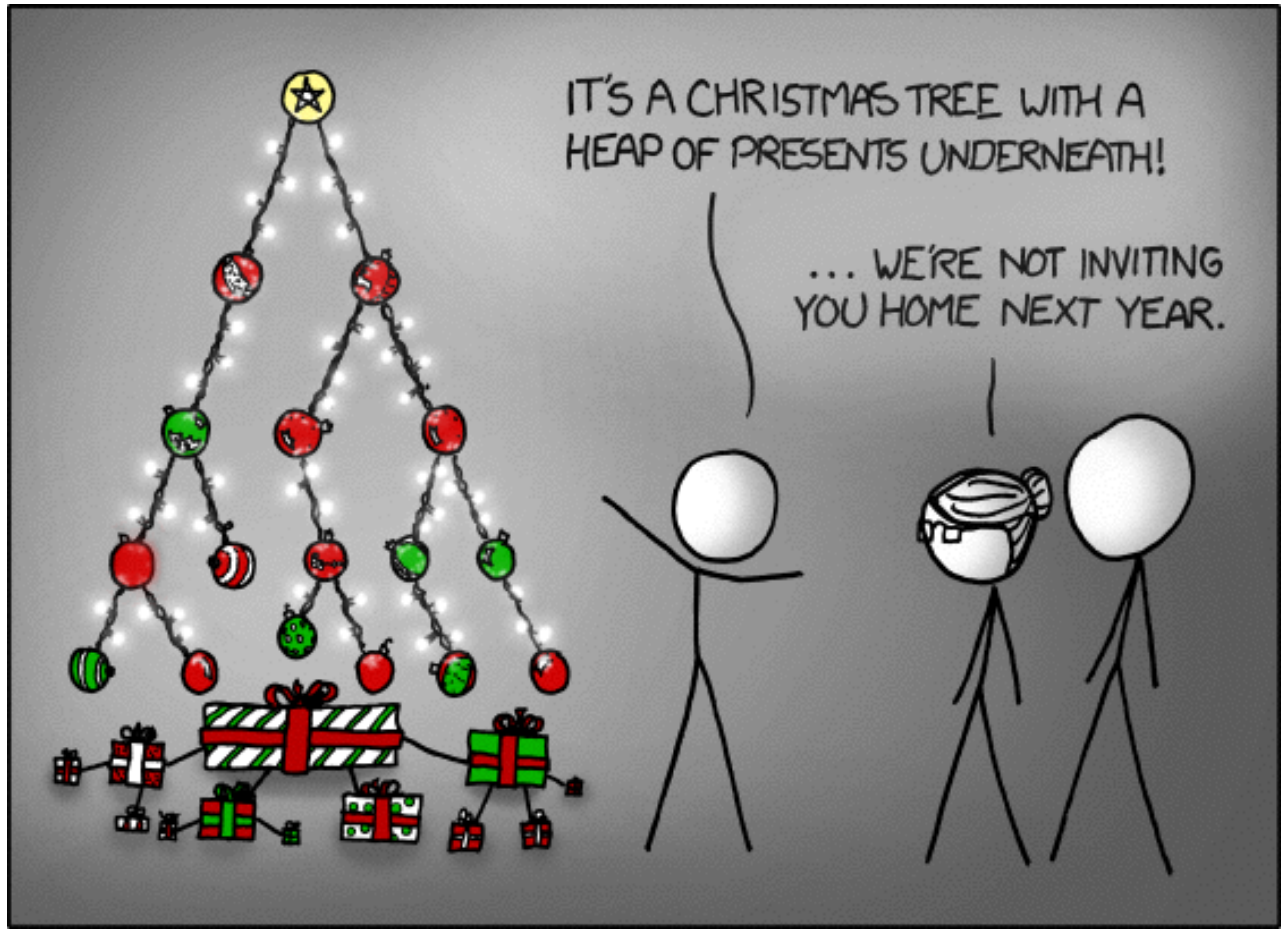
CS171 Visualization

Alexander Lex
alex@seas.harvard.edu

Graphs Part II



[xkcd]

HARVARD
School of Engineering
and Applied Sciences

This Week

Section 7: Data, data, data

Homework 3 due Friday!

Homework 4 due Friday!

- Project Proposal
- Announce project repositories!
- Don’t have a group - e-mail now!

Next Week

Tuesday Lecture: Social Visualization

- Guest Speakers: Fernanda Viegas & Martin Wattenberg. Co-leaders of Google's "Big Picture" data visualization group.

Thursday Lecture: Visualization and Arts

- Guest Speakers: Mark Schifferli and Terrence Fradet from Fathom

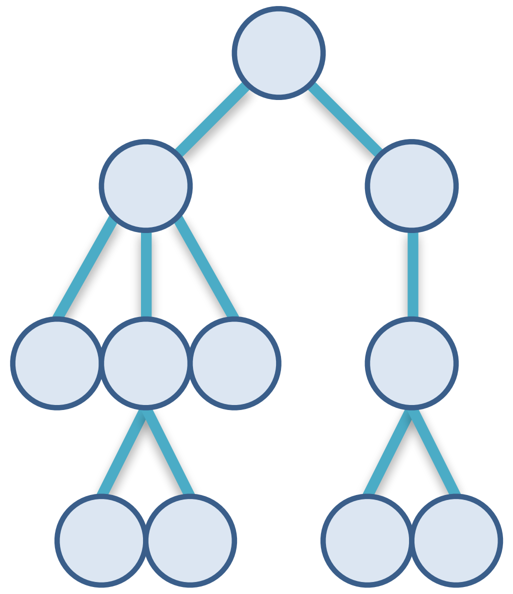
Graph Visualization

Based on Slides by HJ Schulz and M Streit

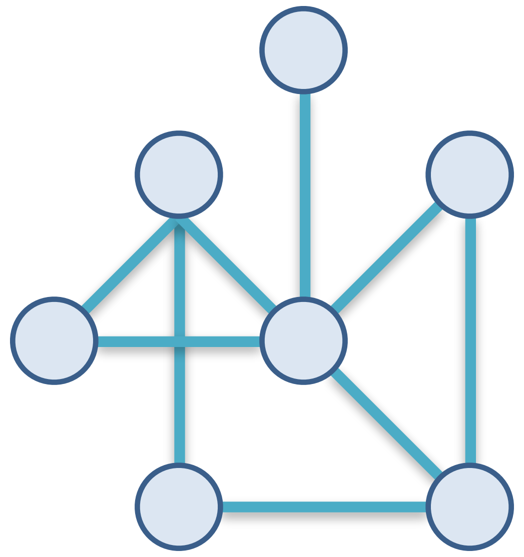

facebook
December 2010

Graph Theory Fundamentals

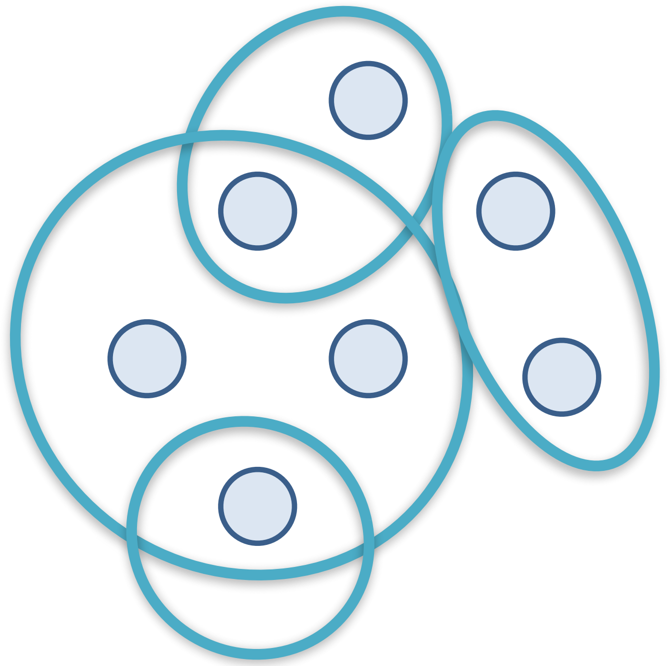
Tree



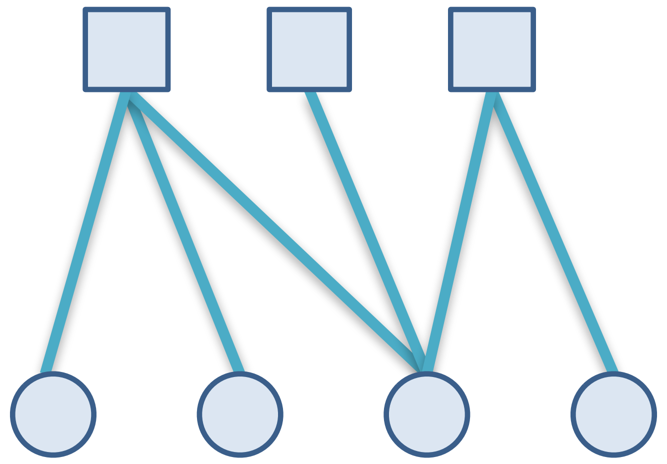
Network



Hypergraph



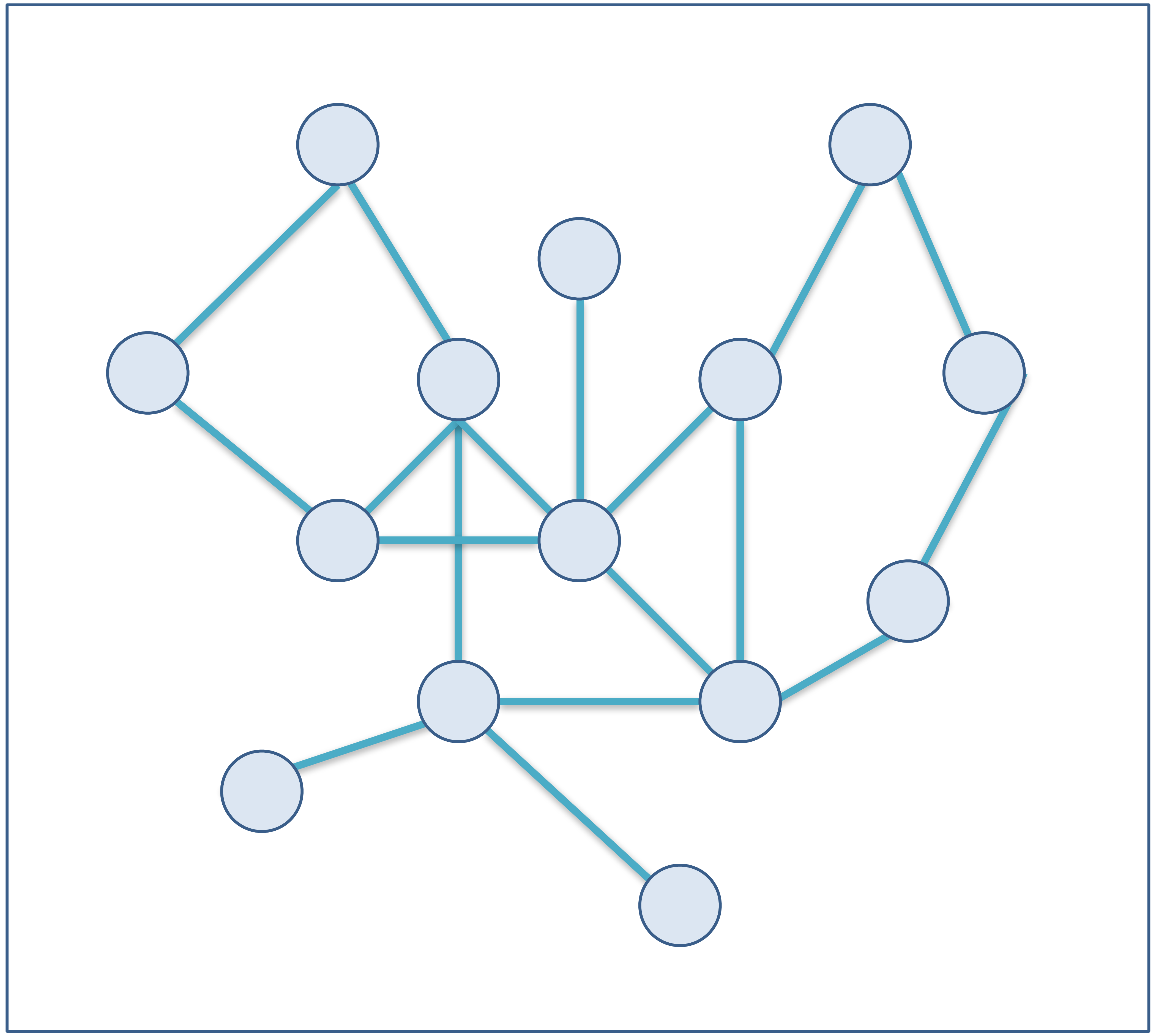
Bipartite Graph



Graph Terms (1)

A graph **G(V,E)** consists of a set of **vertices V** (also called nodes) and a

set of **edges E** connecting these vertices.

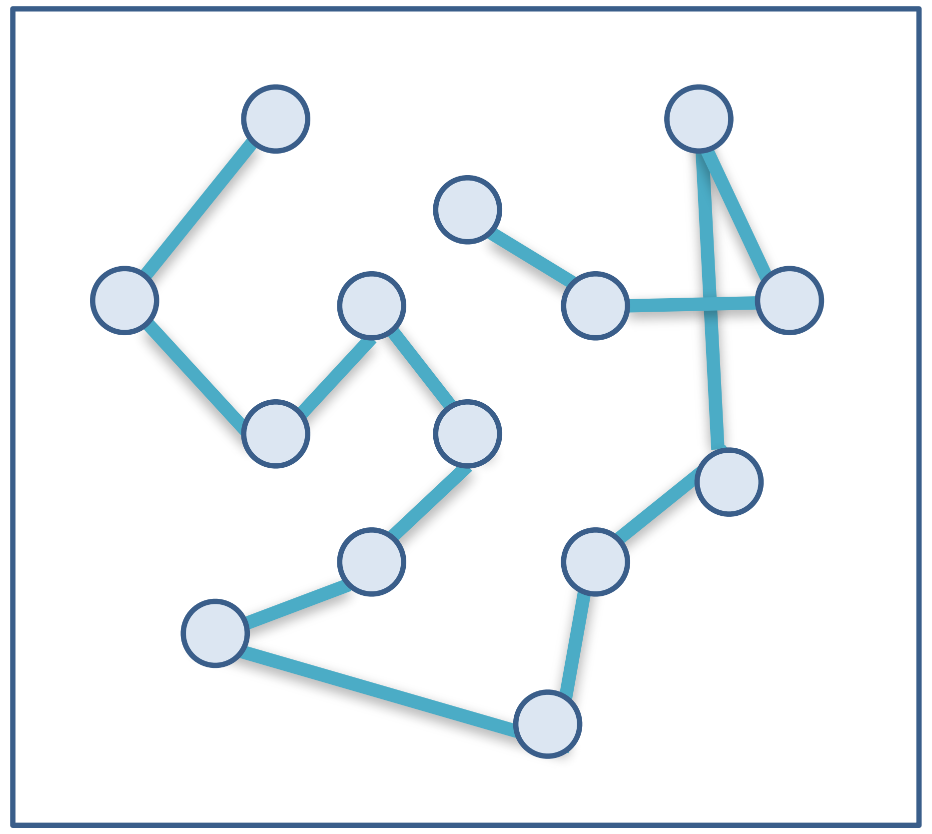


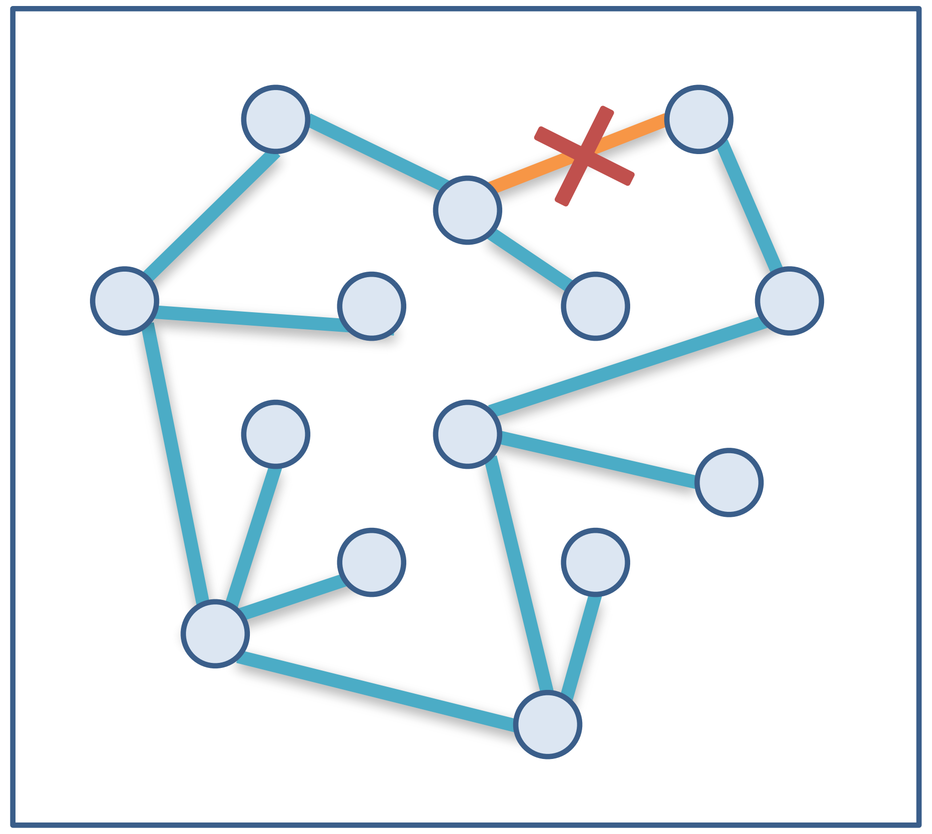
Graph Terms (5)

Path
G contains only edges that can be consecutively traversed

Tree
G contains no cycles

Network
G contains cycles


Path

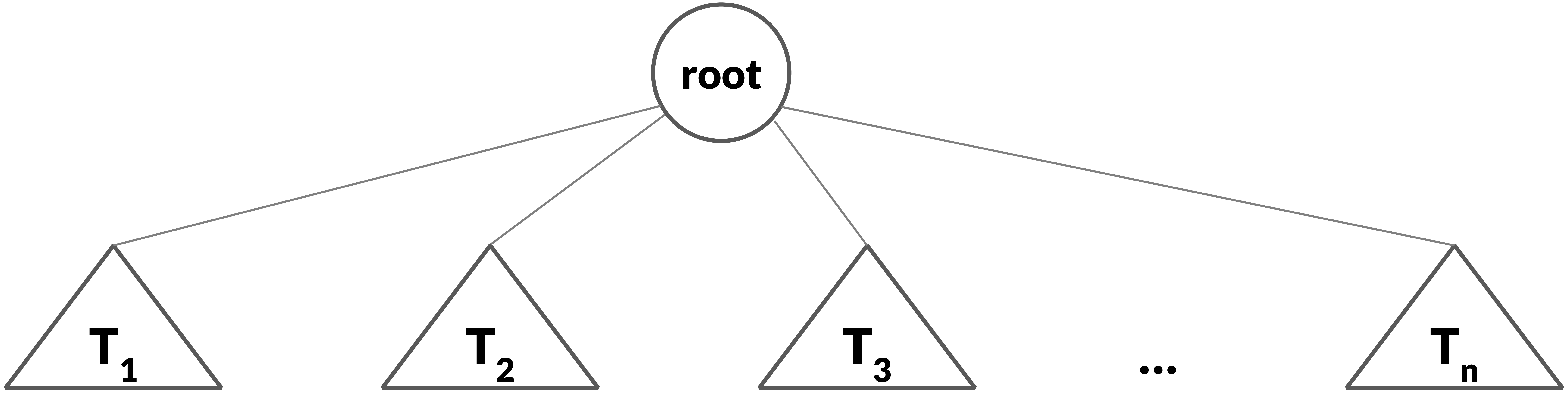

Tree

Tree

A graph with no cycles - or:

A collection of nodes

- **contains a root node and 0-n subtrees**
- **subtrees are connected to root by an edge**

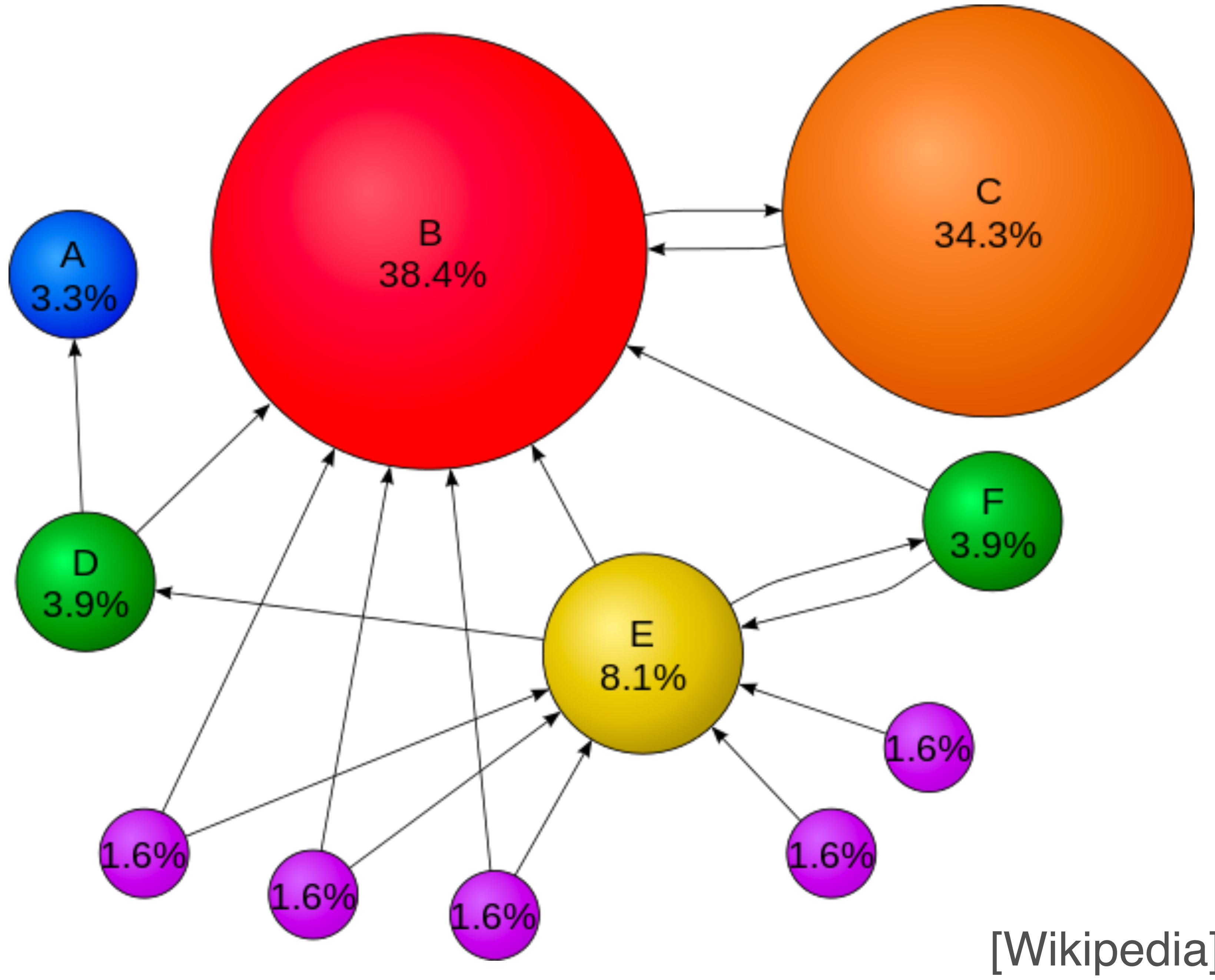


Graph Measures

Node degree deg(x)
The number of edges being incident to this node. For directed graphs indeg/outdeg are considered separately.

Diameter of graph G
The longest shortest path within G.

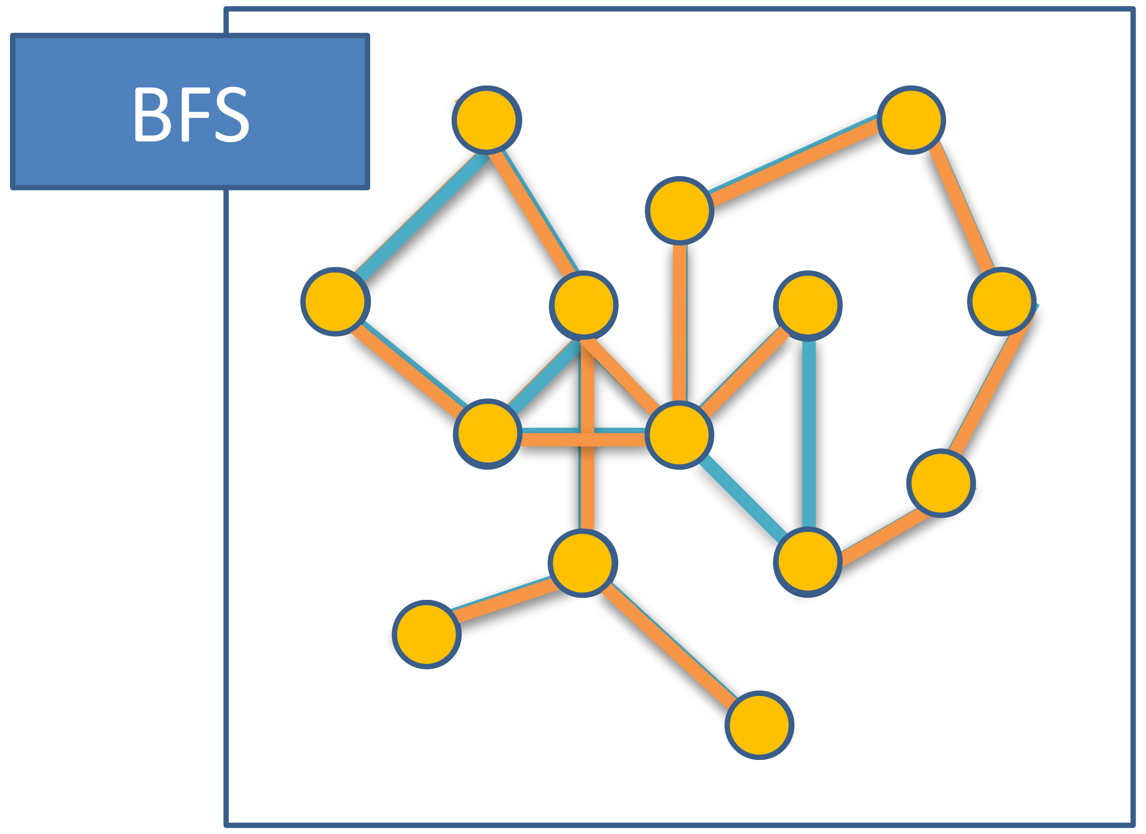
Pagerank
count number & quality of links

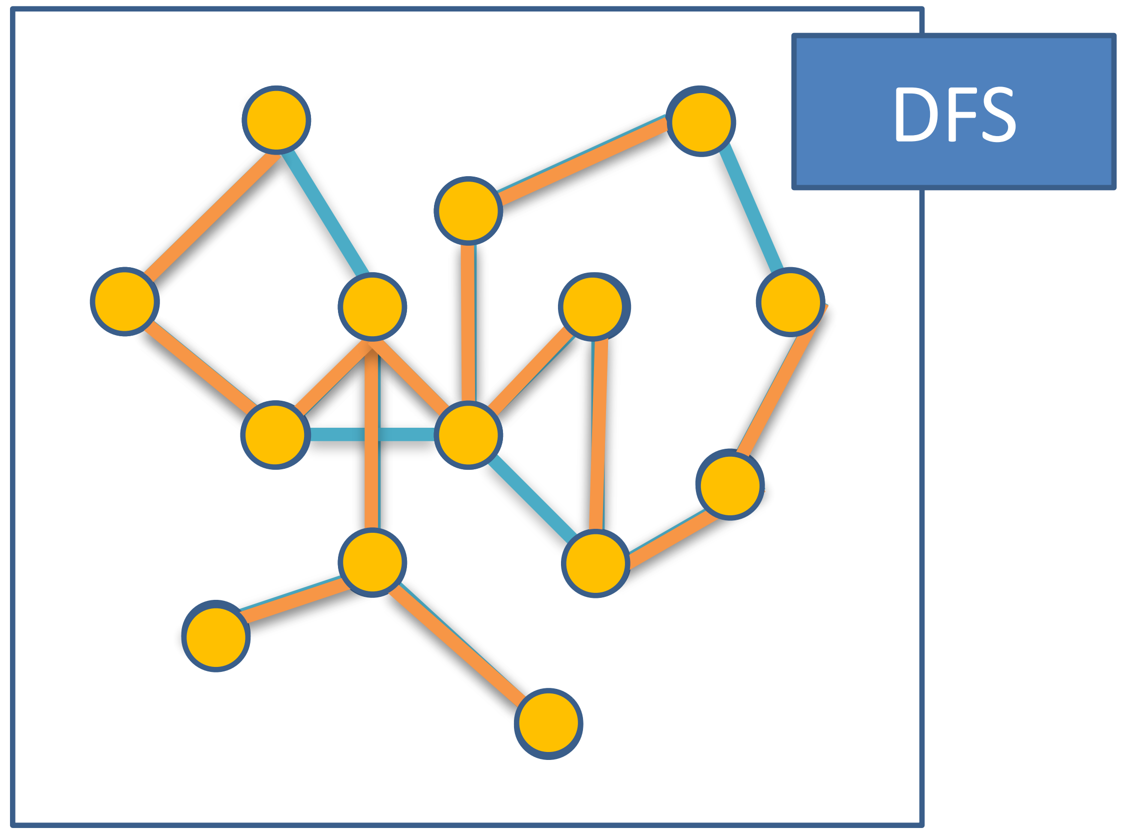


[Wikipedia]

Graph Algorithms (1)

Traversal: Breadth First Search, Depth First Search





- generates neighborhoods
- hierarchy gets rather wide than deep
- solves single-source shortest paths (SSSP)

- classical way-finding/back-tracking strategy
- tree serialization
- topological ordering

Graph and Tree Visualization

Different Kinds of Tasks/Goals

Two principal types of tasks: **attribute-based (ABT)** and **topology-based (TBT)**

Localize – find a single or multiple nodes/edges that fulfill a given property

- ABT: Find the edge(s) with the maximum edge weight.
- TBT: Find all adjacent nodes of a given node.

Quantify – count or estimate a numerical property of the graph

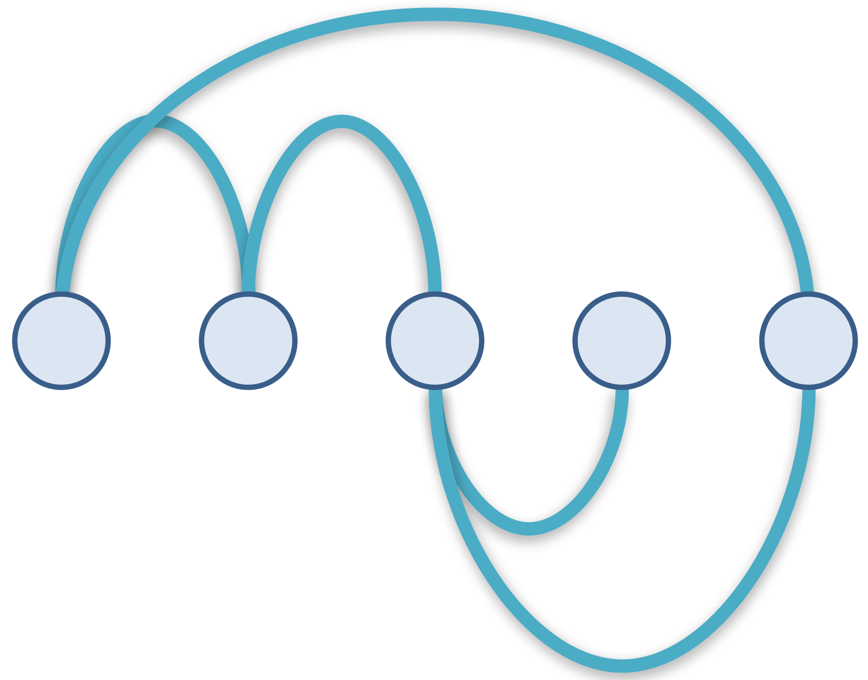
- ABT: Give the number of all nodes.
- TBT: Give the indegree (the number of incoming edges) of a node.

Sort/Order – enumerate the nodes/edges according to a given criterion

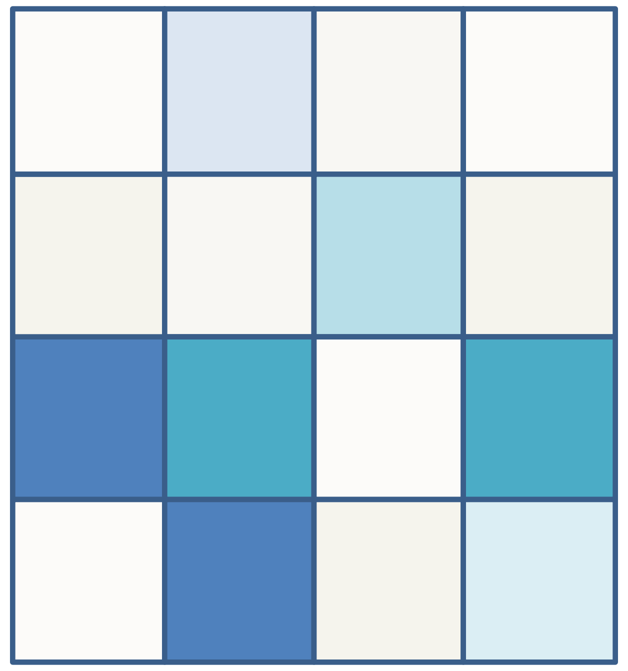
- ABT: Sort all edges according to their weight.
- TBT: Traverse the graph starting from a given node.

list adapted from Schulz 2010

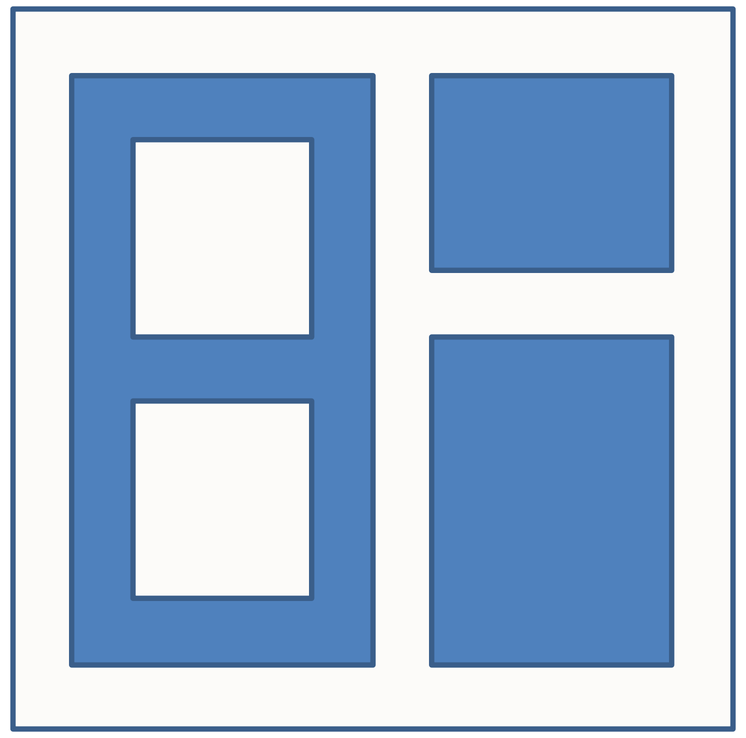
Three Types of Graph Representations



Explicit
(Node-Link)



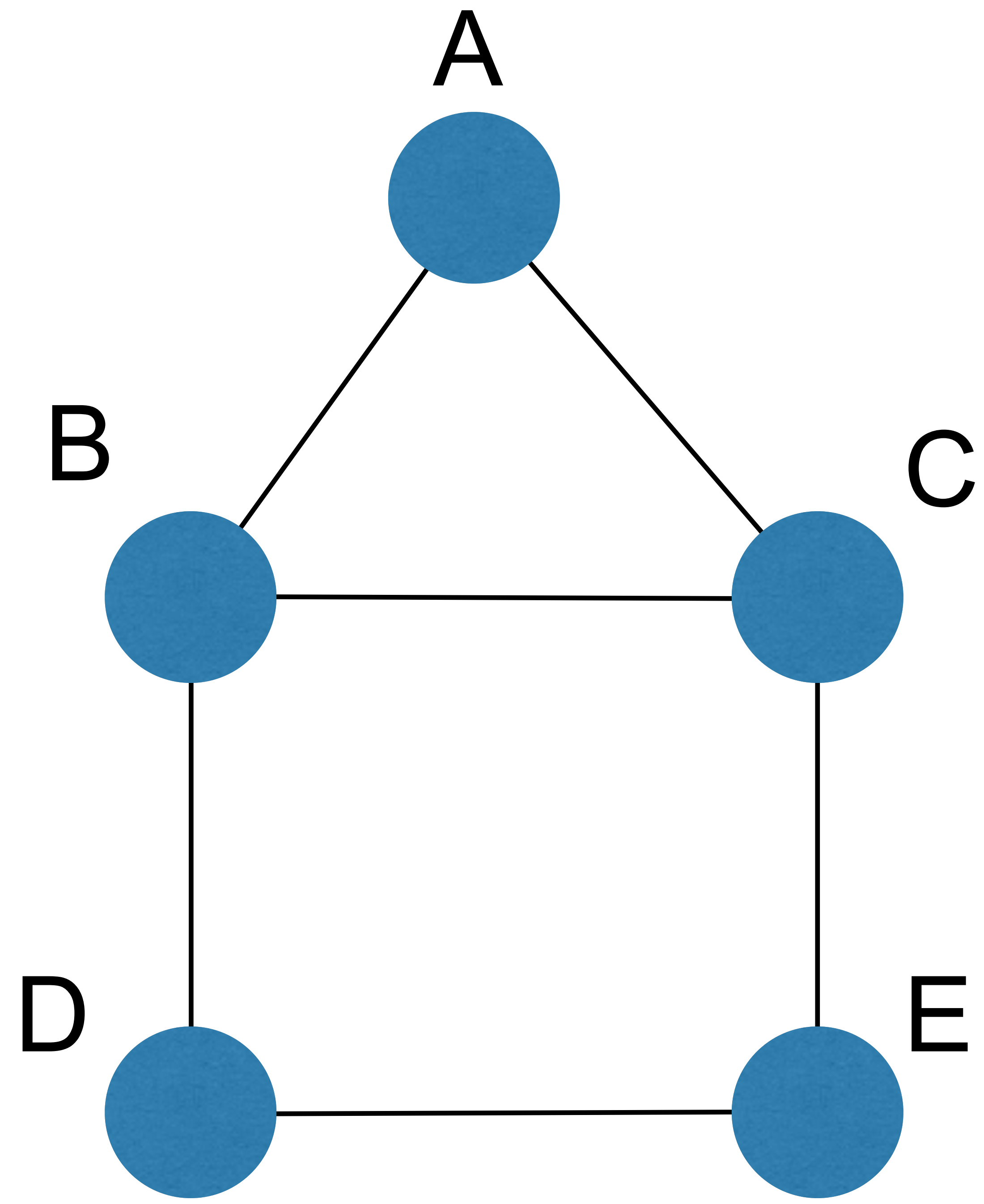
Matrix

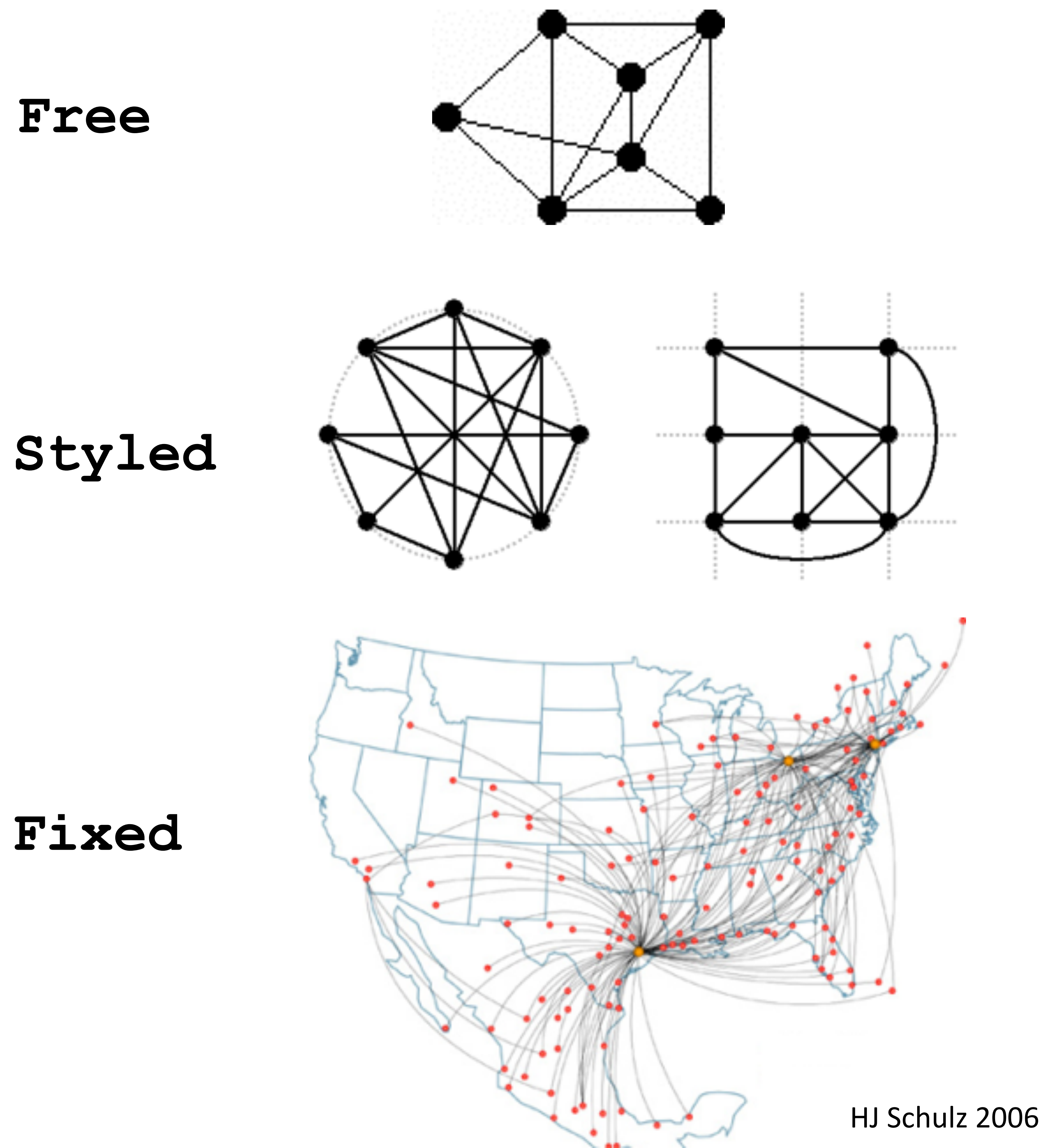


Implicit

Explicit Graph Representations

Node-link diagrams: vertex = point, edge = line/arc





HJ Schulz 2006

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

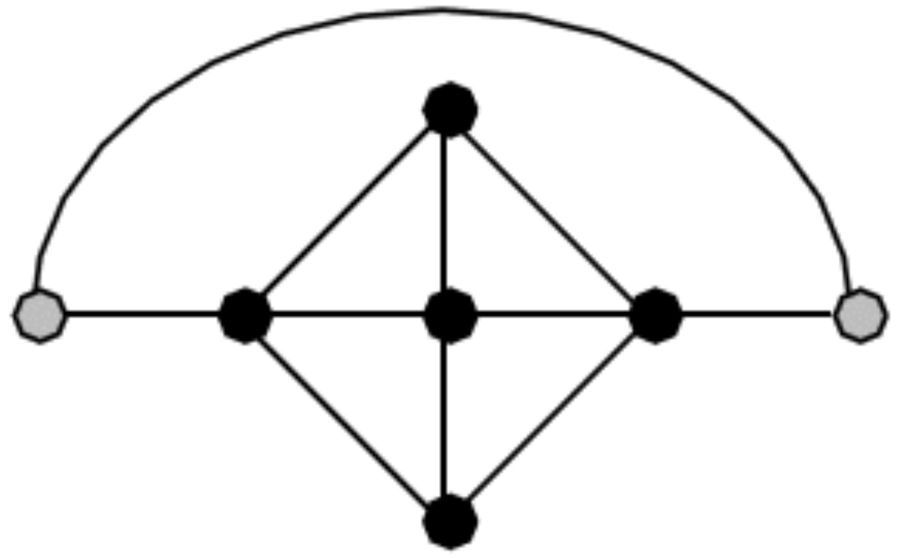
Aspect ratio about 1 (not too long and not too wide)

Symmetry: similar graph structures should look similar

list adapted from Battista et al. 1999

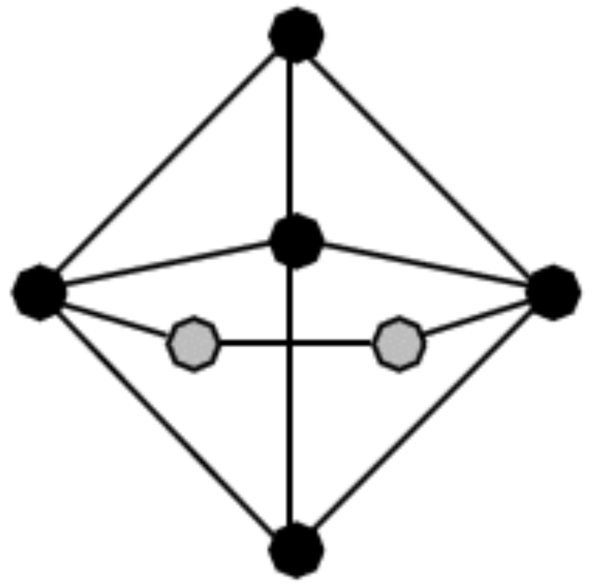
Conflicting Criteria

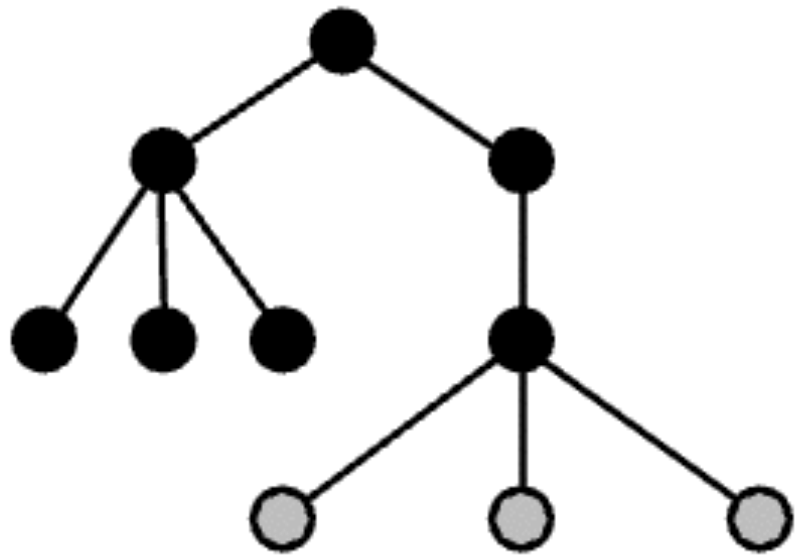
Minimum number
of edge crossings



vs.

Uniform edge
length

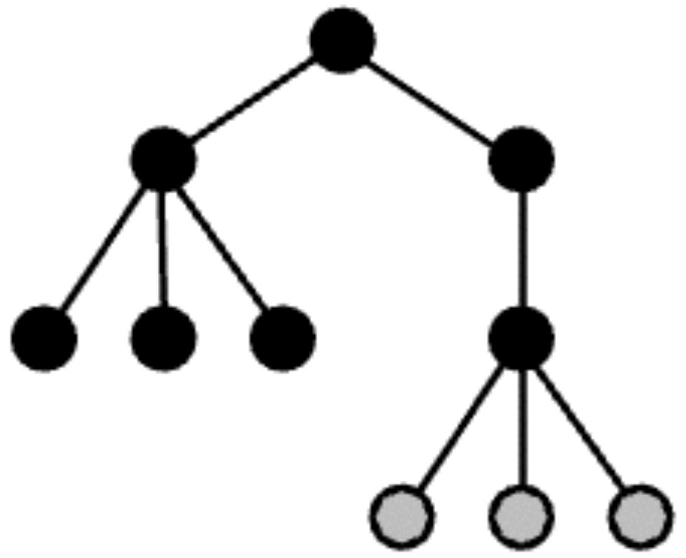




Space utilization

vs.

Symmetry



Schulz 2004

Explicit Representations

Pros:

is able to depict all graph classes

can be customized by weighing the layout constraints

very well suited for TBTs, if also a suitable layout is chosen
[McGrath et al. 1997], [Purchase et al. 2002], and [Huang et al. 2005]

Cons:

computation of an optimal graph layout is in NP
(even just achieving minimal edge crossings is already in NP)

even heuristics are still slow/complex (e.g., naïve spring embedder is in $O(n^2)$)

has a tendency to clutter (edge clutter, "hairball")

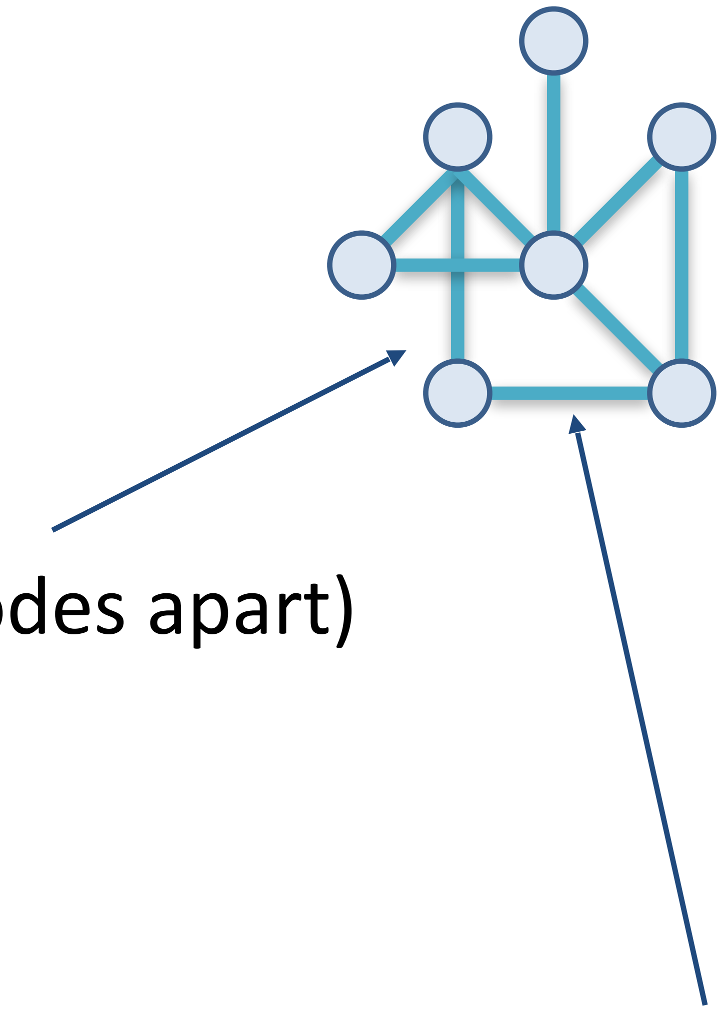
Force Directed Layouts

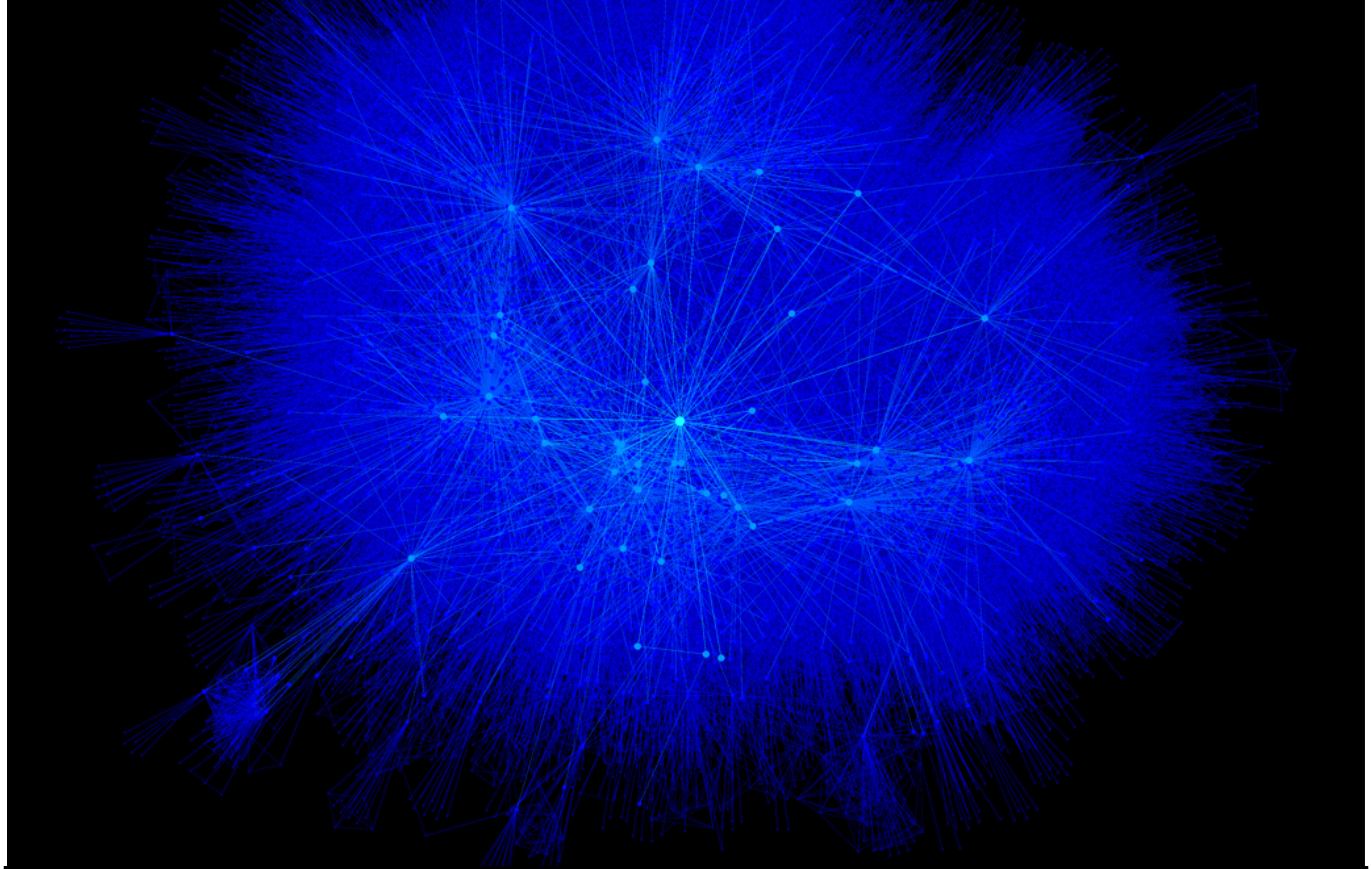
Physics model:
edges = springs,
vertices = repulsive magnets

in practice: damping

Computationally
expensive: $O(n^3)$

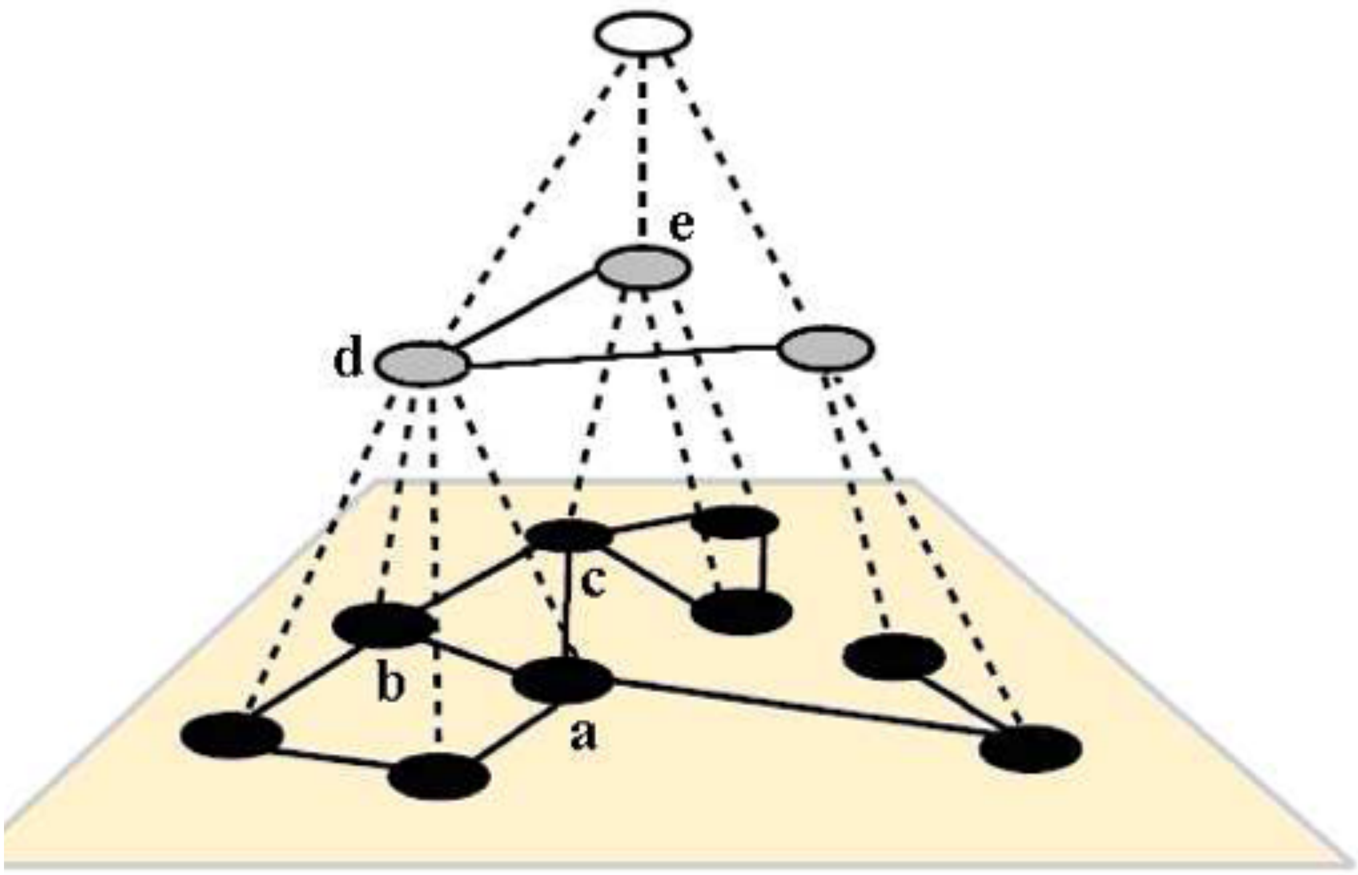
Limit (interactive): ~1000 nodes

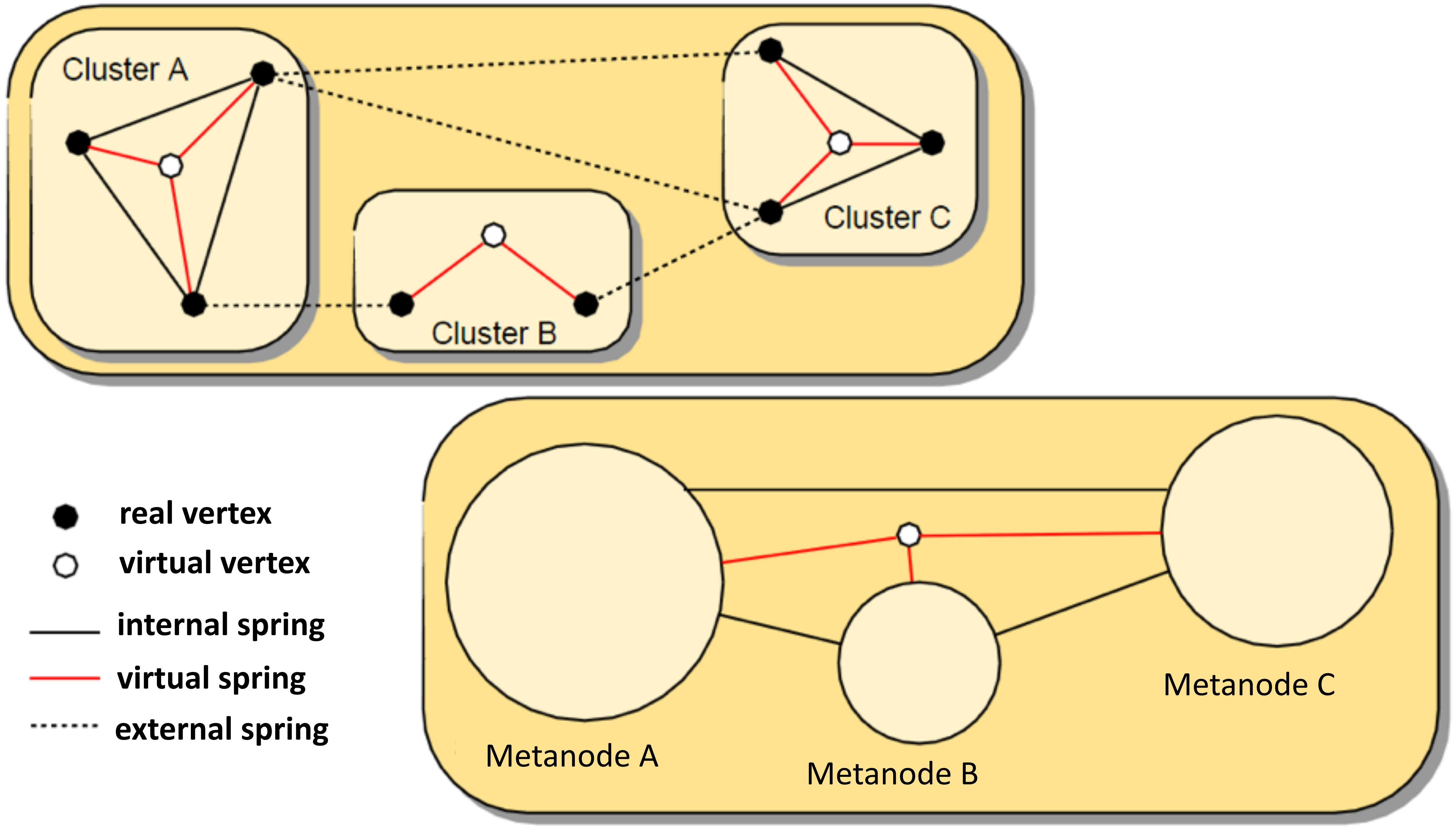




Giant Hairball

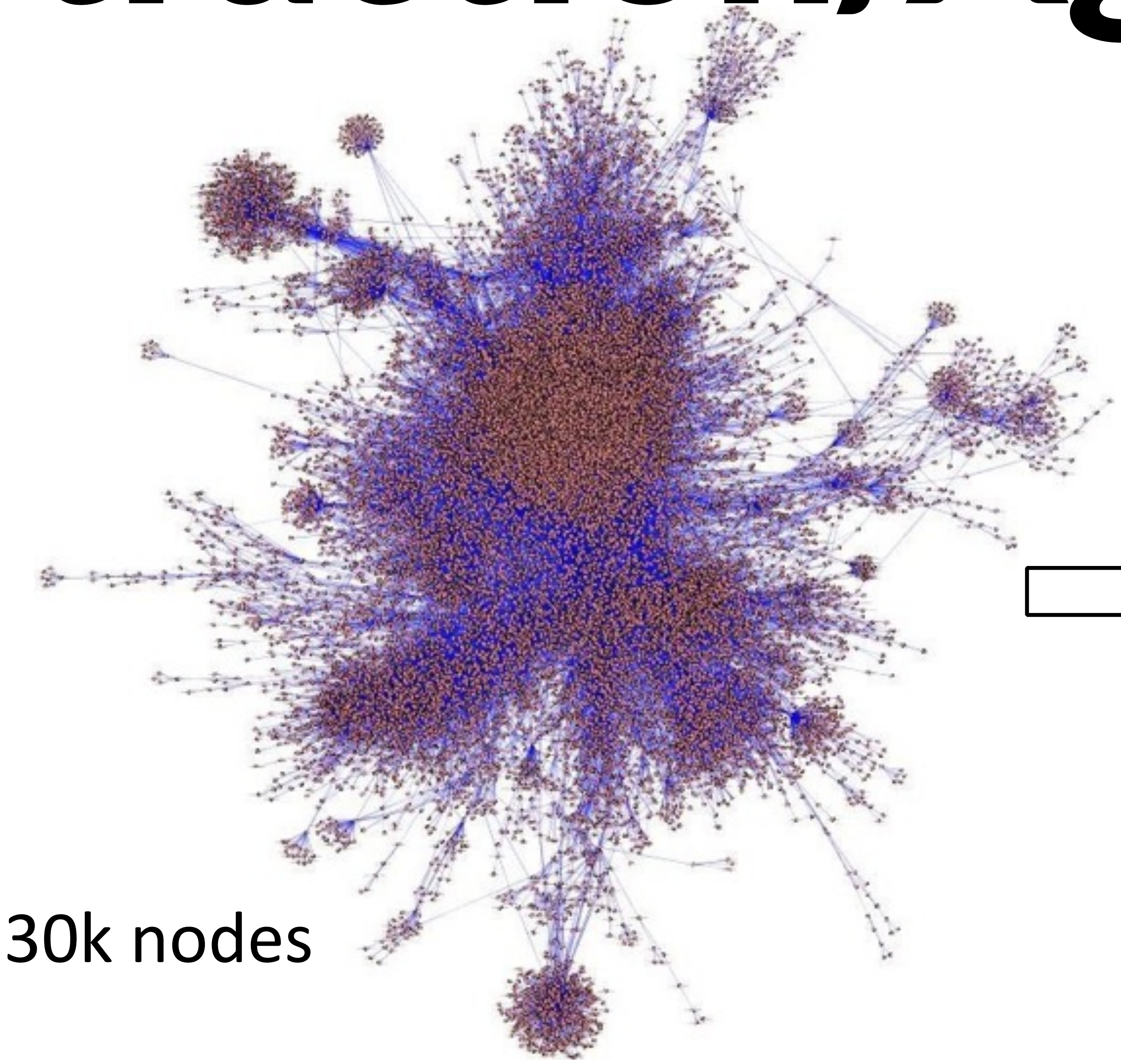
Adress Computational Scalability: Multilevel Approaches



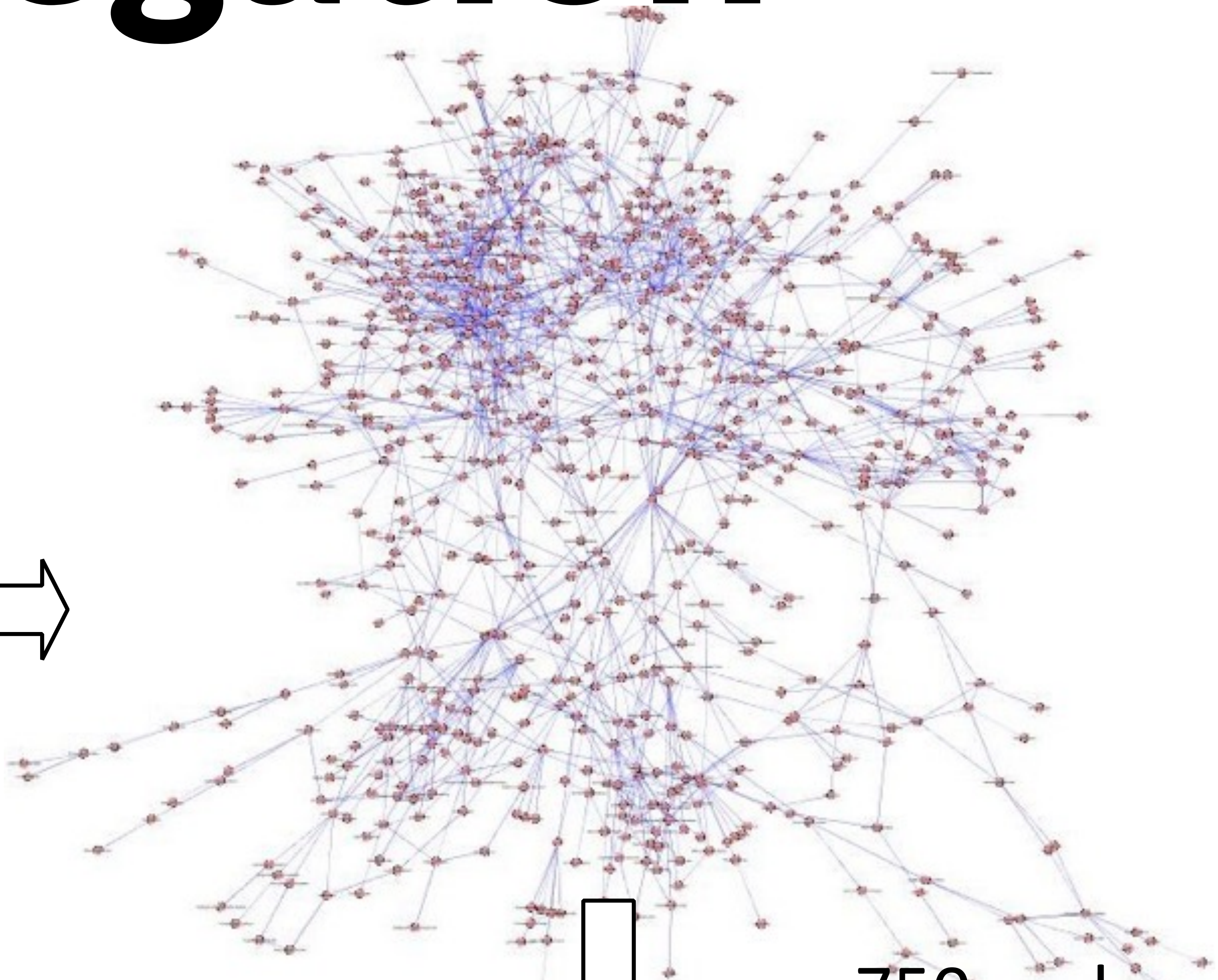


[Schulz 2004]

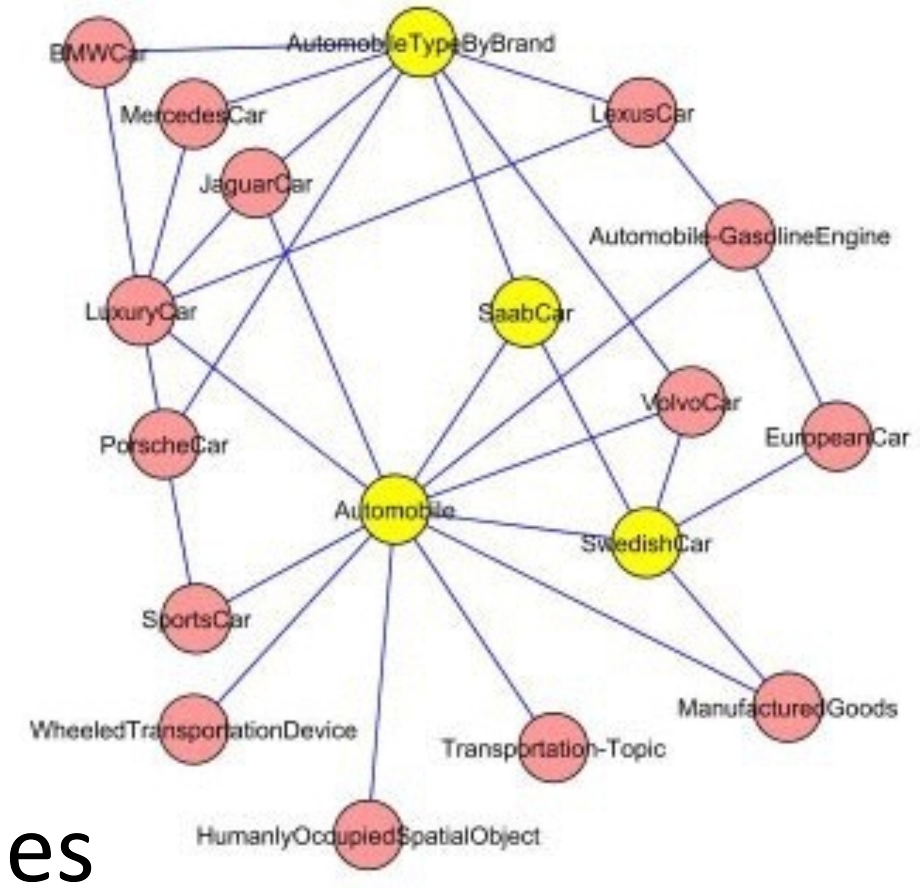
Abstraction/Aggregation



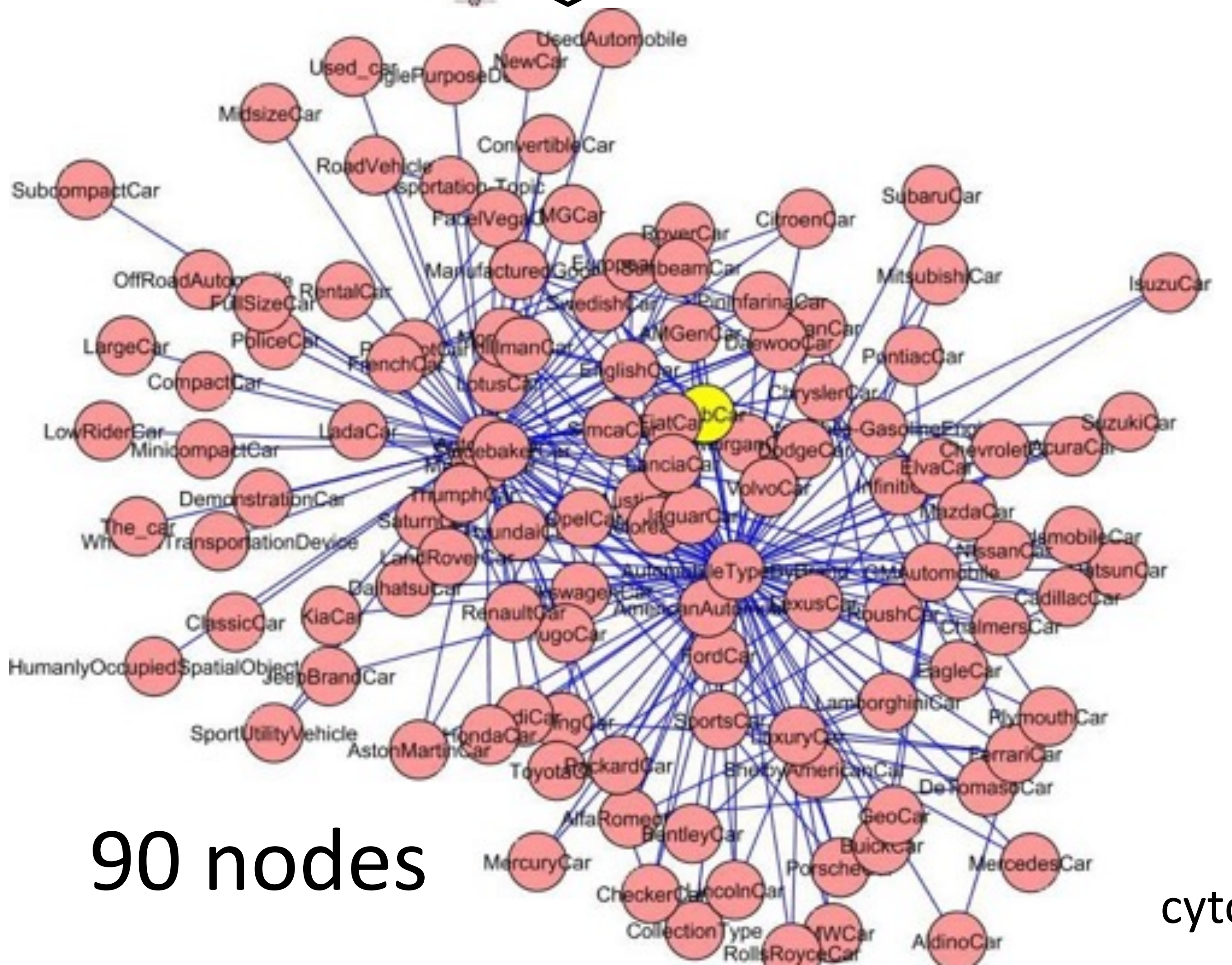
30k nodes



750 nodes



18 nodes



90 nodes

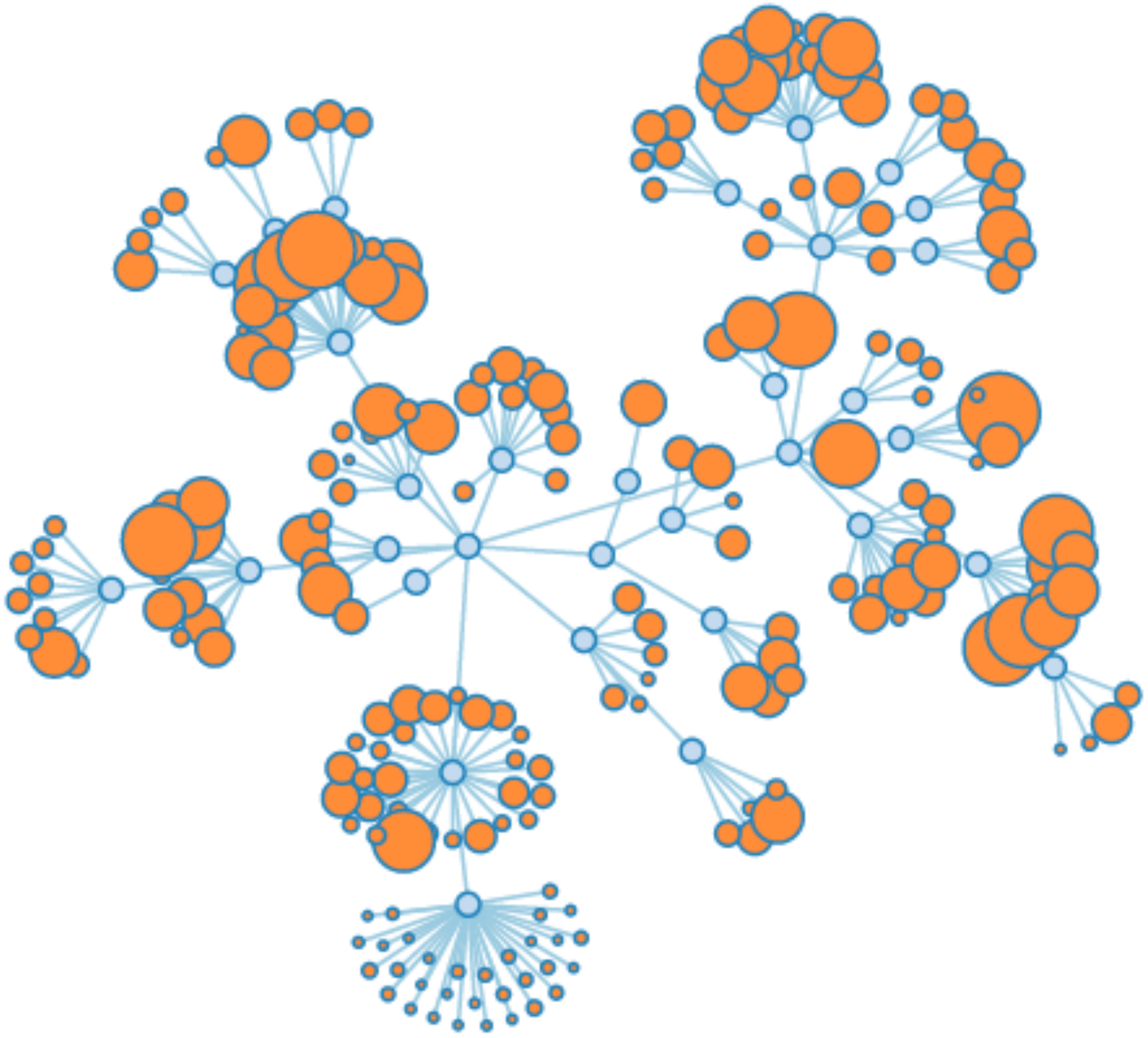
cytoscape.org

Collapsible Force Layout

Supernodes: aggregate of nodes

manual or algorithmic

- clustering

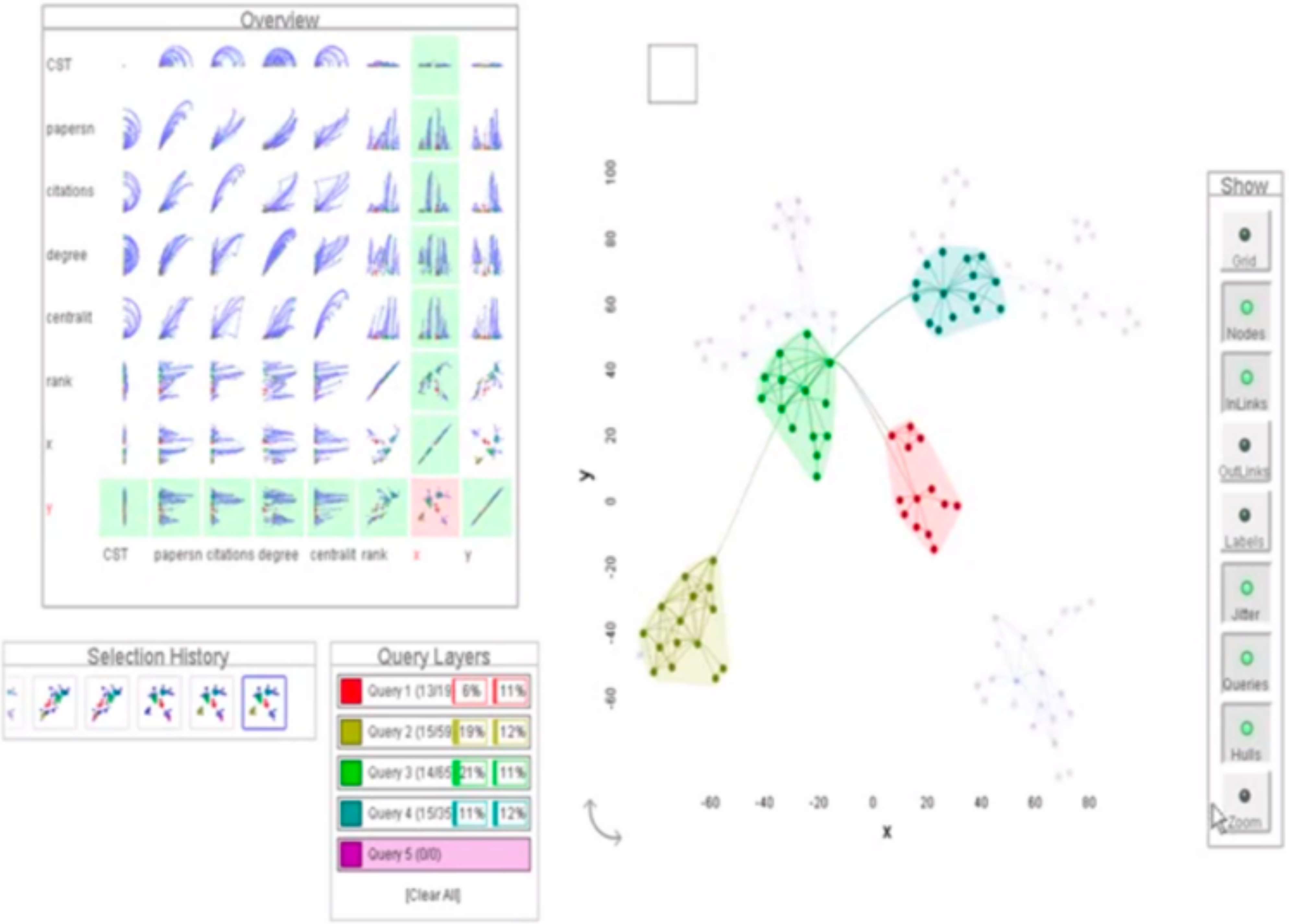


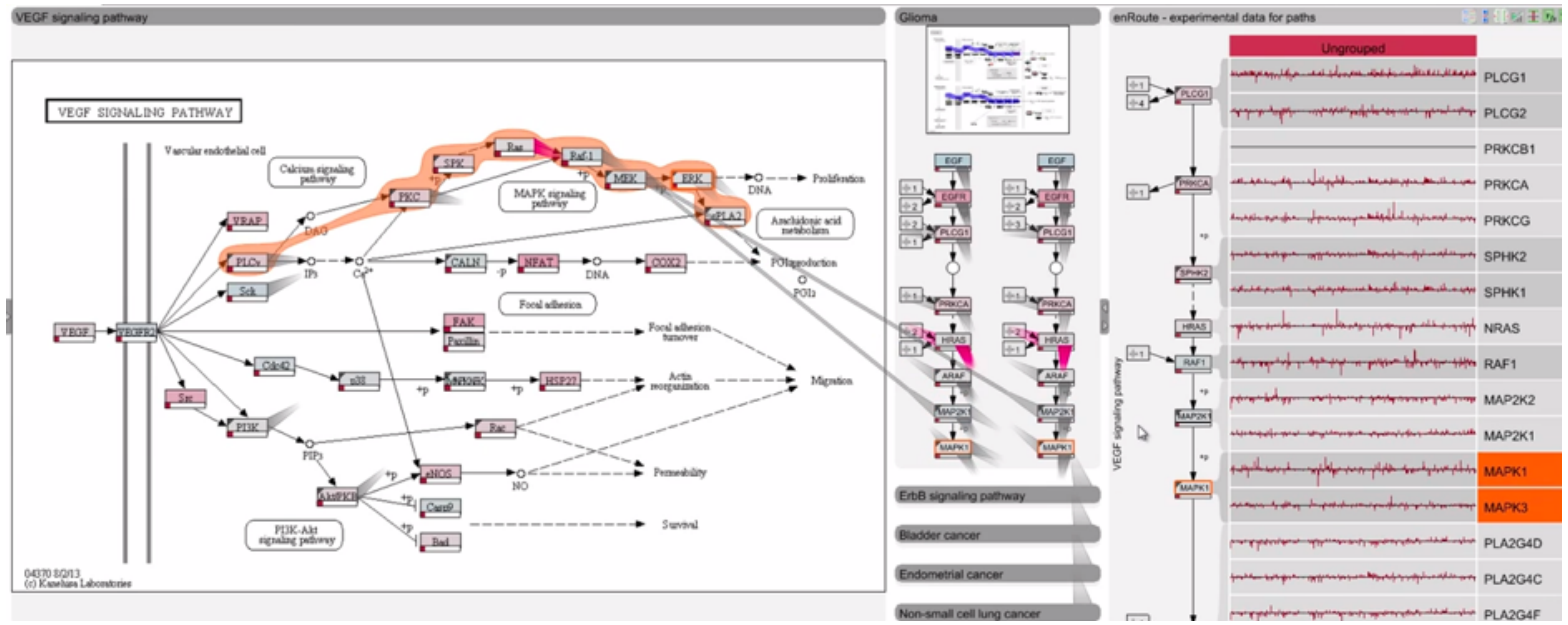
Node Attributes

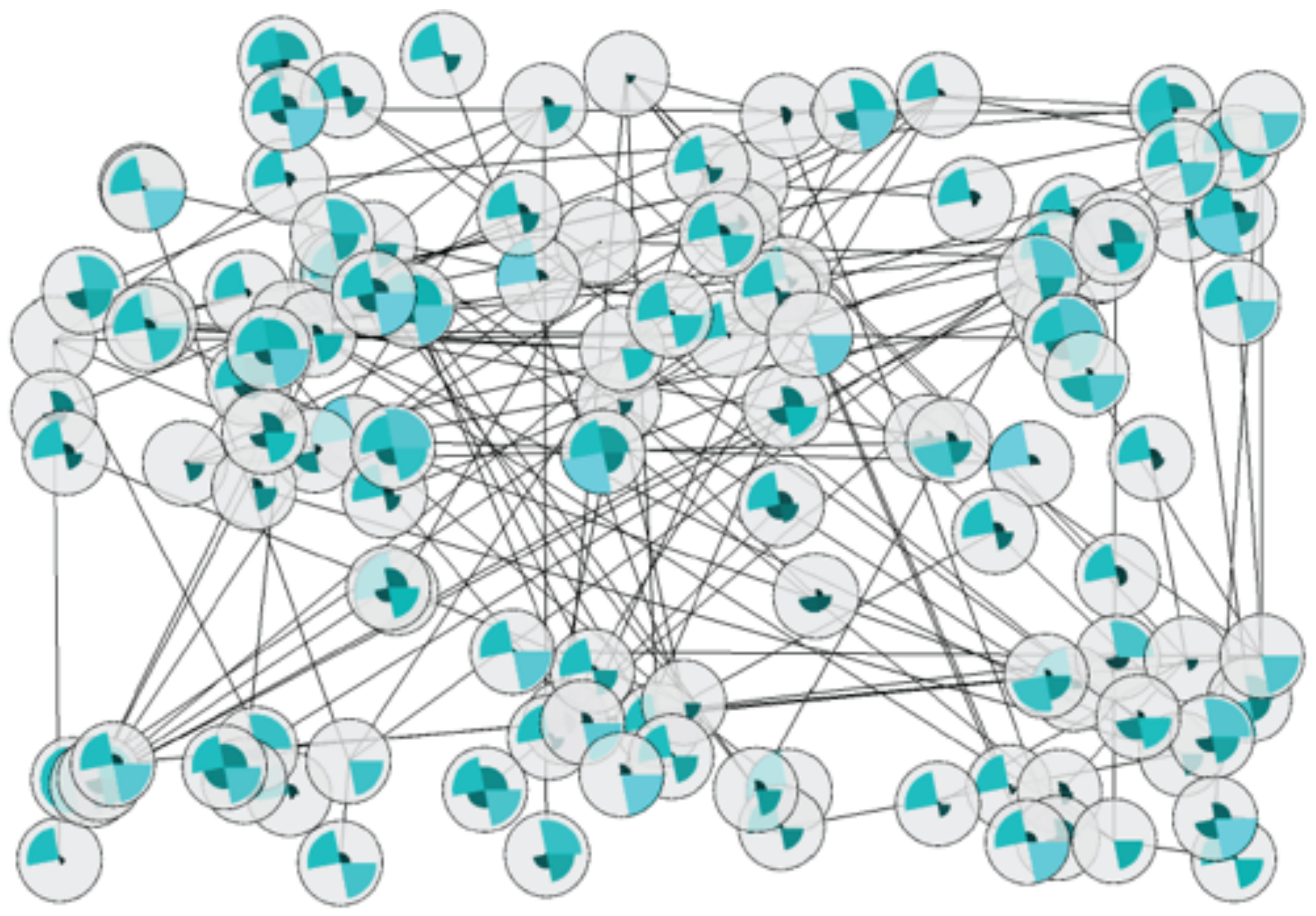
Coloring

Position

Multiple Views /
Path extraction





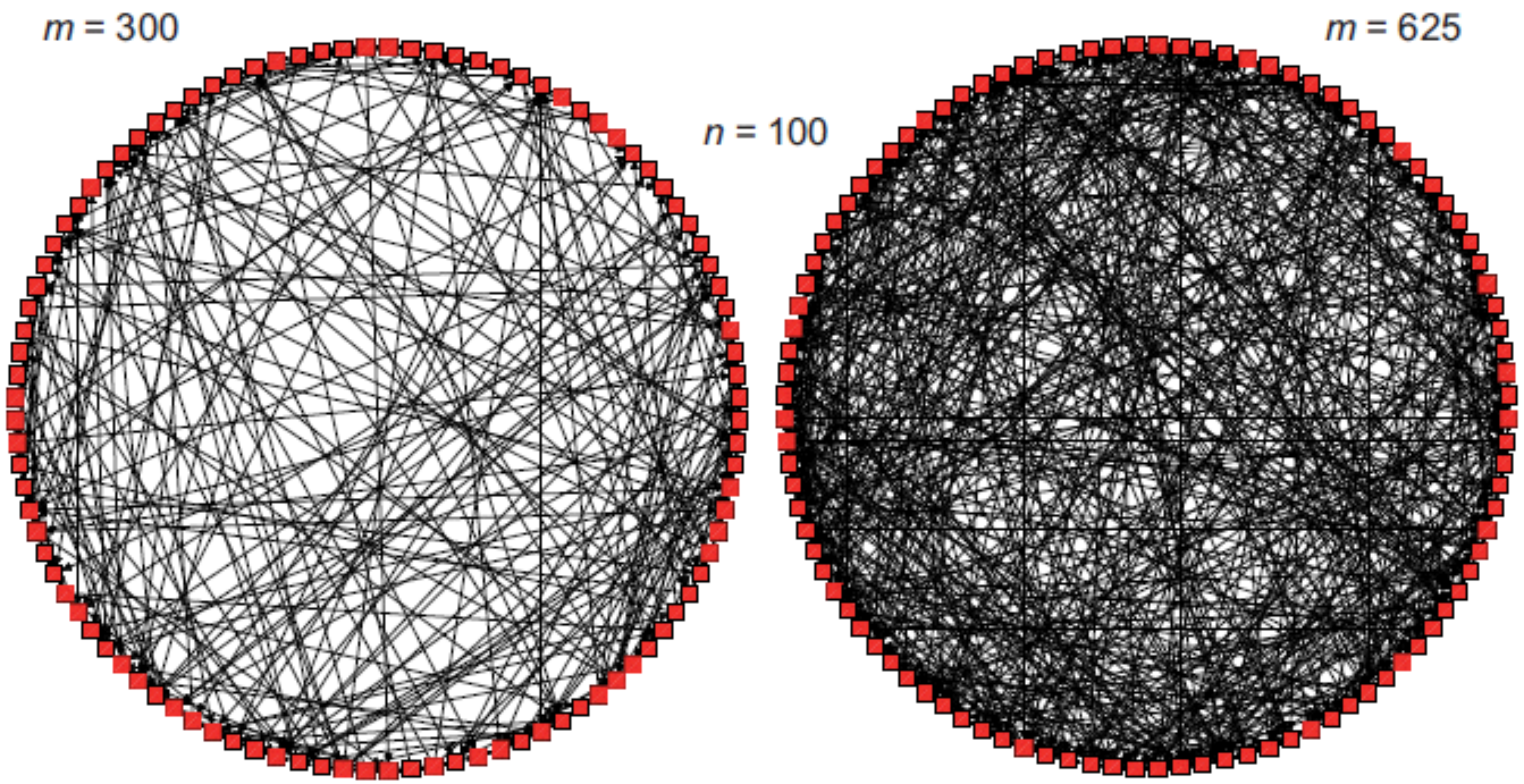


Styled / Restricted Layouts

Circular Layout

Node ordering

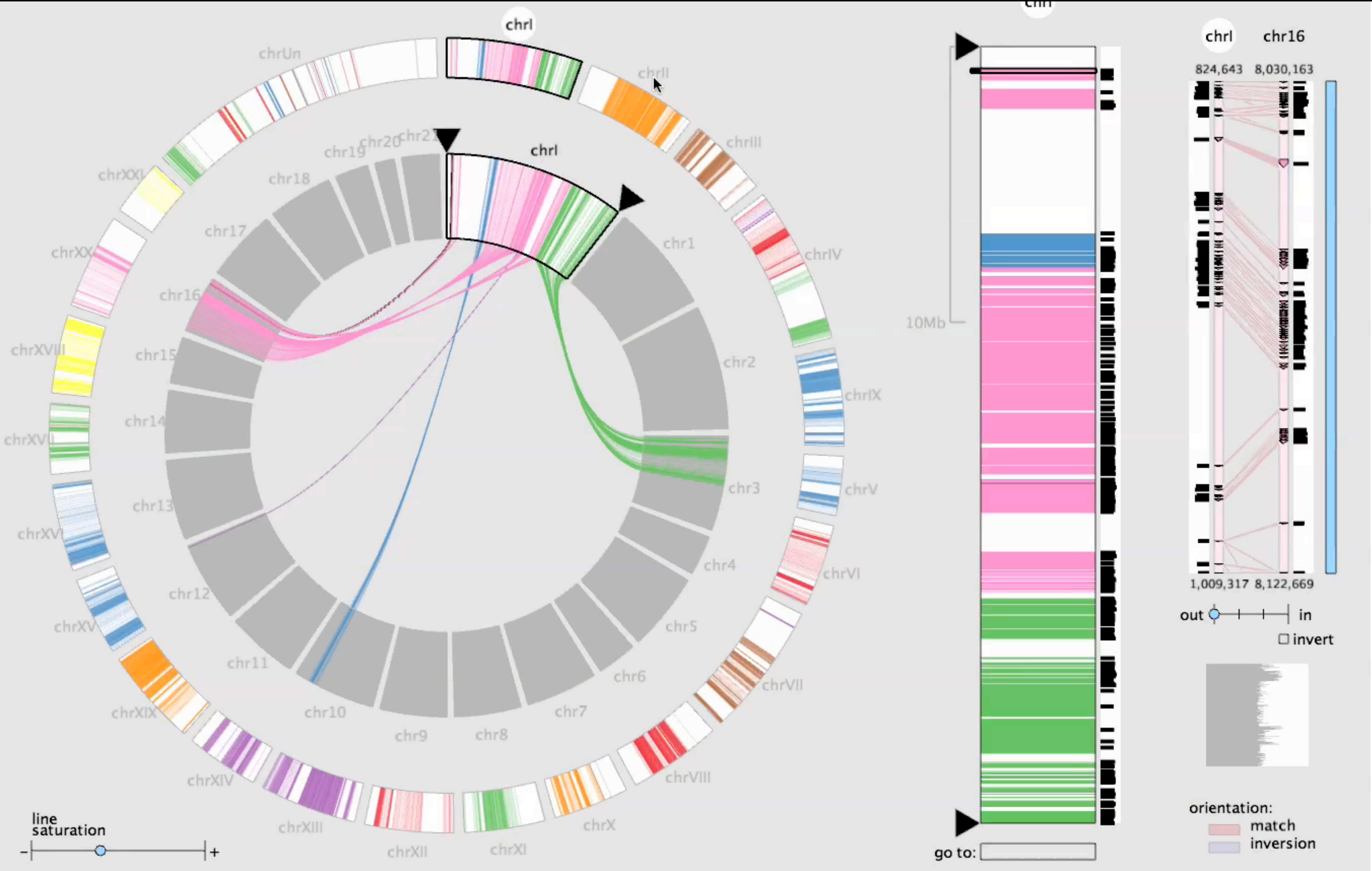
Edge Clutter



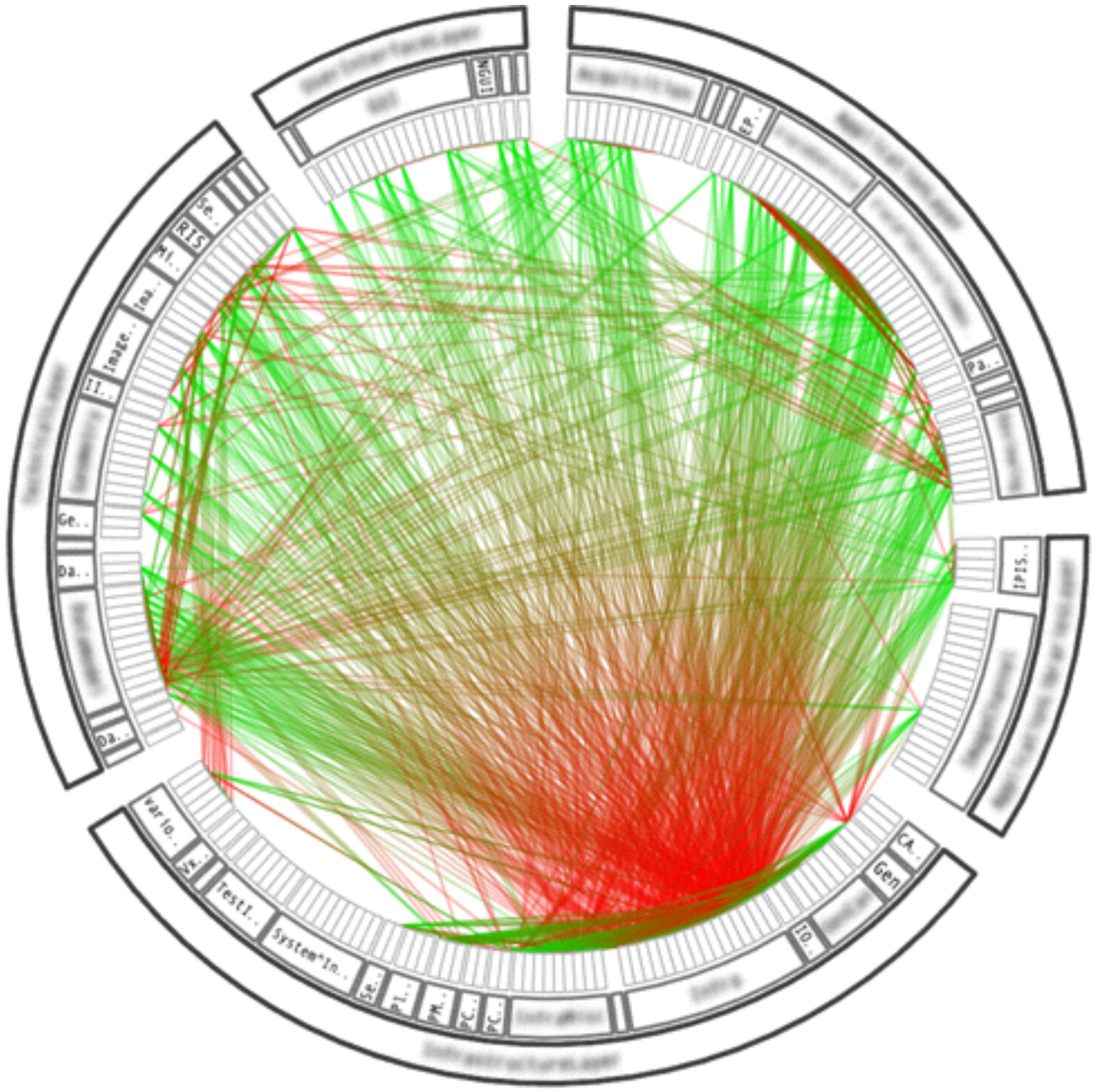
ca. 3% of all possible edges

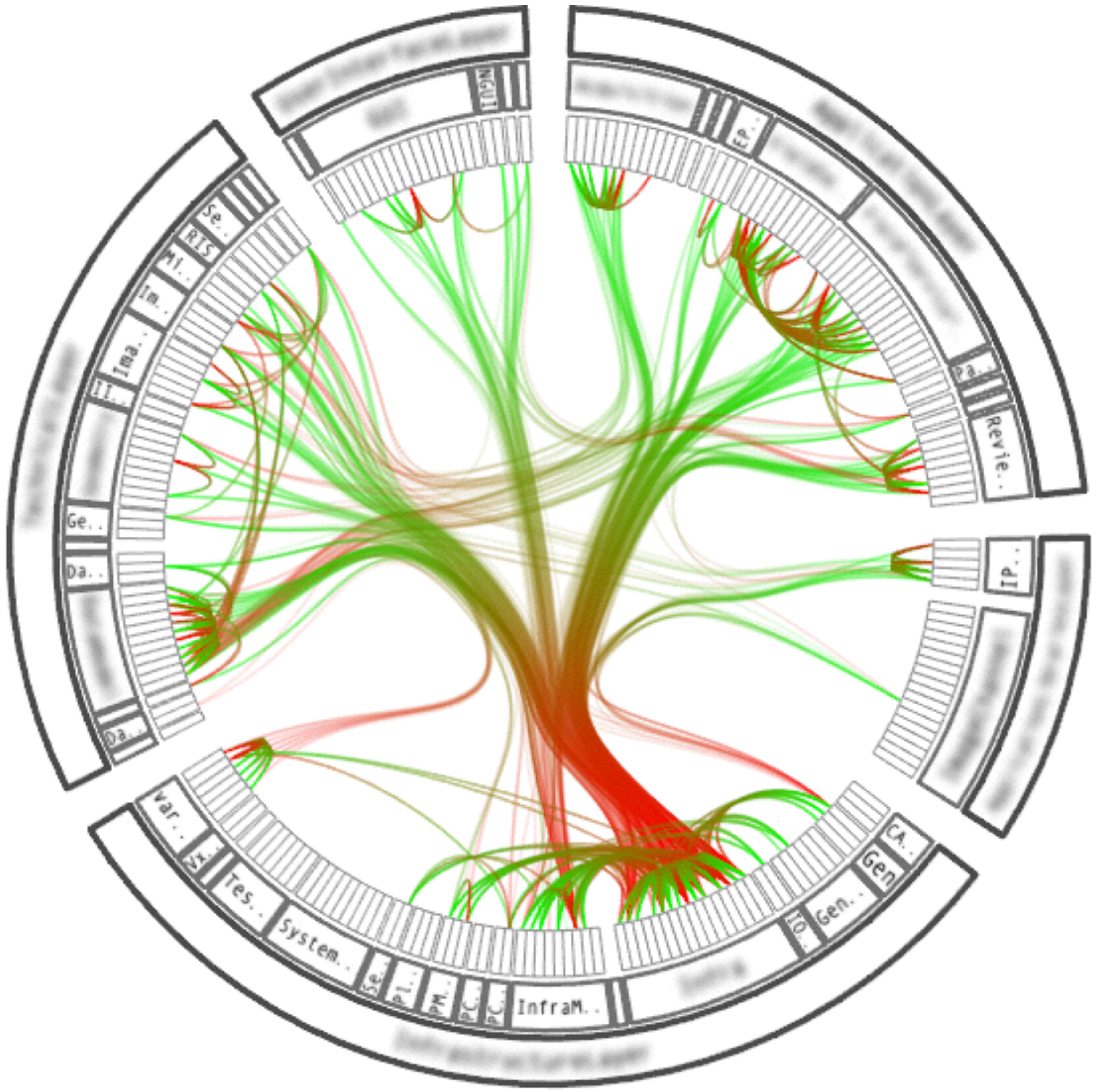
ca. 6,3% of all possible edges

Example: MizBee [Meyer et al. 2009]



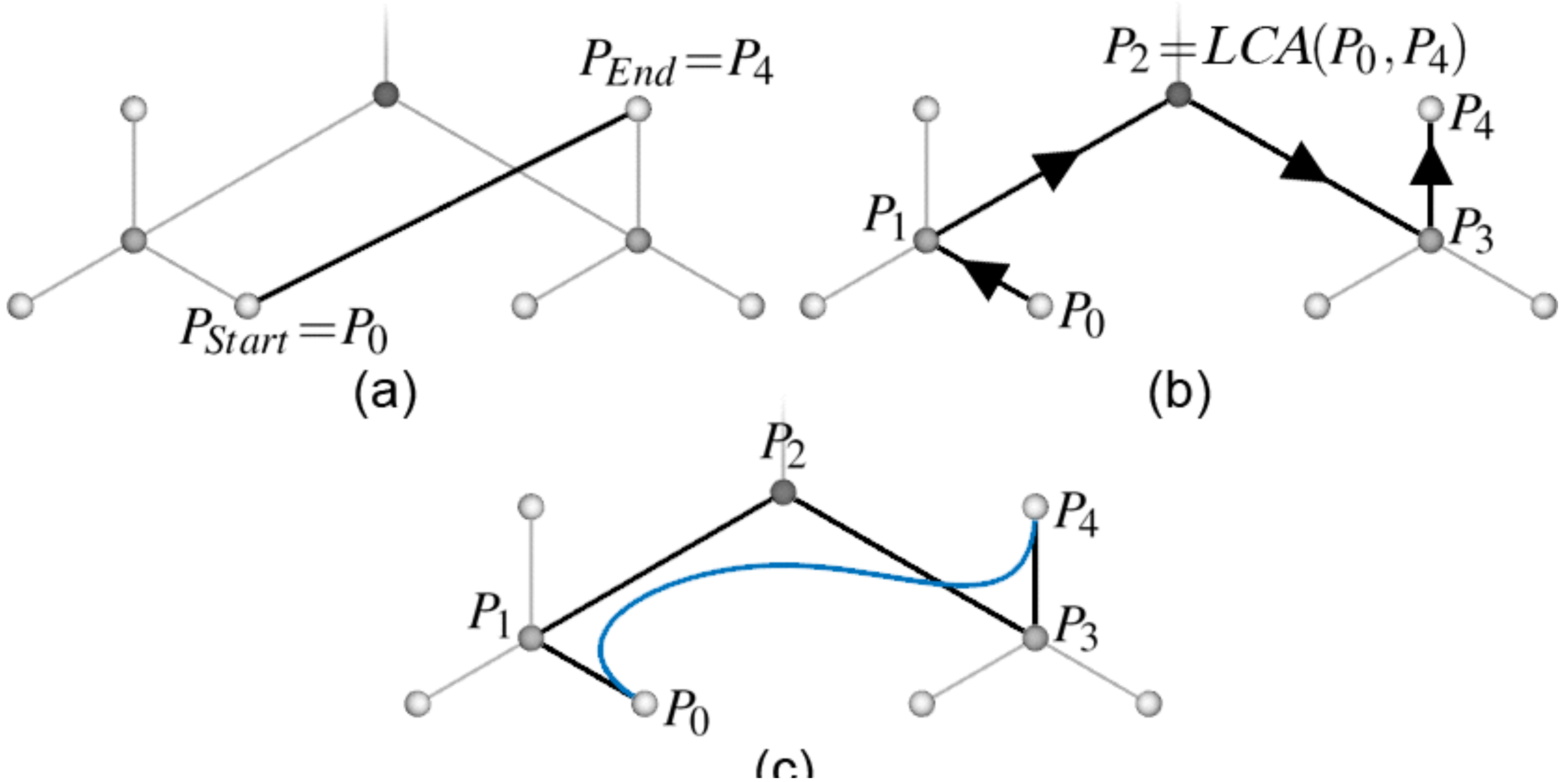
Reduce Clutter: Edge Bundling

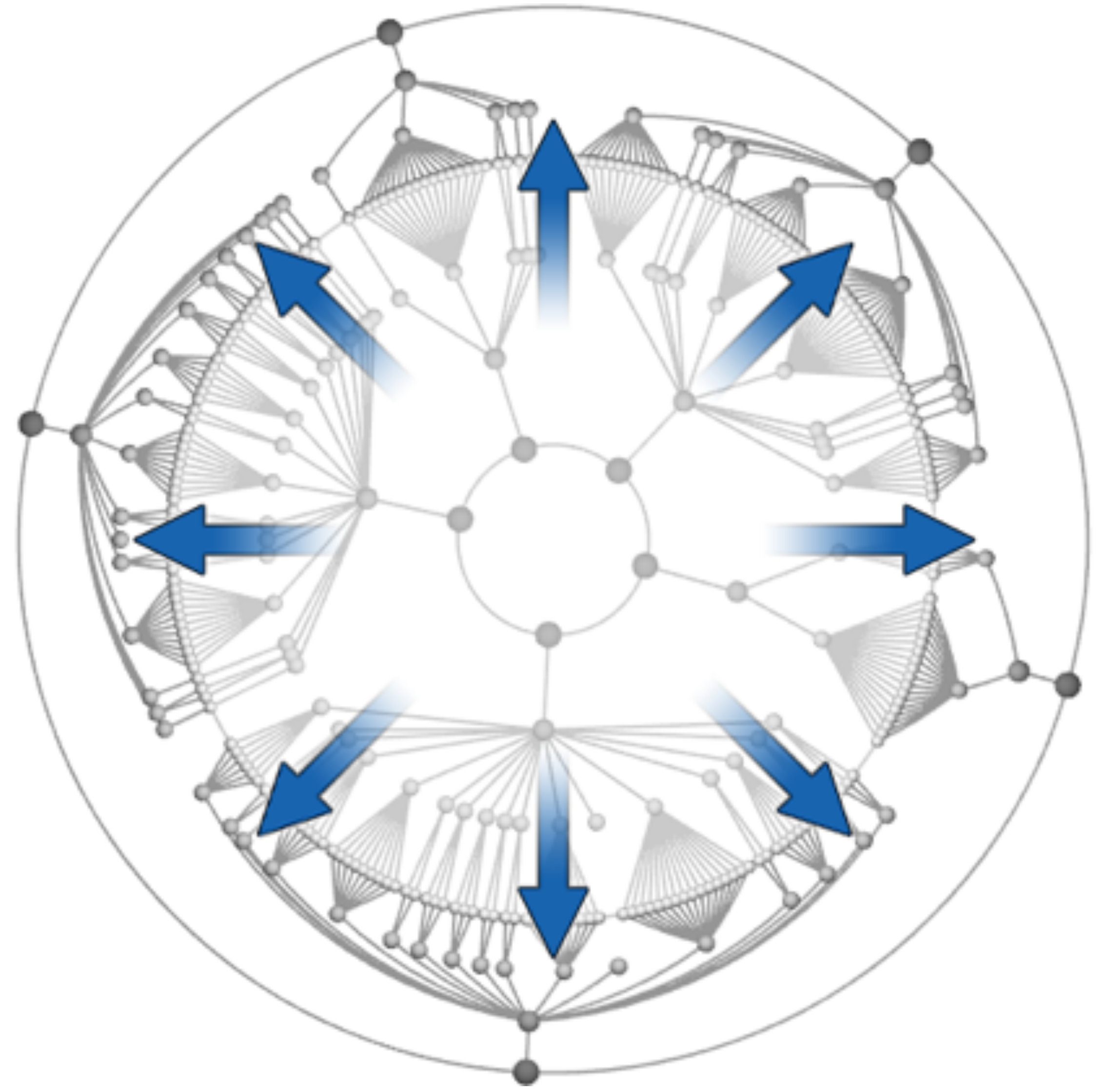


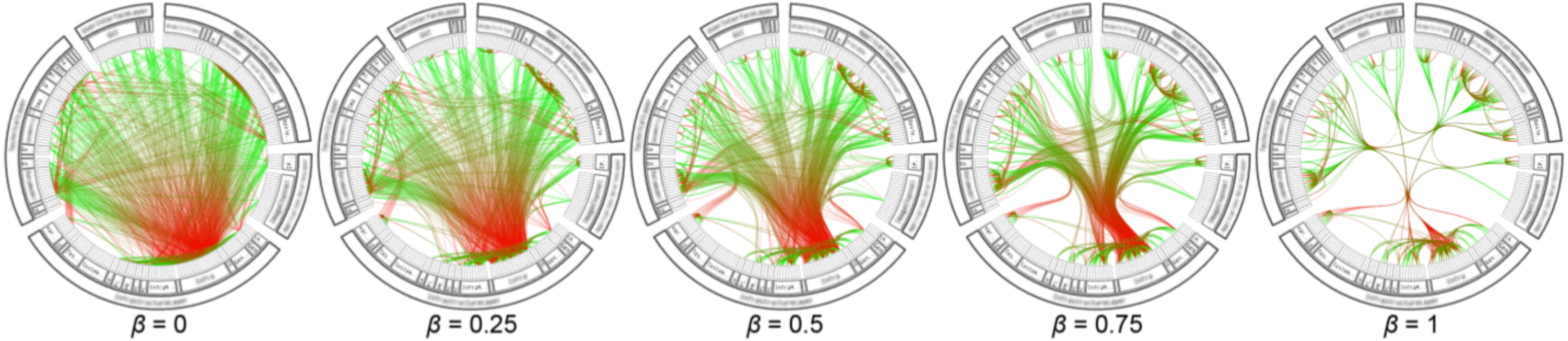


Holten et al. 2006

Hierarchical Edge Bundling







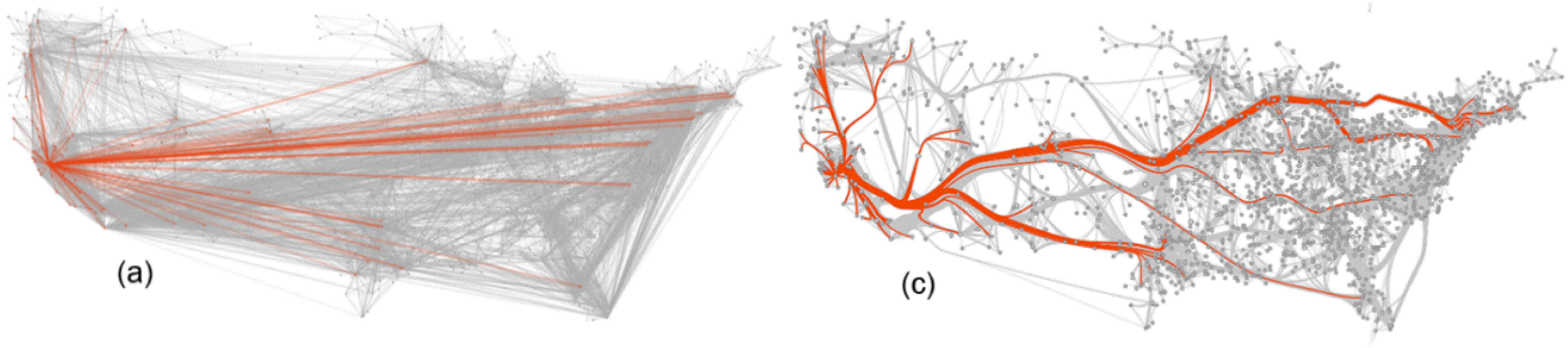
Bundling Strength

Holten et al. 2006

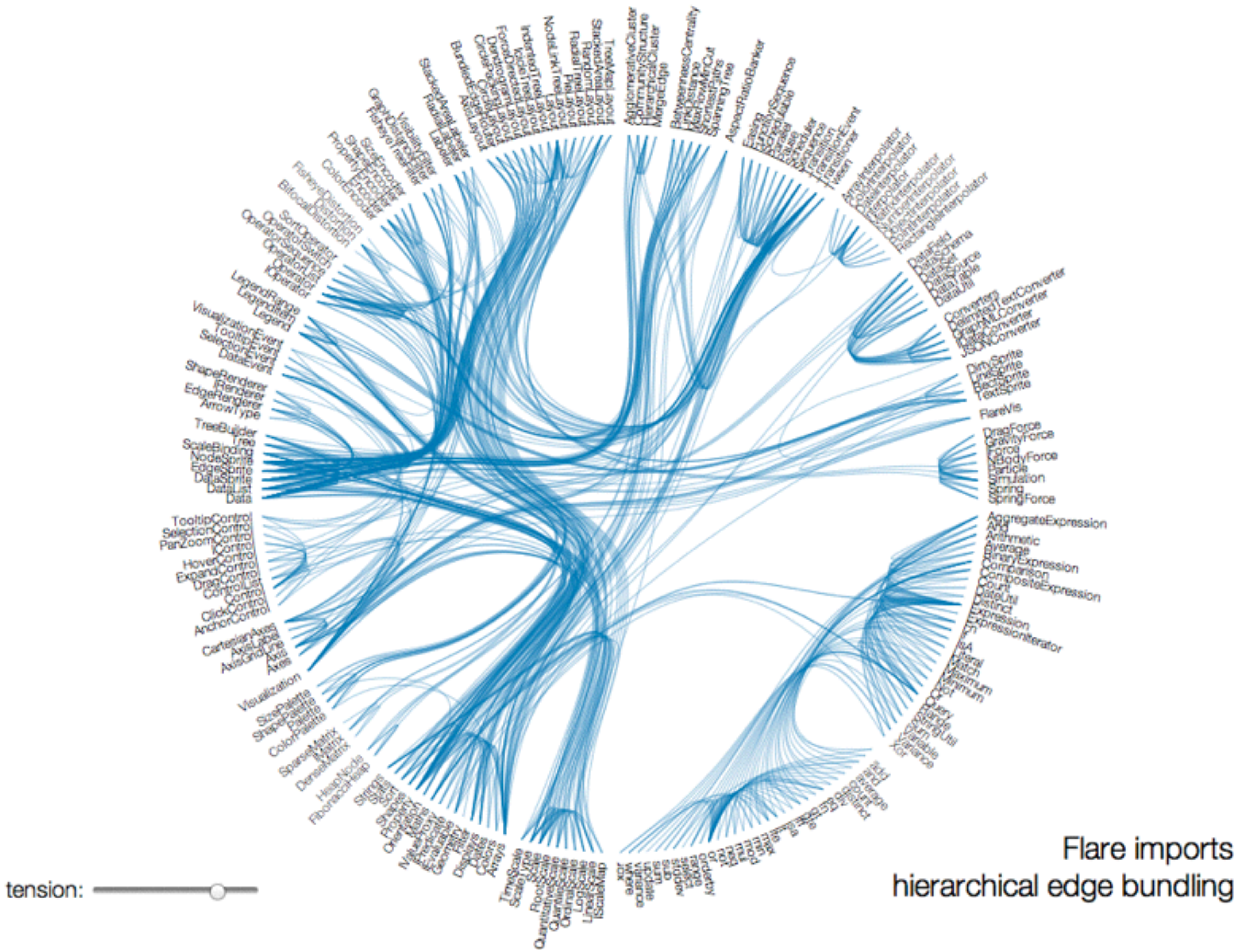
Fixed Layouts

Can’t vary position of nodes

Edge routing important



Bundling Strength



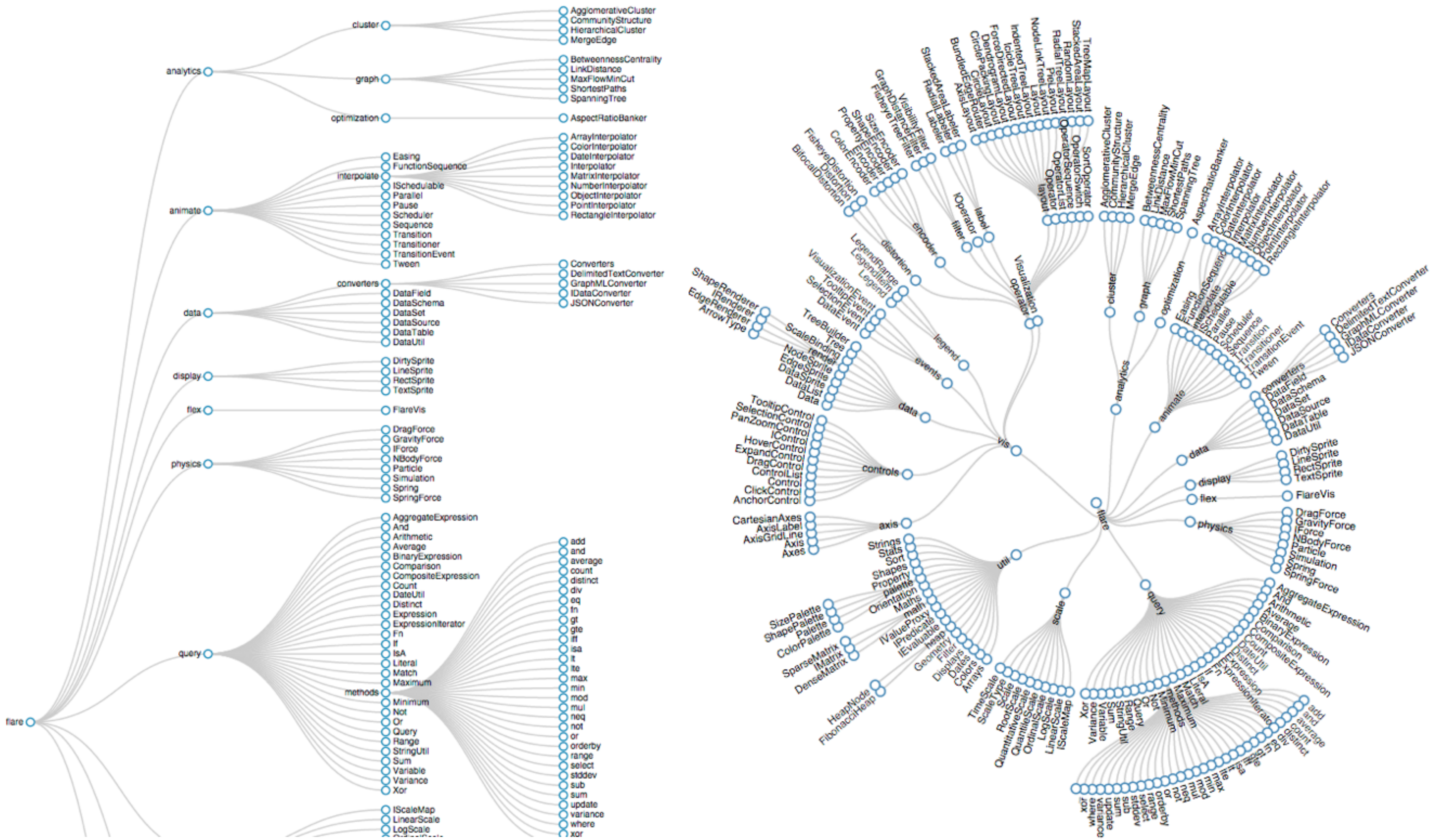
mbostock.github.com/d3/talk/20111116/bundle.html

Michael Bostock

Explicit Tree Visualization

Reingold–
Tilford layout

http://billmill.org/pymag-trees/

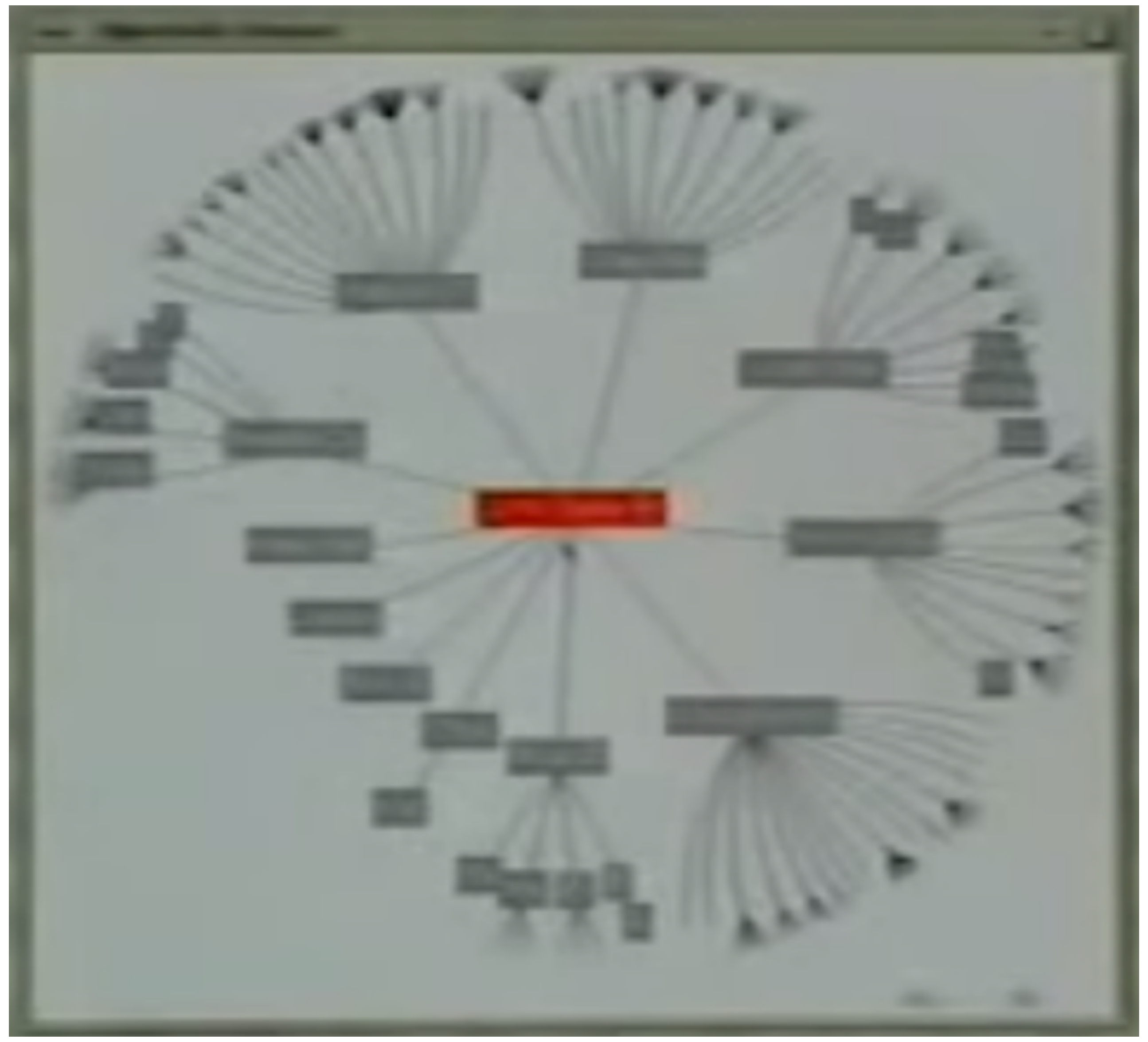


Hyperbolic Tree

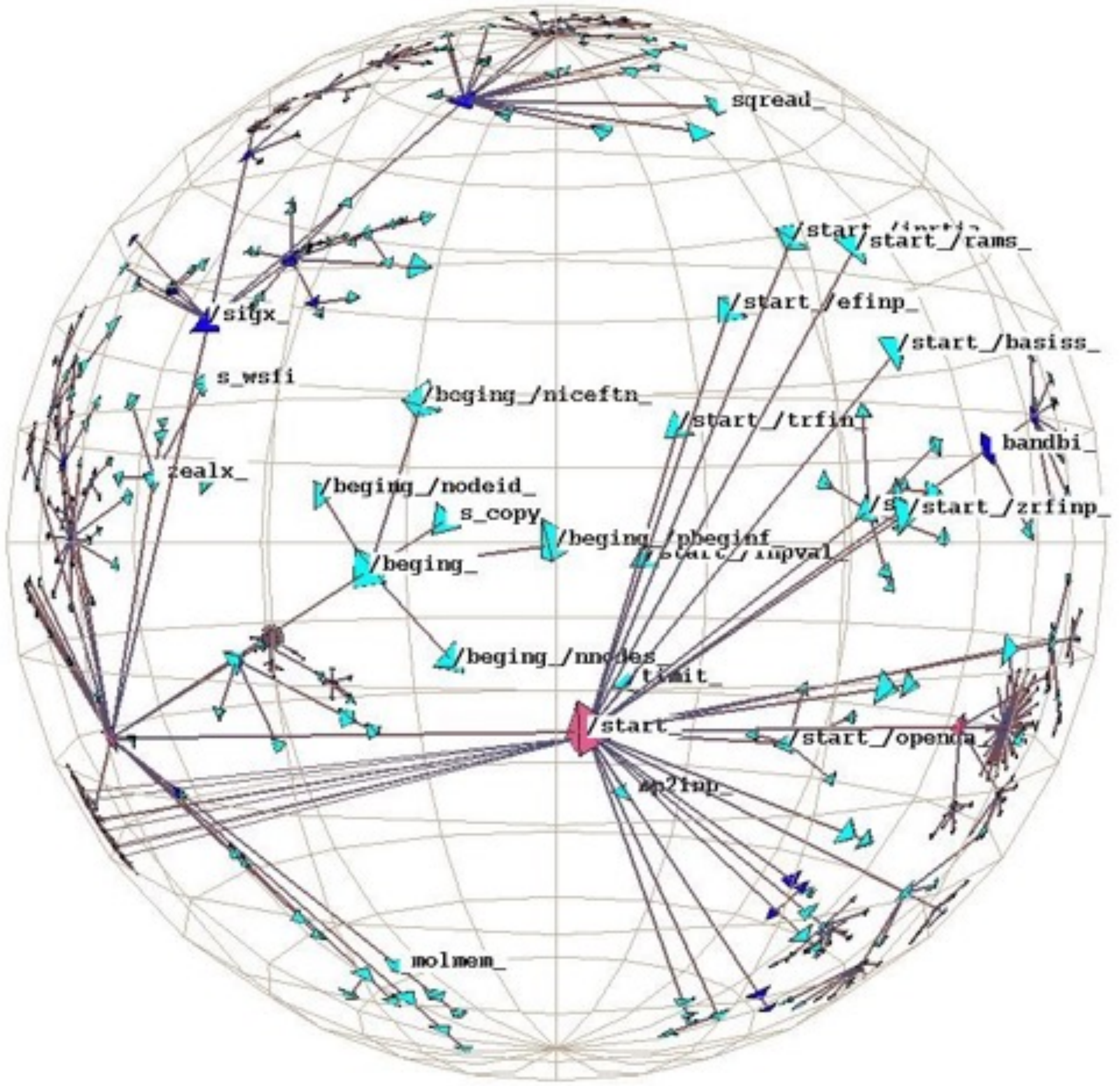
Projection on a sphere (hyperbolic space)

Root initially in the center

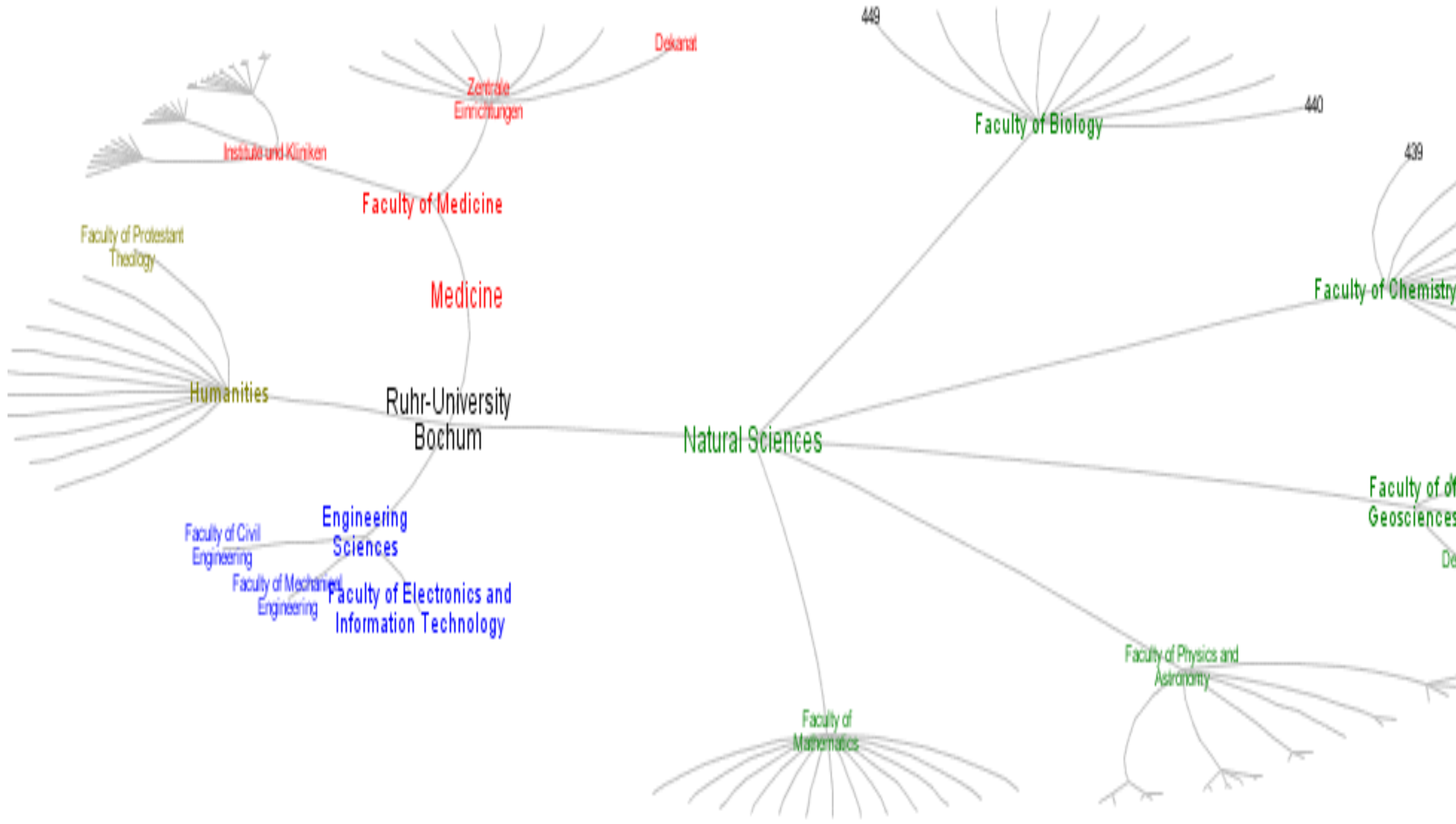
Other nodes can be moved into focus



Lamping and Rao 1995

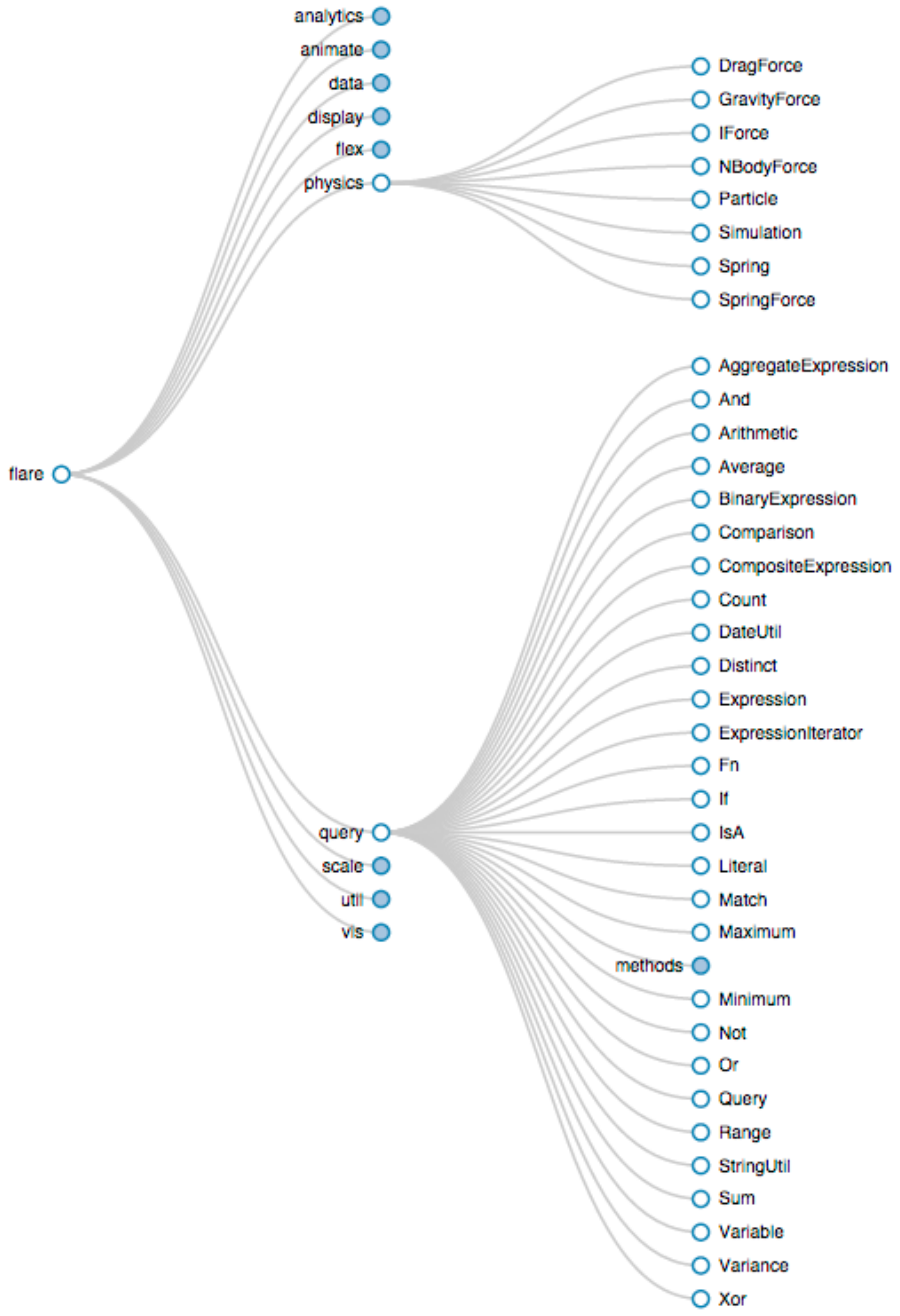


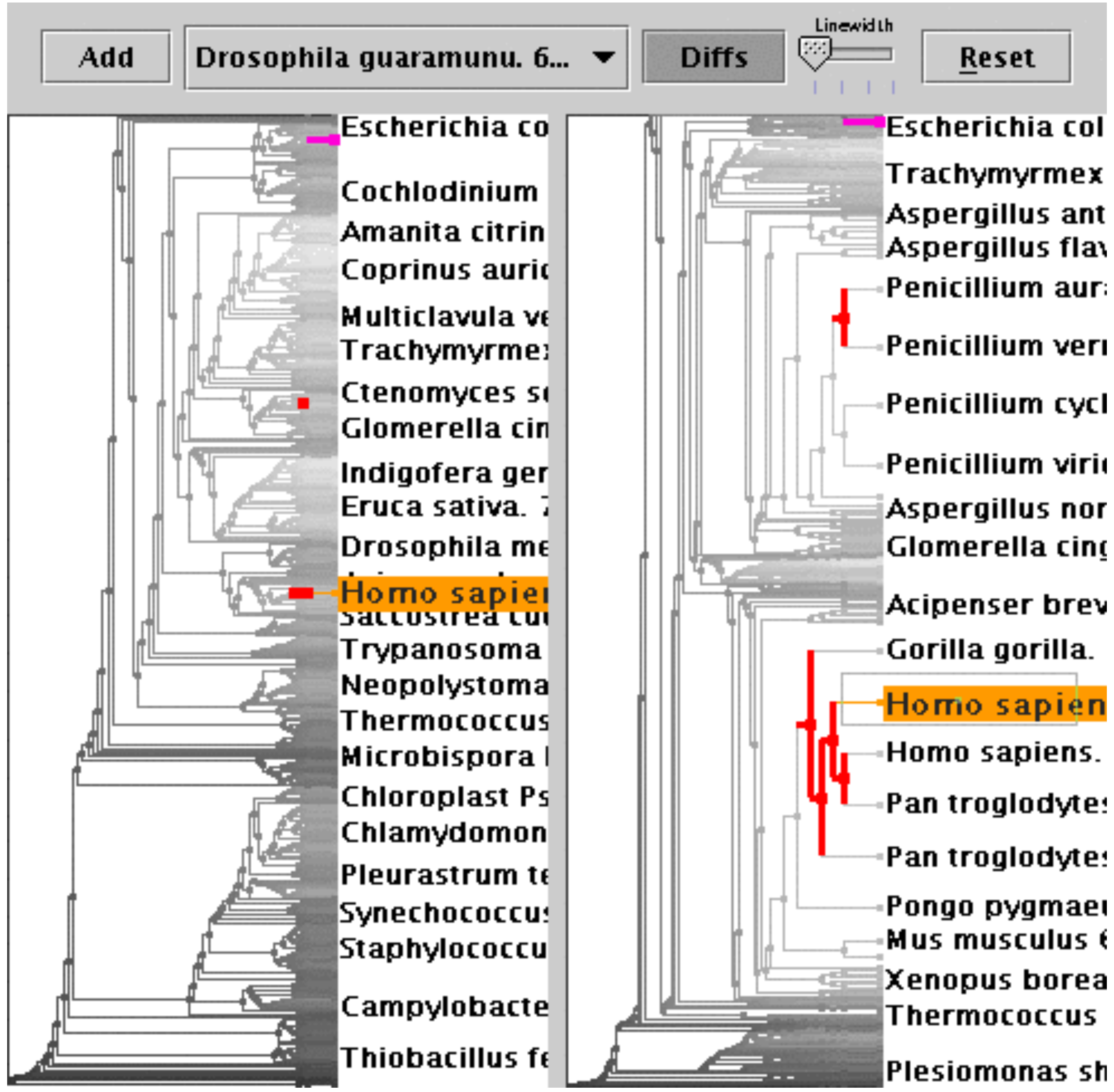
Munzner 1997



http://hypergraph.sourceforge.net/examples-orga.html

Tree Interaction, Tree Comparison

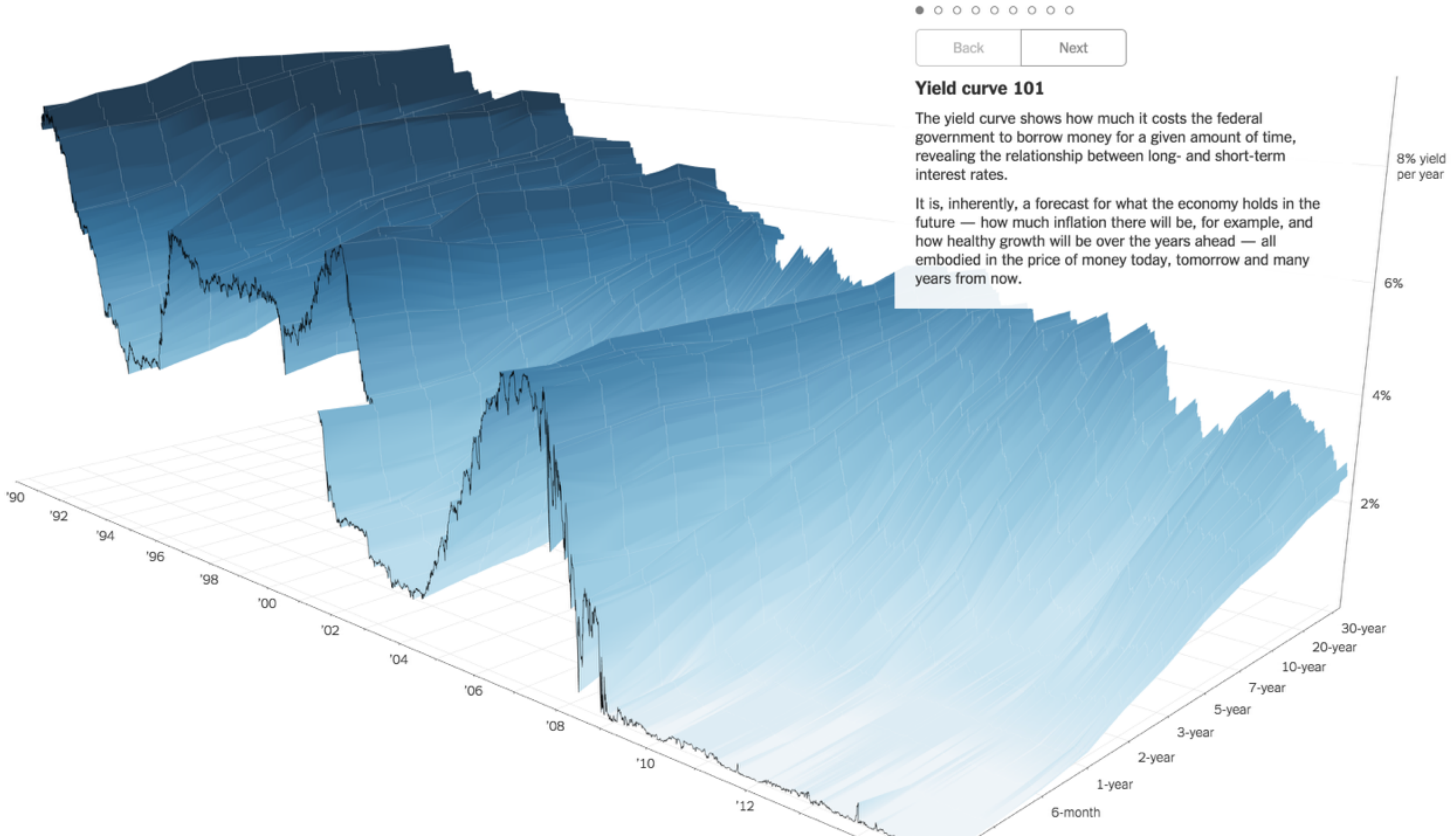




Design Critique

The Yield Curve

http://goo.gl/mt1iQo



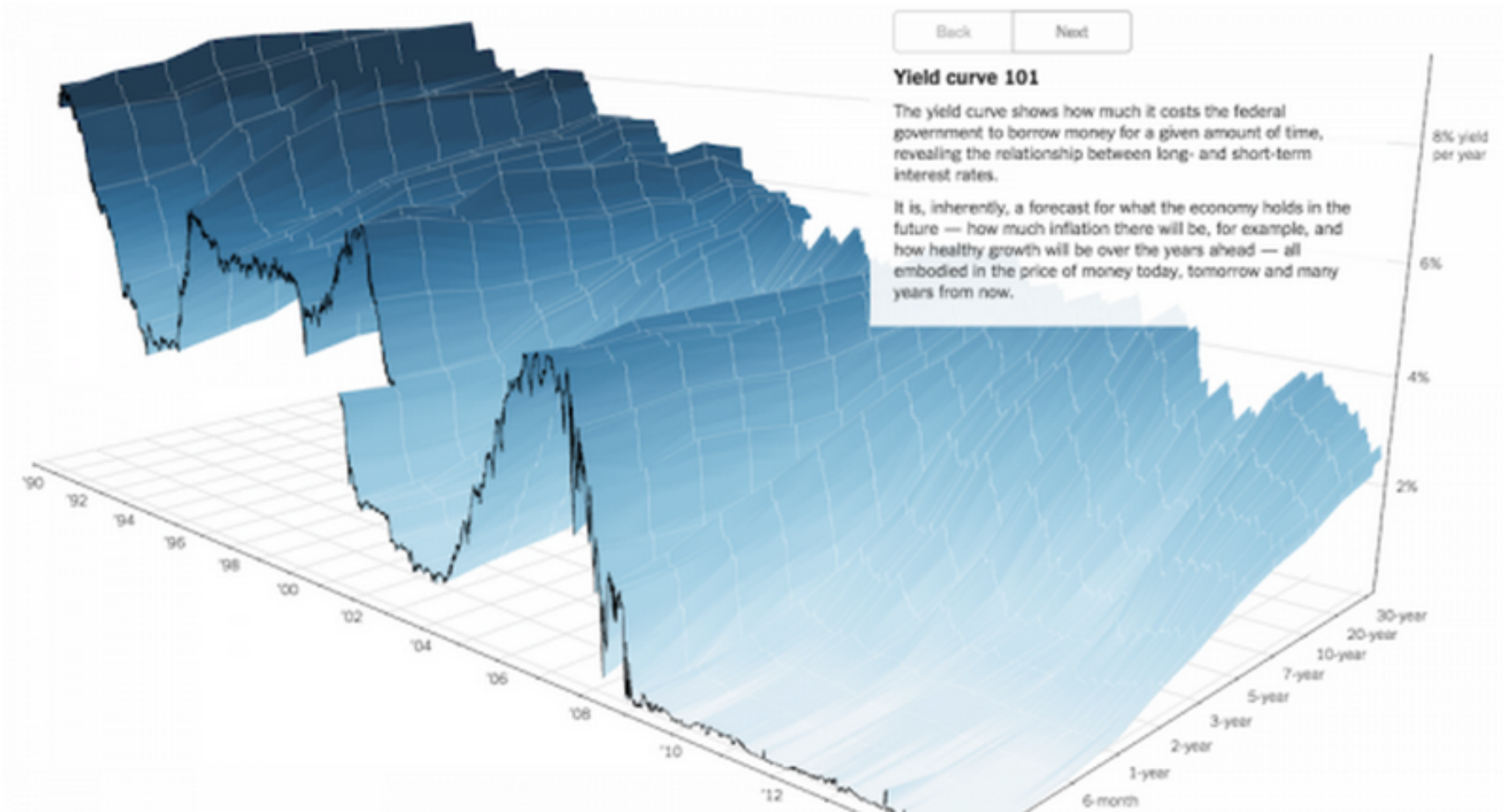
WHEN 3D WORKS

By Andy Kirk | March 20, 2015 | Articles

Earlier this week TheUpshot published a new interactive project visualising the 'Yield Curve'. Created by Gregor Aisch and Amanda Cox the work provides a "3-D view of a chart that predicts the economic future".

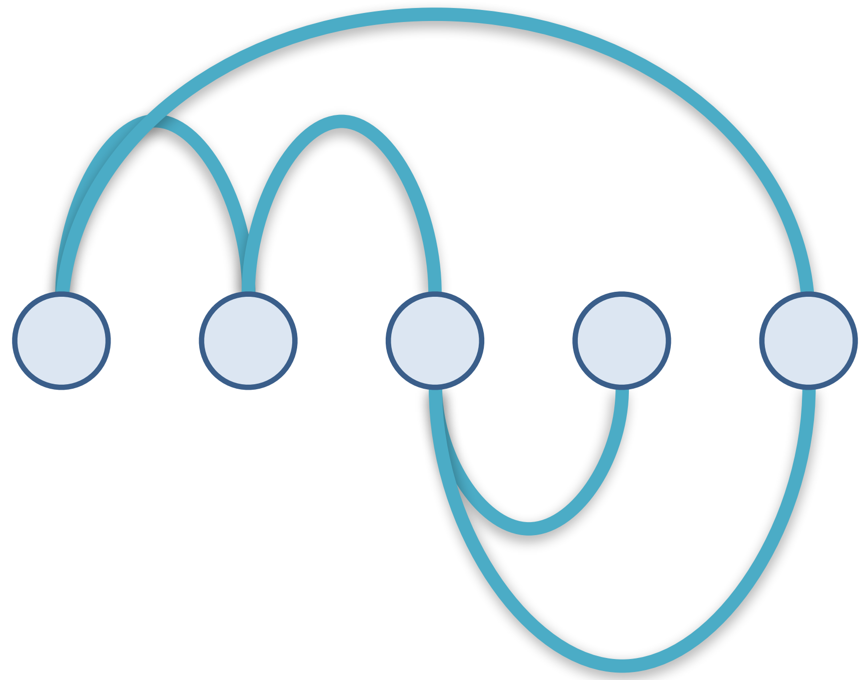
It is a terrific piece of work because, as with any good visualisation, it makes understanding accessible, providing a visual explanation of a potentially (at least for me) complicated subject matter.

The most striking immediate feature is the initial 3D display. Whilst the project received lots of deserved praise online I am conscious that being positive about a 3D work might strike some as going against the grain: as we know, 3D is one of the reliable punching bags for visualisation angst. However, I thought it was important to explain why 3D doesn't just work but is essential in this case.

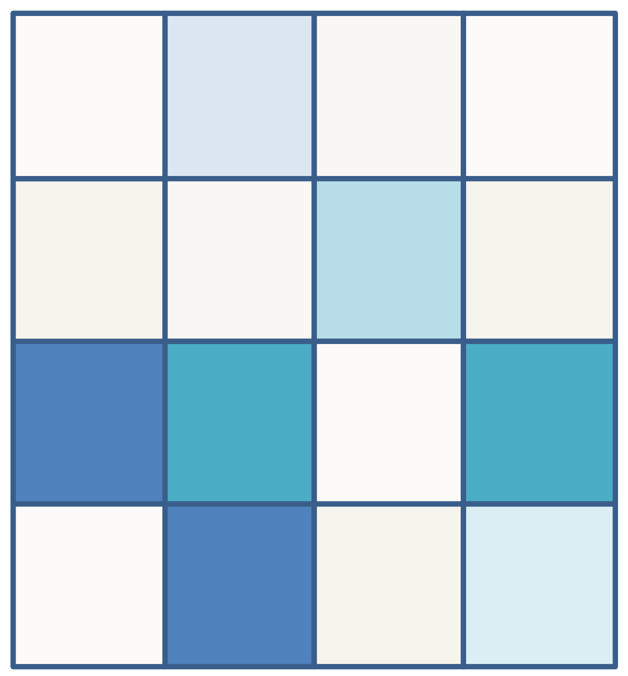


Matrix Representations

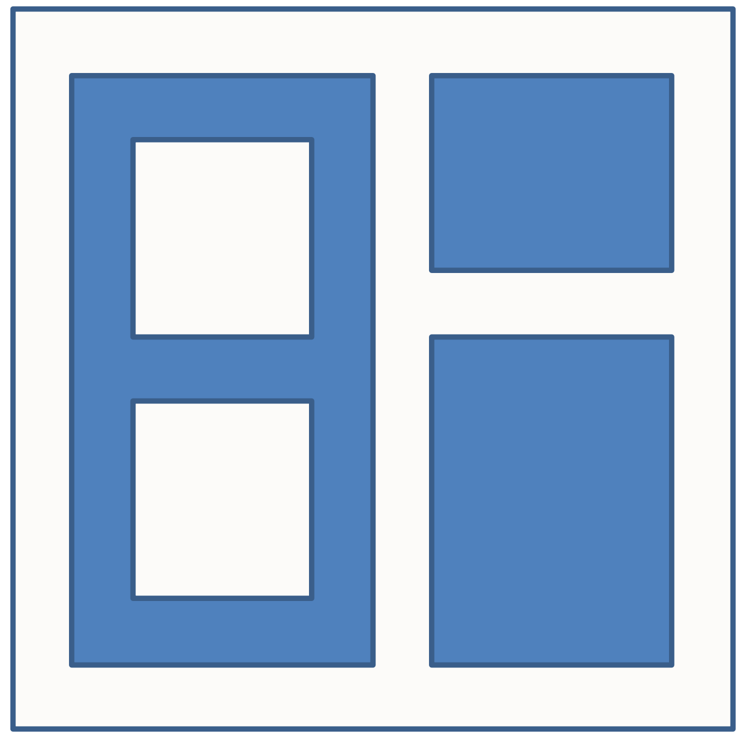
Matrix Representations



Explicit
(Node-Link)



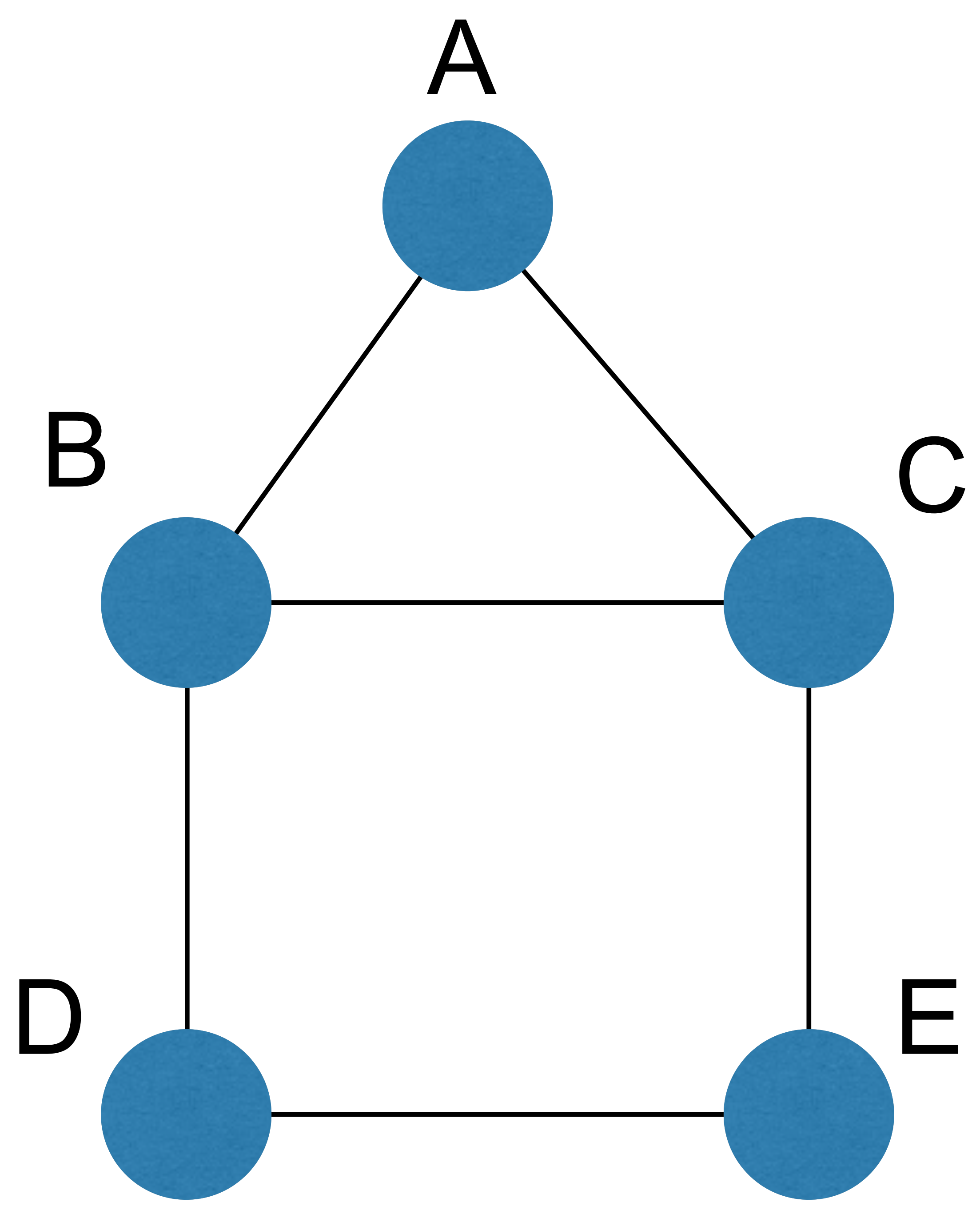
Matrix

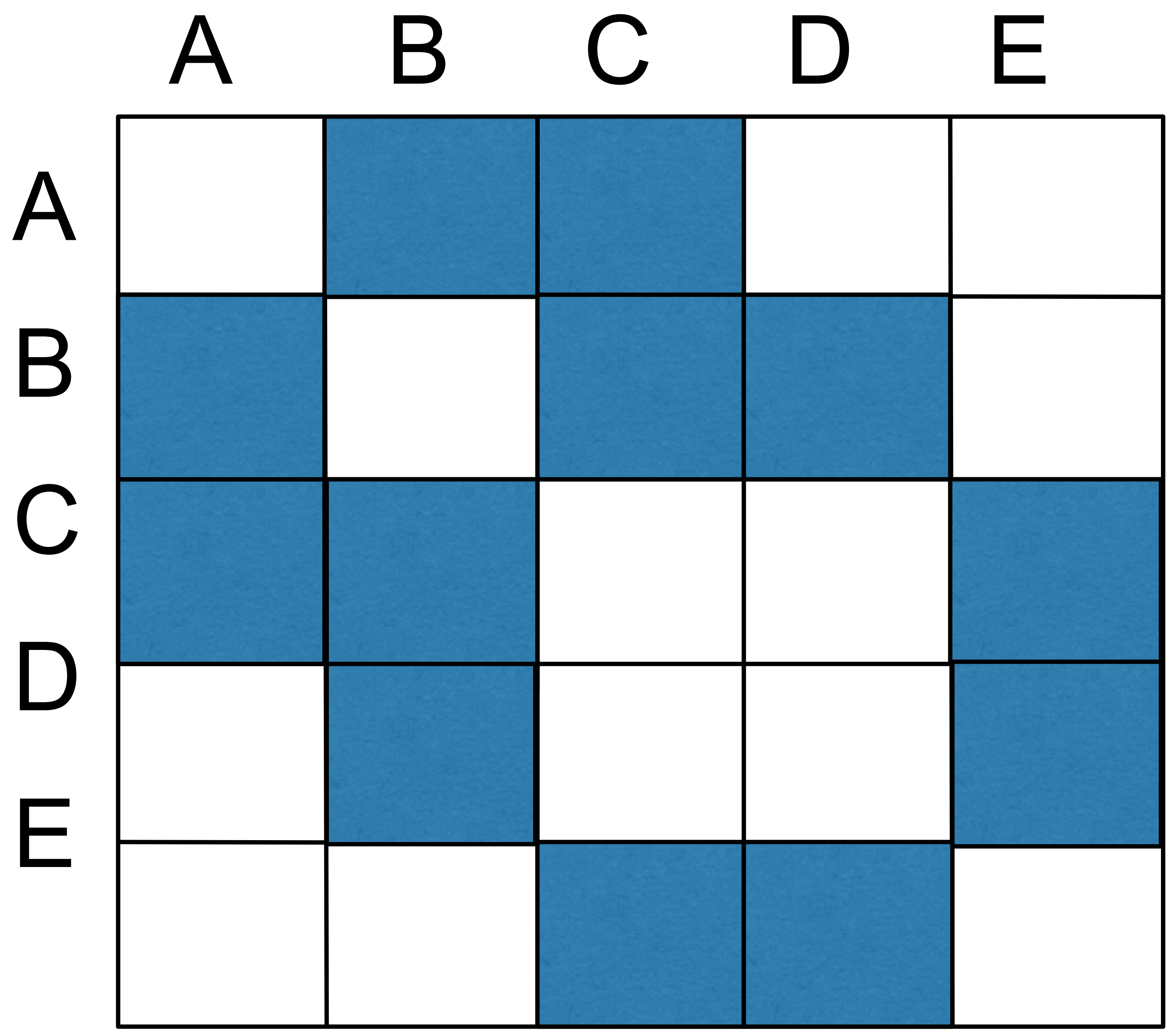


Implicit

Matrix Representations

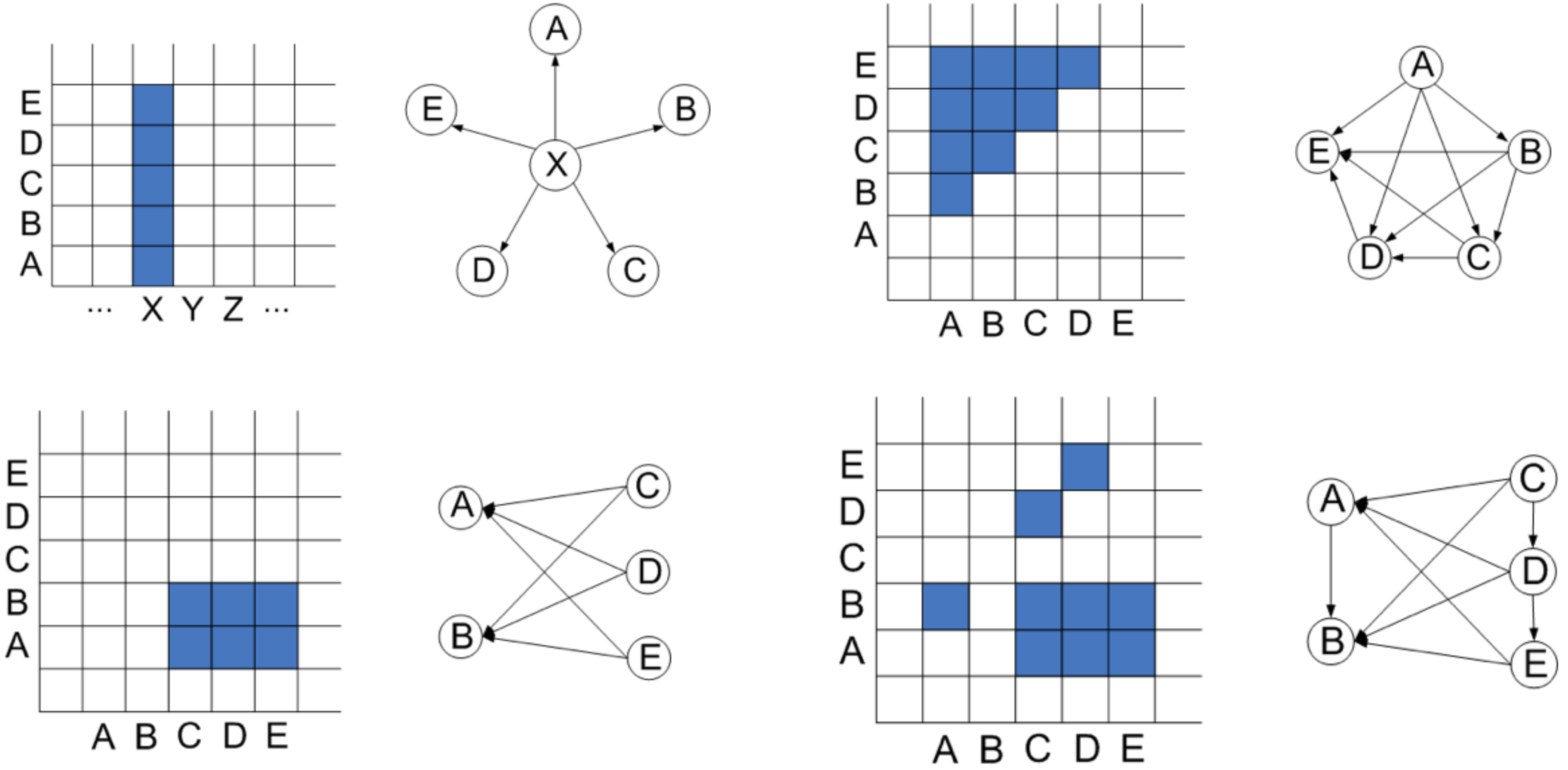
Instead of node link diagram, use adjacency matrix





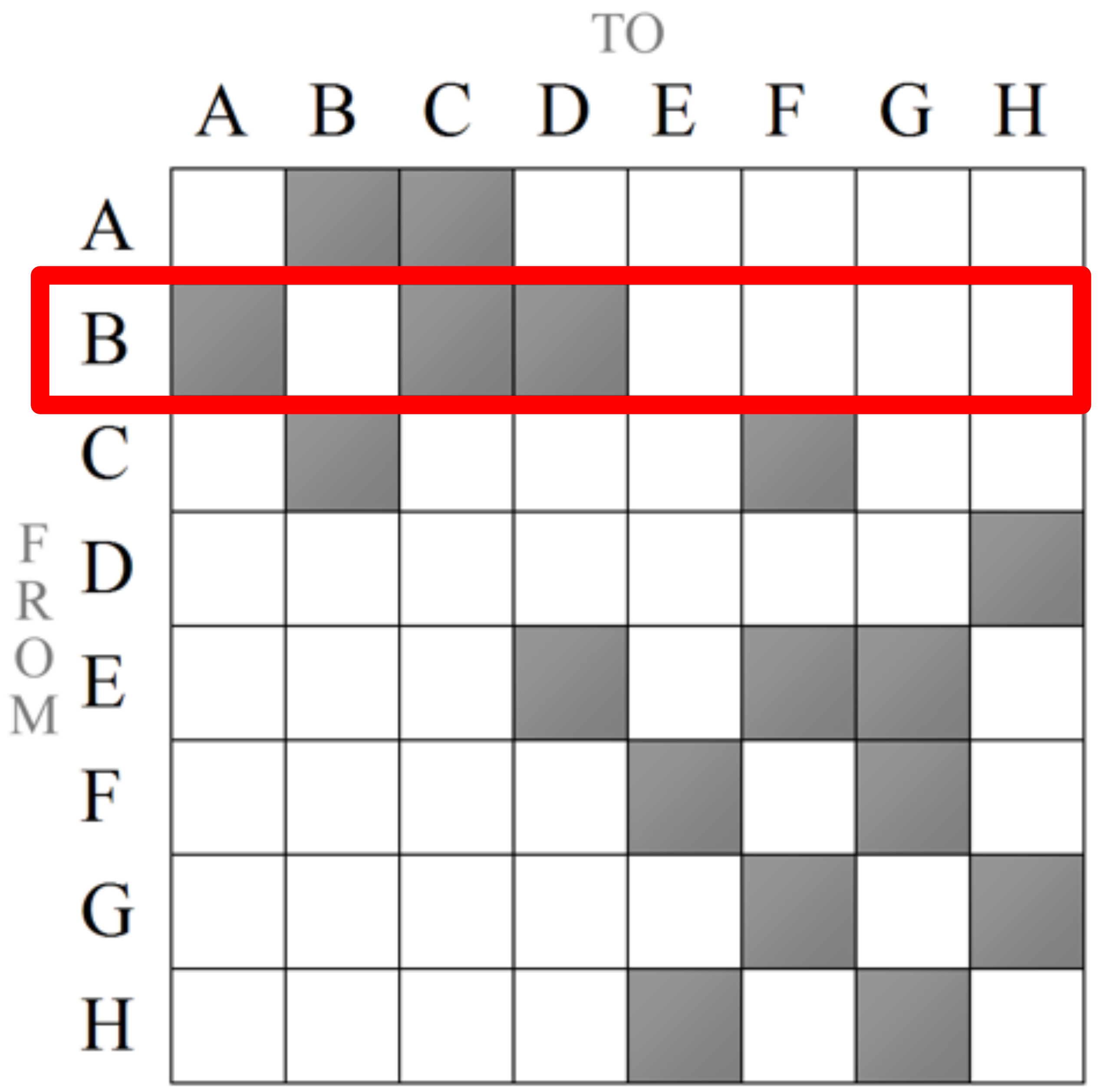
Matrix Representations

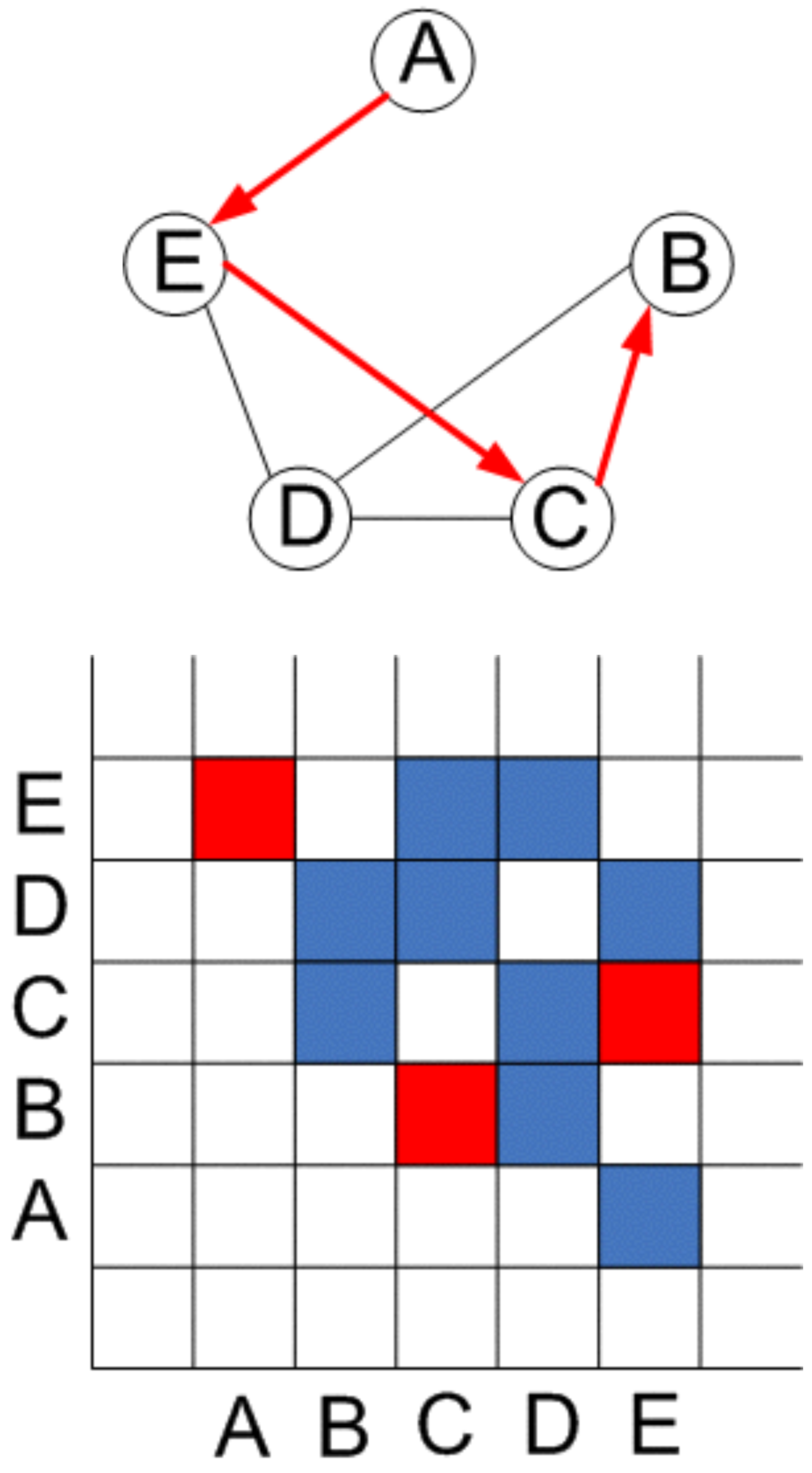
Examples:



HJ Schulz 2007

Matrix Representations

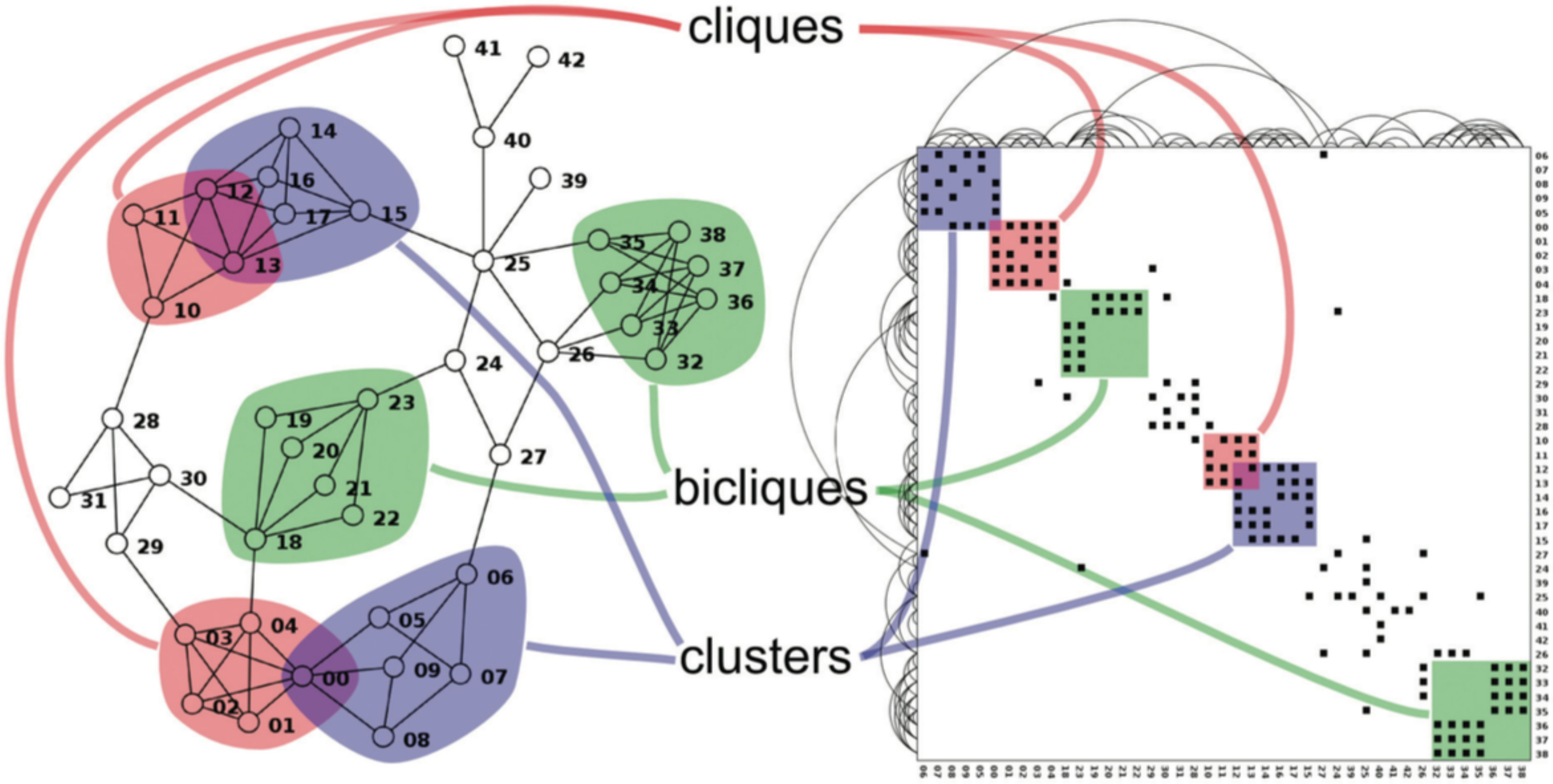




Well suited for
neighborhood-related TBTs

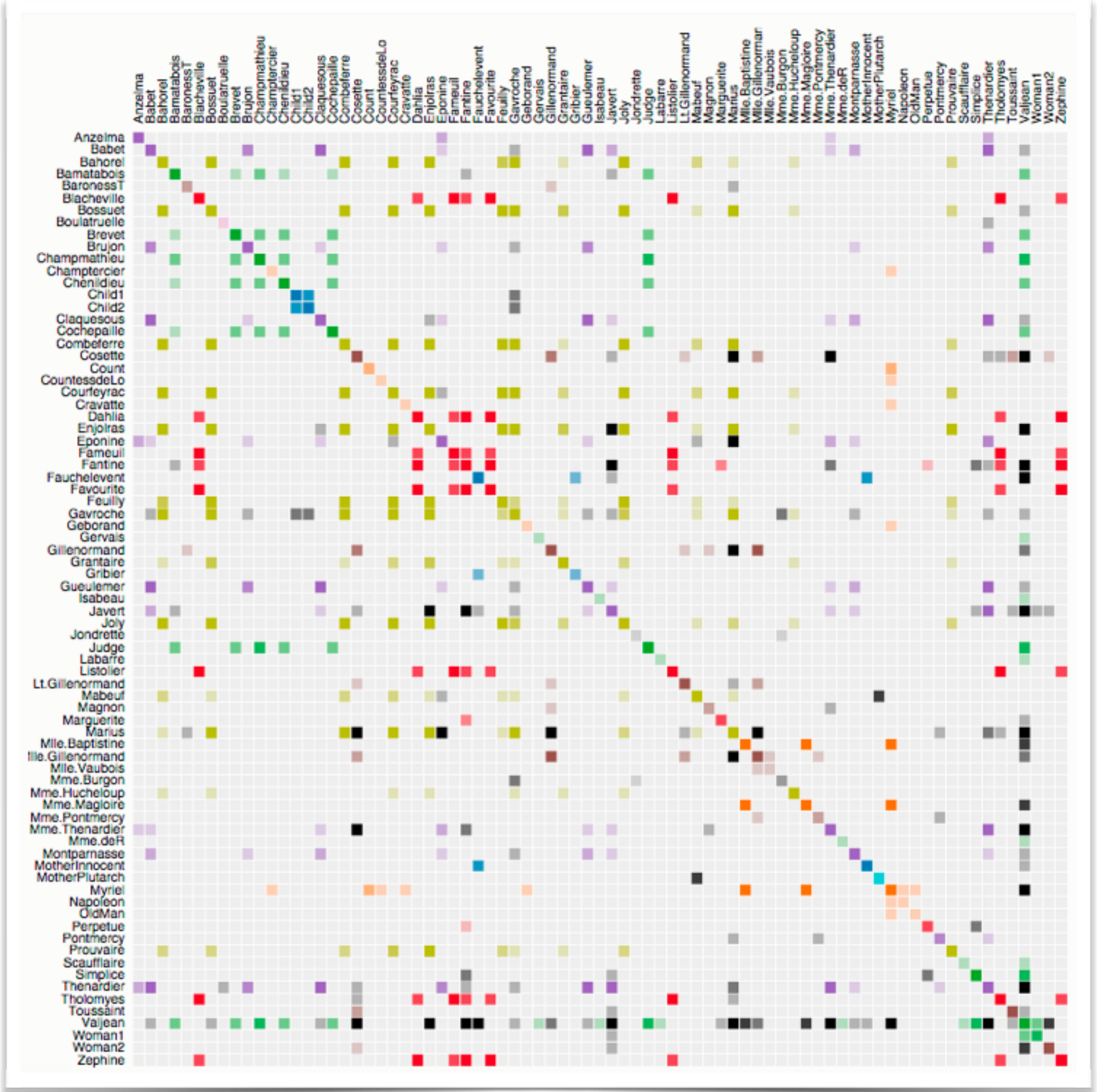
Not suited for
path-related TBTs

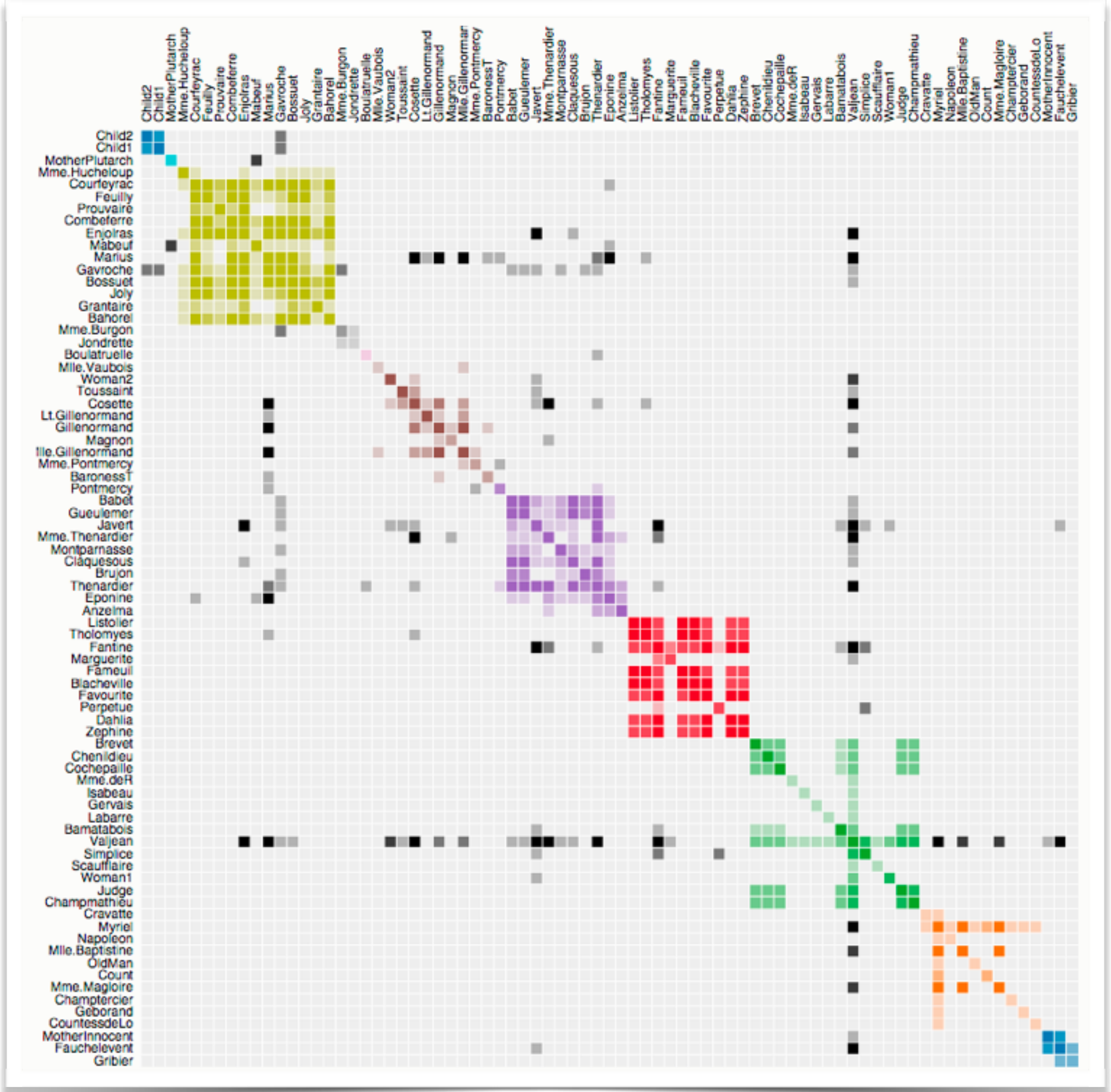
van Ham et al. 2009
Shen et al. 2007



McGuffin 2012

Order Critical!





Matrix Representations

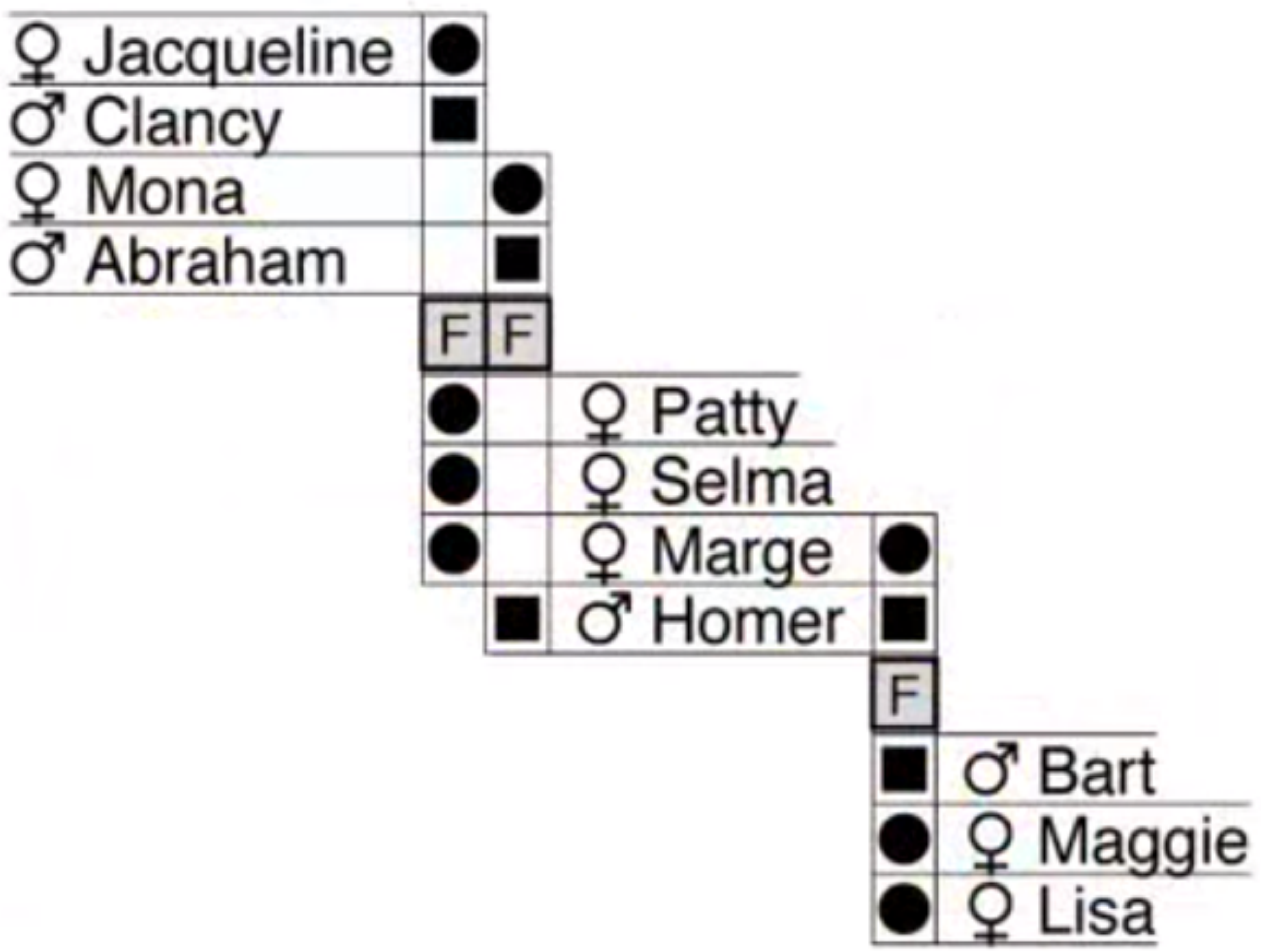
Pros:

- can represent **all graph classes** except for hypergraphs
- puts **focus on the edge set**, not so much on the node set
- simple grid -> **no elaborate layout** or rendering needed
- well suited for **ABT on edges** via coloring of the matrix cells
- well suited for **neighborhood-related TBTs** via traversing rows/columns

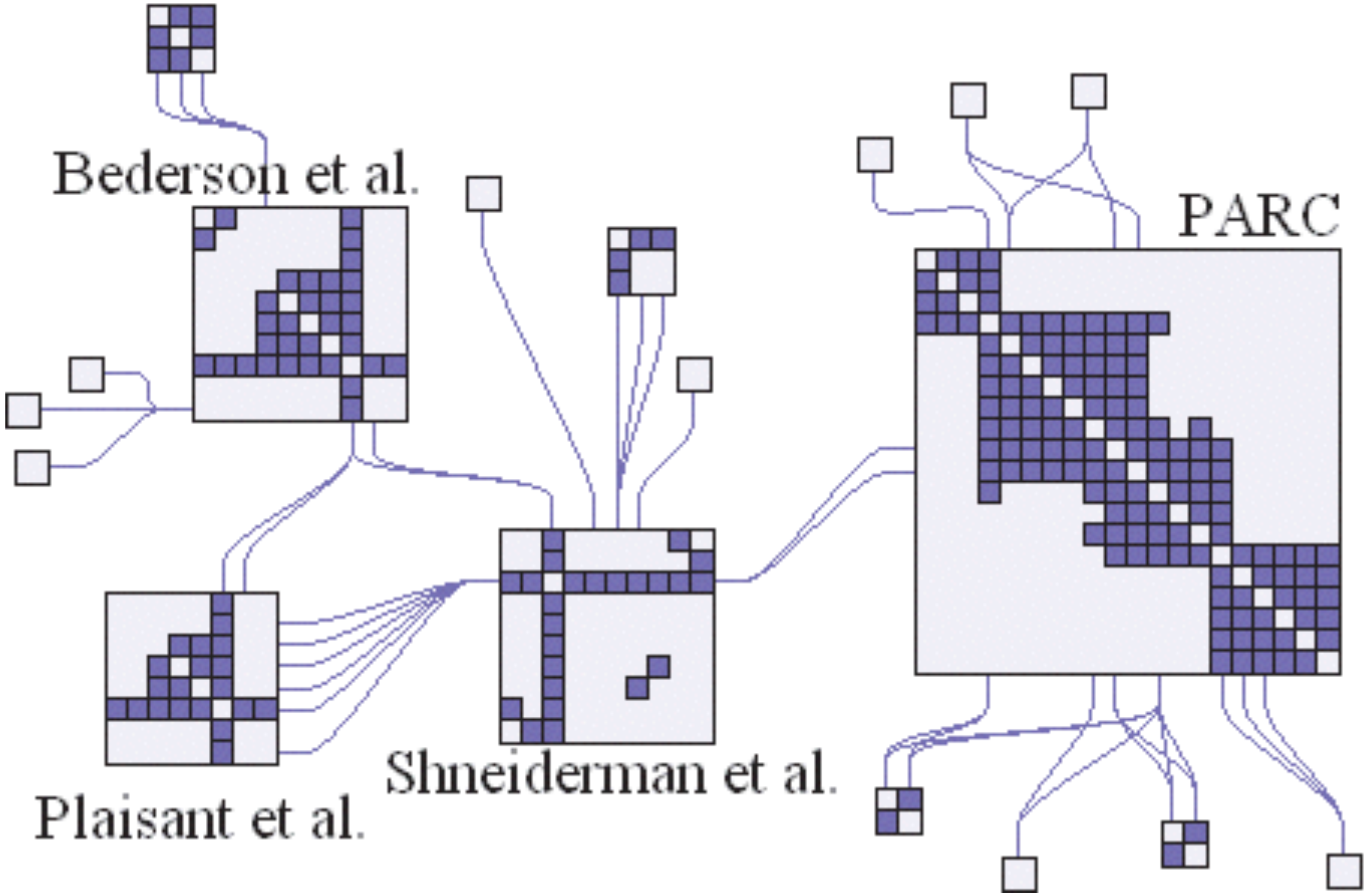
Cons:

- quadratic screen space requirement (any possible edge takes up space)
- not suited for path-related TBTs

Special Case: Genealogy

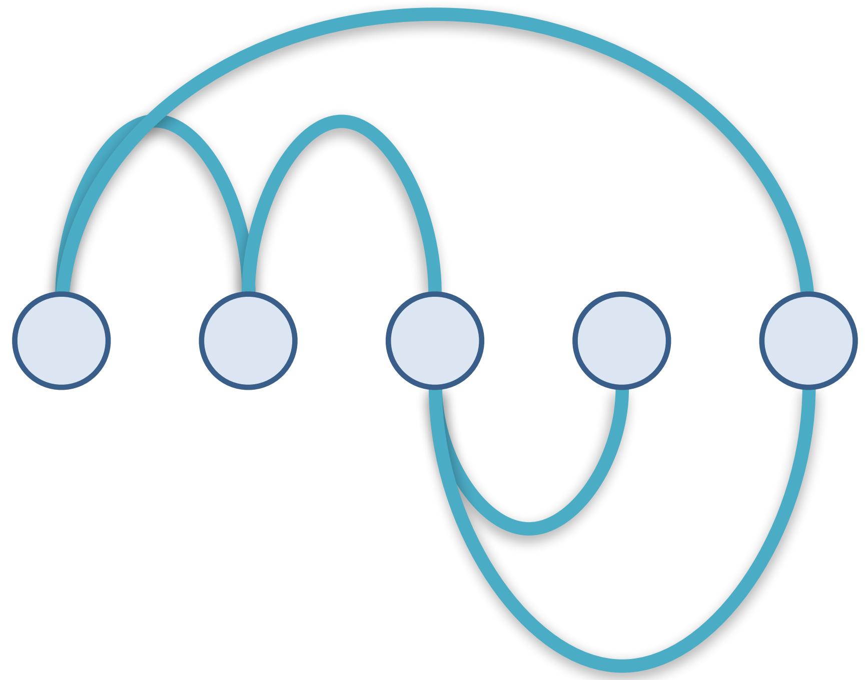


Hybrid Explicit/Matrix

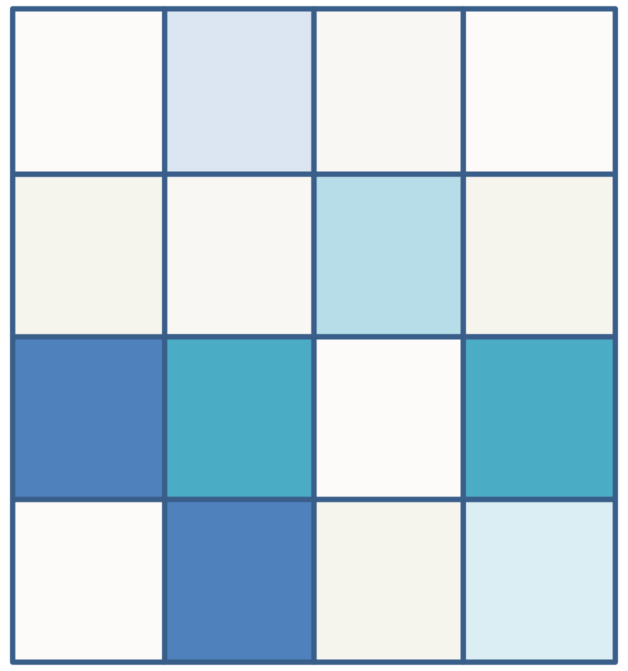


NodeTrix
[Henry et al. 2007]

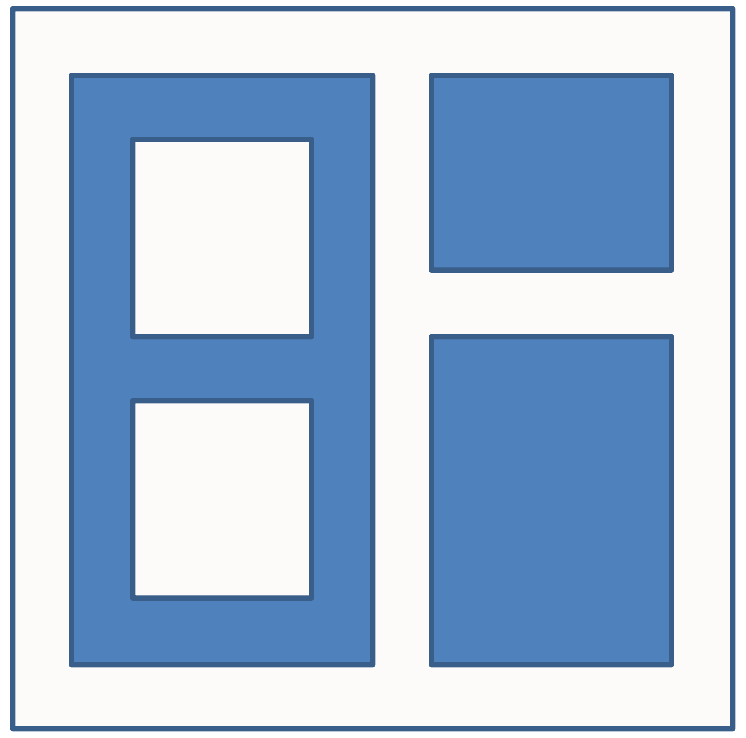
Implicit Layouts



Explicit
(Node-Link)



Matrix



Implicit

Explicit vs. Implicit Tree Vis

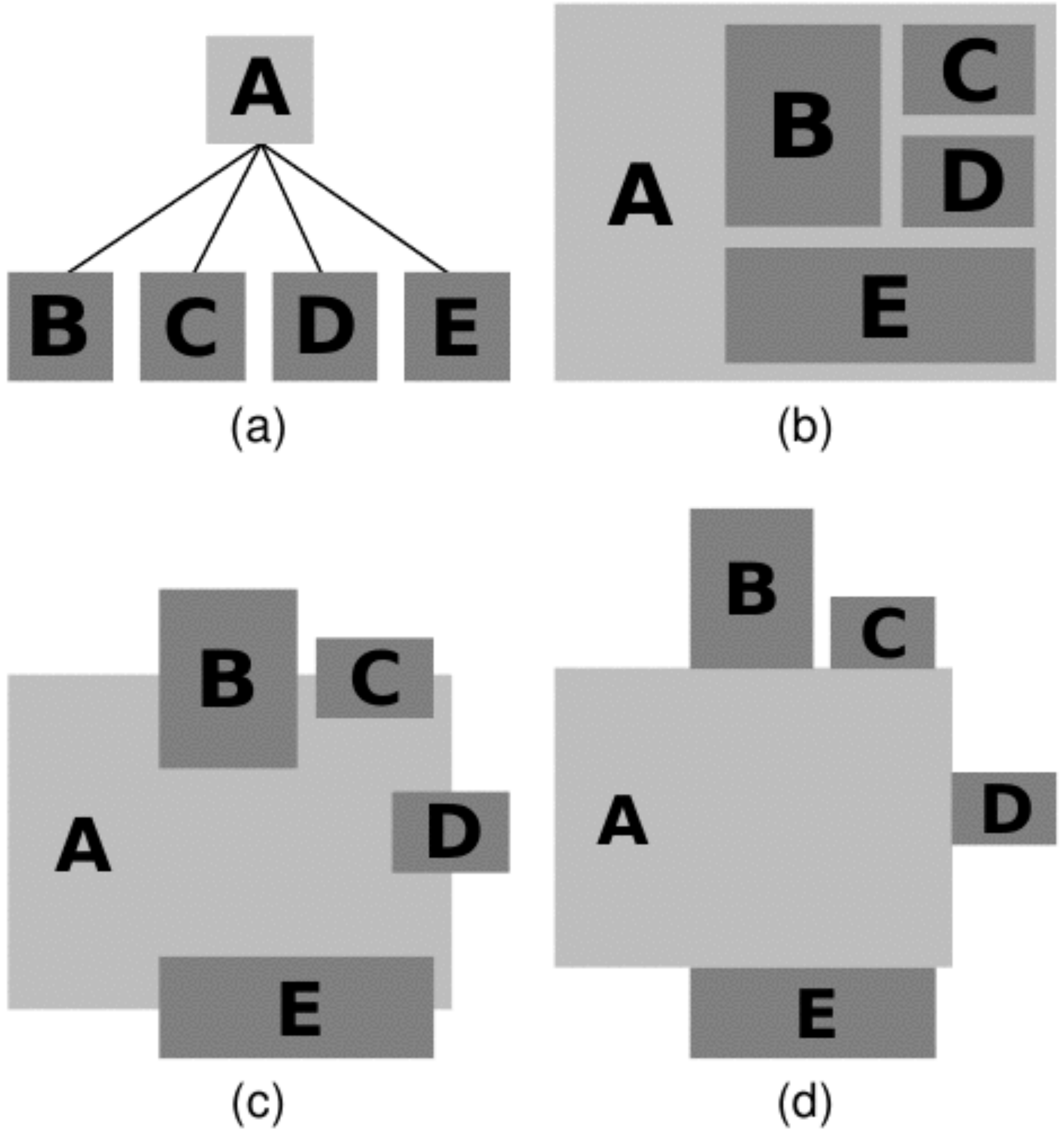
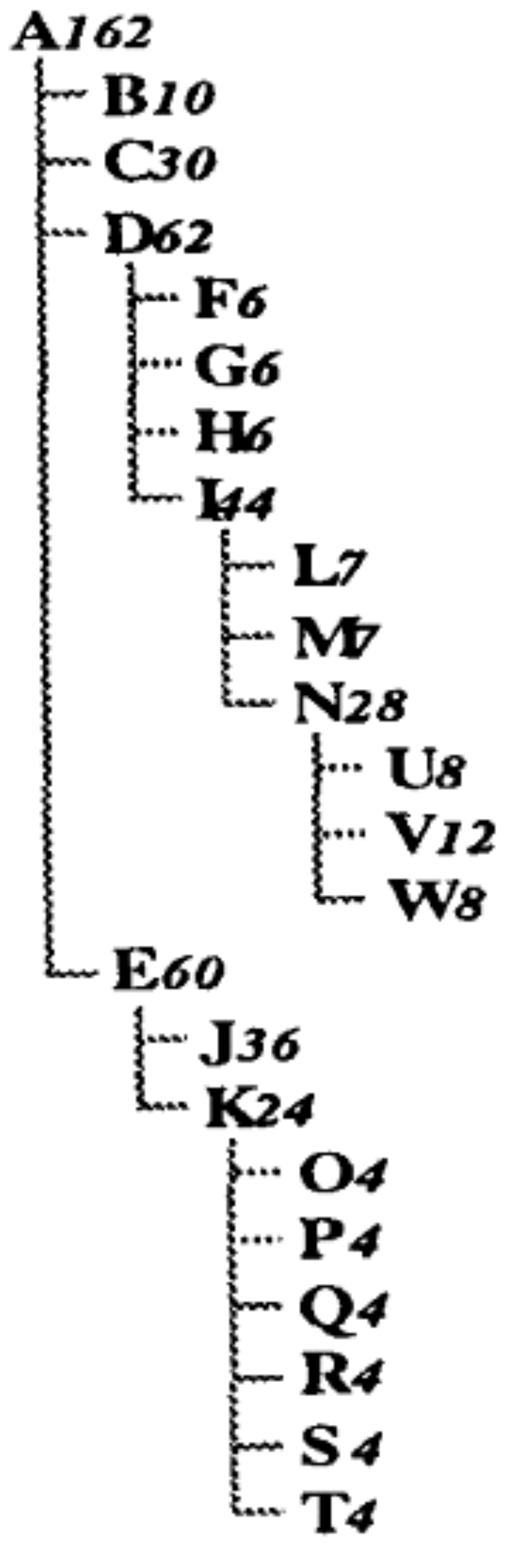
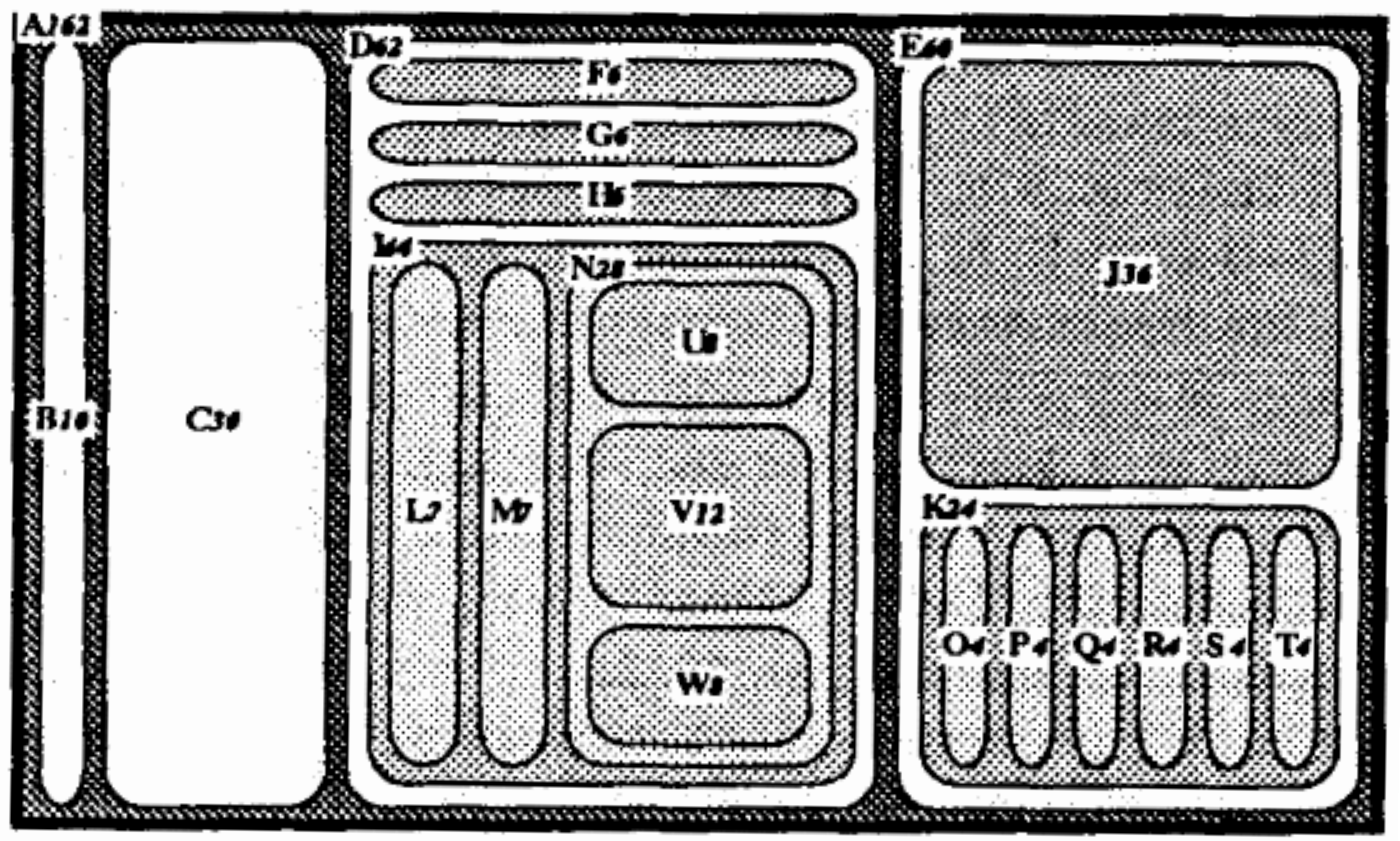


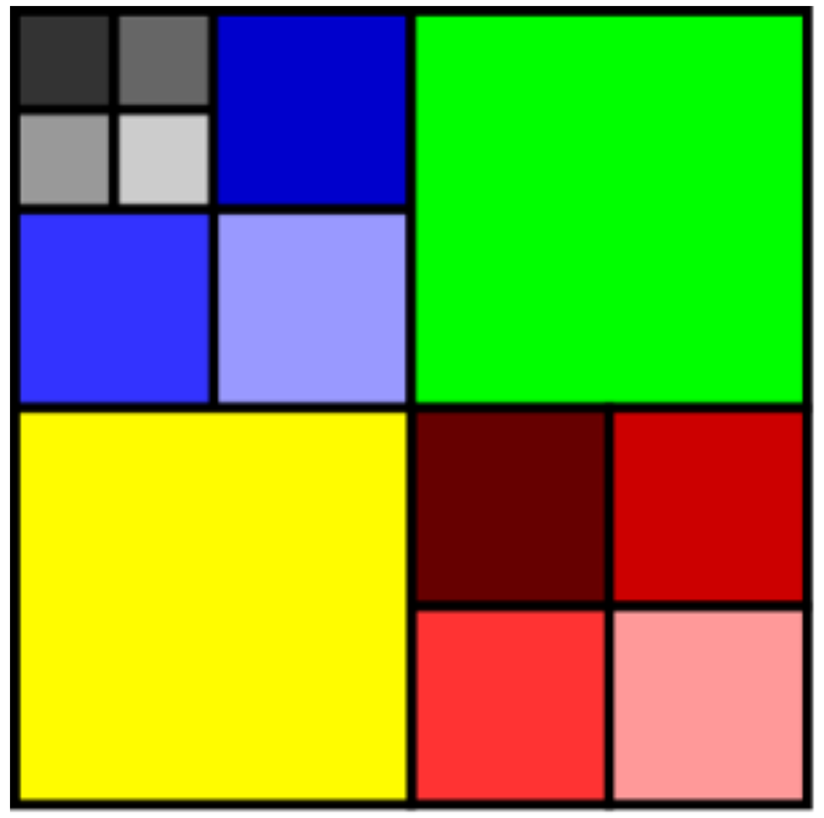
Fig. 2. (a) Explicit, node-link layout, (b) Implicit layout by inclusion, (c) Implicit Layout by overlap, (d) Implicit layout by adjacency.

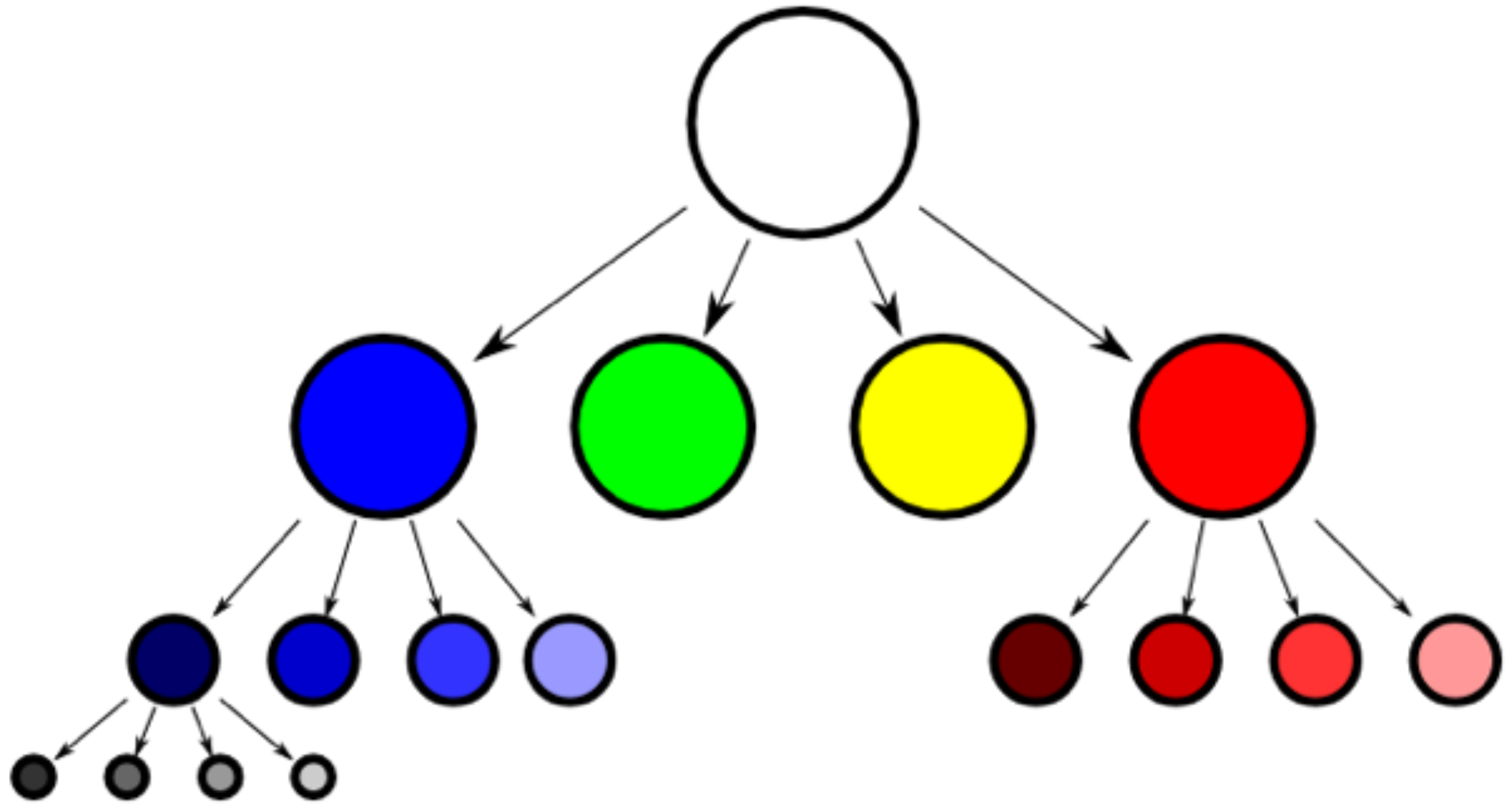
Schulz 2011

Tree Maps



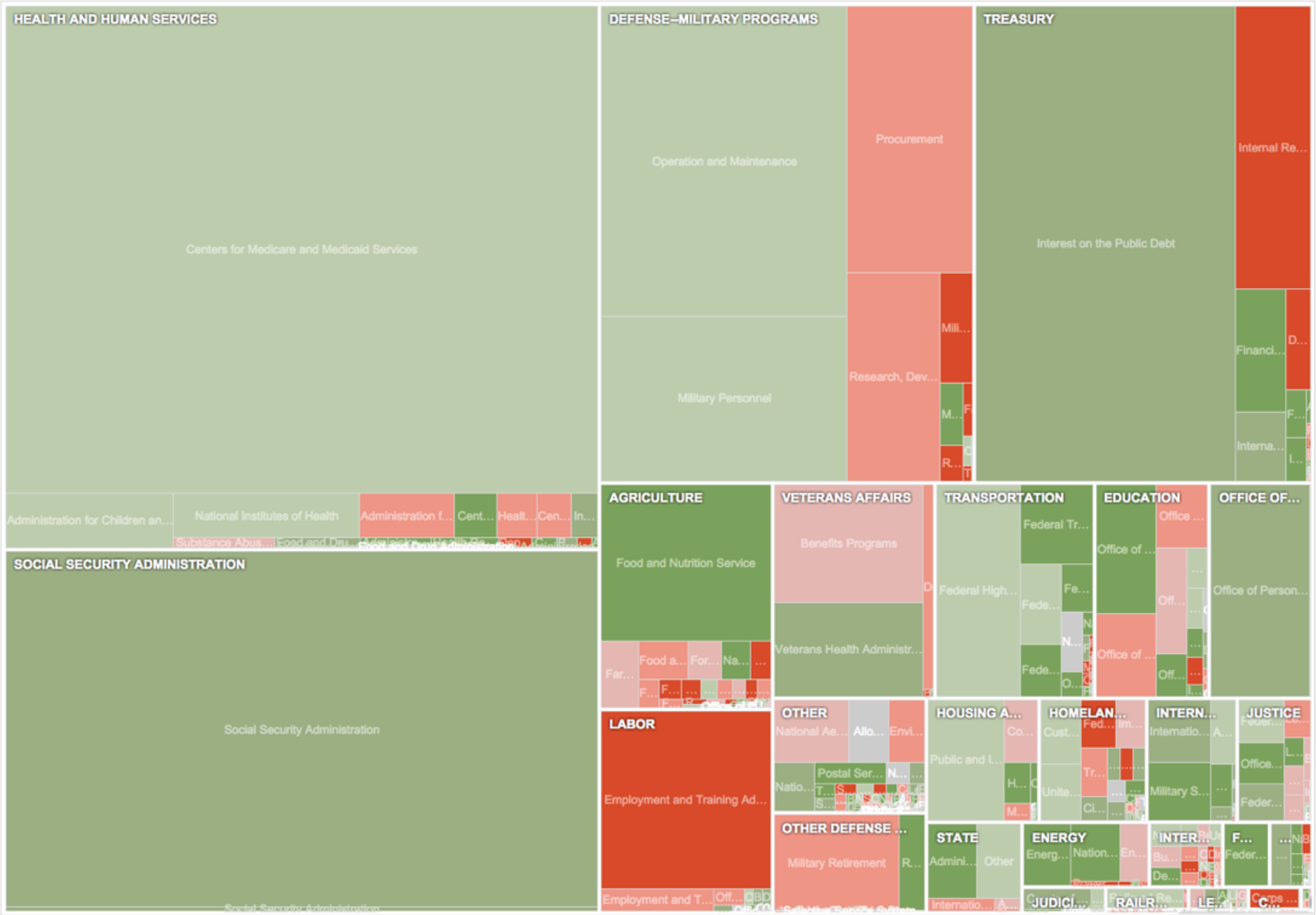




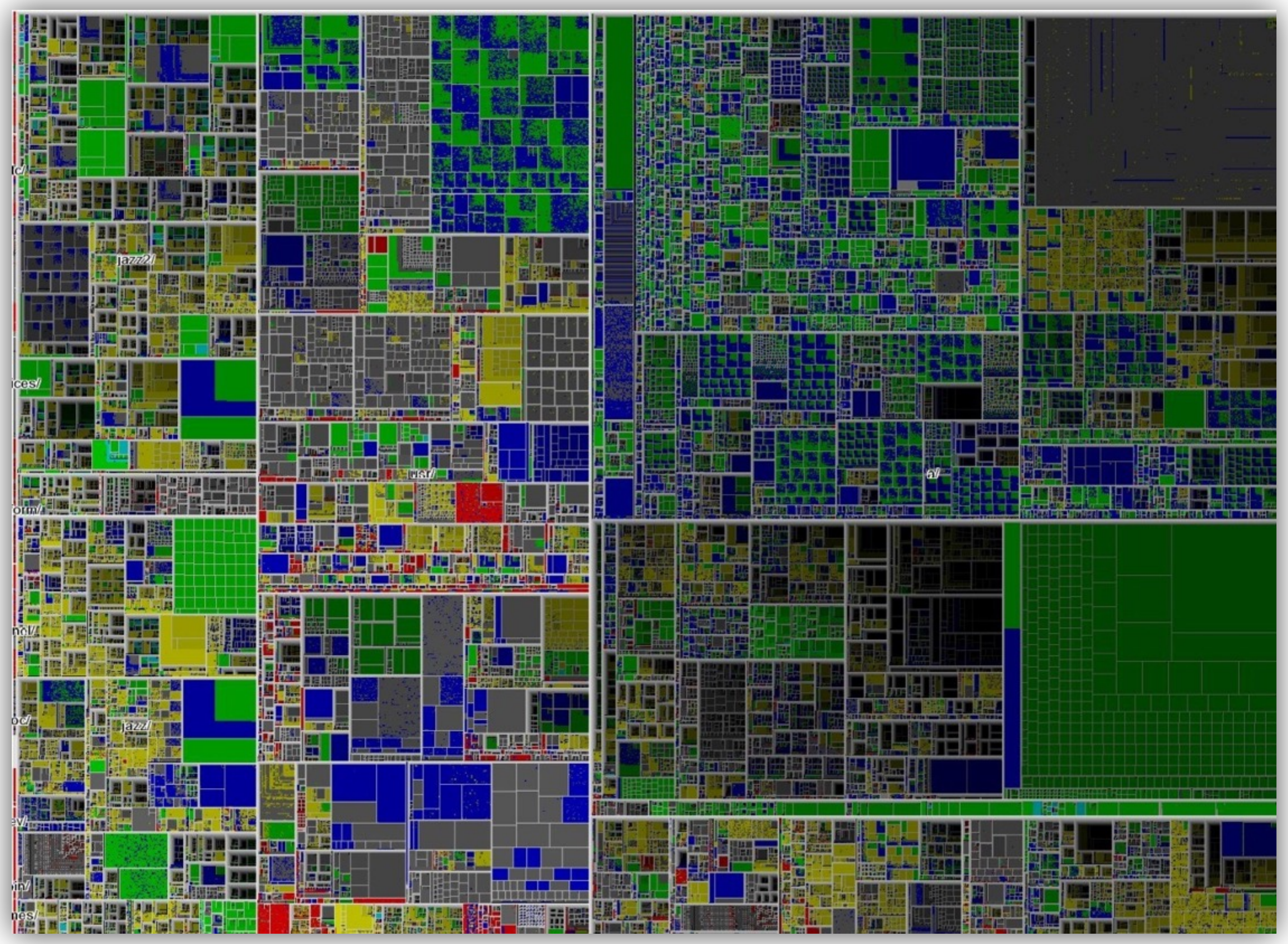


Johnson and Shneiderman 1991

Zoomable Treemap



Example: Interactive TreeMap of a Million Items



Fekete et al. 2002

Sunburst: Radial Layout

[Sunburst by John Stasko, Implementation in Caleydo by Christian Partl]

Implicit Representations

Pros:

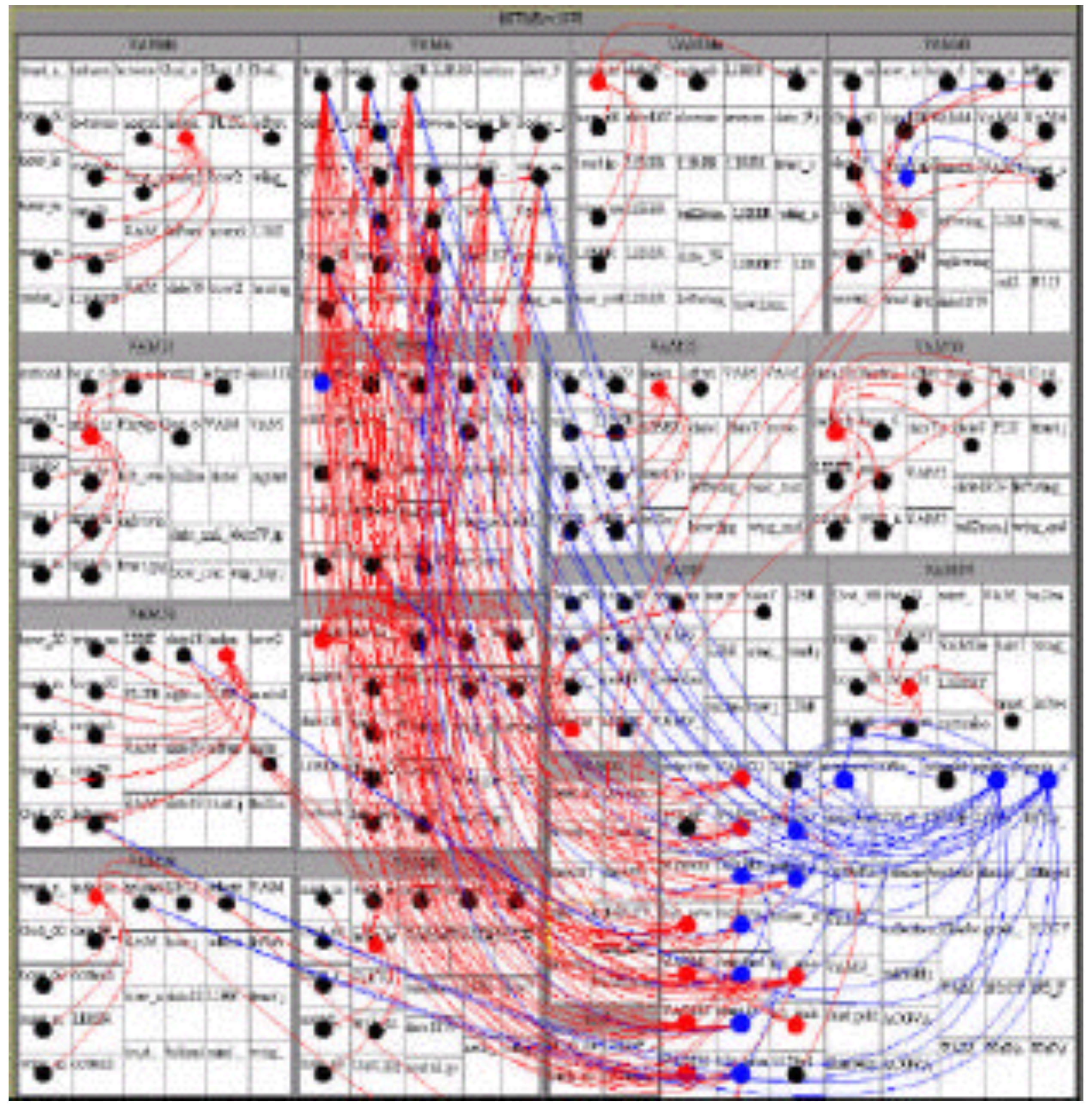
- space-efficient because of the lack of explicitly drawn edges - scale well
- well suited for ABTs on the node set
- also useful for some TBTs

Cons:

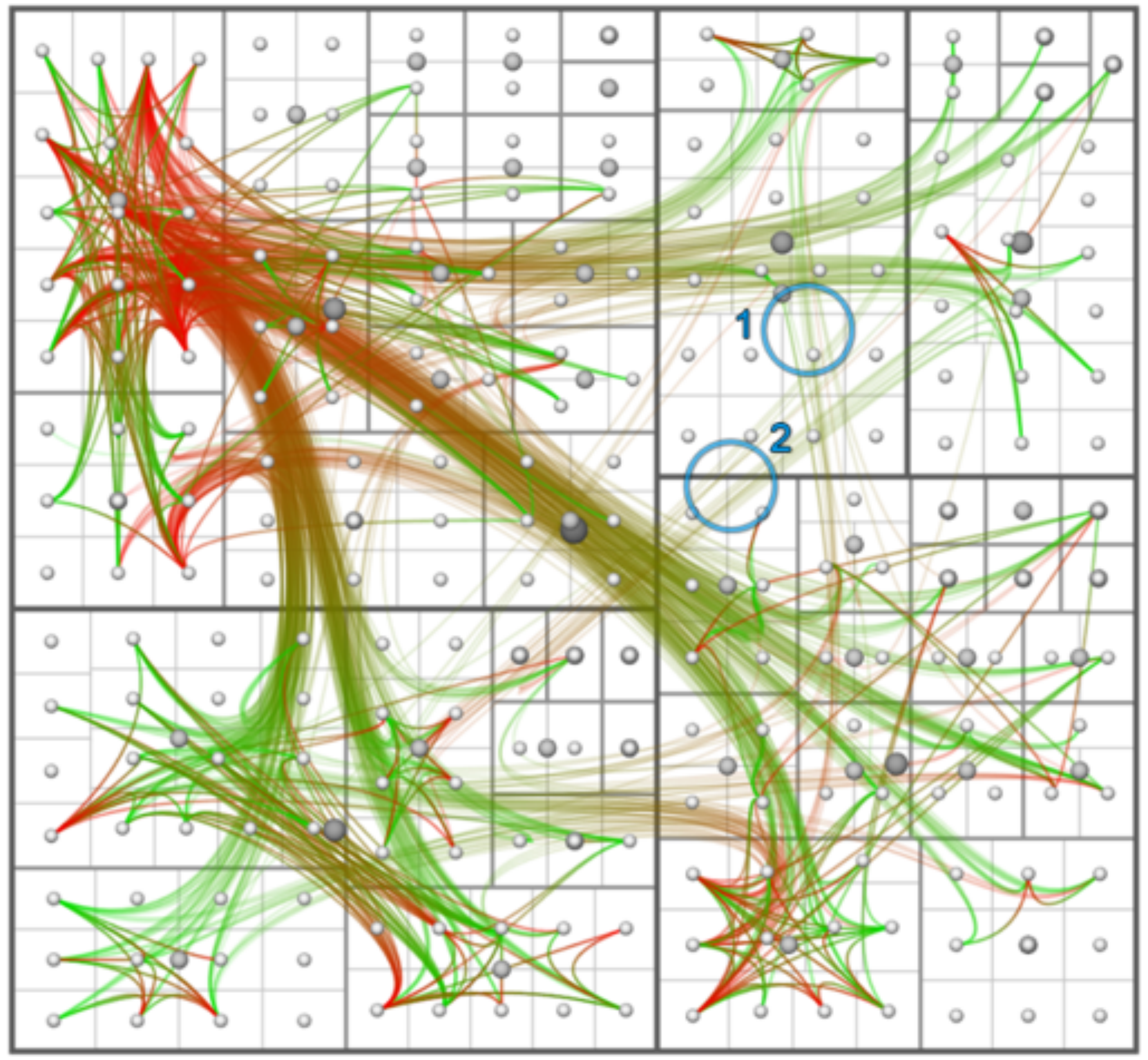
- can only represent trees
- no free arrangement (maps)
- useless for edge task

Adding Edges onto TreeMaps

without edge bundling



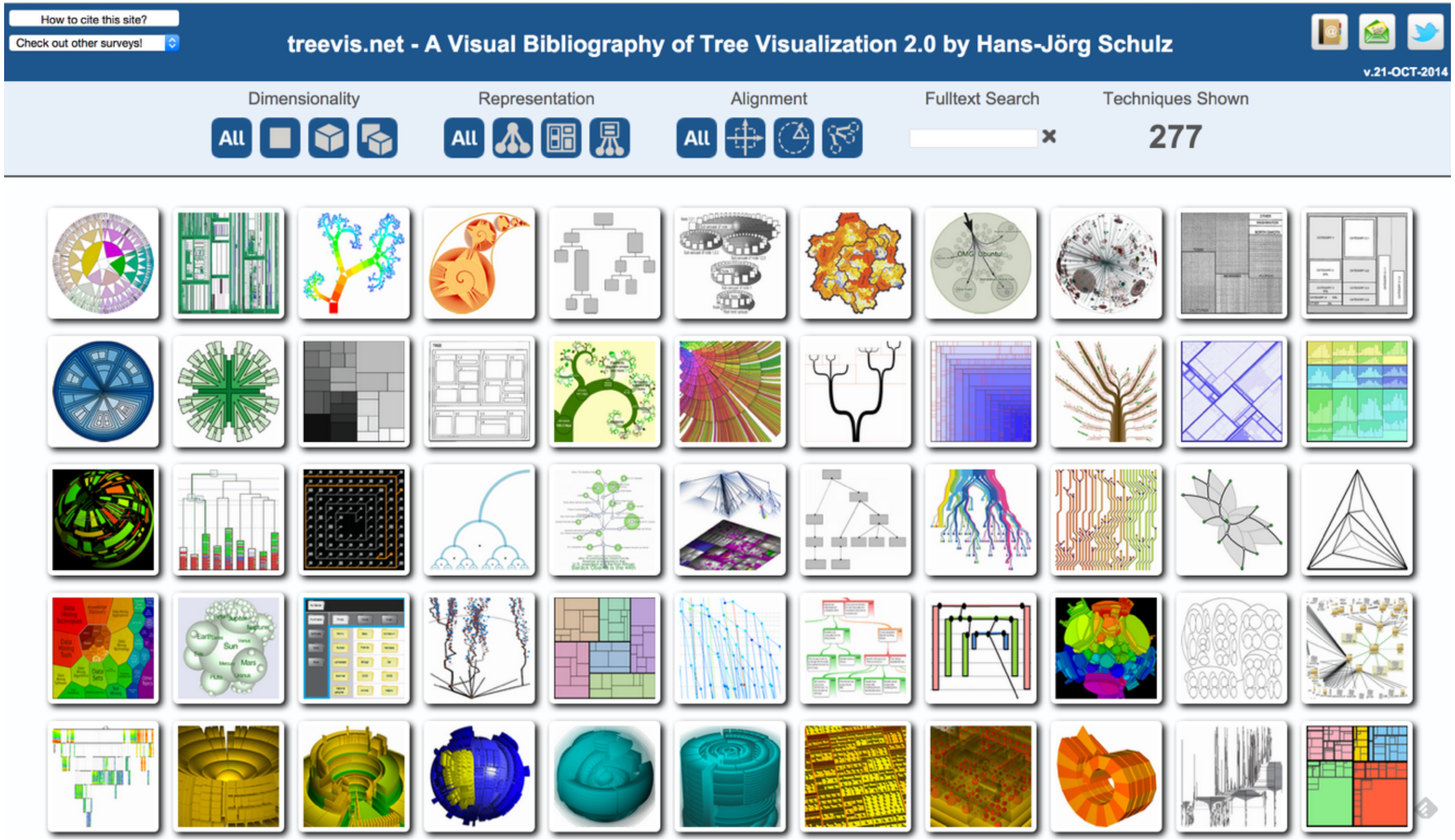
Fekete et al. 2003



Holten 2006

with edge bundling

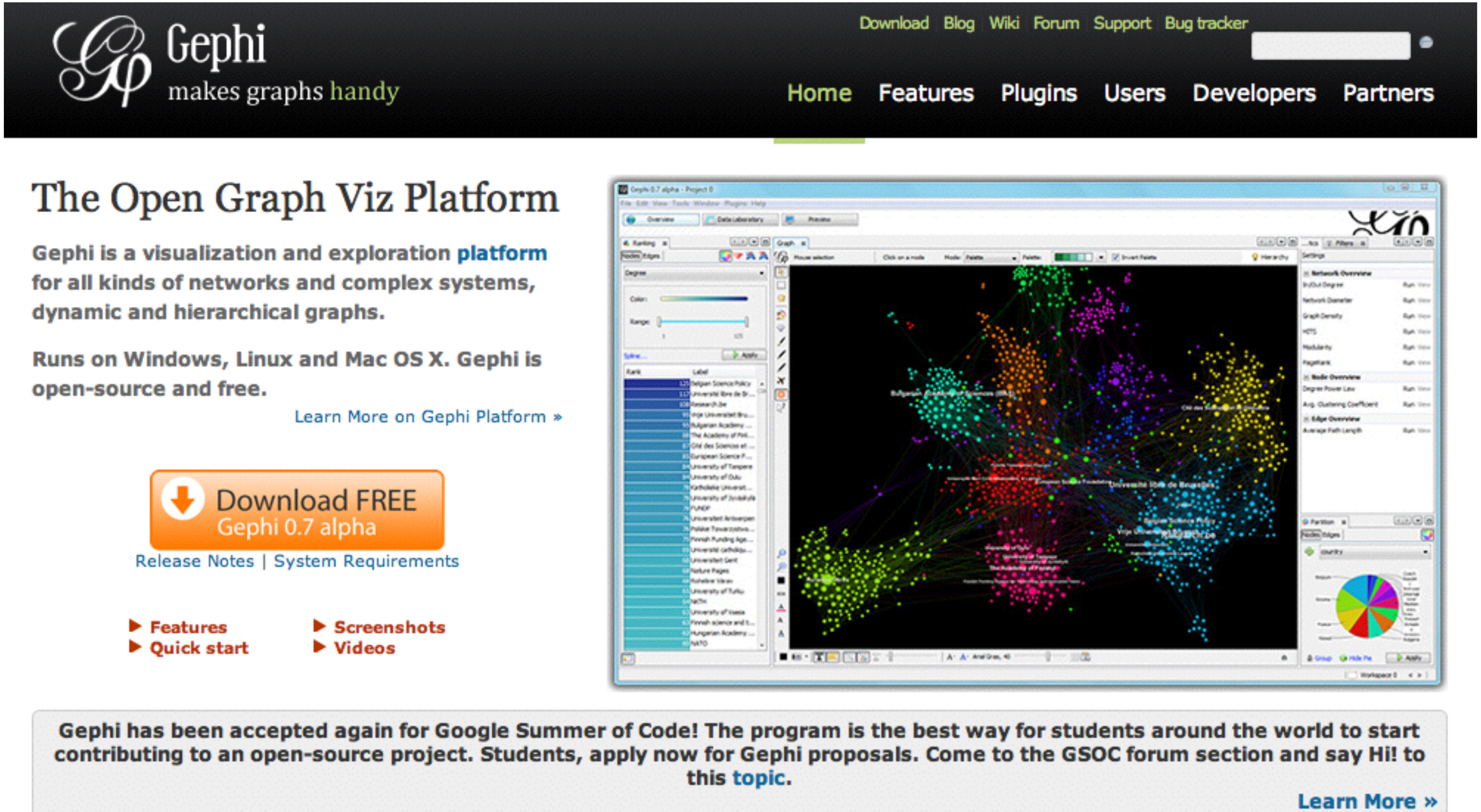
Tree Visualization Reference



Graph Tools & Applications

Gephi

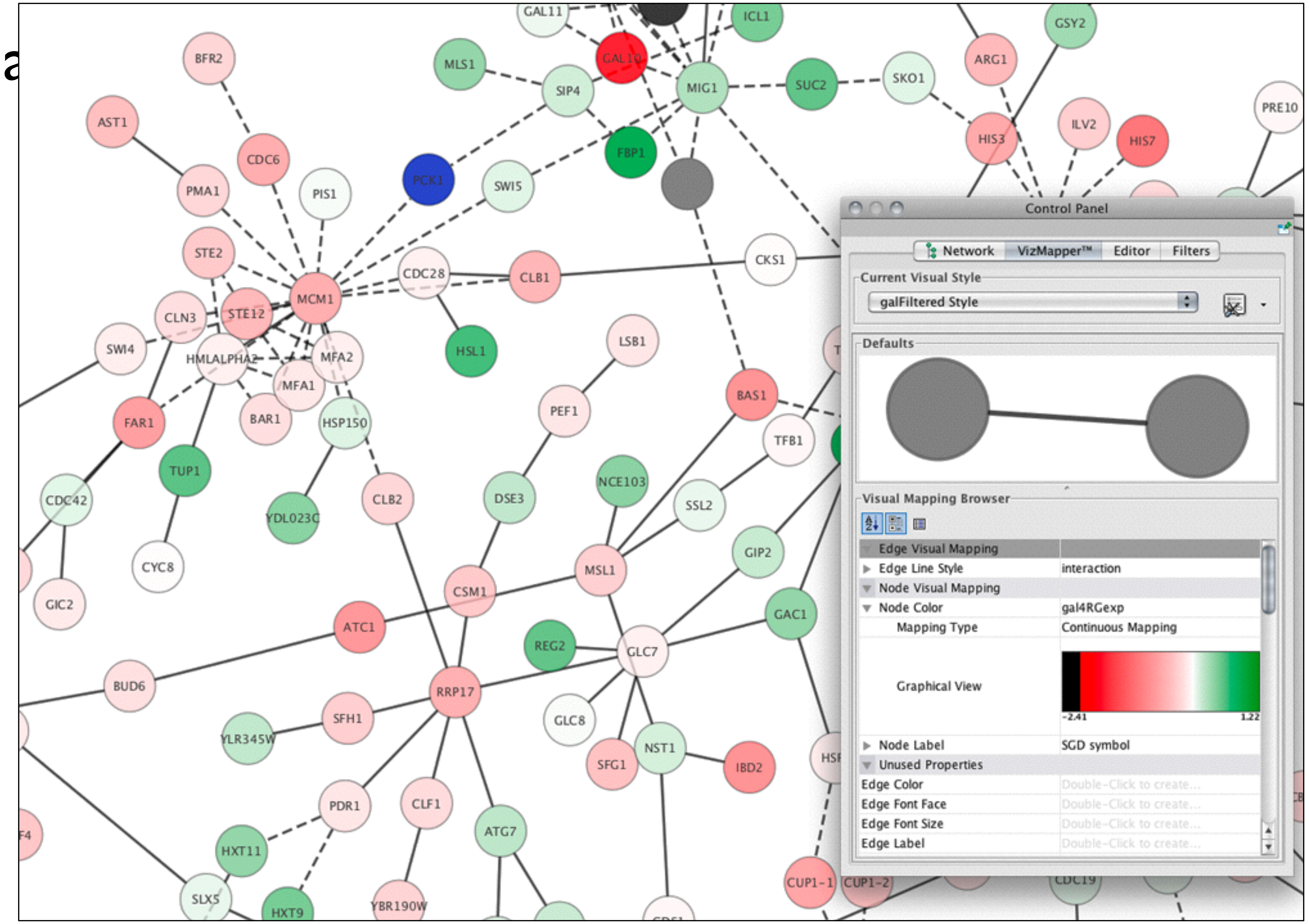
http://gephi.org



Cytoscape

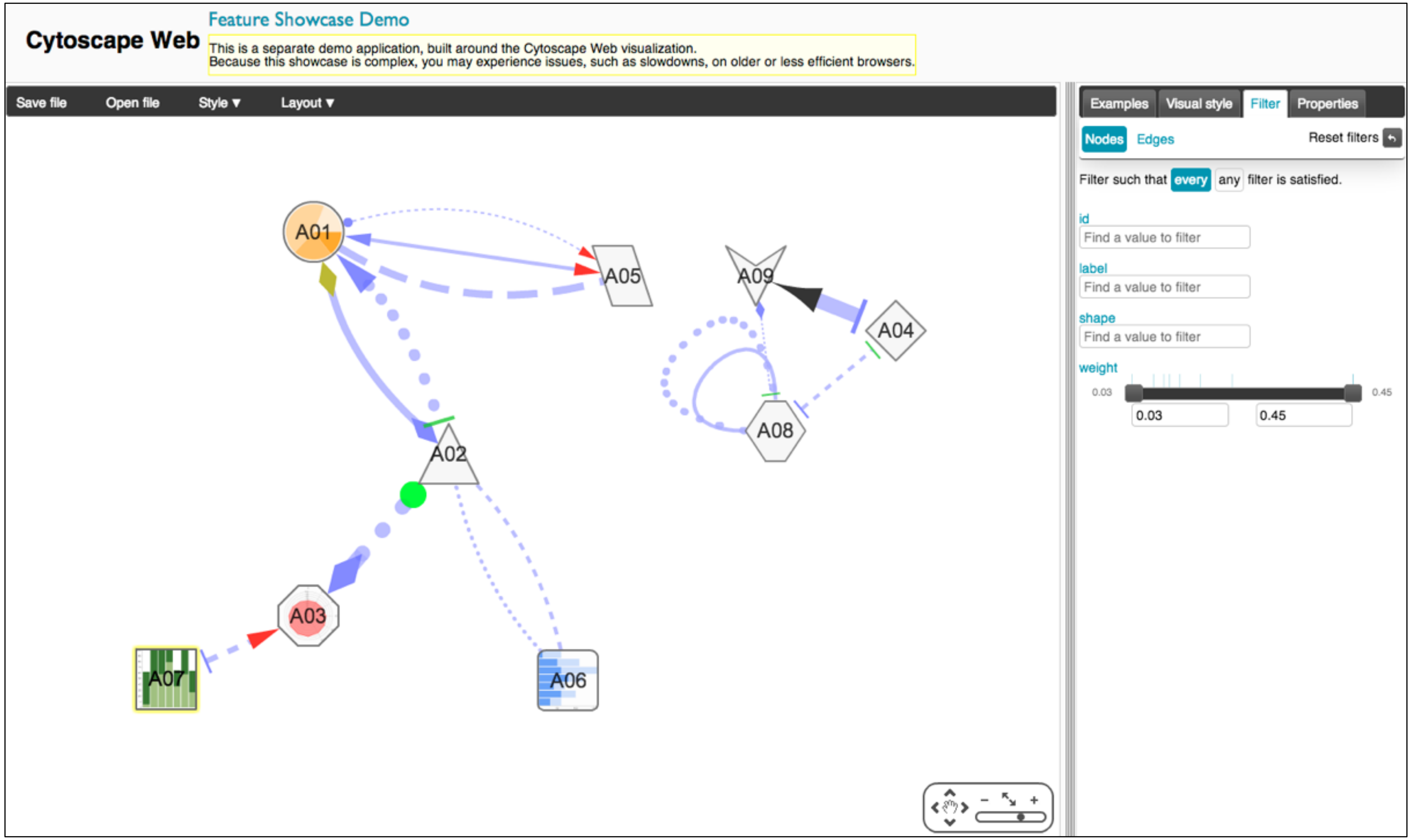
http://www.cytoscape.org/

Open source pla



Cytoscape Web

http://cytoscapeweb.cytoscape.org/



NetworkX

https://networkx.github.io/

NetworkX

NetworkX Home | Documentation | Download | Developer (Github)

High-productivity software for complex networks

NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Documentation
all documentation

Examples
using the library

Reference
all functions and methods

Features

- Python language data structures for graphs, digraphs, and multigraphs.
- Nodes can be "anything" (e.g. text, images, XML records)
- Edges can hold arbitrary data (e.g. weights, time-series)
- Generators for classic graphs, random graphs, and synthetic networks
- Standard graph algorithms
- Network structure and analysis measures
- Open source BSD license
- Well tested: more than 1800 unit tests, >90% code coverage
- Additional benefits from Python: fast prototyping, easy to teach, multi-platform

Versions

Latest Release

1.8.1 - 4 August 2013
downloads | docs | pdf

Development

1.9dev
github | docs | pdf
build passing
coverage 83%

Contact

Mailing list
Issue tracker
Developer guide