



Visualization techniques for hierarchical/network data

Maria Cristina
SCC5836/0252 Visualização Computacional



Visual Mapping

<https://www.data-to-viz.com/>

What kind of data do you have? (*network*)

Which goal? (show part of a whole, show data flow, data connections)

Which techniques are available?

data vs (task) vs visualization technique

What kind of data do you have? Pick the main type using the buttons below. Then let the decision tree guide you toward your graphic possibilities.

Numeric

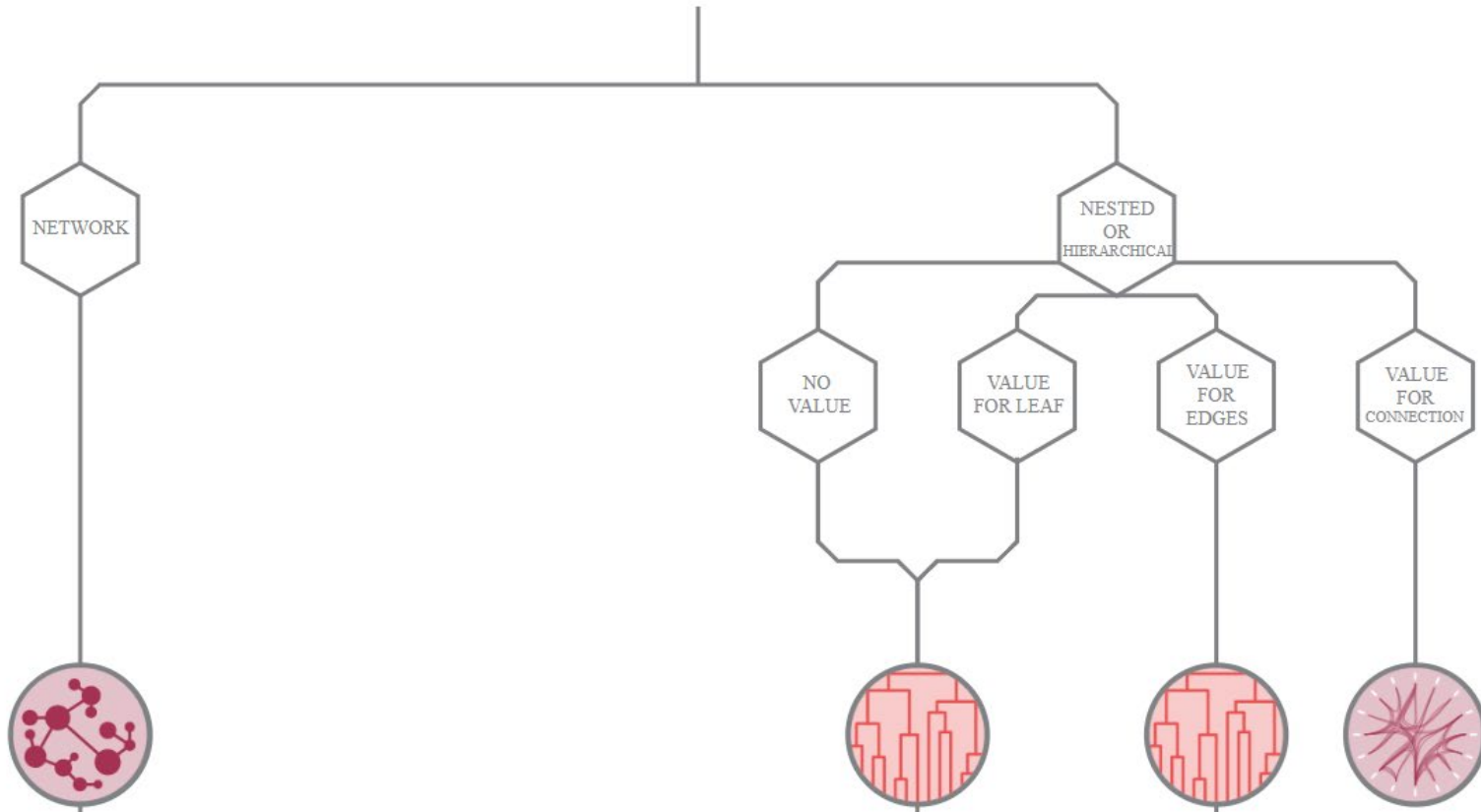
Categoric

Num & Cat

Maps

Network

Time series



A WORLD OF POSSIBILITIES

Here is an overview of all the graph types presented in this website.

Show all

Distribution

Correlation

Ranking

Part of a whole

Evolution

Map

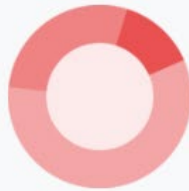
Flow



Treemap



Venn diagram



Doughnut



Pie chart



Dendrogram



Circular packing



Sunburst



A WORLD OF POSSIBILITIES

Here is an overview of all the graph types presented in this website.

Show all

Distribution

Correlation

Ranking

Part of a whole

Evolution

Map

Flow



Chord diagram



Network



Sankey



Arc diagram



Edge bundling



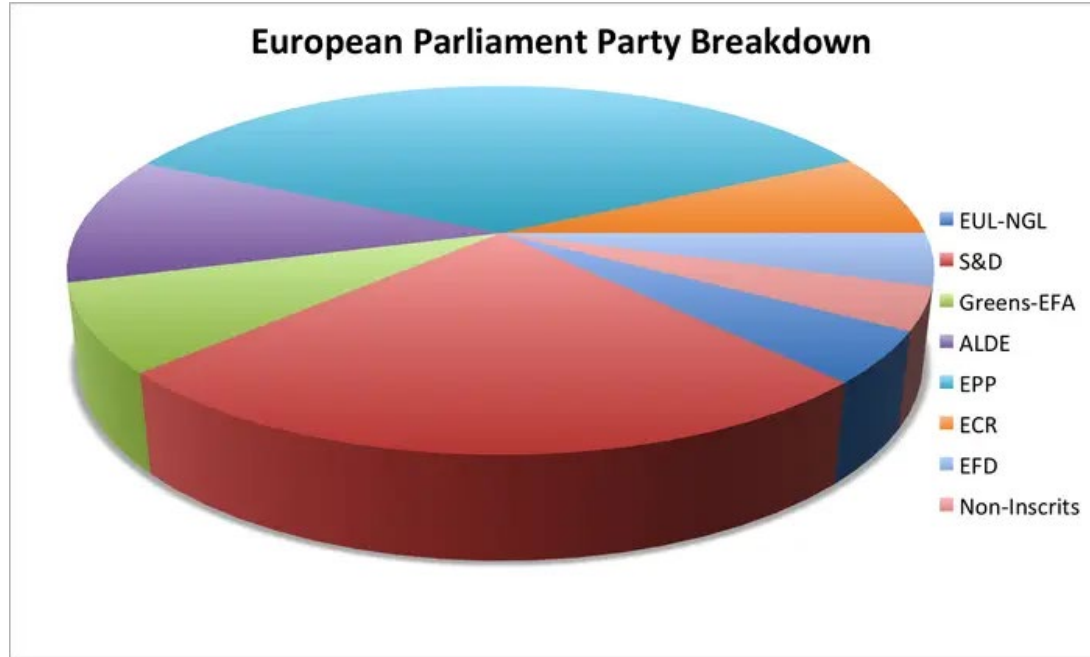
Examples

Task: show **part of a whole** (proportions/hierarchies),
show nested or hierarchical data: **pie chart , doughnut
chart , dendogram , treemap , ...**

3D Pie chart

Looking at this chart, S&D—the red party —appears to be roughly even with EPP, the teal party .

But the reality is, that's because I've distorted your perspective to make it seem as if red is in fact bigger.



source: <https://www.businessinsider.com/pie-charts-are-the-worst-2013-6>



Dendogram

a way to visualize a hierarchy or the results of a hierarchical clustering

what is hierarchical clustering?

<https://www.displayr.com/what-is-hierarchical-clustering/>



Treemap

a treemap chart represents hierarchical data in a tree-like structure

data, organized as branches and sub-branches, is represented using rectangles

the area and color of the rectangles can represent two numerical values

it is possible to drill down within the data to, theoretically, an unlimited number of levels

area filling or spacefilling technique



Treemap

Example

[https://www.fusioncharts.com/resources/chart -primers/treemap -chart](https://www.fusioncharts.com/resources/chart-primers/treemap-chart)

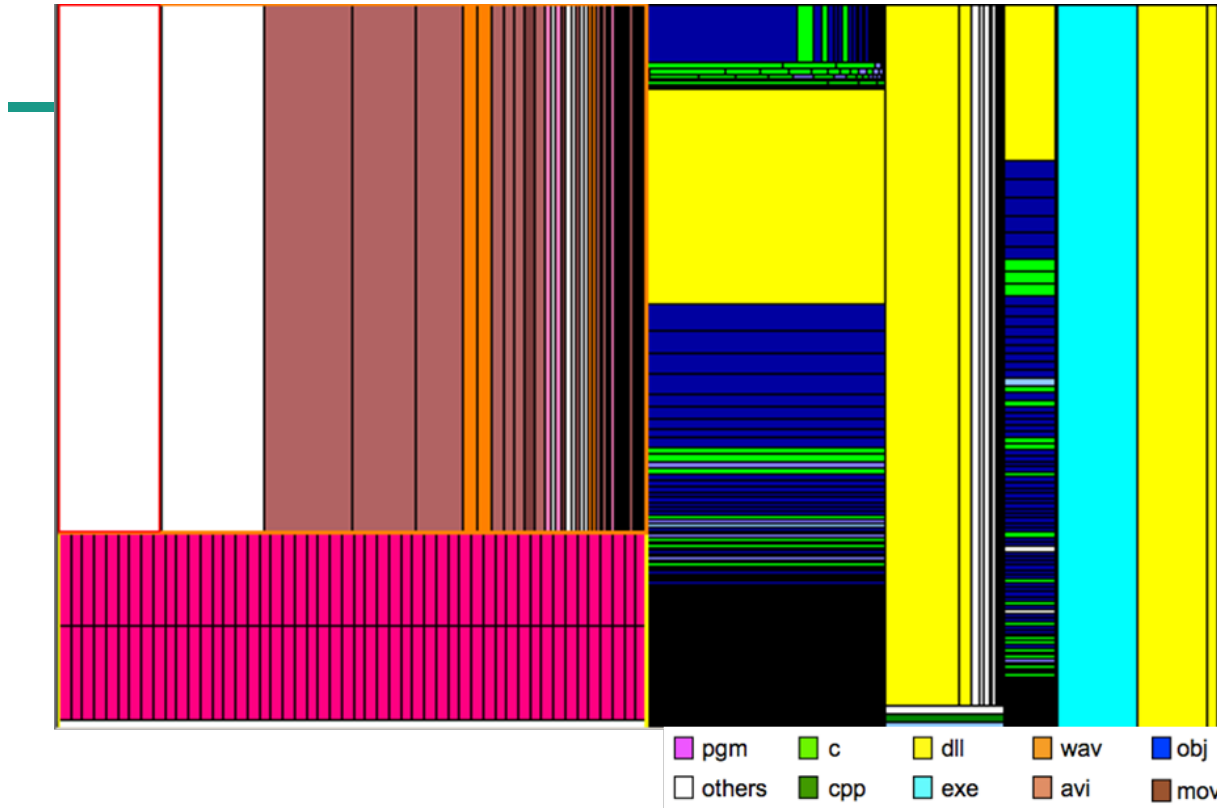
interactivity is very important



Treemaps

Basic idea: 'slice and dice' layout

- 1 node = 1 rectangle
- child node rectangles: *nested* in the parent node rectangle (recursive subdivision)
- leaf rectangle *size* and *color* show data attributes
- edges: *not drawn explicitly!*
- very compact: tens of thousands of nodes on one screen
- aspect ratios are not very good; hierarchy depth unclear



ffmpeg C library

source: Alex Telea



Example

<https://newsmap.ijmacd.com/>



Alternative approaches to visualize `trees`

Many approaches to improve basic idea of treemap

Many alternative approaches to visualize trees...

<https://treevis.net/>



Example

www.onezoom.org



Networks/Graphs

Force-based placement layout



Graph drawing /network visualization

Force-directed graph layout algorithms are a class of algorithms for drawing graphs in an aesthetically-pleasing way.

Also called *spring embedders*

Goal is to position the nodes of a graph in two-dimensional or three-dimensional space so that all the edges are of more or less equal length and there are as few crossing edges as possible.



Graph drawing /network visualization

General layout problem

Input: Graph $G = (V, E)$

Output: clear and readable node-link drawing of G (straight line edges)

Aesthetics criteria ?



Graph drawing /network visualization

Aesthetics criteria

Adjacent vertices close

Non-adjacent vertices far apart

Edges short, straight lines, similar length

Densely connected parts (clusters) form `communities`

As few line crossings as possible

Even distribution of vertices in space



Graph drawing /network visualization

Optimization criteria partially contradict each other ... nooptimal solution meets all

Aesthetics criteria

- Adjacent vertices close

- Non-adjacent vertices far apart

- Edgesshort, straight lines, similar length

- Densely connected parts (clusters) form `communities`

- As few line crossings as possible

- Even distribution of vertices in space



Graph drawing /network visualization

Peter Eades⁸⁴ - “analogy with a mechanical system”

Vertices as steel rings, edges as springs

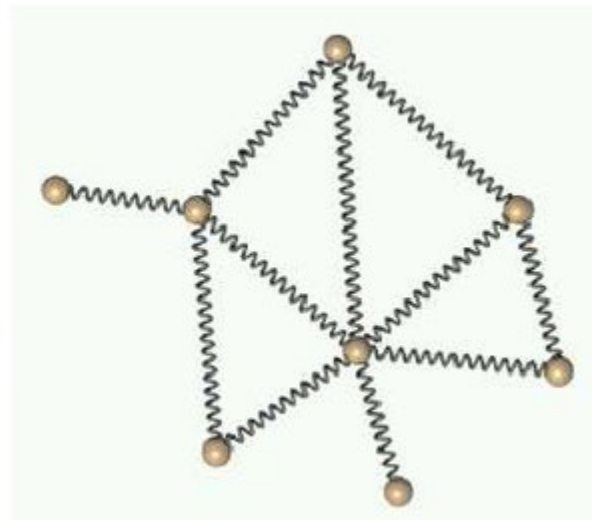
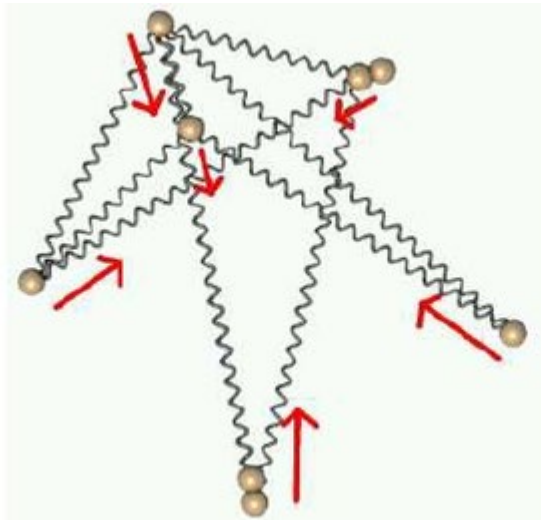
Initial layout with the rings placed at arbitrary places

Adjacent vertices connected by springs (edges)

Let go, spring forces act on the rings and move them, until the overall system reaches a stable minimum energy state



Generic spring embedder



<https://arxiv.org/abs/1201.3011>



Graph drawing /network visualization

Peter Eades84 - “analogy with a mechanical system”

Vertices as steel rings, edges as springs

Attractive spring forces alone would collapse the system

Also needs repulsive forces – imagine rings are magnets all positive (or negative) which repel each other

$F_{\text{attraction}}?$

$F_{\text{repulsion}}?$



Force -directed layouts

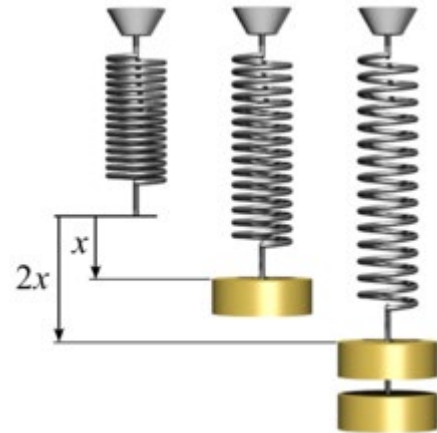
Typically, the attractive force between the nodes modeled as springs are calculated using [Hooke's law](#).

On the other hand, the magnet forces to push two nodes away from each other are modeled using [Coulomb's law](#).

Hooke's law

the **force** (F) needed to extend or compress a **spring** by some distance (x) **scales linearly** with respect to that distance—that is, $F_s = kx$, where k is a constant factor characteristic of the spring (i.e., its **stiffness**), and x is small compared to the total possible deformation of the spring.

https://en.wikipedia.org/wiki/Hooke%27s_law





Coulomb's law

the magnitude of the electrostatic **force** of attraction or repulsion between two point **charges** is directly proportional to the product of the magnitudes of charges and inversely proportional to the square of the distance between them

$$|F| = k_e \frac{|q_1||q_2|}{r^2}$$

k_e is **Coulomb's constant** ($k_e \approx 8.988 \times 10^9 \text{ N}\cdot\text{m}^2\cdot\text{C}^{-2}$),^[1] q_1 and q_2 are the signed magnitudes of the charges, and the scalar r is the distance between the charges.



Example D3

<https://observablehq.com/@d3/force-directed-graph>

force-layout: many algorithms, basis for node-link graph visualizations...

in Python: NetworkX and PyVis



Example yFiles

<https://www.yworks.com/pages/force-directed-graph-layout>



Force-directed layouts - algorithm

The attractive `spring` forces attract pairs of nodes linked by edges towards each other, while simultaneously the repulsive `electrical` forces tend to pull all pairs of nodes apart

The behavior of the entire graph under these forces is simulated as if it were a physical system: the forces are applied to all the nodes, pulling them closer together or pushing them further apart

This is repeated iteratively until the system comes to a **mechanical equilibrium** state; i.e., their relative positions do not change anymore from one iteration to the next.

The positions of the nodes in this equilibrium are used to generate a drawing of the graph



Force-directed layouts

```
Given  $G(V, E)$            {the vertices are assigned random initial positions}
At each iteration        {parameters: # iterations, threshold value}
    for each vertex  $v$  in  $V$ 
        compute the repulsive forces relative to all other vertices
        update resultant force in  $v$ 
    for each edge  $e = (u, v)$  in  $E$ 
        compute the attractive forces relative to end nodes  $u$  and  $v$ 
        update resultant forces in  $u$  and  $v$ 
    for each vertex  $v$  in  $V$ 
        adjust placement of  $v$  according to final force in  $v$ 
    update parameters for iteration
end
```



Force-directed layouts

See example (graph-based visualization) here:

<https://colah.github.io/posts/2014-10-Visualizing-MNIST/>



Force-directed layouts

design an *energy function* $E : \mathbf{R}^m \rightarrow \mathbf{R}_+$ which is low when layout is ‘good’

connected nodes should be close \rightarrow layout distance should reflect graph-theoretic distance

nodes should not overlap

aspect ratio should be balanced



Force-directed layouts

Decisions to make

how to measure the quality of a layout (define E , (e.g., edge length, node overlap, edge crossing...))

what elements of the layout we parameterize (e.g., node position)

how to efficiently and effectively minimize the energy function

Force-directed layouts

define E in terms of forces ($F = -\nabla E$)

$$\begin{aligned} \mathbf{F}_a(n_i, n_j) &= \frac{|\mathbf{p}_i - \mathbf{p}_j|^2}{k}, \\ \mathbf{F}_r(n_i, n_j) &= -\frac{k^2}{|\mathbf{p}_i - \mathbf{p}_j|} \end{aligned}$$

[Fruchterman and Reingold '91]

$$\begin{aligned} \mathbf{F}_a(n_i, n_j) &= k \log |\mathbf{p}_i - \mathbf{p}_j| \\ \mathbf{F}_r(n_i, n_j) &= -\frac{k}{|\mathbf{p}_i - \mathbf{p}_j|^2}. \end{aligned}$$

[Eades '84]

or else directly:

$$= \frac{1}{2} \sum_{i=1}^{N-1} \sum_{j=i+1}^N c_{ij} (|\mathbf{p}_i - \mathbf{p}_j| - d_{ij})^2.$$

[Kamada and Kawai '89]

find node positions p_i by minimizing E :

- move nodes iteratively along F (since $F = -\nabla E$) with some small distance

- compute F_r only w.r.t. close nodes (to save time)

- use spatial search data structures (e.g. octrees) to find close nodes

- stop when E low enough or max #iterations reached

Given $G(V, E)$, N nodes

```
for (int i=0; i<N; i++)           /* assign nodes to random initial positions
    pi = random position

float t = t0                       /* initial maximal allowed move
for (int i=1; i< ITER; i++) {      /* compute layout
    for (int i=0; i<N; i++) {
        fi = 0;
        for (int j=0; j<N; j++)    /* compute repulsive forces
            if (j!=i) fi += Fr(i,j);
        }
        for (int edge=0; edge<|E|; edge++) { /* compute attractive forces
            int i = edge.first; j = edge.second; /* get nodes adjacent to edge
            fi -= Fa(i,j); fj += Fa(i,j);
        }
    }
    for (int i=0; i<N; i++) {      /* move the nodes by applying forces
        pi += fi/|| fi || *min(delta, t*|| fi ||);
    }
    t -= t*deltat;                /* reduce maximal allowed move t
}
```



Force-directed layouts

Problems

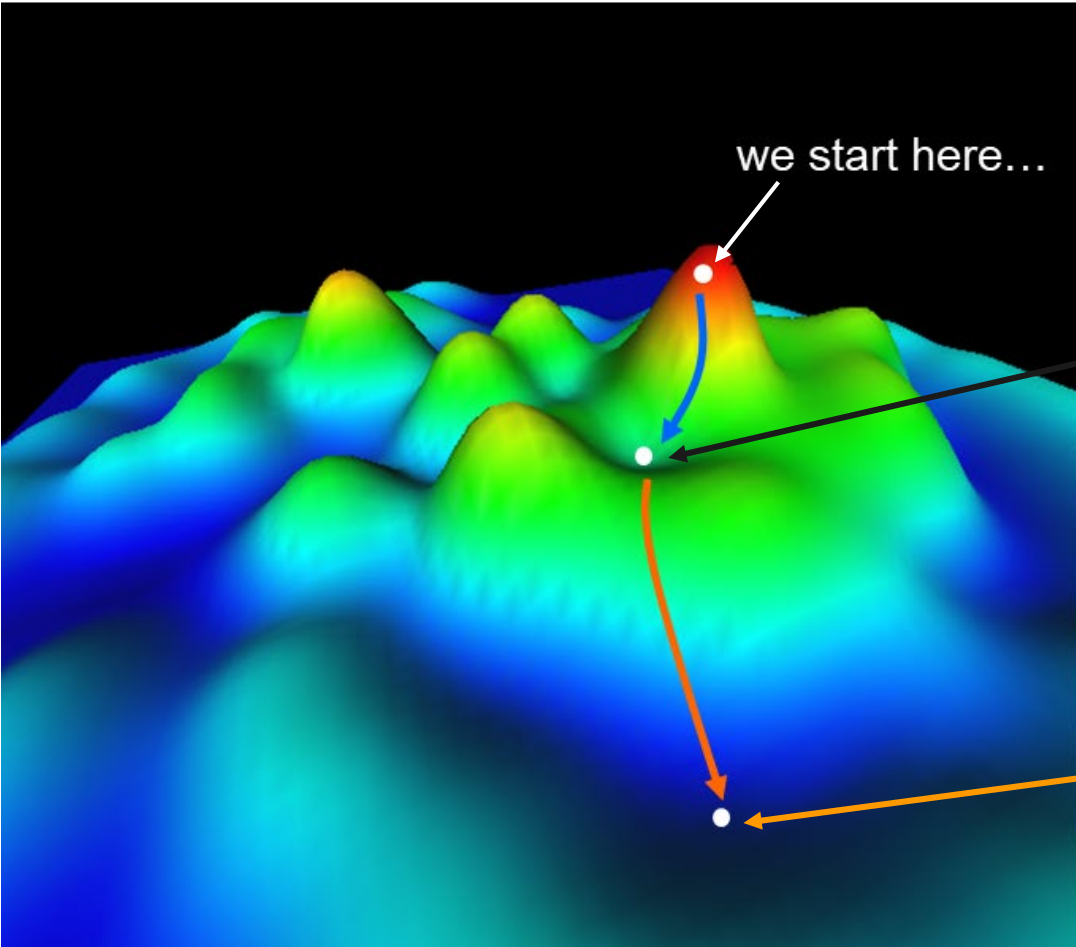
the energy function is not monotonic!

minimizers such as gradient descent work locally -> can get stuck in local minima

solve this by more advanced minimizers (see e.g. [Di Battista et al '94])

drawings not intuitive; no clear ordering -> where to start reading the drawing??

energy 2D height plot



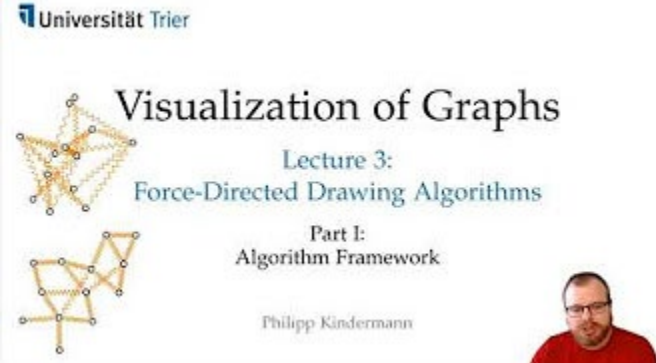
...minimizer may get stuck here (local minimum)

...instead of arriving here (global minimum)

Force-directed layouts

Excelente série de aulas:

<https://www.youtube.com/em/playlist?list=PLubYOWSI9mlvtnRjCCHP3wqNETTHYjQex>



Universität Trier

Visualization of Graphs

Lecture 3:
Force-Directed Drawing Algorithms

Part I:
Algorithm Framework

Philipp Kindermann

The slide features two graph visualizations on the left: a top one with nodes and edges in a complex, somewhat circular arrangement, and a bottom one with a more linear, zig-zag structure. A small portrait of Philipp Kindermann is in the bottom right corner.



Limitation of node link views

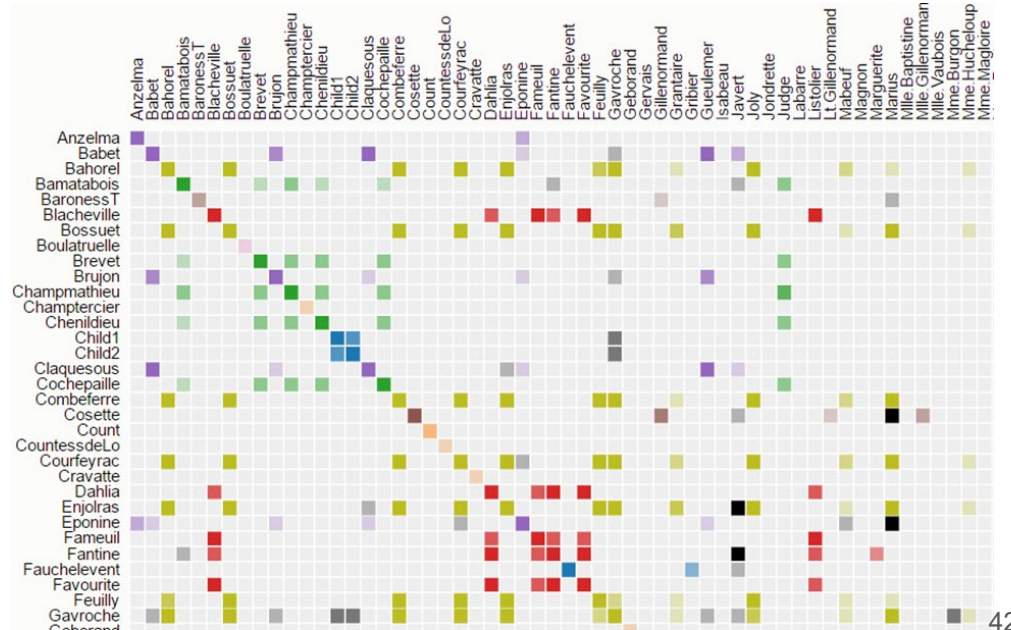
`hairball` effect on large graphs

alternatives ?

<https://eagereyes.org/techniques/graphs-hairball>

Alternative approaches to visualize `graphs/networks`

<https://bost.ocks.org/mike/miserables/>





Alternative layouts

radial/circular layouts, e.g. chord diagram and others



Graph visualization libraries

Cytoscape, software for visualising biological networks. The basepackage includes force-directed layouts as one of the built-in methods.

Gephi, an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs.

Graphviz, software that implements a multilevel force-directed layout algorithm (among many others) capable of handling very large graphs.

Tulip, software that implements most of the force-directed layout algorithms (GEM, LGL, GRIP, FM³).

Prefuse, a Java based toolkit for building interactive information visualization applications.
<https://github.com/prefuse/Prefuse>



Improvement

edge bundling (networks, trees, trajectories ,)

<https://vega.github.io/vega/examples/edge-bundling/>



References

Eades, Peter (1984), "A Heuristic for Graph Drawing", *Congressus Numerantium*, **42** (11): 149–160.

Fruchterman, Thomas M. J.; Reingold, Edward M. (1991), "Graph Drawing by Force-Directed Placement", *Software: Practice and Experience*, Wiley, **21** (11): 1129–1164, doi:10.1002/spe.4380211102, S2CID 31468174.