Universitat de Girona
Fundació UdG: Innovació i Formació

ViT Visualization for Transparency Foundation

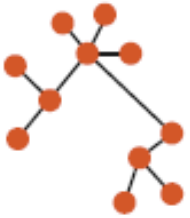
MVTEC

Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera **Data Visualization Programming** Anton Bardera Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera Data Visualization Programming Anton Bardera Data Visualization

Graph representations

- Three types:

Node–Link Diagrams
Connection Marks
✔ NETWORKS ✔ TREES



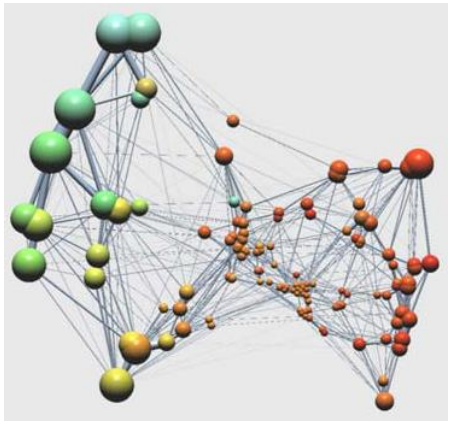
Adjacency Matrix
Derived Table
✔ NETWORKS ✔ TREES

Enclosure
Containment Marks
✘ NETWORKS ✔ TREES

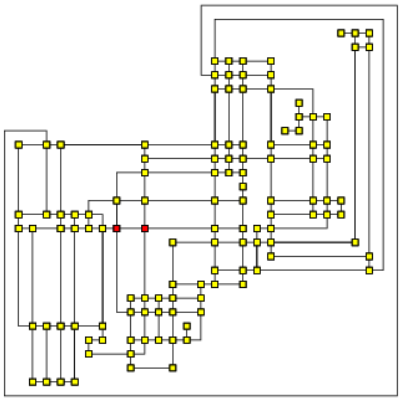
From T. Munzer, 2014

Graph representations

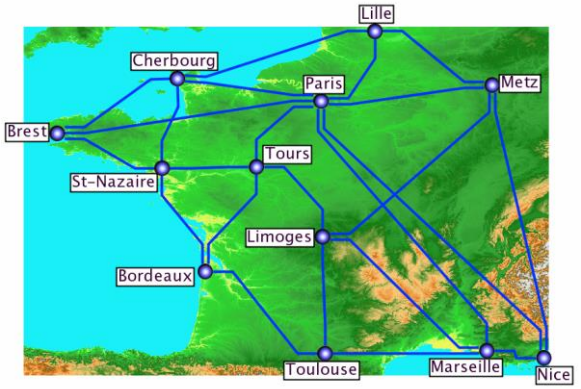
- Node-Link Layout
  - Vertex represented by points
  - Edges represented by lines or arcs



Free



Styled



Fixed

From HJ Schulz's PhD Thesis

Graph representations

- Criteria for good node-link layout
  - Minimized **edge crossings**
  - Minimized **distance** of neighboring nodes
  - Minimized **drawing area**
  - Uniform edge **length**
  - Minimized edge **bends**
  - Maximized **angular distance** between different edges
  - Aspect ratio about 1 (not too long and not too wide)
  - **Symmetry**: similar graph structures should look similar

list adapted from Battista et al. 1999

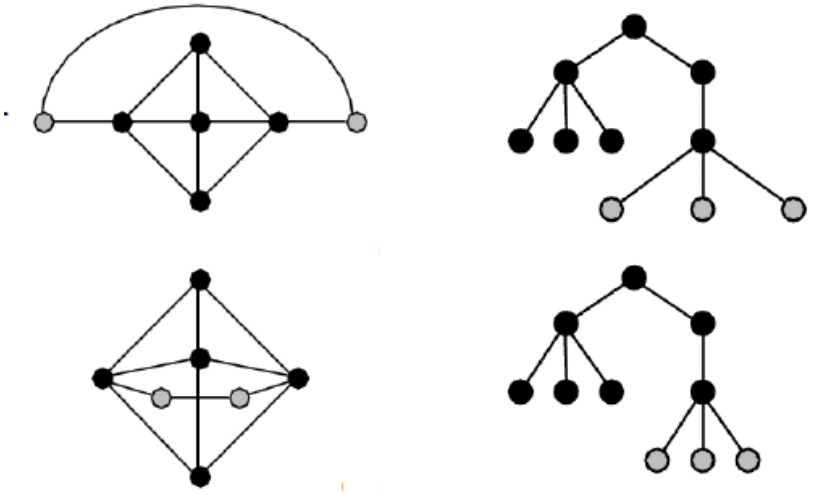
Graph representations

- Conflicting Criteria

Minimum number of edge crossings

vs.

Uniform edge length



Space utilization

vs.

Symmetry

From HJ Schulz 2006, based on A. Lex lectures

Graph vis: Force directed layouts

- A “physical” simulation is performed to determine the node positions
- Several forces are defined depending on the node and edge features
- Solved by iterative methods
- Computationally demanding
- Limit (interactive): ~1000 nodes

Graph vis: Force directed layouts

- Force directed layout in D3.js

Repulsion

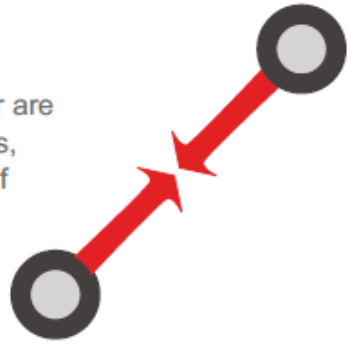
All nodes push each other away. Sometimes this force is set to be based on an attribute of a node. Larger nodes can be given more space by setting their repulsion higher, or they can act as anchors by setting their repulsion lower. In D3, this is defined using .charge().

Canvas Gravity

Nodes are pulled toward the layout center to keep the interplay of forces from pushing them out of sight. In D3, this is defined using.gravity().

Attraction

Nodes that are connected to each other are pulled toward each other. Sometimes, this force is based on the strength of connection, so that more strongly connected nodes are closer. In D3, this is defined using .linkDistance() and .linkStrength().



From E. Meeks 2015, “D3.js in Action”

Graph vis: Force directed layouts

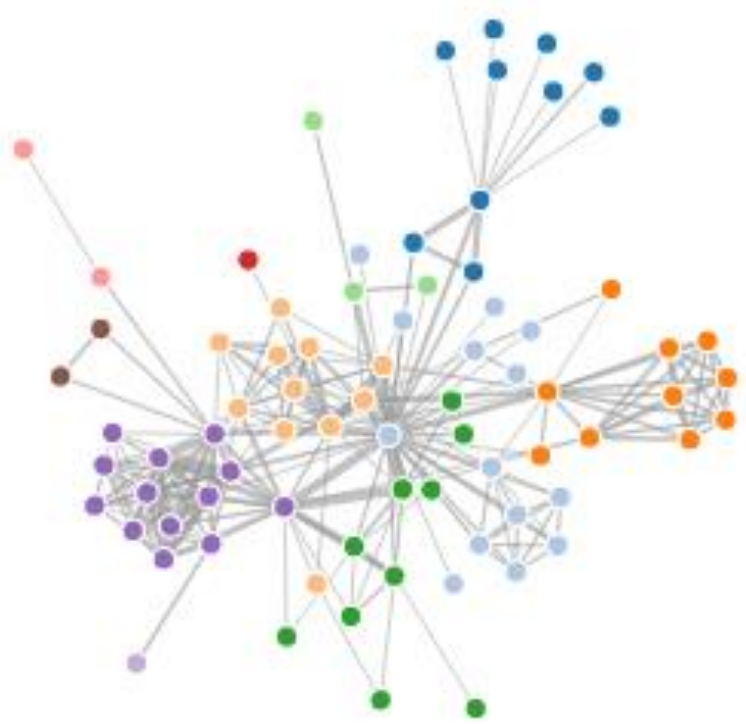
- Algorithm:
  - start from random layout
  - (global) loop:
    - for every node pair compute repulsive force
    - for every edge compute attractive force
    - accumulate forces per node
    - update each node position in direction of accumulated force
  - stop when layout is ‘good enough’

- Seems to be $o(n^2)$, but can be implemented on o(nlog(n)) using quadtrees or k-d trees
  - Barnes–Hut simulation

Graph vis: Force directed layouts

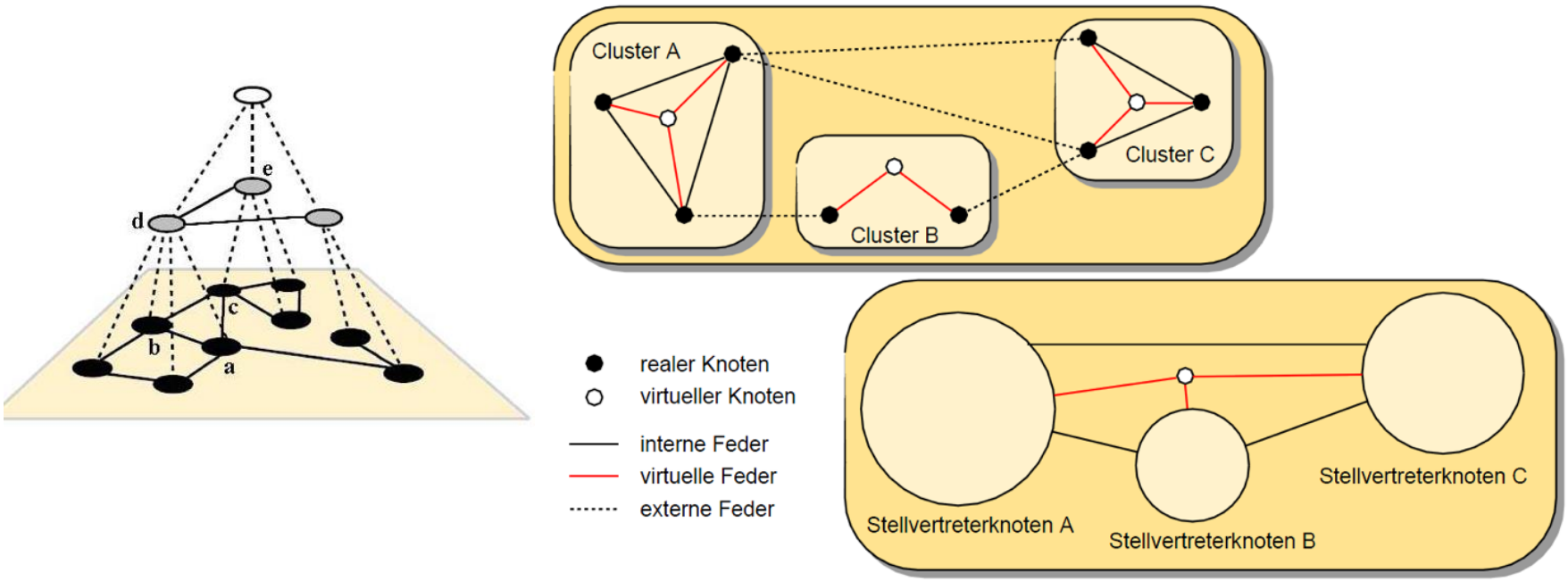
- Force directed layout example in D3.js:

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



Graph vis: Force directed layouts

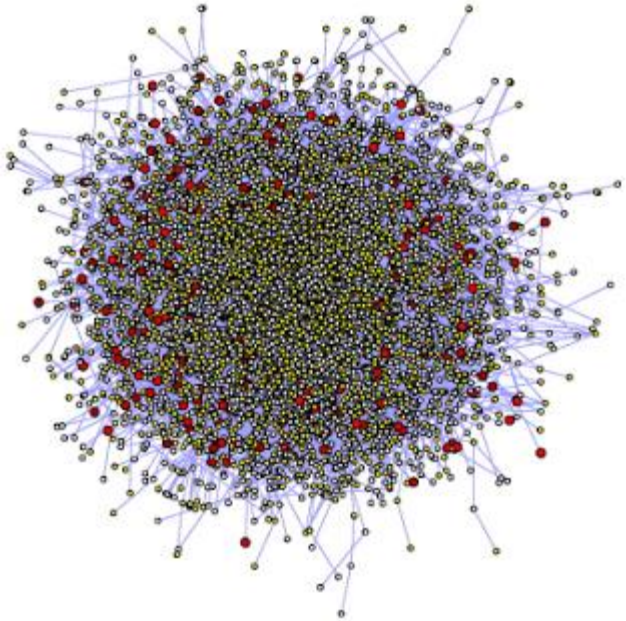
- Multilevel approaches



From HJ Schulz 2006

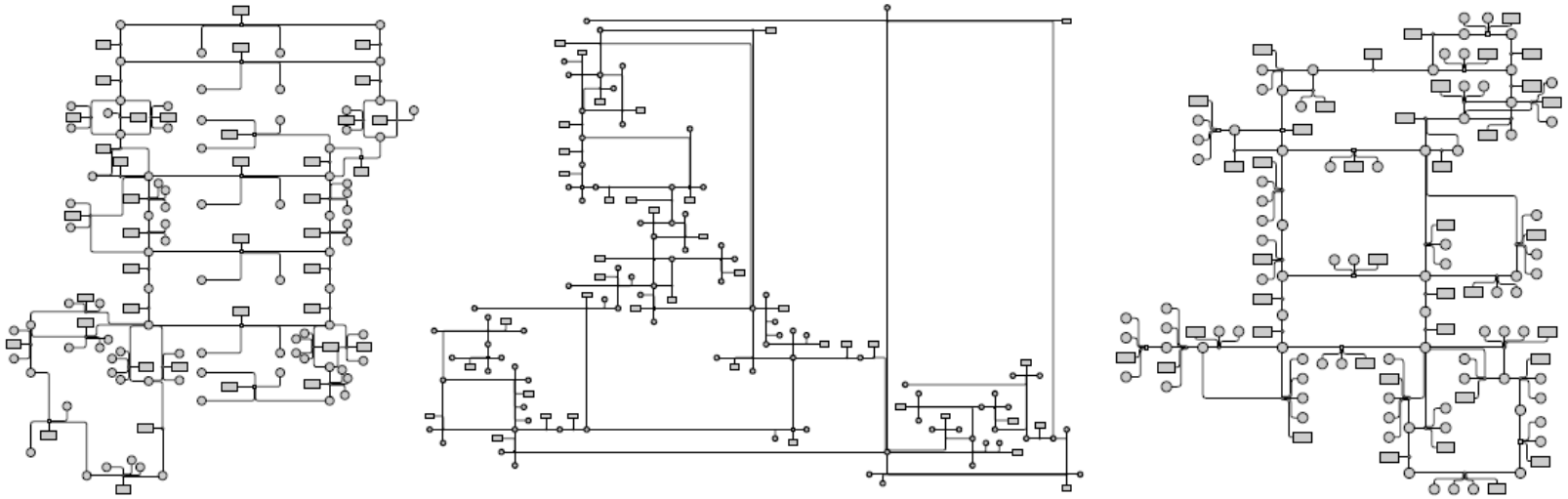
Graph vis: Force directed layouts

- Take care!
- Hairball graph



Graph vis: Orthogonal layouts

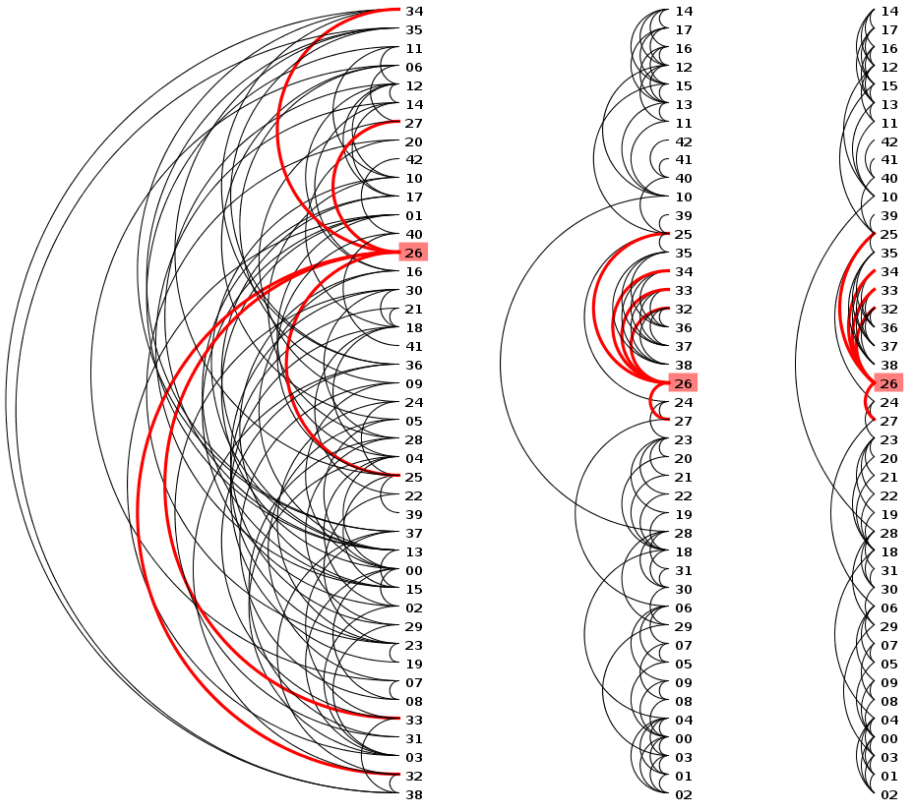
- HOLA: Human-like Orthogonal Network Layout
- Styled representation
- Analysis on human preferences
- Used in electrical engineering and software engineering



Kieffer, S.; Dwyer, T.; Marriott, K.; Wybrow, M., "HOLA: Human-like Orthogonal Network Layout," in Visualization and Computer Graphics, IEEE Trans. on , 22(1), pp.349-358, 2016

Graph vis: Restricted layouts

- Arc diagram
  - Node ordering: barycenter heuristic



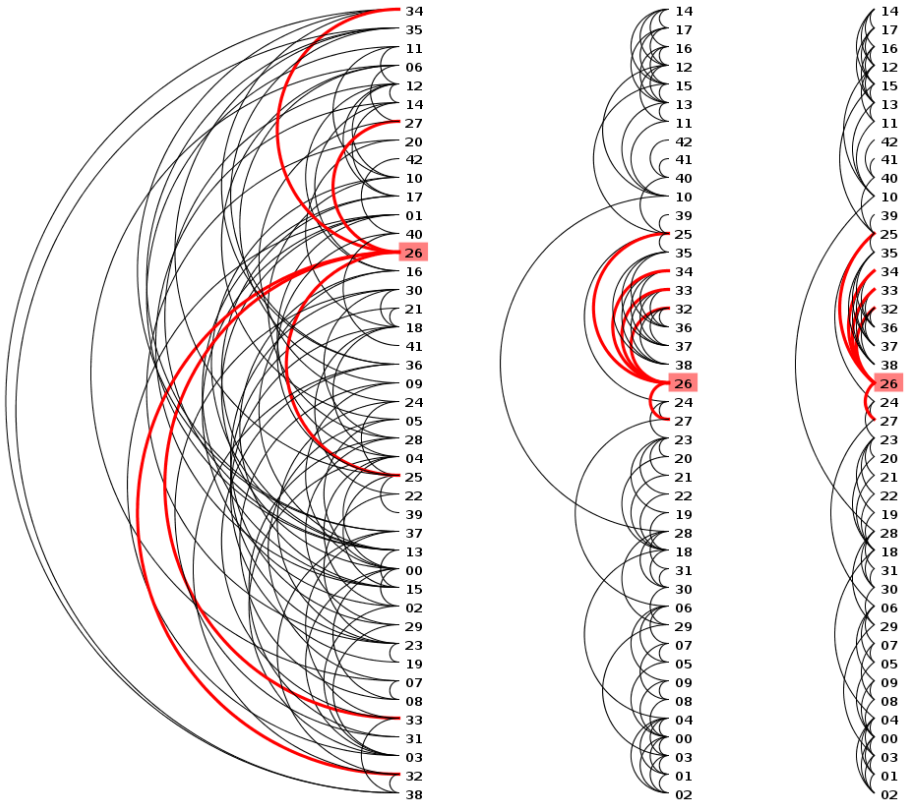
Graph vis: Restricted layouts

- Arc diagram
  - Node ordering: barycenter heuristic
  - Iterative technique:
    - Compute the average position (or “barycenter”) of the neighbors of each node
    - sort the nodes by this average position
    - repeat until convergence

- Intuitively, this should move nodes closer to their neighbors, making the arcs shorter

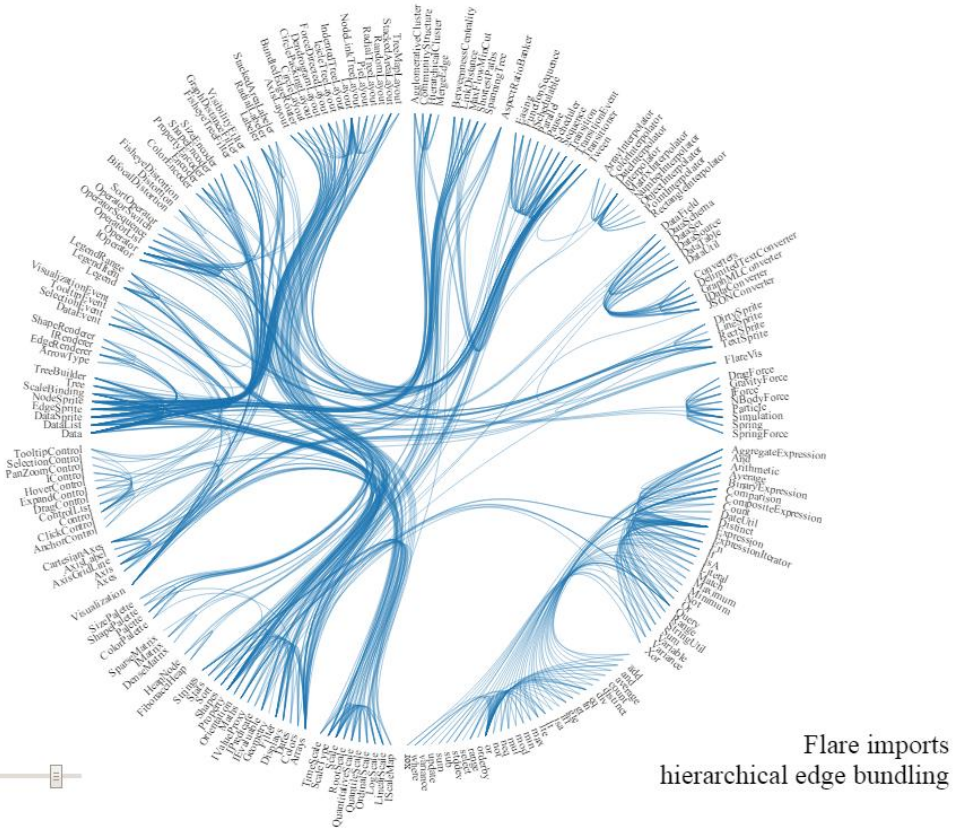
Graph vis: Restricted layouts

- Arc diagram
  - Node ordering: barycenter heuristic



Graph vis: Restricted layouts

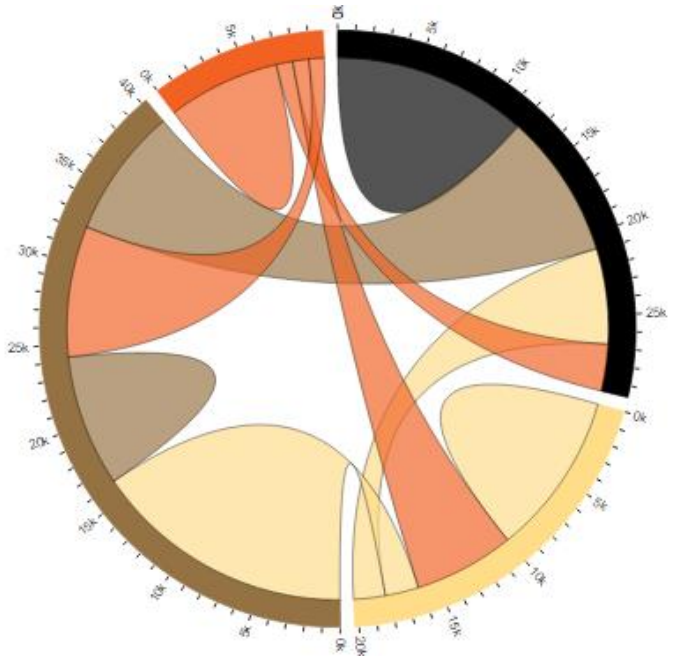
- Circular Layout, also chord diagram
  - Node ordering
  - Edge Clutter



- Example d3.js:
  https://observablehq.com/@d3/hierarchical-edge-bundling

Graph vis: Restricted layouts

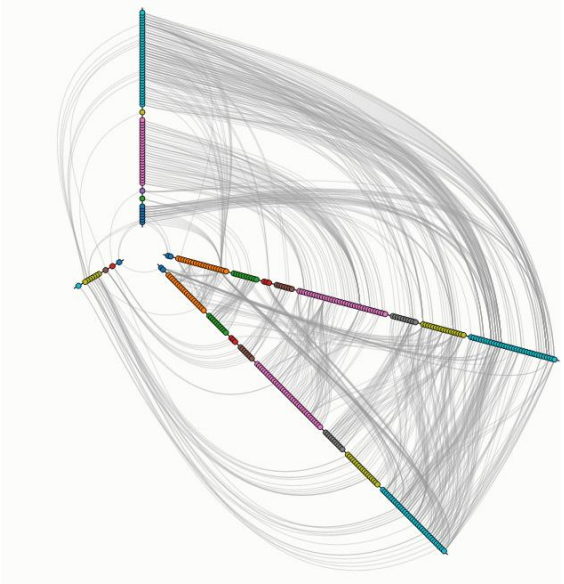
- Chord diagram can also be extended to sets and their relationships



- Example d3.js: https://observablehq.com/@d3/chord-diagram

Graph vis: Restricted layouts

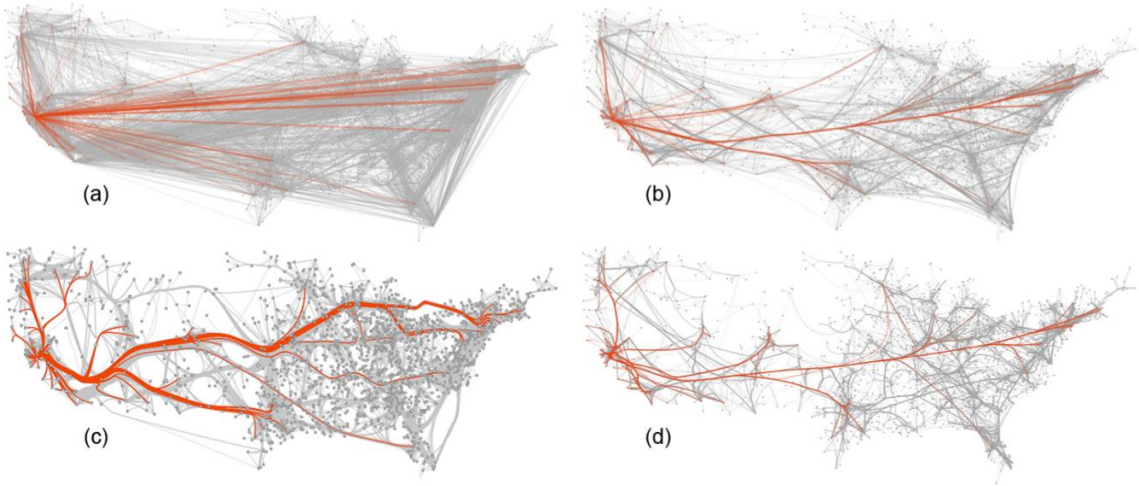
- Hive plot: http://www.hiveplot.net/
  - Nodes are positioned on radially distributed linear axes based on network structural properties
  - Edges are drawn as curved links



- Example d3.js: http://bost.ocks.org/mike/hive/

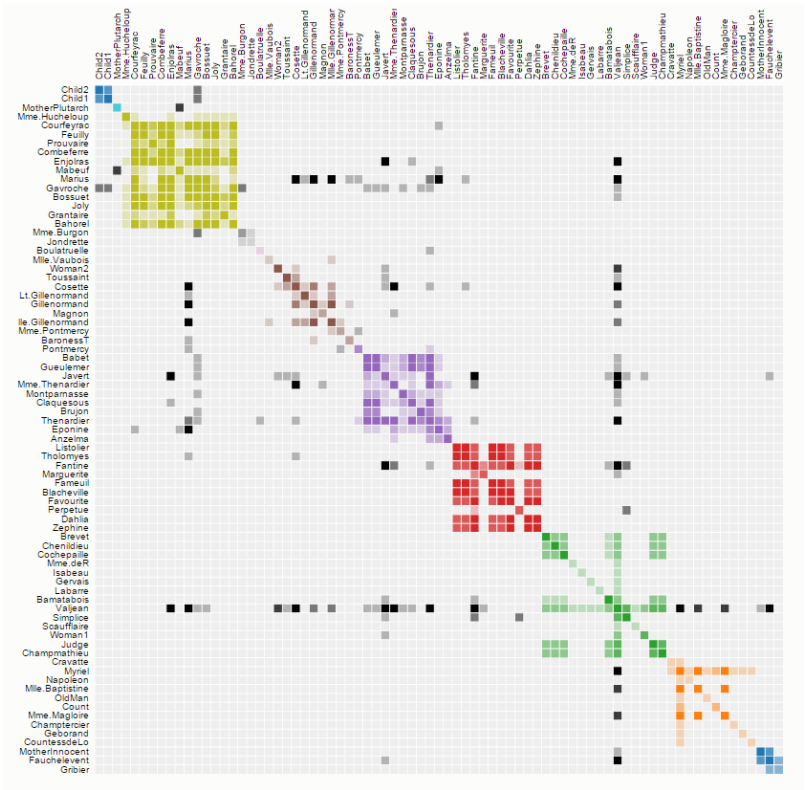
Graph vis: Fixed layouts

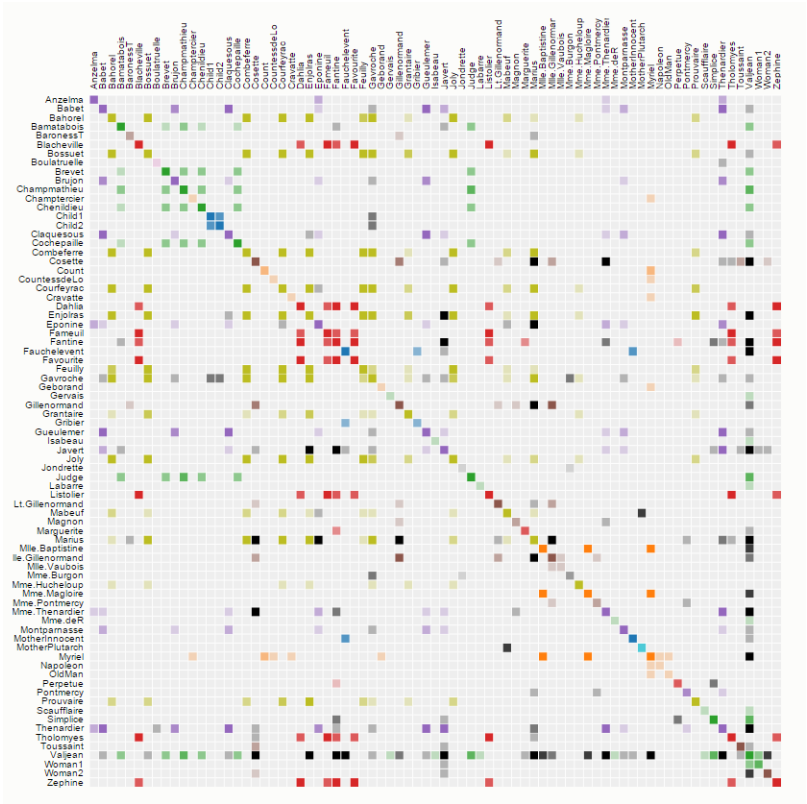
- Can’t vary position of nodes
- Edge routing important
- Edge boundling
  (next week talk!)



Graph vis: Adjacency matrix

- Order is critical!





- D3.js exemple: https://observablehq.com/@bstaats/matrix-diagram

Graph vis: Adjacency matrix

- Interpretation needs certain training