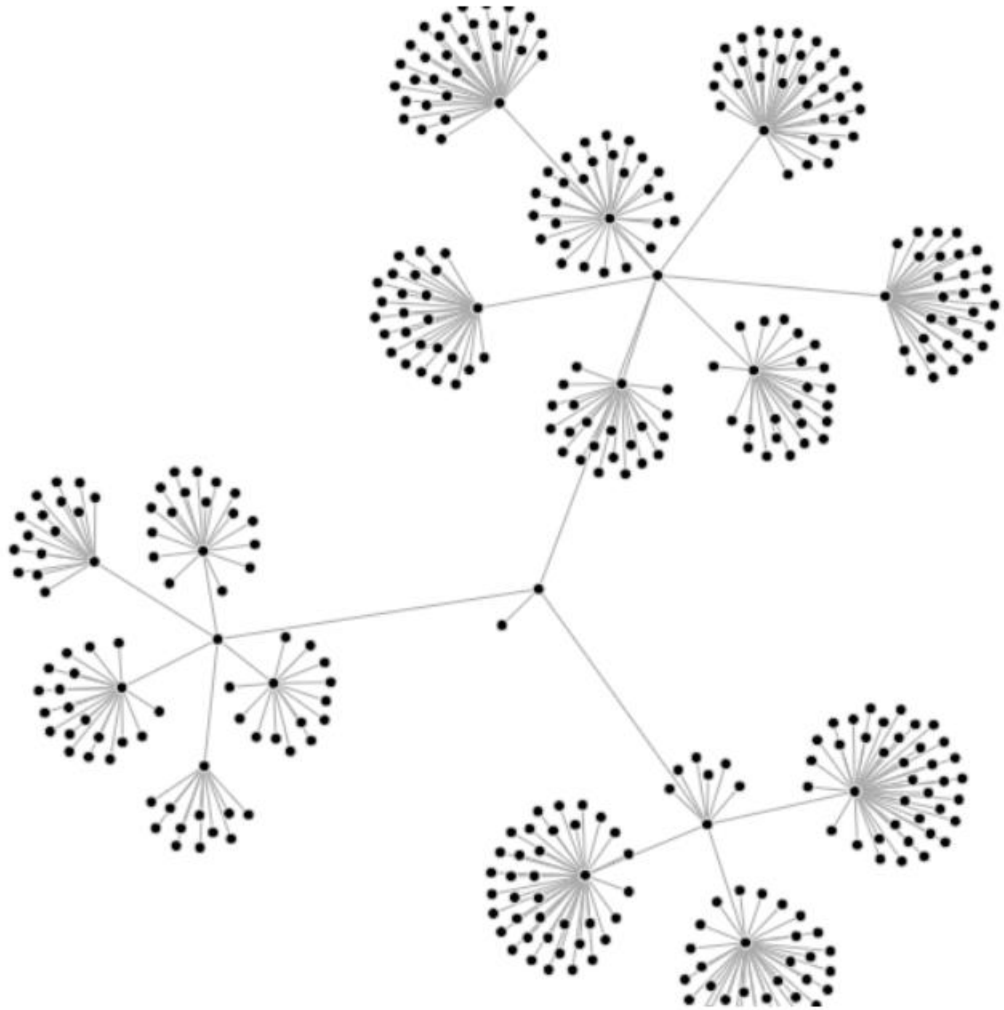
D3js/ var simulation =
d3.forceSimulation(nodes)



Voir demo/tp06.html

4.4 Liabrairies

D3js

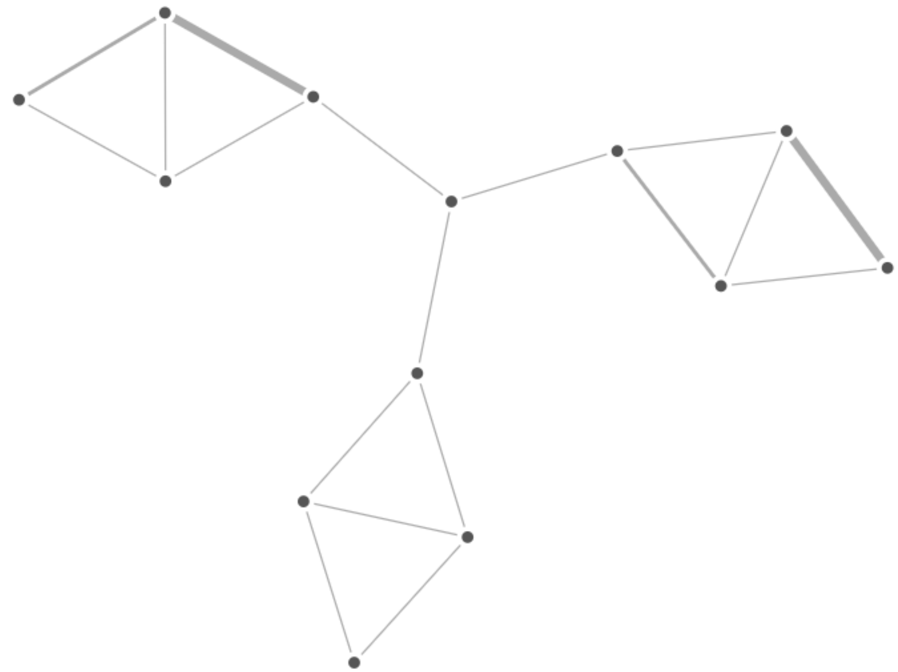


Voir demo/tp06.html

4.4 Liabrairies

D3js

```
var force = d3.layout.force()
    .gravity(.01)
    .distance(100)
    .charge(-100)
    .size([width, height]);
```

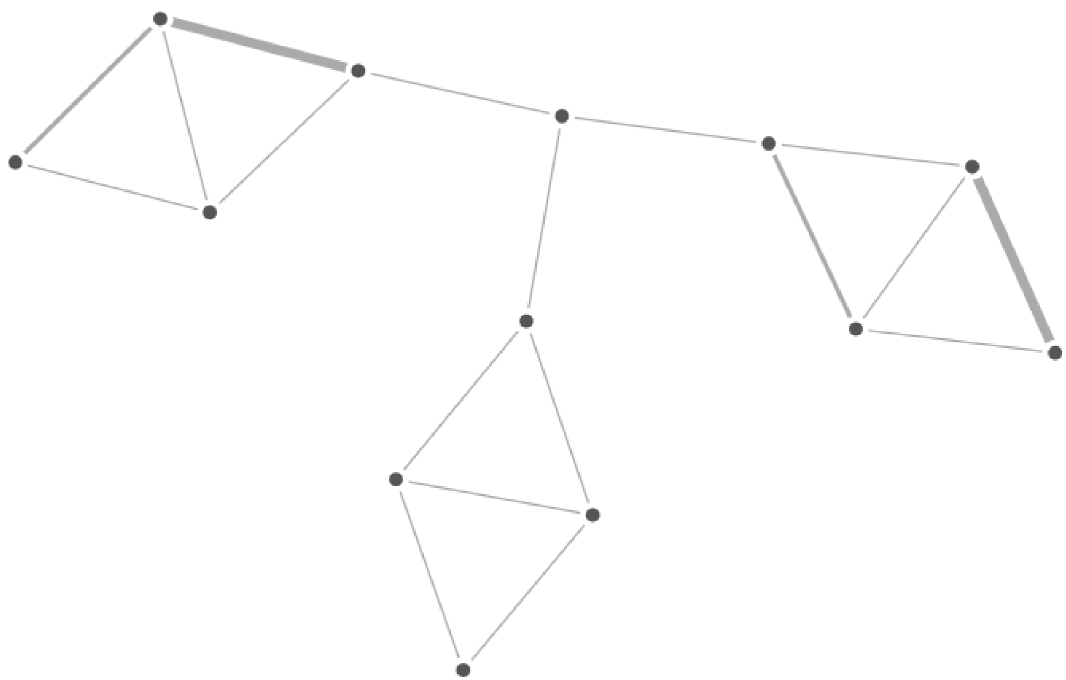


Voir demo/tp07.html

4.4 Liabrairies

D3js

```
var force = d3.layout.force()
   .gravity(.01)
   .distance(100)
   .charge(-100)
   .size([width, height]);
```

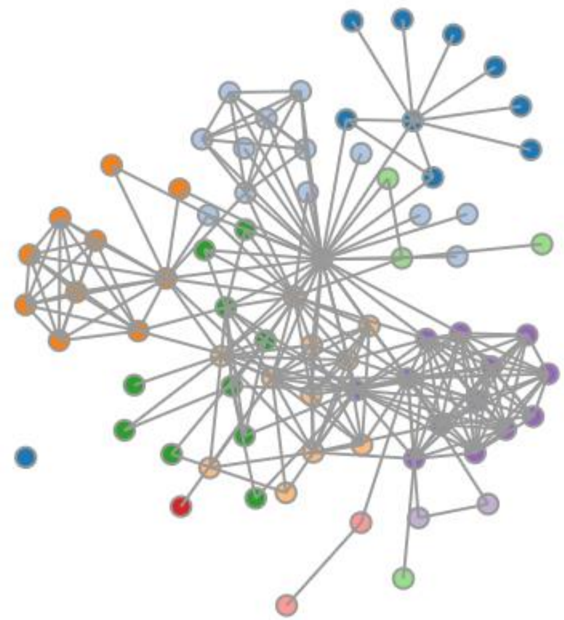


Voir demo/tp07.html

4.4 Liabrairies

D3js

```
var force = d3.layout.force()
.size([width, height])
.nodes(graph.nodes)
.links(graph.links)
.gravity(.5)
   .distance(40)
   .charge(-200);
```



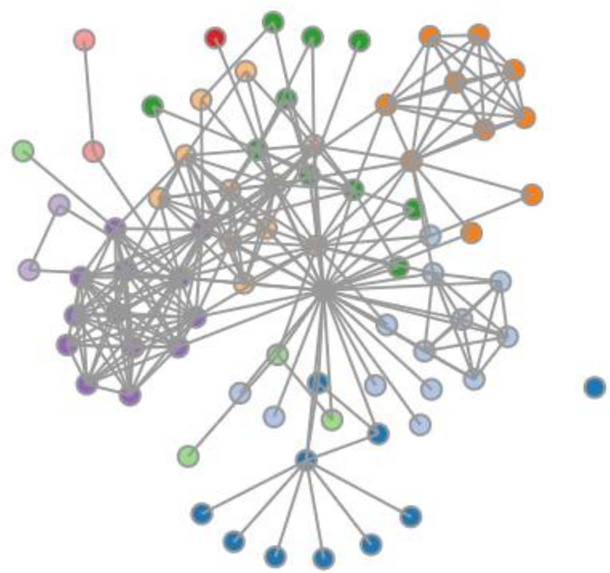
Voir demo/tp07.html

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.4 Liabrairies

D3js

```
var force = d3.layout.force()
.size([width, height])
.nodes(graph.nodes)
.links(graph.links)
.gravity(.5)
   .distance(40)
   .charge(-200);
```



Voir demo/tp07.html

4.4 Liabrairies

D3js

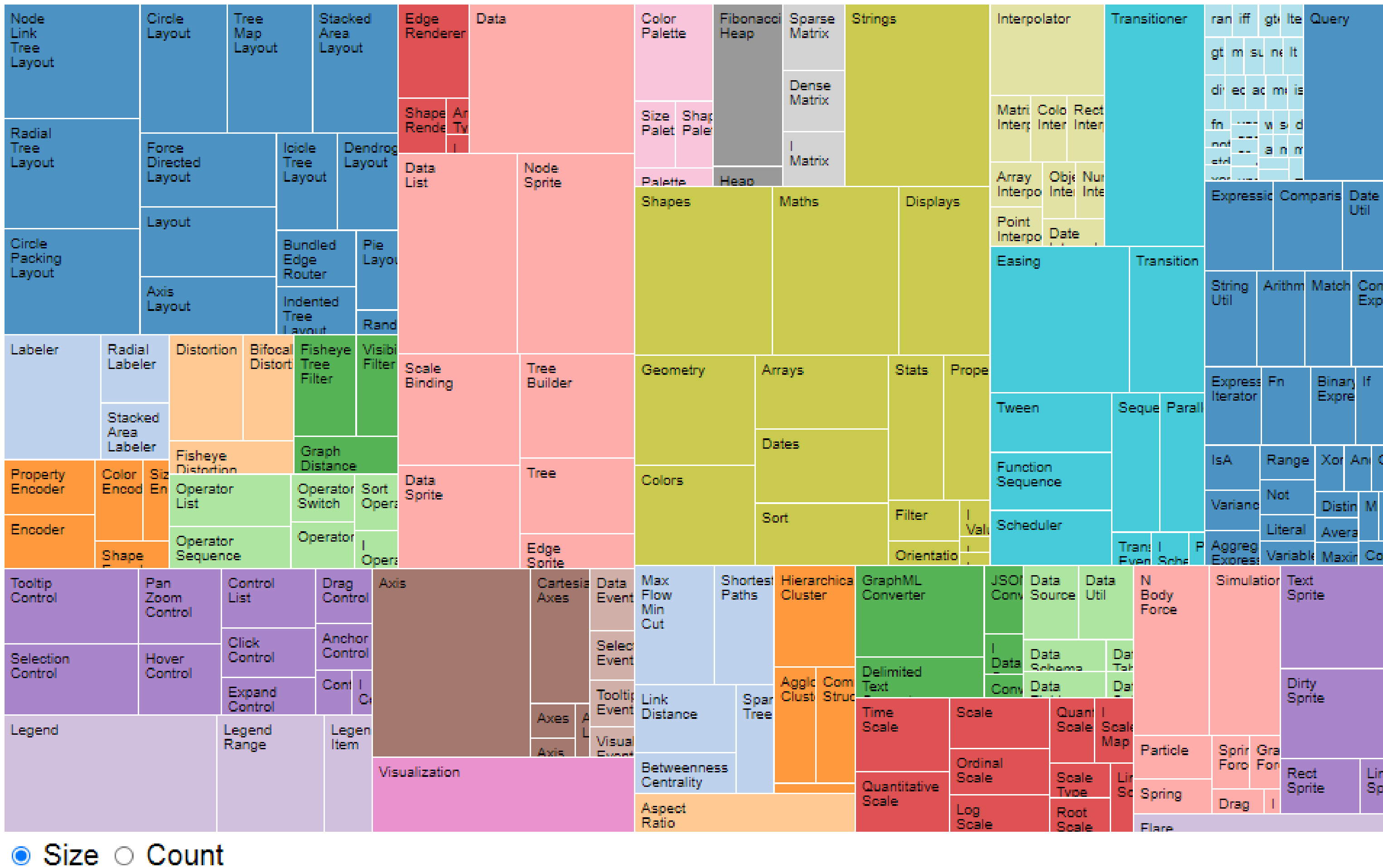
```
var treemap = d3.treemap()
    .tile(d3.treemapResquarify)
    .size([width, height]);
```

Voir demo/tp09.html

Chapitre 4: VISUALISATION DE GRAPHES ET ARBRES

4.4 Liabrairies

D3js

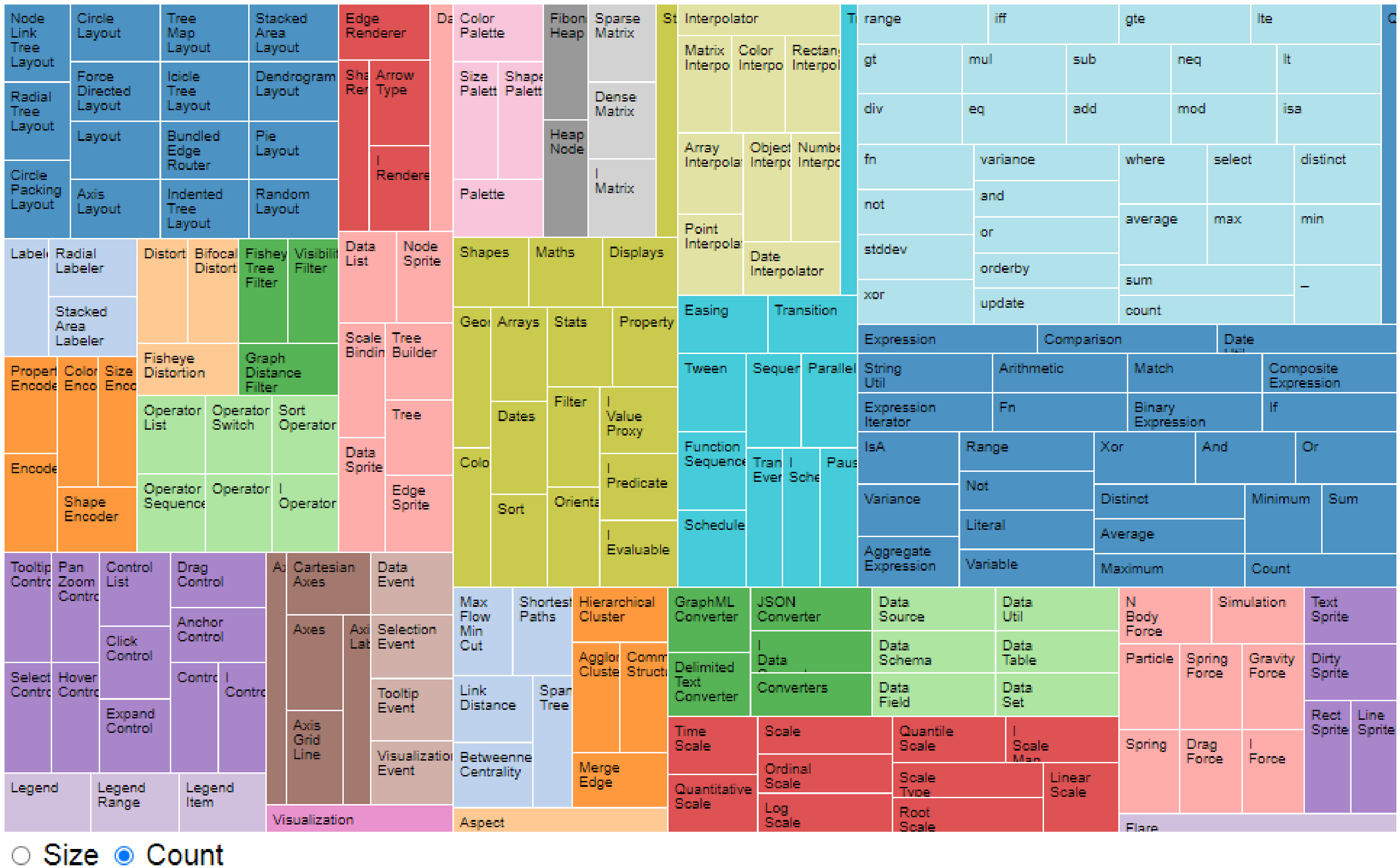


Voir demo/tp09.html

Chapitre 4: *VISUALISATION DE GRAPHES ET ARBRES*

4.4 Liabrairies

D3js



Voir demo/tp09.html