Visualisation de graphes

Graphes – 3e partie

Raphaël Fournier-S'niehotta

CNAM Paris, fournier@cnam.fr

HTT-FOD
RCP216
2020-2021

le cnam

Plan

Objectifs et principes

Objectifs

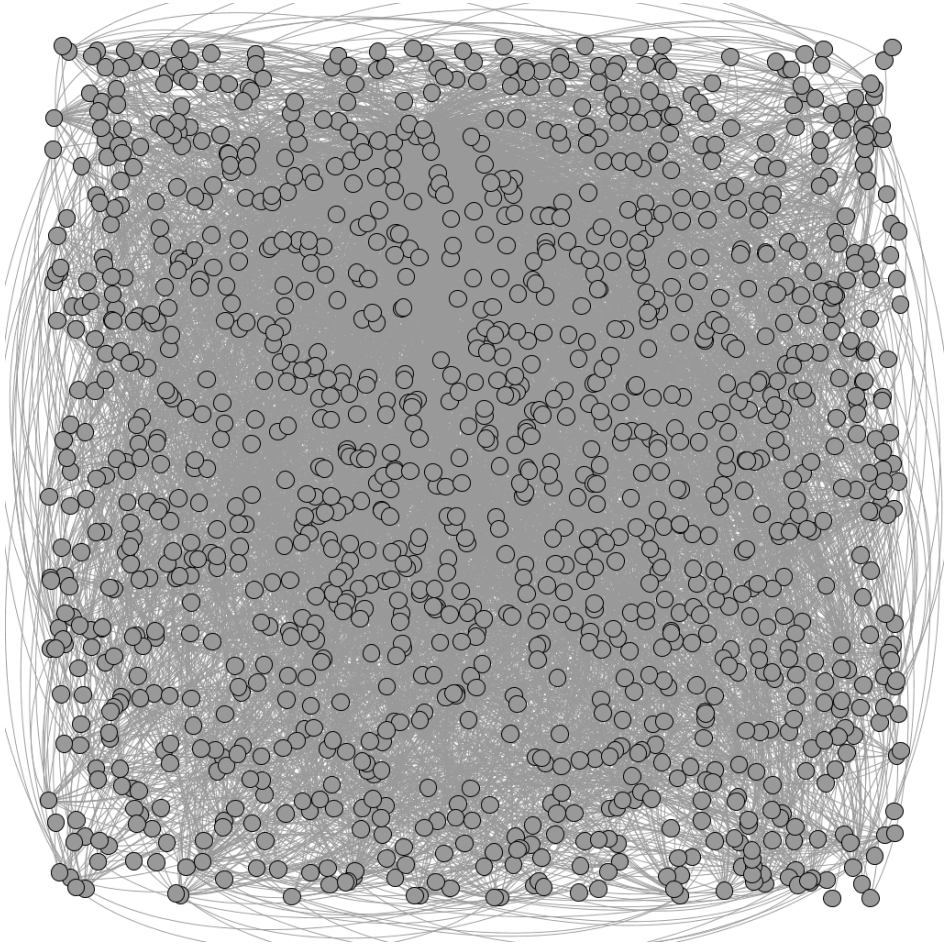
Comprendre le comportement des entités
qui interagissent dans le système étudié, et les lois qui les gouvernent

- On cherche donc :
  - quelle est la structure des graphes
  - quelle est l'évolution de cette structure
  - quels sont les phénomènes reposant sur l'existence de ce réseau

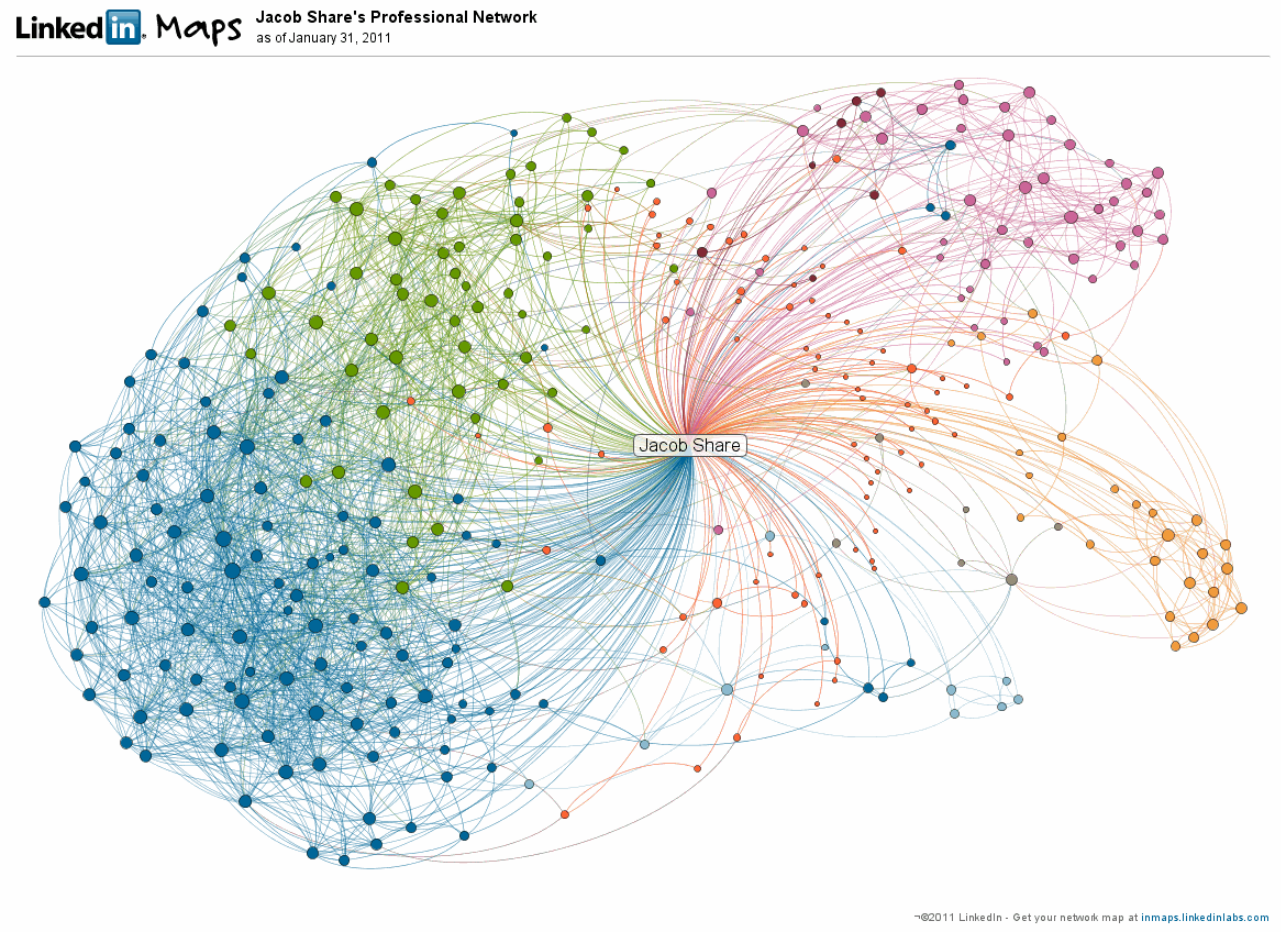
Plus précisément

- Identification du rôle des individus :
  - Leader
  - Suiveur
  - Intermédiaire
- Identification de l'importance d'un groupe en analysant :
  - la taille
  - la cohésion
  - les profils
  - les relations internes et externes

Motivation de la visualisation



Motivation de la visualisation



Motivation de la visualisation

Un outil de visualisation bien conçu peut permettre à l'utilisateur d'observer des structures dans un graphe qu'il serait difficile ou impossible à voir dans le graphe sous sa forme brute.

Rappels

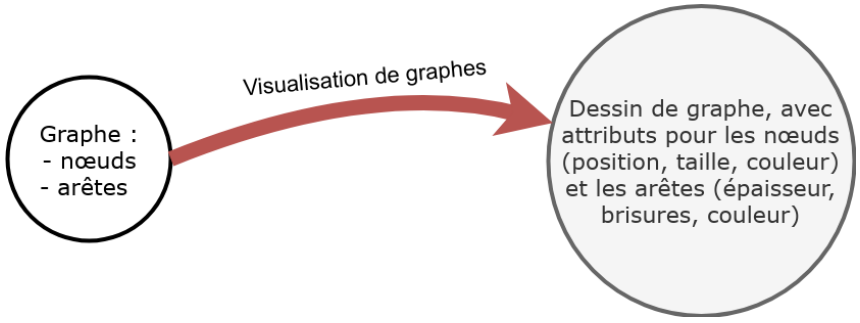
Un graphe est défini par un couple G = (V,E) tel que :

- V (pour l'anglais *vertices*) est un ensemble fini de sommets
- E (pour l'anglais *edges*) est un ensemble fini d'arêtes

Un graphe peut être orienté, ou non :

- si oui, les couples $(v_i, v_j) \in E$ sont ordonnés, v_i est le sommet initial, v_j est le sommet terminal.
- on appelle alors le couple (v_i, v_j) un *arc*, représenté graphiquement par $v_i \rightarrow v_j$.
- si non, les couples ne sont pas orientés et (v_i, v_j) est équivalent à (v_j, v_i), et on l'appelle *arête*, représenté par $v_i - v_j$

Principes de la représentation



- Pour chaque nœud dans le graphe, il faut :
  - une position
  - une taille (fixe, variable)
  - une forme (cercle, disque, carré)
  - une couleur
- Pour chaque lien dans le graphe, il faut :
  - une position (chevauchant d'autres arêtes ou non)
  - une taille (longueur, épaisseur du trait)
  - une forme (courbe, droit)
  - une couleur

Historiquement, la couleur est venue après. Comme la taille, elle sert souvent à indiquer une valeur de propriété pour le nœud/lien (degré, communauté, etc.). La forme sert généralement dans un contexte *biparti*.

Terminologie

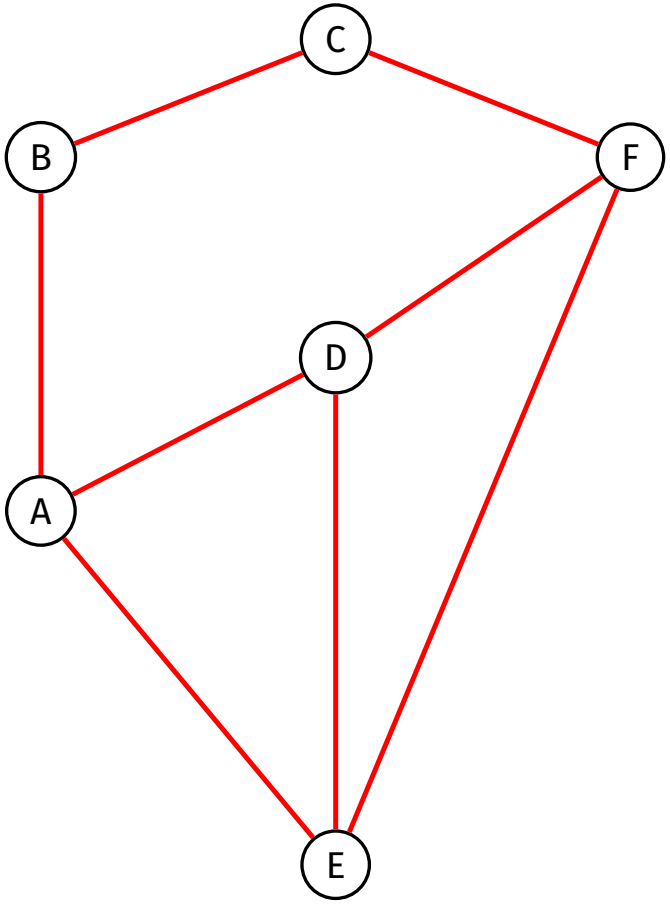
Le champ académique de la visualisation de graphes comporte pas ou peu de publications en français. Il n'y a donc pas vraiment de terminologie établie. On utilisera dans ce cours un vocabulaire proche de l'anglais.

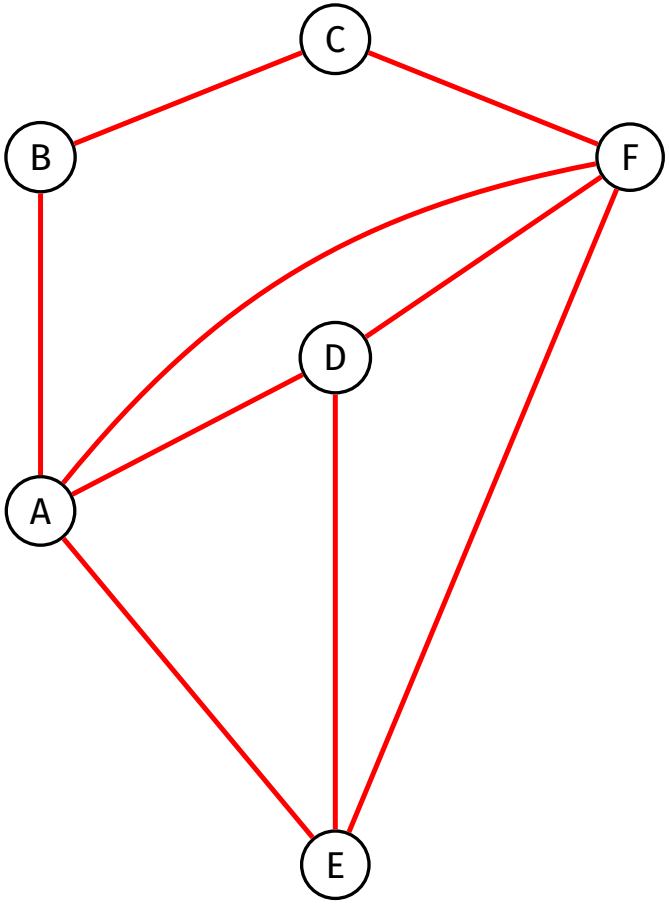
On emploiera de façon presque indifférenciée les termes :

- dessin
- représentation
- image

Straight-line drawing

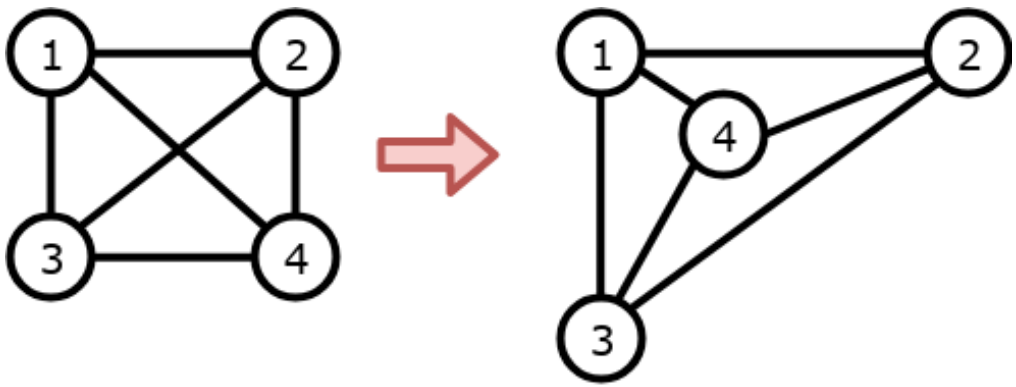
Un dessin de graphe est dit ”aux arêtes droites” (*straight-line drawing*) si toutes ses arêtes sont des segments de droite.

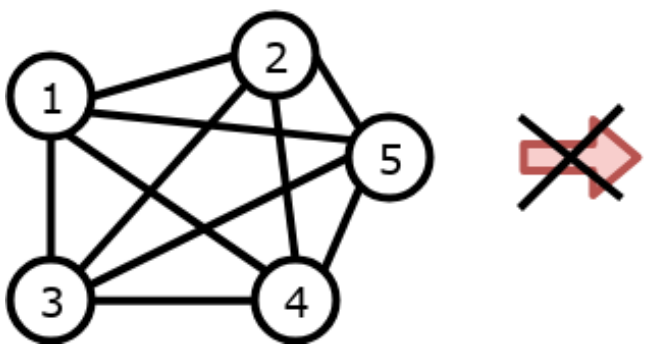




Planarité

- Un graphe est dit planaire s'il peut être dessiné sans chevauchement d'arêtes.
- Un graphe est non-planaire si **toutes** ses représentations ont **au moins un** chevauchement d'arête





Qualité d'un dessin

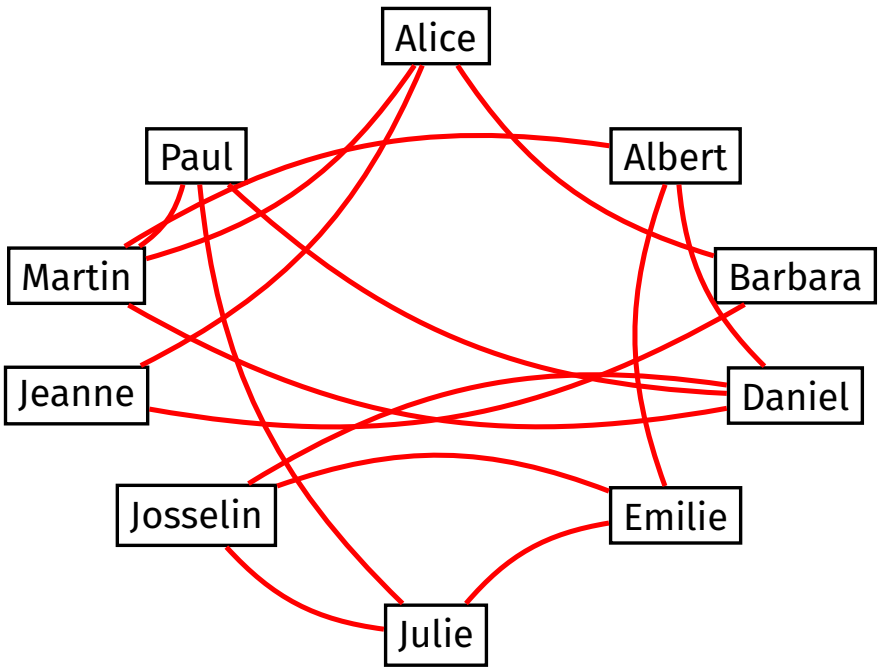
Une bonne représentation de graphe devrait être :

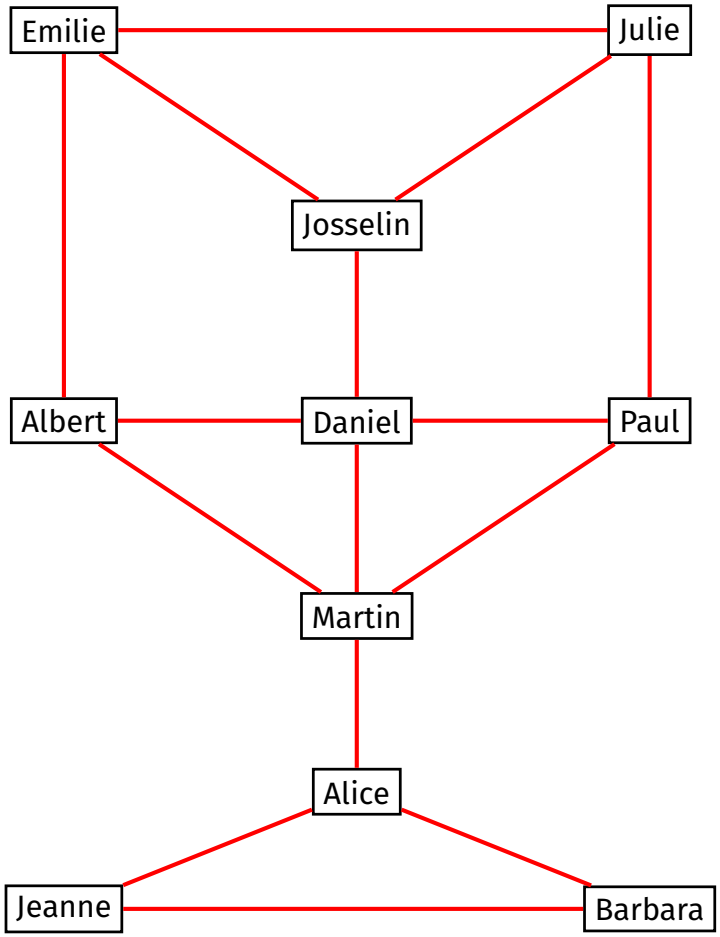
- facile à comprendre
- facile à mémoriser
- esthétiquement plaisante

Quelques critères :

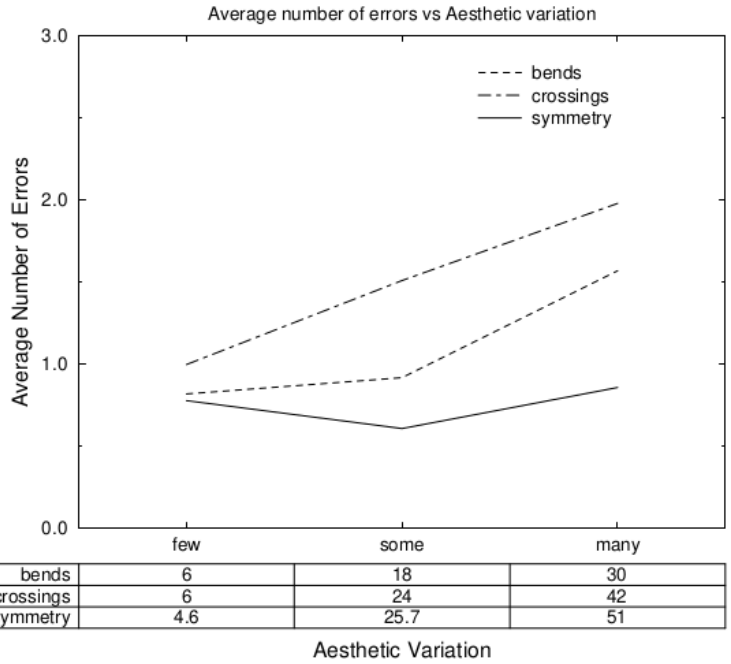
- éviter les chevauchements d'arêtes
- arêtes droites

Qualité d'un dessin : intuition



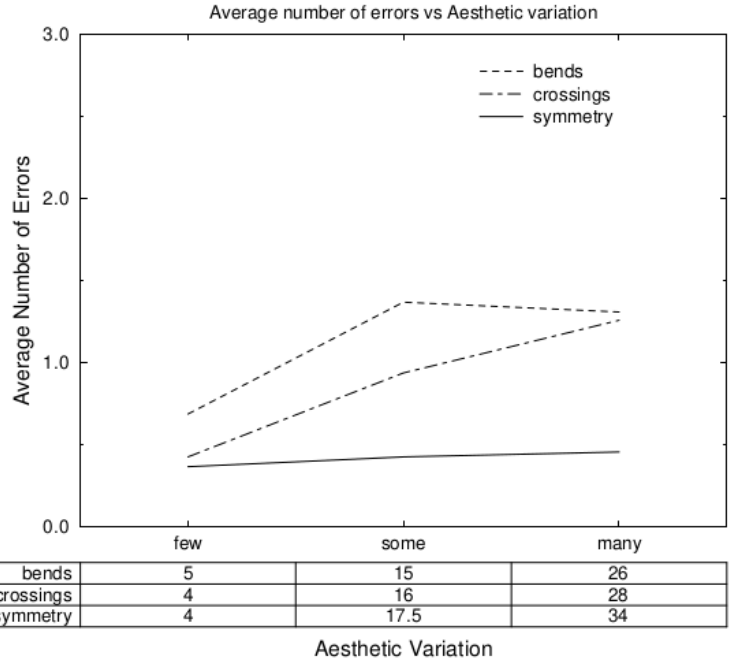


Qualité d'un dessin : chevauchements et courbures



	few	some	many
bends	6	18	30
crossings	6	24	42
symmetry	4.6	25.7	51

Fig. 3. Results for the dense graph



	few	some	many
bends	5	15	26
crossings	4	16	28
symmetry	4	17.5	34

Fig. 4. Results for the sparse graph

1

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

- les croisements d'arêtes
- les lignes brisées

Histoire des algorithmes de spatialisation

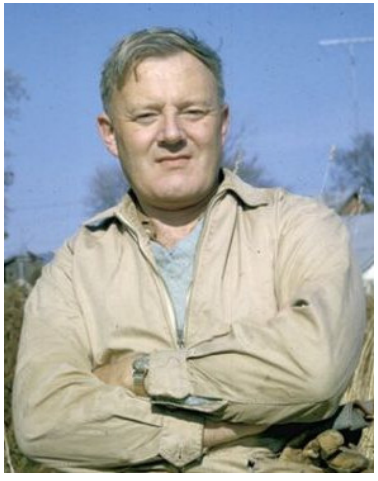
Théorème de Fáry

Théorème (István Fáry[2])

Un graphe simple planaire peut être dessiné sans chevauchements d'arêtes, avec des lignes droites.

- Cependant, beaucoup de graphes que l'on rencontrera ne seront pas planaires
- Et qu'un théorème assurant l'existence d'une représentation sans croisements ne donne pas immédiatement accès à celle-ci : peu d'utilité pratique

William Thomas Tutte



mathématicien et britannique (1917-2002).

- Cryptanalyste à Bletchley Park avec Alan Turing
- Pionnier de la théorie des matroïdes
- pionnier en théorie des graphes
- inventeur du premier algorithme de dessin de graphes
- article séminal en 1963 ”How to draw a graph”[3]

Algorithme barycentrique

- L'algorithme prend en entrée un graphe $G = (V, E)$.
- La sortie de l'algorithme est une représentation avec les lignes droites dudit graphe, que l'on appellera p (et on note $p(u)$ la position de u dans p).

Algorithme

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 :

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

Algorithme barycentrique

Algorithme

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 :

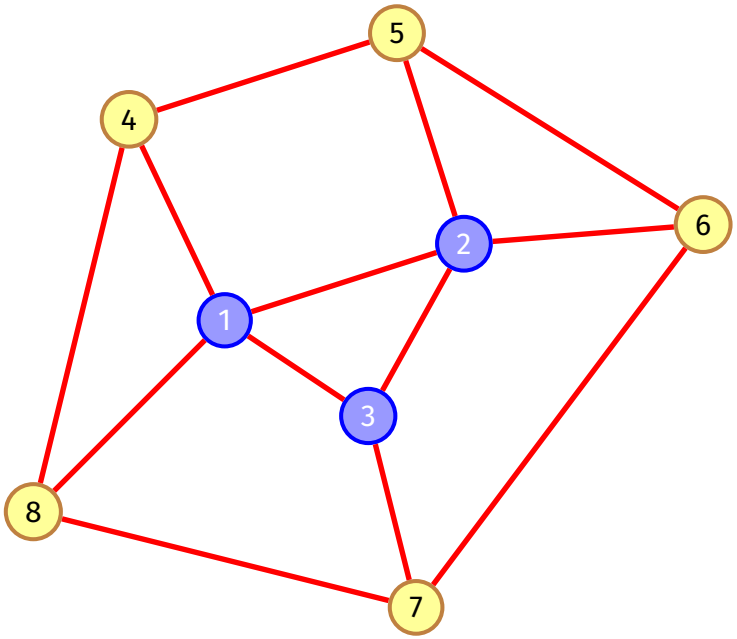
$$p(u) = \frac{1}{deg(u)} \sum_{v \in N(u)} p(v) \quad \text{avec } N(u) \text{ l'ensemble des voisins de } u.$$

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

$$x(u) = \frac{1}{deg(u)} \sum_{v \in N(u)} x(v)$$

$$y(u) = \frac{1}{deg(u)} \sum_{v \in N(u)} y(v)$$

Algorithme barycentrique



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 :

$$\begin{cases} x_1 &= \frac{1}{4}(x_2 + x_3 + x'_4 + x'_8) \\ x_2 &= \frac{1}{4}(x_1 + x_3 + x'_5 + x'_6) \\ x_3 &= \frac{1}{3}(x_1 + x_2 + x'_7) \end{cases} \qquad \begin{cases} y_1 &= \frac{1}{4}(y_2 + y_3 + y'_4 + y'_8) \\ y_2 &= \frac{1}{4}(y_1 + y_3 + y'_5 + y'_6) \\ y_3 &= \frac{1}{3}(y_1 + y_2 + y'_7) \end{cases}$$

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

Algorithme barycentrique

On obtient :

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

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

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

Vision matricielle de l'algorithme

Il s'agit de trouver les vecteurs $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ et $\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$ tels que :

$$\begin{cases} Mx = c \\ My = d \end{cases} \quad \text{avec } M = \begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 3 \end{bmatrix} \quad \text{et} \quad \begin{cases} c_u = \sum_{w \in A} x_w \\ d_u = \sum_{w \in A} y_w \end{cases}$$

Algorithme barycentrique

- le cœur de l'algorithme de Tutte consiste à inverser une matrice
- plus précisément, de la sous-matrice Laplacienne correspondant à l'ensemble $V \setminus A$
- elle est définie ainsi :

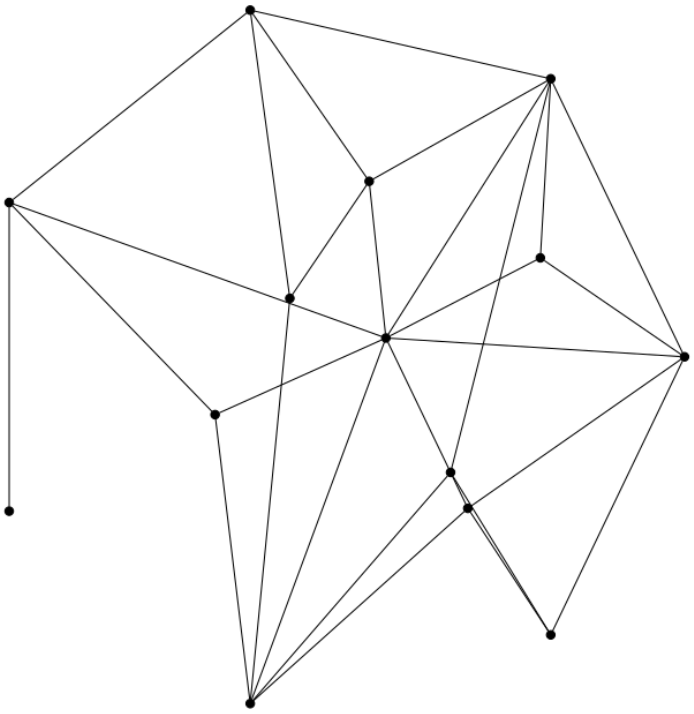
$$M_{uv} = \begin{cases} deg(u) & \text{si } u = v \\ -1 & \text{si } (u,v) \in E \\ 0 & \text{sinon.} \end{cases}$$

- Les coordonnées x et y de chaque sommet de A sont alors obtenues par :

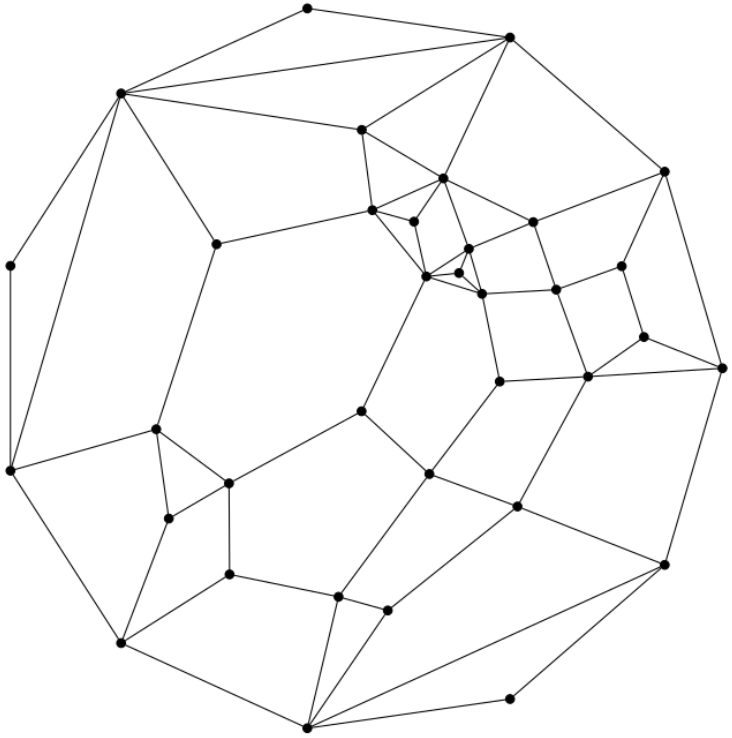
$$\begin{cases} x &= M^{-1}c \\ y &= M^{-1}d \end{cases}$$

- Enfin, $p(u) = (x_u, y_u)$

Exemples de dessins



Graphe non planaire



Graphe planaire

Vision énergétique

L'algorithme

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. on place tous les autres sommets afin de minimiser l'énergie

Quelle est l'énergie d'un dessin p ?

Vision énergétique

L'algorithme

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. on place tous les autres sommets afin de minimiser l'énergie

Quelle est l'énergie d'un dessin p ?

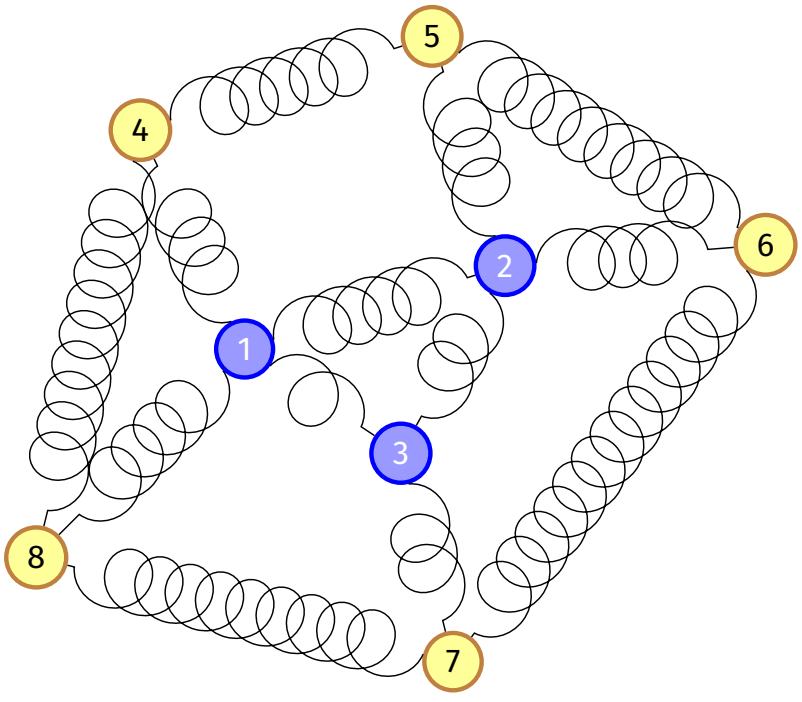
- La distance euclidienne entre u et v vaut :

$$d(u,v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$$

- l'énergie de l'arête vaut : $E_{uv} = d(u,v)^2$
- L'énergie totale est la somme de l'énergie des arêtes :

$$E_p = \sum_{(u,v)\in E} d(u,v)^2 = \sum_{(u,v)\in E} (x_u - x_v)^2 + (y_u - y_v)^2$$

Vision énergétique



Analogie avec la mécanique :

- chaque arête du graphe est un ressort ($l_0 = 0$)
- chaque sommet du graphe est un anneau

- La 2e étape de l'algorithme, consistant à choisir des positions pour un ensemble de sommet, revient à clouer ces anneaux au sol.
- La 3e étape, minimiser l'énergie, revient à laisser le système évoluer librement jusqu'à atteindre une position d'équilibre.

Vision énergétique

La minimisation de l'énergie correspond à un état unique, tel que :

$$\begin{cases} \frac{\partial E(p)}{\partial x_u} = 0 \\ \frac{\partial E(p)}{\partial y_u} = 0 \end{cases} \text{, pour chaque } u \in V - A.$$

Ce qui se traduit (en x, même chose en y) par :

$$\begin{aligned} & \frac{\partial E(p)}{\partial x_u} && = 0 \\ \text{soit :} \quad & \frac{\partial}{\partial x_u} \left(\sum_{(u,v)\in E} (x_u - x_v)^2 + (y_u - y_v)^2 \right) && = 0 \\ \text{puis :} \quad & \sum_{v \in Vt.q.(u,v)\in E} 2(x_u - x_v) && = 0 \\ \text{d'où :} \quad & x_u = \frac{1}{deg(u)} \sum_{v \in Vt.q.(u,v)\in E} x_v \end{aligned}$$

On retrouve l'équation vue dans la vision classique de l'algorithme.

Qualité

Est-ce que cet algorithme est **bon** ?

- Performance
  - La complexité de l'algorithme est théoriquement en $\mathcal{O}(n^{1.5})$, pour les graphes planaires.
  - En pratique, avec les méthodes de résolution numérique, il est assez rapide.
- Élégance
  - simple à comprendre
  - plutôt facile à implémenter
  - des bibliothèques numériques efficaces existent pour les parties difficiles
- Adéquation
  - on obtient des représentations planaires (graphes planaires) et des arêtes droites.

Remarque

- L'algorithme souffre d'un défaut, une limite de résolution sommitale ("poor vertex resolution").
- C'est-à-dire que les sommets peuvent se placer très près les uns des autres.

Limite de résolution sommitale

L'algorithme de Tutte a généralement une mauvaise résolution sommitale

Exemple (cas particulier) :

- A est placé à $(0.5, 0)$
- B à $(0, 0)$
- C à $(1, 0)$
- pour $0 < j < n$, les sommets j sont à (x_j, y_j)

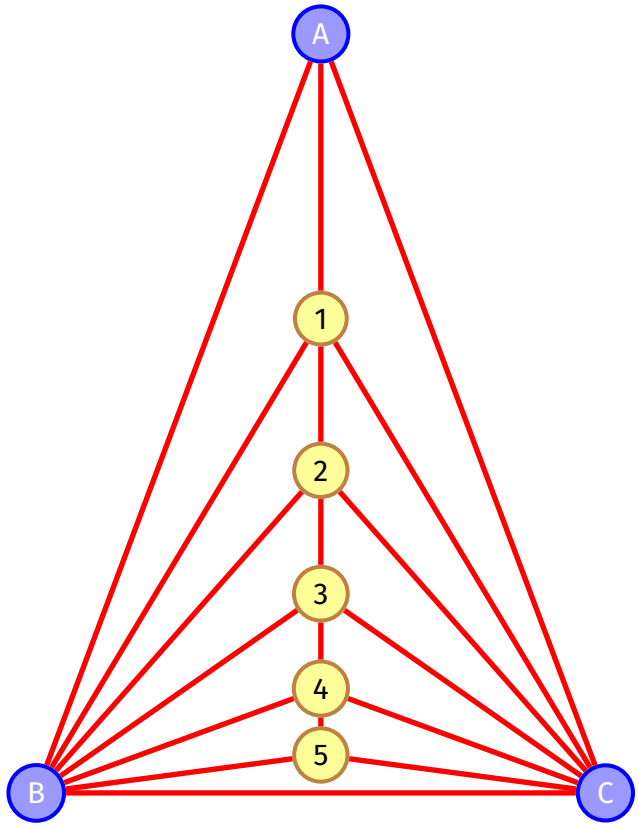
Avec les équations barycentriques,

$$\begin{cases} x_j &= 0.5 \\ y_j &= \frac{1}{4}(y_{j-1} + y_{j+1}) \end{cases}$$

D'où :

$$|y_{j-1} - y_j| < \frac{1}{2^j}$$

Les sommets peuvent être arbitrairement proches les uns des autres.

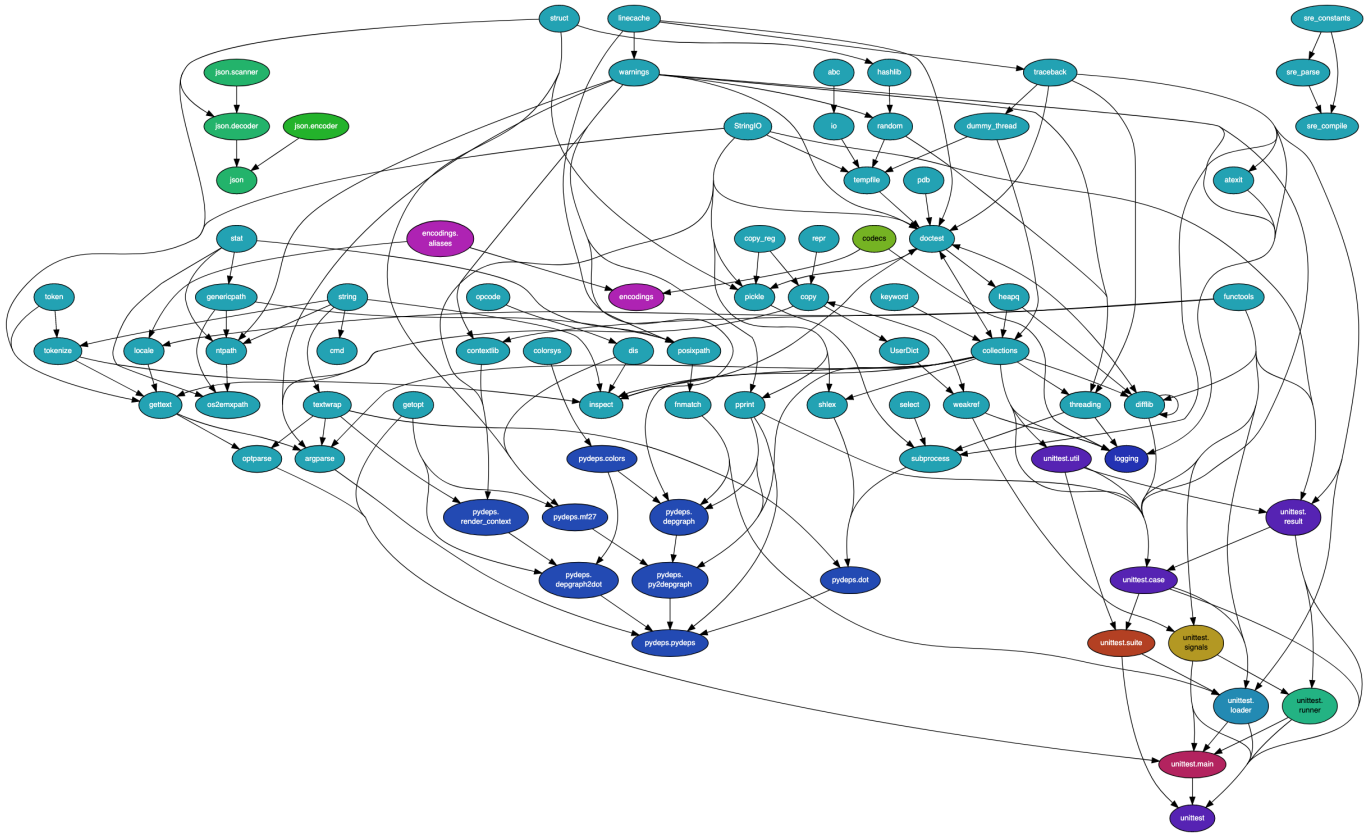


Changements d'usage

- Dans les années 1980, probablement avec l'augmentation des performances des machines, le dessin de graphe est passé de simple curiosité mathématique à un outil de fouille de données
- la demande industrielle a cru significativement :
  - développement logiciel : Computer-aided software engineering (CASE), reverse engineering, etc.
  - biologie : réseaux de régulation génétiques
  - sécurité : gestion de risques, transferts d'argent
  - étude de réseaux sociaux (Social Network Analysis, SNA)
- de nombreuses entreprises achètent des logiciels de dessin de graphe, beaucoup en programment

Nouveaux usages

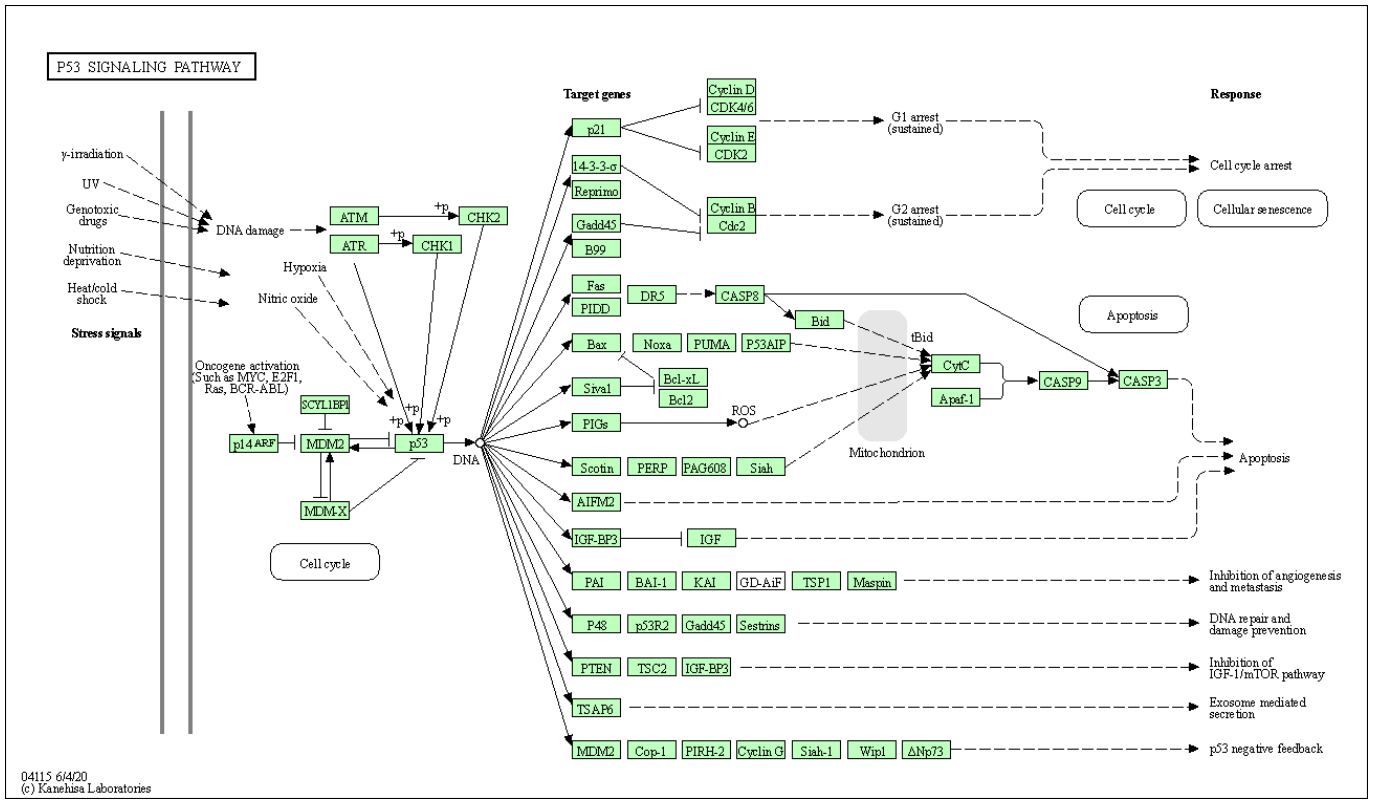
Dépendances logicielles



Source : DANJOU, *Dependencies Handling in Python*

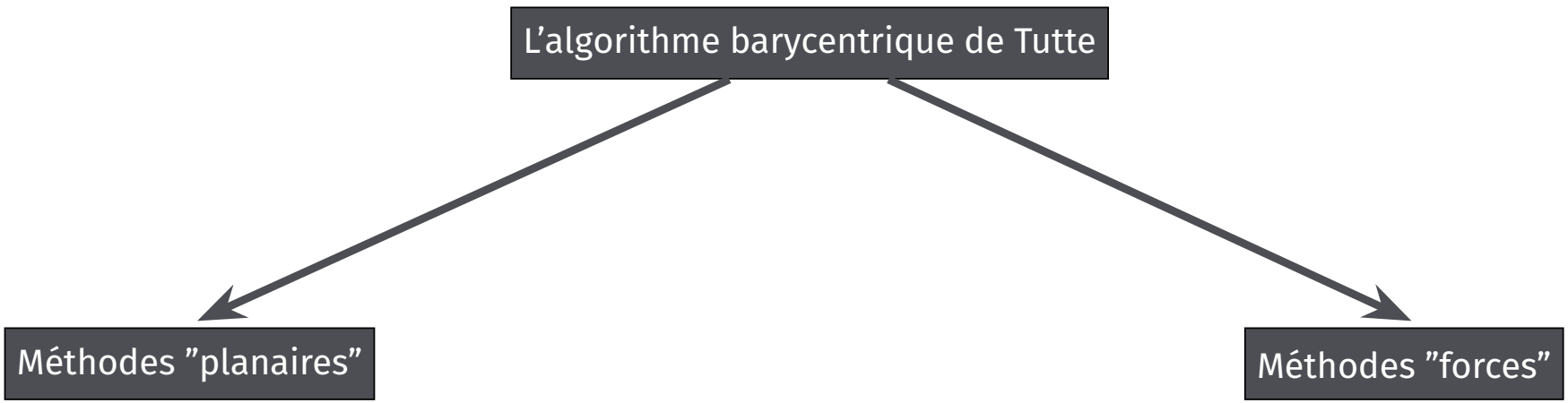
Nouveaux usages

Biologie (Génome)



Source : LABORATORIES, *KEGG project : p53 signaling pathway - Homo sapiens (human)*

Deux voies de travail



L'impasse de la planarité

- Les méthodes autour de la planarité ont abouti d'abord à des algorithmes linéaires efficaces
- mais qui souffraient toujours du problème de résolution sommitale limitée[4].
- En 1990, de Fraysseix, Pach et Pollack[5] ont énoncé le théorème suivant :

Théorème

Tout graphe planaire peut être représenté avec des lignes droites avec les sommets sur une grille de $2n$ x $4n$.

- Cela résout la question de la résolution, puisque la distance minimale entre deux sommets est au minimum de $\frac{\text{taille écran}}{4n}$.

L'impasse de la planarité

- Un algorithme linéaire a rapidement été trouvé[6]
- Suivi de nombreuses améliorations dans la décennie qui suivit.

- Cependant, ces algorithmes souffrent d'un (autre) problème de résolution angulaire, et aboutissent à des positions peu conventionnelles.

Remarque

Bien que la planarité soit un critère esthétique fondamental, aujourd'hui presque aucun algorithme reposant sur la planarité n'est adopté commercialement, au profit des méthodes "orientées forces", que nous allons voir maintenant.

Principe de la spatialisation avec des forces

- En parallèle du travail sur la planarité, des améliorations de l'algorithme de Tutte ont été proposées, en poursuivant l'analogie avec la mécanique
- de nouvelles forces ont été introduites :
  - une force reposant sur une longueur naturelle non nulle pour les ressorts (raideur)
  - une force répulsive, proportionnelle à l'inverse du carré de la distance entre les sommets

Précisément :

- Si u et v sont adjacents, $\overrightarrow{F}_{ressort}(u,v) = k_{uv}(d(u,v) - l_0)\,\overrightarrow{i}$ avec
  - k_{uv} est la constante de raideur (ou raideur) du ressort
  - $d(u,v)$ est la distance euclidienne entre u et v
  - l_0 est la longueur à vide du ressort
  - $\overrightarrow{i}$ est un vecteur unitaire (orienté de u vers v)
- Si u et v ne sont pas adjacents, $\overrightarrow{F}_{nonadj}(u,v) = \frac{r_{uv}}{d(u,v)^2}\,\overrightarrow{i}$ avec
  - r_{uv} est l'intensité de la répulsion

Principe de la spatialisation avec des forces

- La force totale exercée sur chaque sommet u du graphe est donc :

$$\overrightarrow{F}(u) = \sum_{(u,v)\in E} \overrightarrow{F}_{ressort}(u,v) + \sum_{(u,v)\notin E} \overrightarrow{F}_{nonadj}(u,v)$$

- Un minimum (local) d'énergie satisfait $\overrightarrow{F}(u) = 0$
- C'est un système d'équations non-linéaires

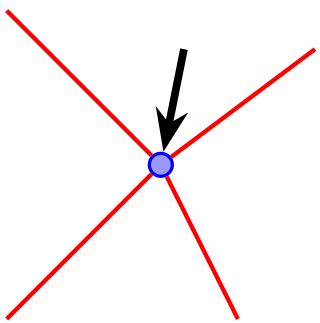
- la solution est en général **non unique**, des minima locaux ne sont pas des minima globaux
- de multiples méthodes de résolutions existent, rapides ou lentes, selon les spécificités des équations

Flexibilité de ces méthodes

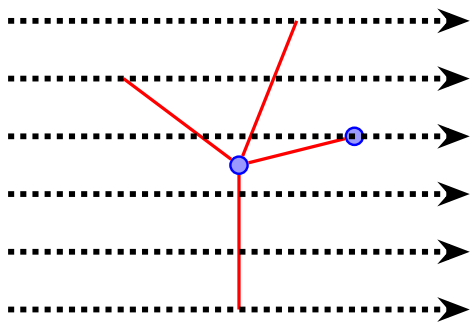
- Ces méthodes peuvent être assez facilement **contraintes**
- les constantes r_{uv}, k_{uv} et l_0 peuvent être ajustées pour s'adapter aux contraintes du domaine étudié
- Exemple : si une arête (u, v) est importante, la raideur k_{uv} peut être élevée et la longueur l_0 faible (ainsi, u et v resteront proches)

- Autres exemples :

des clous tiennent des nœuds en place



des champs magnétiques alignent nœuds et arêtes



des forces d'attraction maintiennent les communautés resserrées

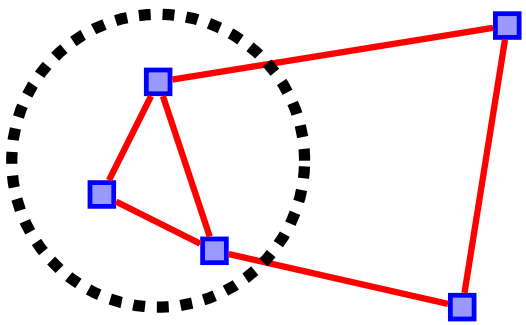
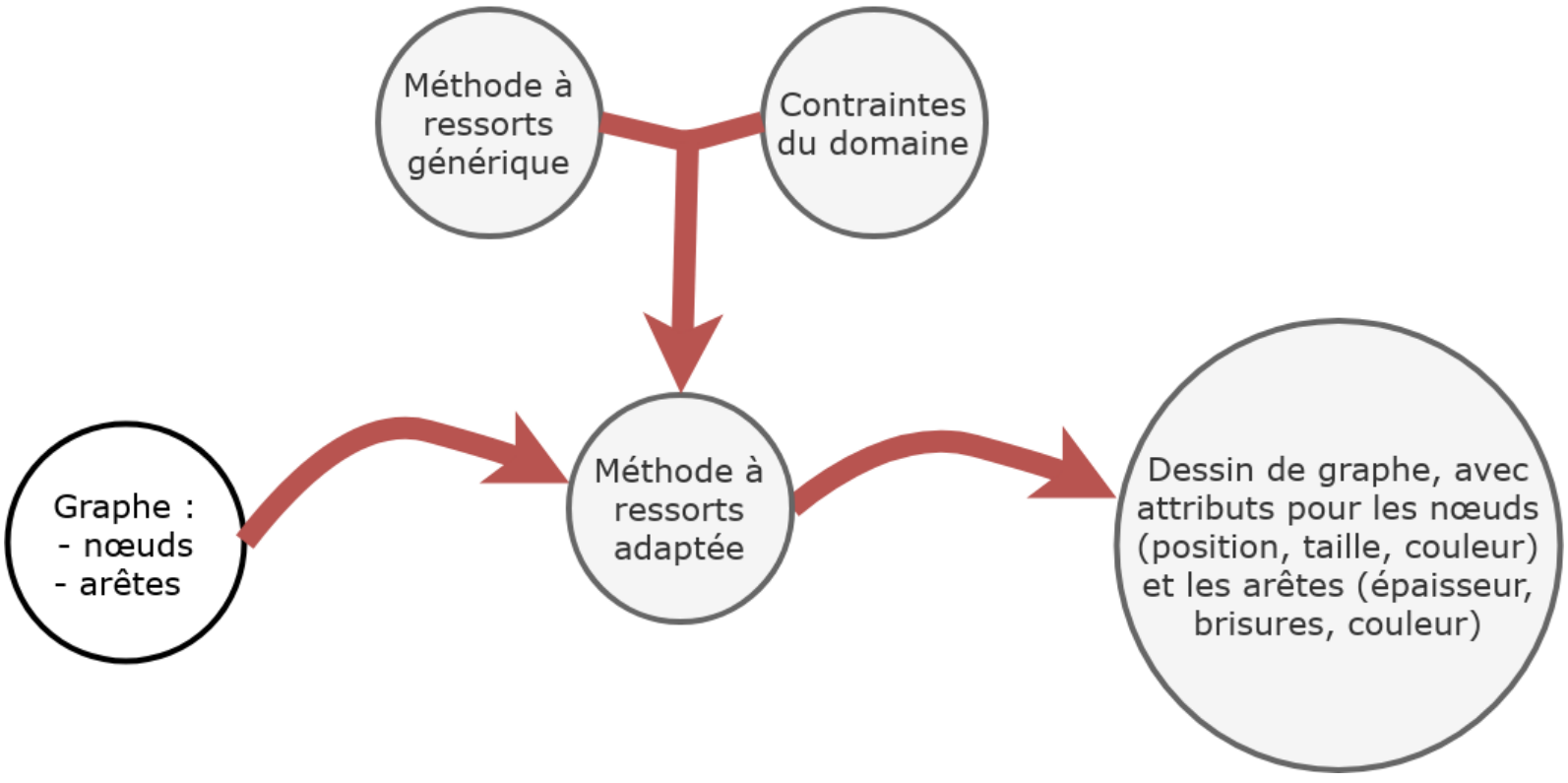


Schéma de principe



Exemples d'algorithmes

Il existe aujourd'hui de très nombreux algorithmes utilisant de façon sous-jacente le principe des équations dérivées sur des forces.

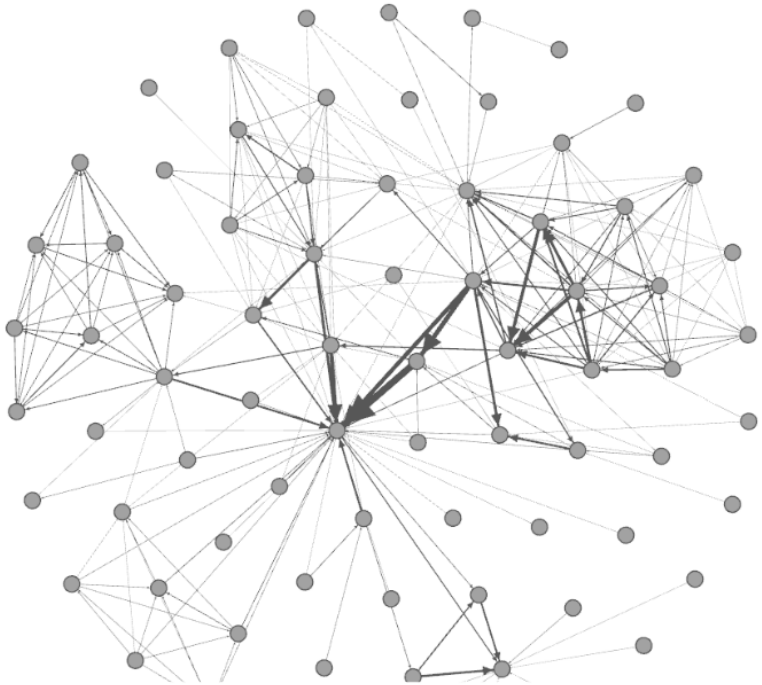
Voici quelques uns parmi les plus connus :

- Fruchterman-Reingold
- OpenOrd
- Yifan Hu
- ForceAtlas et ForceAtlas 2

Algorithme Fruchterman-Reingold

- Auteurs : Thomas Fruchterman et Edward Reingold[a]
- Date : 1991
- Complexité : $O(n^2)$
- Taille du graphe : $<$ 1000 nœuds
- Poids : Non

Considère le graphe comme une masse de particules, avec des ressorts entre les particules. Cherche une minimisation simple de l'énergie : c'est un standard, mais aujourd'hui assez lent/peu adapté aux grands graphes.

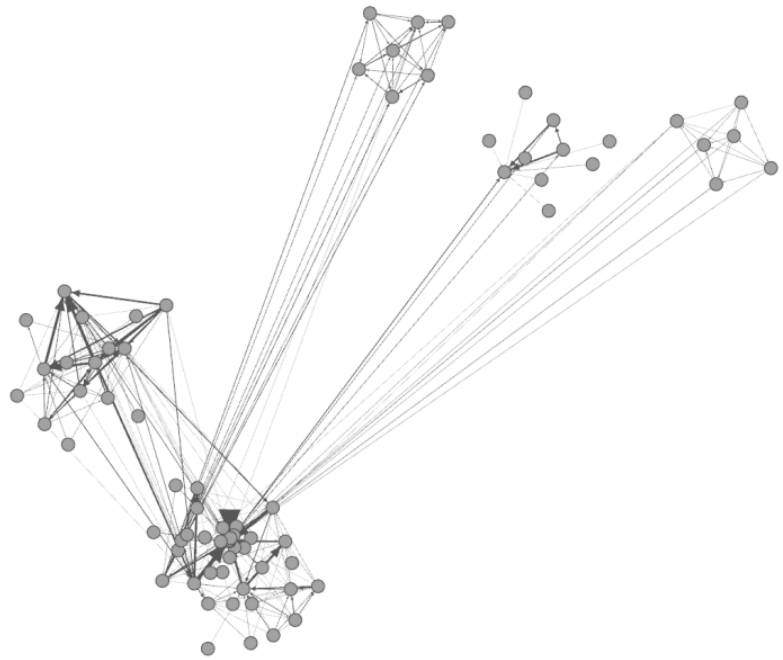


[a] FRUCHTERMAN et REINGOLD, « Graph drawing by force-directed placement ».

Algorithme OpenOrd

- Auteur : S. Martin and M. Brown and R. Klavans and KW Boyack[a]
- Date : 2011
- Complexité : $O(n * log(n))$
- Taille du graphe : $<$ 1000000 nœuds
- Poids : oui

A pour but de bien distinguer les communautés. Peut se paralléliser, s'arrête seul. Est parti d'une base Fruchterman-Reingold, marche avec un nombre fixé d'itérations controlée par un processus de recuit simulé (phase liquide, expansion, refroidissement, *crunch*, *simmer*)

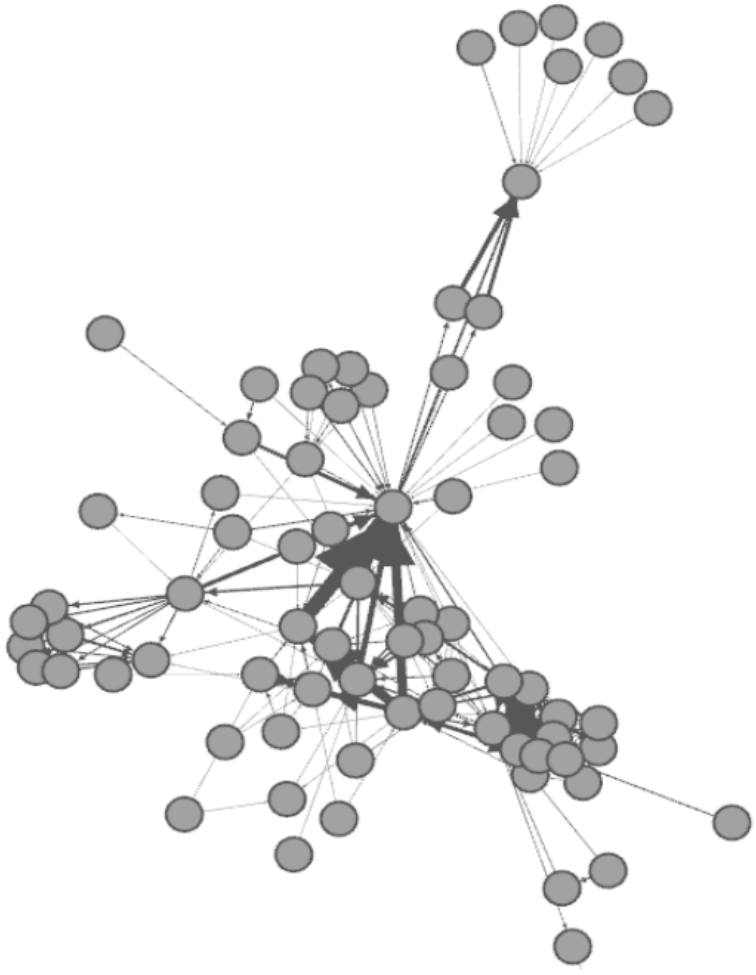


[a] Martin et al., « OpenOrd : an open-source toolbox for large graph layout ».

Algorithme Yifan Hu

- Auteur : Yifan Hu[a]
- Date : 2005
- Complexité : $O(n * log(n))$
- Taille du graphe : 100 000 nœuds
- Poids : Non

Algorithme très rapide, de bonne qualité sur les grands graphes. Combine un modèle de forces avec une technique de vision multi-échelle du graphe, pour réduire la complexité. Approximation des forces répulsives sur un nœud par une communauté de nœuds en considérant qu'il s'agit d'un *super-nœud*. S'arrête automatiquement.

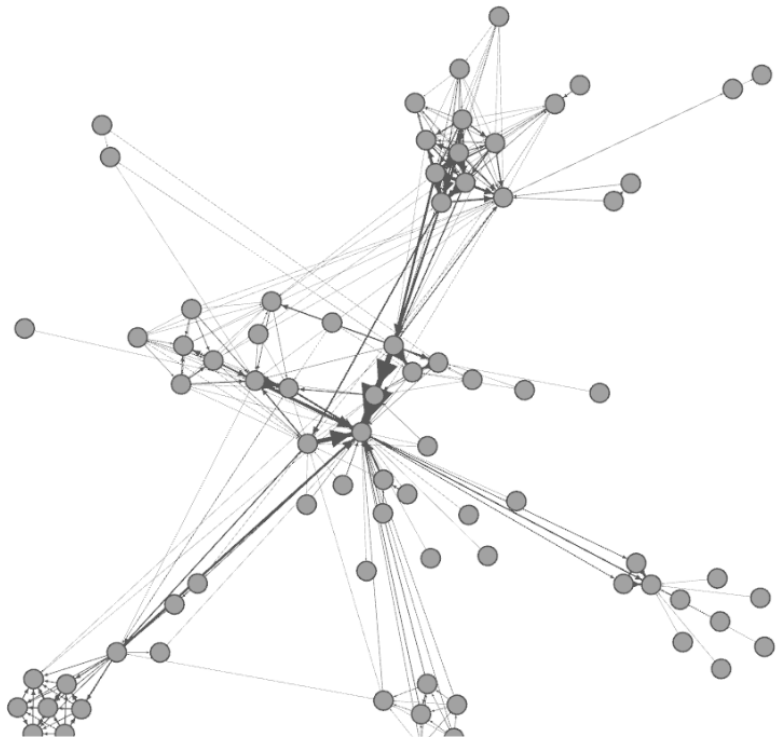


[a] Hu, « Efficient, high-quality force-directed graph drawing ».

Algorithme ForceAtlas2

- Auteur : M. Jacomy, T. Venturini, S. Heymann, M. Bastian
- Date : 2011
- Complexité : $O(n * log(n))$
- Taille du graphe : 1 000 000 nœuds
- Poids : Oui

Considère le graphe comme une masse de particules, avec des ressorts entre les particules. Cherche une minimisation simple de l'énergie : c'est un standard, mais aujourd'hui assez lent/peu adapté aux grands graphes.



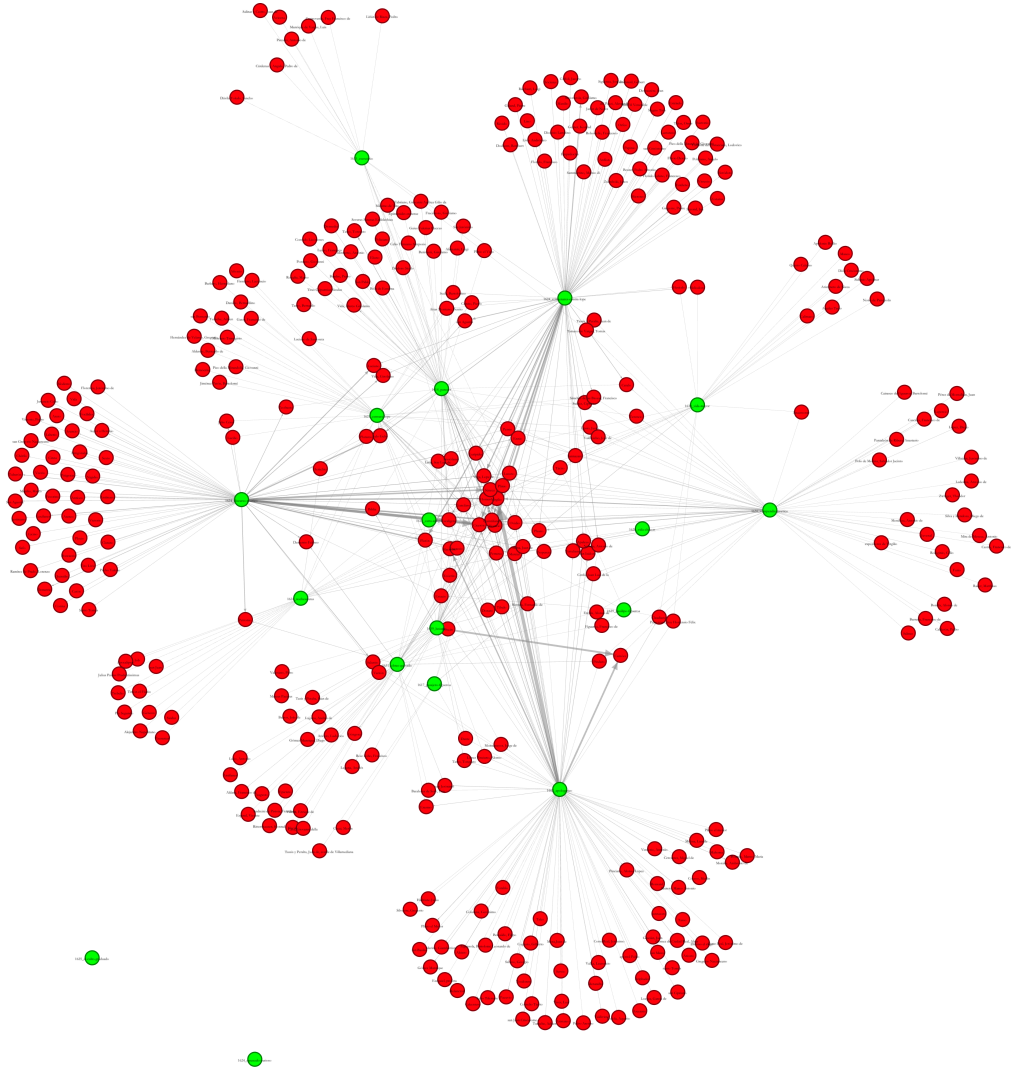
Autres spatialisations

Attention : on ne détaillera pas les algorithmes ici (ils sont plus simples), mais il existe d'autres manières de placer automatiquement les sommets d'un graphe. Citons par exemple :

- layout Radial
- layout Circulaire
- layout Géographique

Quelques exemples en image

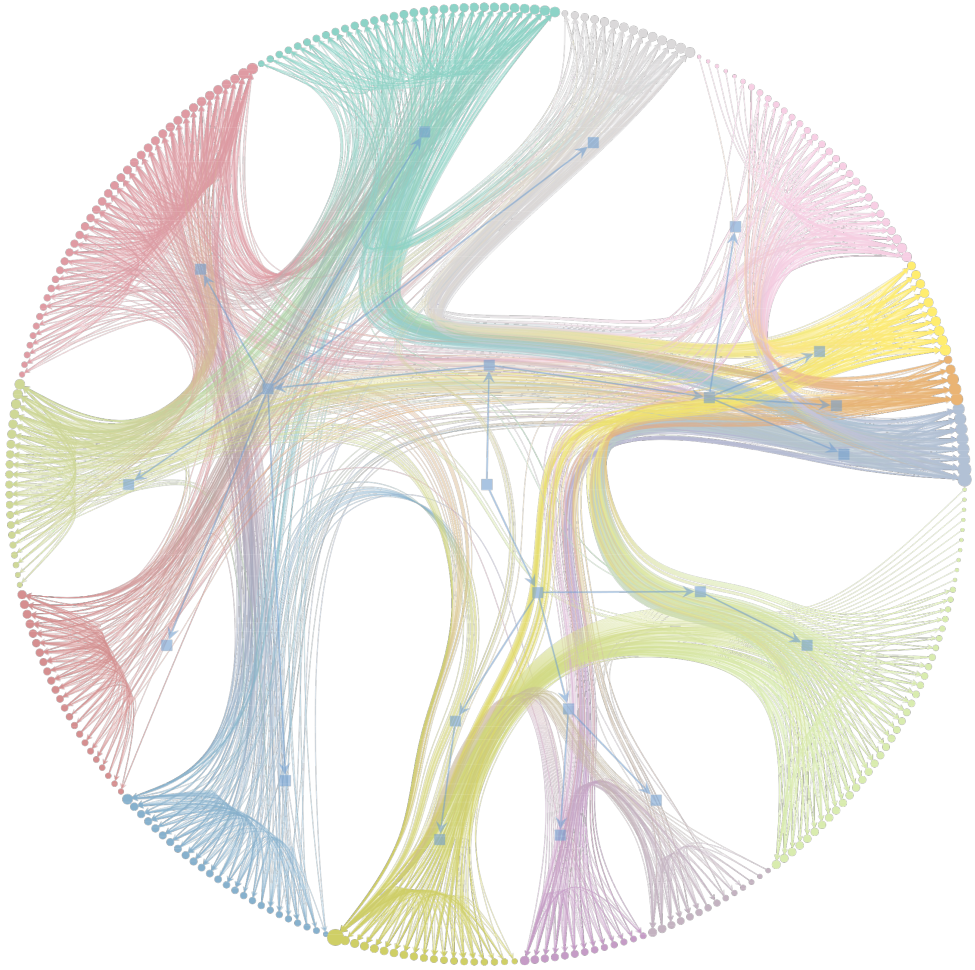
Analyse d'une polémique littéraire



Source : Plagnard et Ruiz, « Polemos2nodes/polemos2edges : première lecture de la polémique gongorine par l'analyse de réseau »

Quelques exemples en image

Réseau de neurones de *C. elegans*



Source : Watts et Strogatz, « Collective dynamics of 'small-world'networks »

Qualité

Est-ce que ces algorithmes sont **bons** ?

- Élégance
  - des méthodes simples
  - faciles à implémenter
  - des bibliothèques numériques efficaces existent
- Adéquation
  - ☑ très flexible
  - ☑ arêtes droites
  - ☒ les graphes planaires ne sont généralement pas planaires
  - ☒ les grands graphes peuvent être très emmêlés
- Performance
  - OK pour les petits graphes
  - Parfois OK pour les grands graphes (résolution numérique)

État de l'art scientifique

- Très peu de résultats sur les dessins obtenus par des spatialisation avec forces
- Très peu de théorèmes prouvés sur ces méthodes
  - le théorème de Tutte
  - un théorème de symétrie du résultat
  - quelques théorèmes de "positionnement multidimensionnel" s'appliquent
- Quelques comparaisons empiriques ont été réalisées
- Beaucoup d'analyses peu formalisées
  - faisant appel à l'intuition du développeur
  - des cas d'études spécifiques

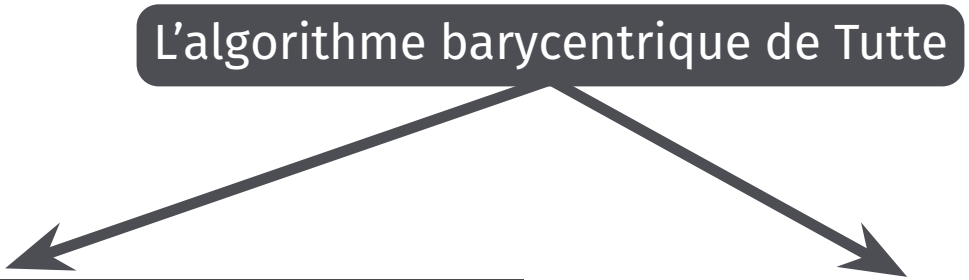
État de l'art commercial

Cette technique reposant sur des "forces" a engendré

- de nombreuses méthodes dérivées
- de nombreux outils développés (cf plus loin)
- beaucoup de brevets

$\Rightarrow$ omniprésence actuelle

Deux voies de travail



Méthodes "planaires"

- bien étayées par la théorie
  - mathématiquement
  - psychologiquement
- Qualité démontrée empiriquement
- Rarement utilisées en pratique
- Peu de brevets
- Algorithmes parfois délicats à coder

Méthodes "forces"

- Qualité scientiquement peu étayée
- Validation empirique faible
- Peu de fondements théoriques
- Utilisées universellement
- Nombreux brevets
- La plupart sont simples à coder
- Flexibilité

Outils et formats

Logiciels et bibliothèques

Logiciels intégrés :

- Gephi (Win/Mac/Linux, cf TP)
  http://gephi.org
- Tulip
  https://tulip.labri.fr/TulipDrupal/
- Cytoscape (Win/Mac/Linux)
  https://cytoscape.org/
- UCINET (Win)
  https://sites.google.com/site/ucinetsoftware/home
- Pajek
  http://mrvar.fdv.uni-lj.si/pajek/
- GraphViz (Win/Mac/Linux)
  https://graphviz.org/
- ...

Bibliothèques :

- igraph (Python, R)
- networkx (Python)
- graph-tool (Python)
- graphX / GraphFrames (Spark)
  https://graphframes.github.io/graphframes/docs/_site/index.html
- Jung (Java)
- cytoscape.js

Formats

On a vu dans la première séance diverses manières de représenter les graphes (listes de liens, matrices d'adjacence) Elles se traduisent dans divers formats

numériques d'encodage :

- GraphML
- GEXF
- DOT (GraphViz)
- DL (UCINET)
- .Net (Pajek)
- TLP (Tulip)
- et : CSV / Feuille de calcul

Formats : Pajek

```
*Network DGG.net [2-Mode]
% Davis, Gardner & Gardner (1941): Deep South. Chicago UP.
*Vertices  32 18
 1 "Evelyn"     0.2941 0.5098
 2 "Laura"      0.2941 0.5686
 3 "Theresa"    0.2941 0.6863
[...]
32 "E14"        0.8824 0.6471
*Arcs
*Edges
 1 19
 1 20
 1 21
 1 22
 1 23
[...]
18 27
18 29
```

Formats : GraphViz

```
digraph D {

  A [shape=diamond]
  B [shape=box]
  C [shape=circle]

  A -> B [style=dashed, color=grey]
  A -> C [color="black:invis:black"]
  A -> D [penwidth=5, arrowhead=none]

}
```

Formats : GraphML

```
<?xml version='1.0' encoding='utf-8'?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
   http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<key attr.name="weight" attr.type="long" for="edge" id="d2"/>
<key attr.name="name" attr.type="string" for="node" id="d1"/>
<key attr.name="name" attr.type="string" for="graph" id="d0"/>
<graph edgedefault="undirected"><data key="d0">Les Miserables</data>
<node id="0">
  <data key="d1">Myriel</data>
</node>
<node id="1">
  <data key="d1">Napoleon</data>
</node>

[...]

<edge source="0" target="1">
  <data key="d2">1</data>
</edge>
<edge source="0" target="2">
  <data key="d2">8</data>
</edge>
<edge source="0" target="3">
  <data key="d2">10</data>
</edge>
```

Formats : GEXF

```
<?xml version='1.0' encoding='utf-8'?>
<gexf version="1.2" xmlns="http://www.gexf.net/1.2draft"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/XMLSchema-instance">
  <graph defaultedgetype="directed" mode="static"
    name="Wikipedia for school">
    <attributes class="node" mode="static">
      <attribute id="0" title="name" type="string" />
      <attribute id="1" title="category" type="string" />
    </attributes>
    <meta>
      <creator>NetworkX 2.3</creator>
      <lastmodified>10/10/2019</lastmodified>
    </meta>
    <nodes>
      <node id="0" label="0">
        <attvalues>
          <attvalue for="0" value="Áedán mac Gabráin" />
          <attvalue for="1" value="History" />
        </attvalues>
      </node>

      [...]

    </nodes>
    <edges>
      <edge id="0" source="0" target="1" />
      <edge id="1" source="0" target="2" />
    </edges>
  </graph>
</gexf>
```

Formats : synthèse

	Edge List / Matrix Structure	XML structure	Edge weight	Attributes	Visualisation Attributes	Attribute default value	Hierarchical Graphs	Dynamics
CSV	x	-	x	-	-	-	-	-
UCInet	x	-	x	x	-	-	-	-
GraphViz	-	-	x	-	x	-	-	-
GDF	-	-	x	x	x	x	-	-
GEXF	-	x	x	x	x	x	x	x
GML	-	-	x	x	x	-	-	-
GraphML	-	x	x	x	x	x	x	-
Pajek	x	-	x	-	x	-	-	-
TLP Tulip	-	-	-	-	-	-	-	-
VNA Netdraw	-	-	x	x	-	-	-	-
Feuille de calcul	-	-	x	x	-	-	x	-

Gephi

- Projet open source démarré par des étudiants à UTC
- Outil de visualisation et d'analyse
- Écrit en Java, maintenu par une communauté large (développeurs seuls, entreprises, universités, etc.)
- http ://gephi.org

Références

Références

Ce cours repose notamment sur les travaux de :

- Peter Eades (Université de Sydney)
- Rushed Kanawati (Sorbonne Paris Nord)


Chiba, Norishige, Tadashi Yamanouchi et Takao Nishizeki. « Linear algorithms for convex drawings of planar graphs ». In : *Progress in graph theory* 173 (1984), p. 153-173.


Danjou, Julien. *Dependencies Handling in Python*. https ://julien.danjou.info/dependencies-handling-in-python-automatic-update/. 2019.


De Fraysseix, Hubert, János Pach et Richard Pollack. « How to draw a planar graph on a grid ». In : *Combinatorica* 10.1 (1990), p. 41-51.


Fruchterman, Thomas MJ et Edward M Reingold. « Graph drawing by force-directed placement ». In : *Software : Practice and experience* 21.11 (1991), p. 1129-1164.


Hu, Yifan. « Efficient, high-quality force-directed graph drawing ». In : *Mathematica journal* 10.1 (2005), p. 37-71.


Laboratories, Kanehisa. *KEGG project : p53 signaling pathway - Homo sapiens (human)*. `https://www.genome.jp/kegg-bin/show_pathway?hsa04115`. 2020.


Martin, Shawn et al. « OpenOrd : an open-source toolbox for large graph layout ». In : *Visualization and Data Analysis 2011*. T. 7868. International Society for Optics et Photonics. 2011, p. 786806.


Plagnard, Aude et Hector Ruiz. « Polemos2nodes/polemos2edges : première lecture de la polémique gongorine par l'analyse de réseau ». In : *e-Spania. Revue interdisciplinaire d'études hispaniques médiévales et modernes* 29 (2018).

Purchase, Helen C., Robert F. Cohen et Murray I James. « An experimental study of the basis for graph drawing algorithms ». In : *Journal of Experimental Algorithmics (JEA)* 2 (1997), 4-es.

Read, Ronald Cedric. *A new method for drawing a planar graph given the cyclic order of the edges at each vertex*. Faculty of Mathematics, University of Waterloo, 1986.

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

Watts, Duncan J et Steven H Strogatz. « Collective dynamics of 'small-world'networks ». In : *nature* 393.6684 (1998), p. 440-442.