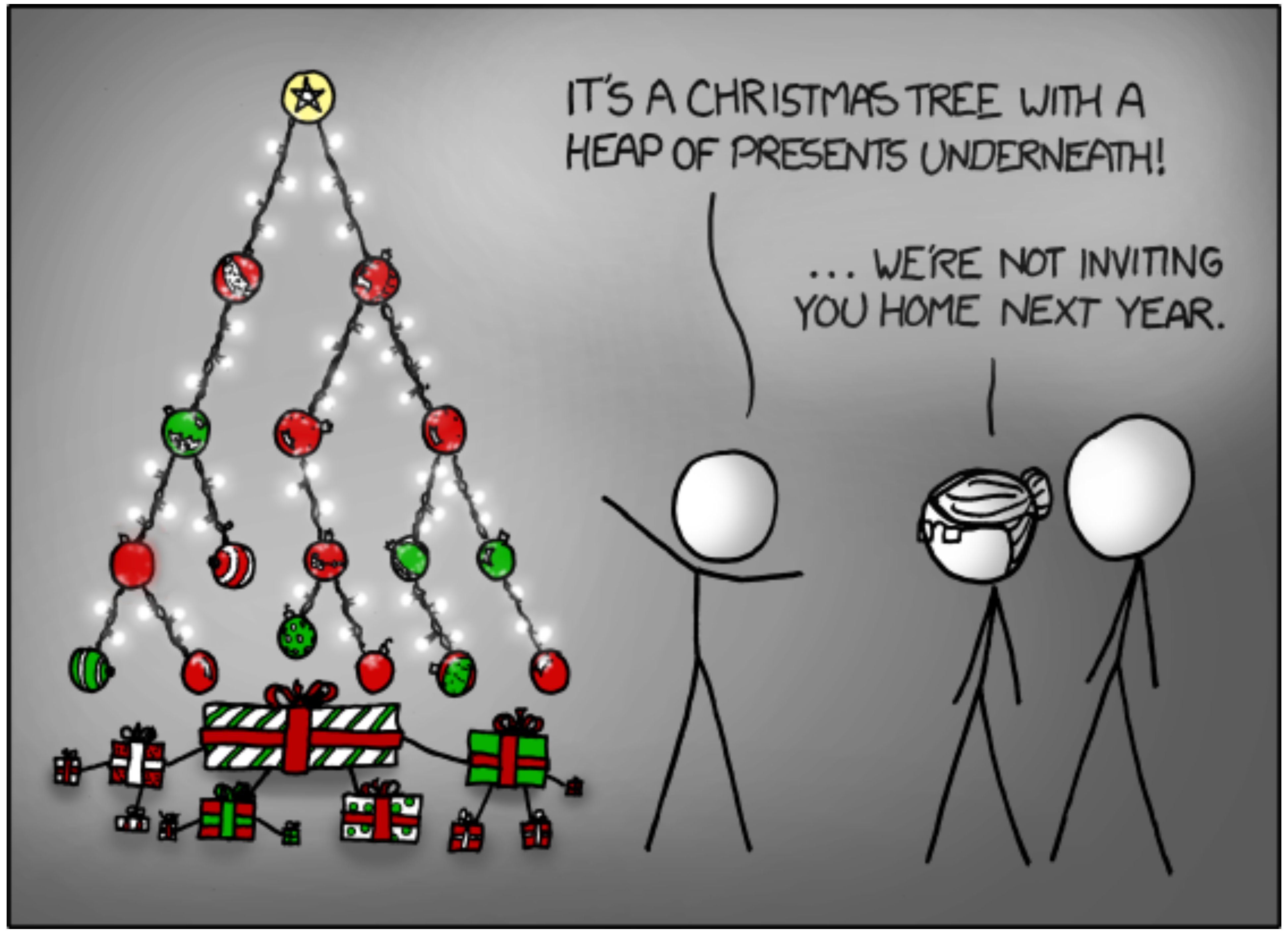
CS-5630 / CS-6630 Visualization for Data Science
Networks

Alexander Lex
alex@sci.utah.edu

THE UNIVERSITY OF UTAH



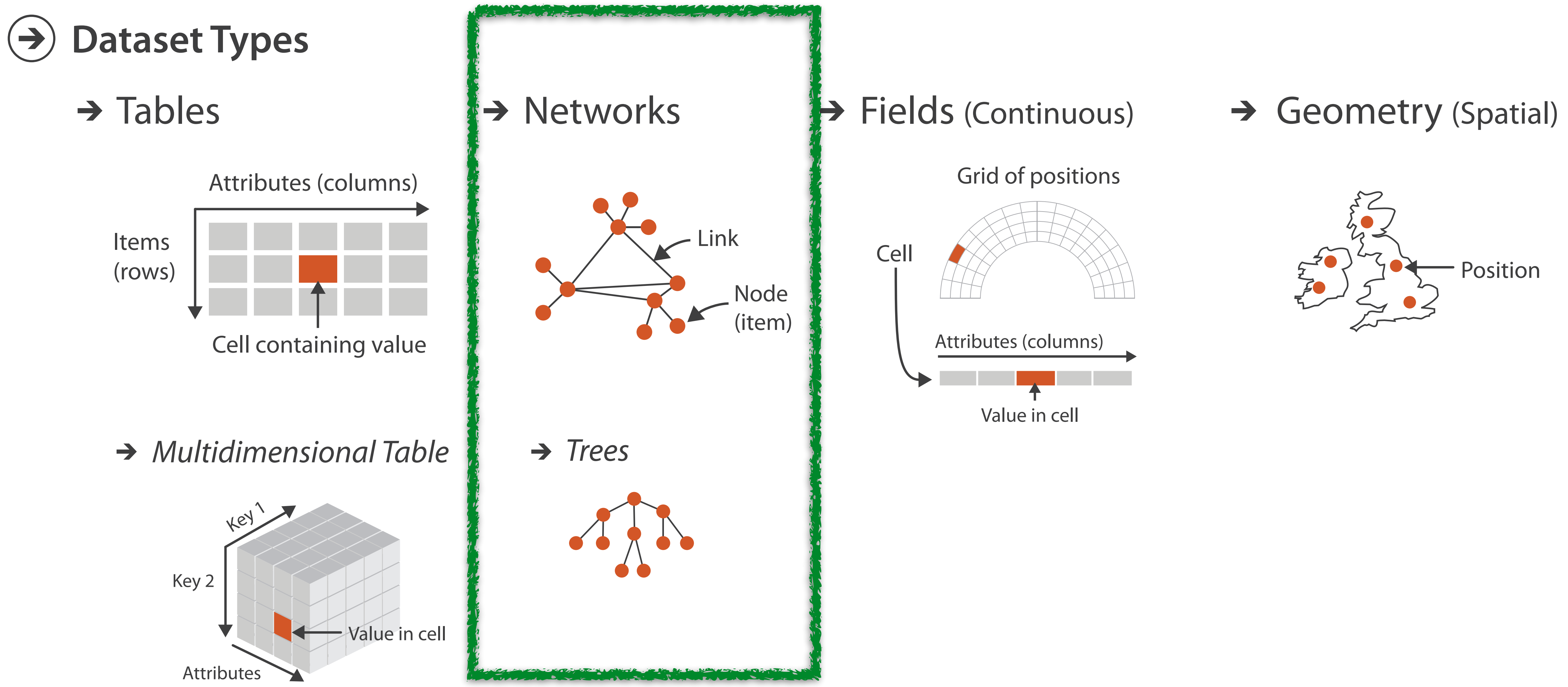
[xkcd]

Networks and Graphs

Networks model relationships between items

Network vs Graph

- Network: a specific instance
  - social network…
- Graph: the generic term
  - graph theory…



Network Exercise

Nodes and Node Attributes

Author (# papers)

Carolina (6),

Miriah (42)

Alex (36),

Sean (8),

Marc (40)

Nils (51),

Silvia (110)

Links and Link Attributes

Co-author, co-author - # joint papers

Carolina, Alex - 2

Sean, Miriah - 7

Miriah, Alex - 2

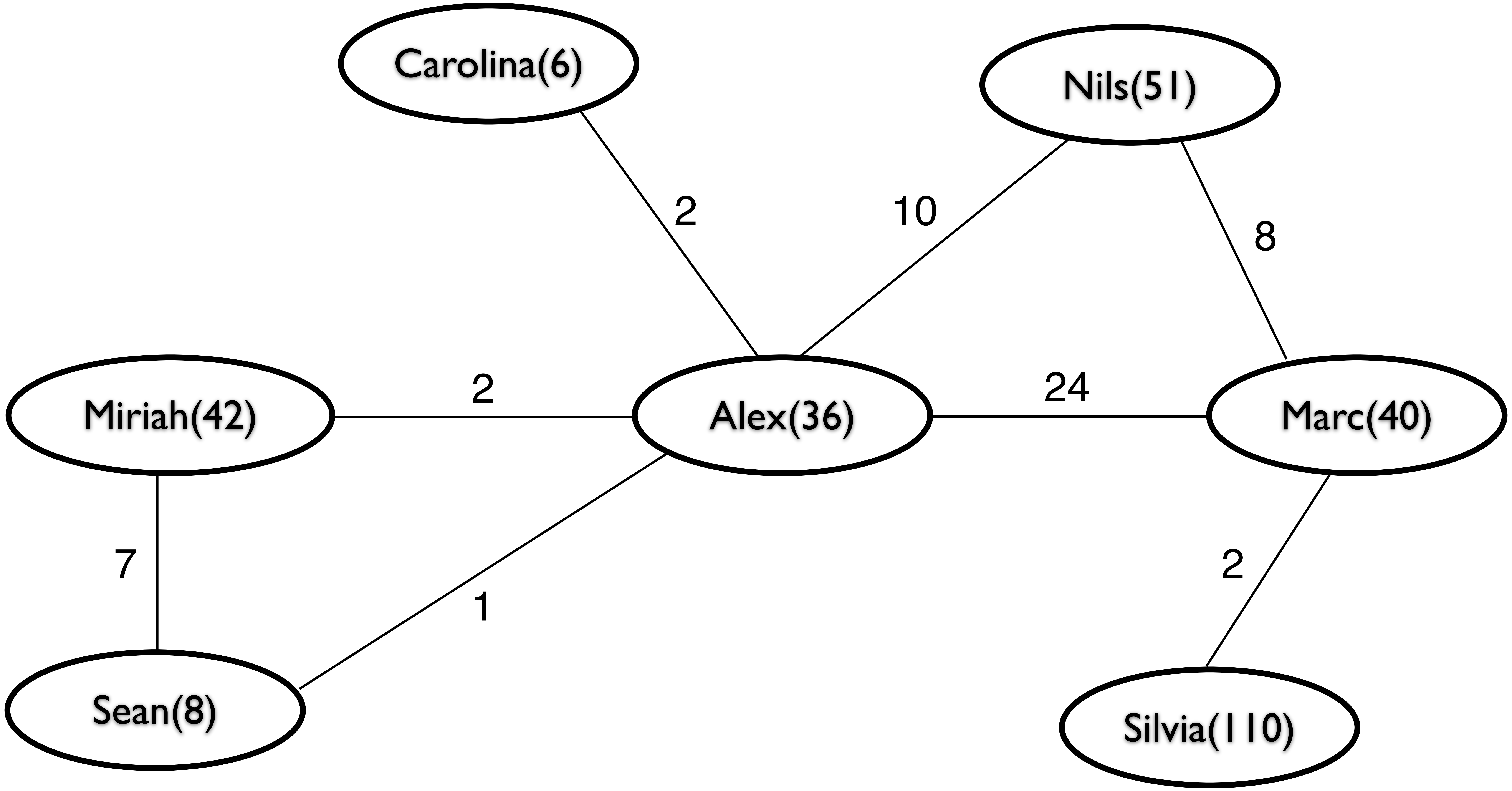
Alex, Sean - 1

Alex, Nils - 10

Alex, Marc - 24

Marc, Silvia - 1

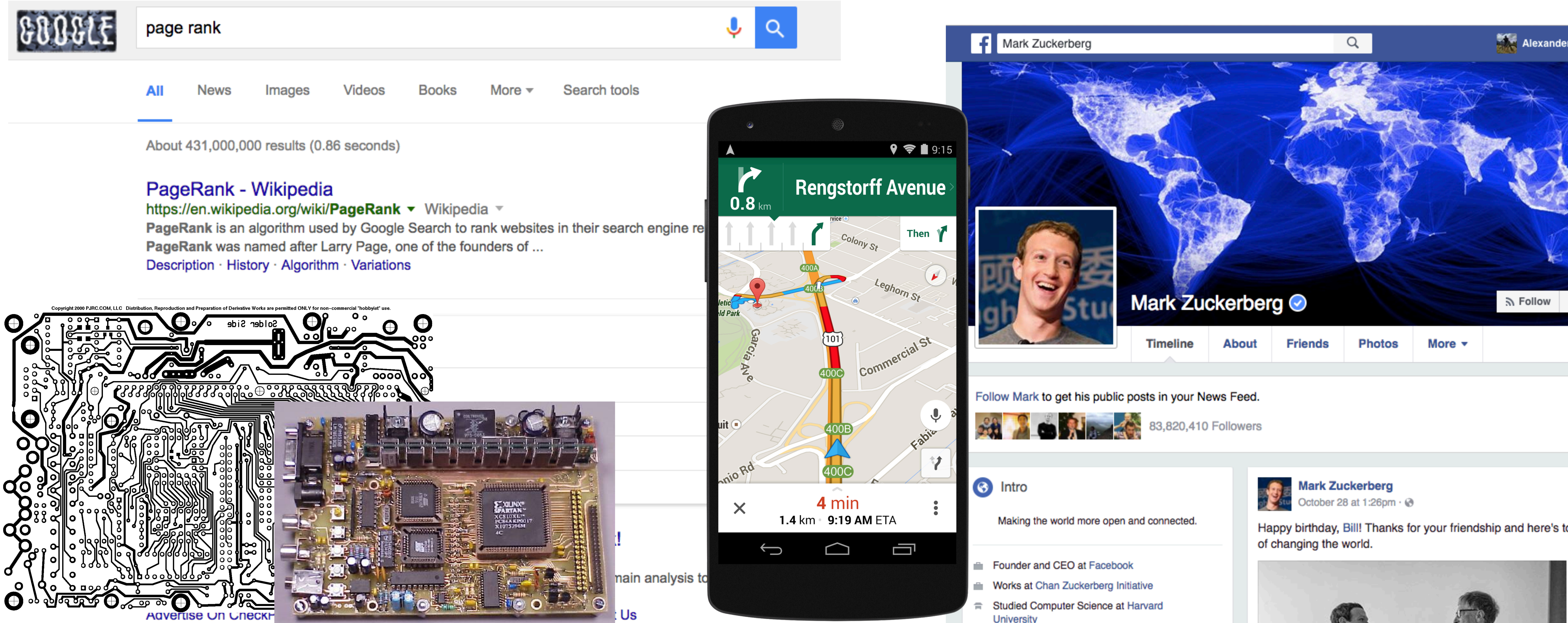
Marc, Nils - 8


Carolina(6)
Nils(51)
2
10
8
Miriah(42)
2
Alex(36)
24
Marc(40)
7
1
2
Sean(8)
Silvia(110)

	Carolina (6)	Miriah (42)	Alex (36)	Sean (8)	Marc (40)	Nils (51)	Silvia (110)
Carolina (6)			2				
Miriah (42)			2	7			
Alex (36)	2	2		1	14	10	
Sean (8)		7	1				
Marc (40)			14			8	1
Nils (51)			10		8		
Silvia (110)					1		

Applications of Networks

Without graphs, there would be none of these:




facebook
December 2010

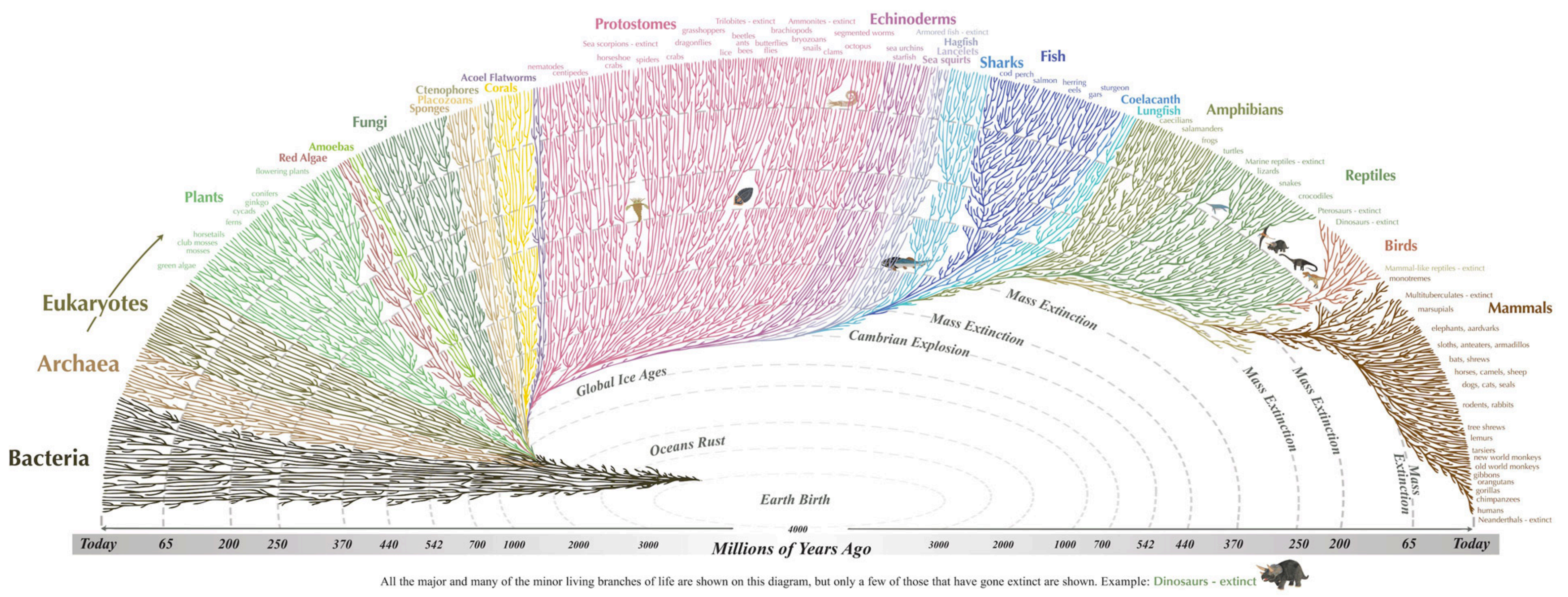
Biological Networks

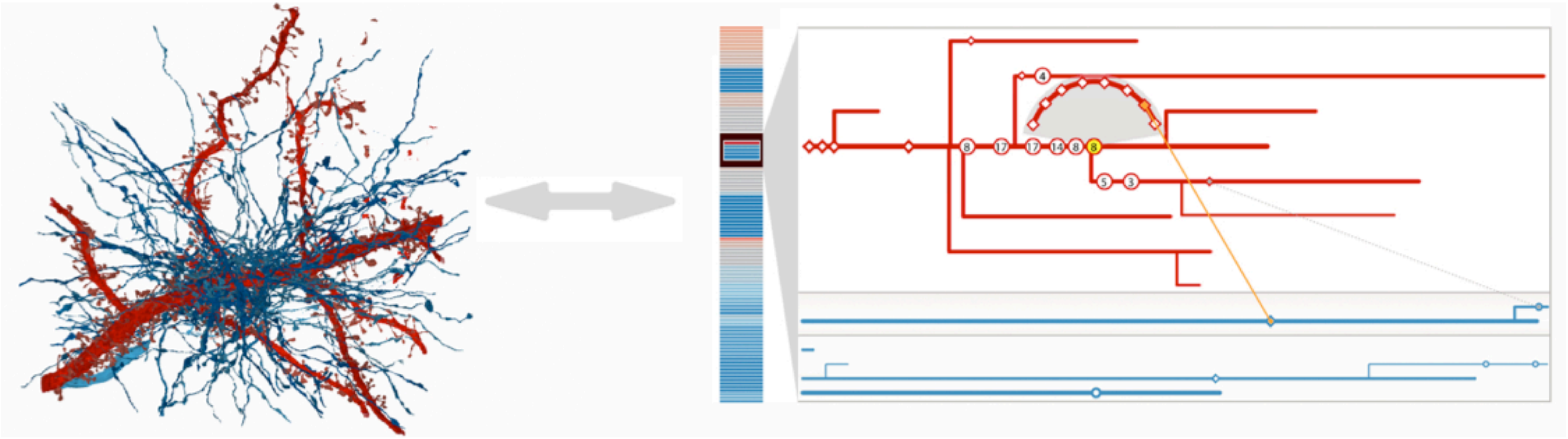
Interaction between genes, proteins and chemical products

The brain: connections between neurons

Your ancestry: the relations between you and your family

Phylogeny: the evolutionary relationships of life



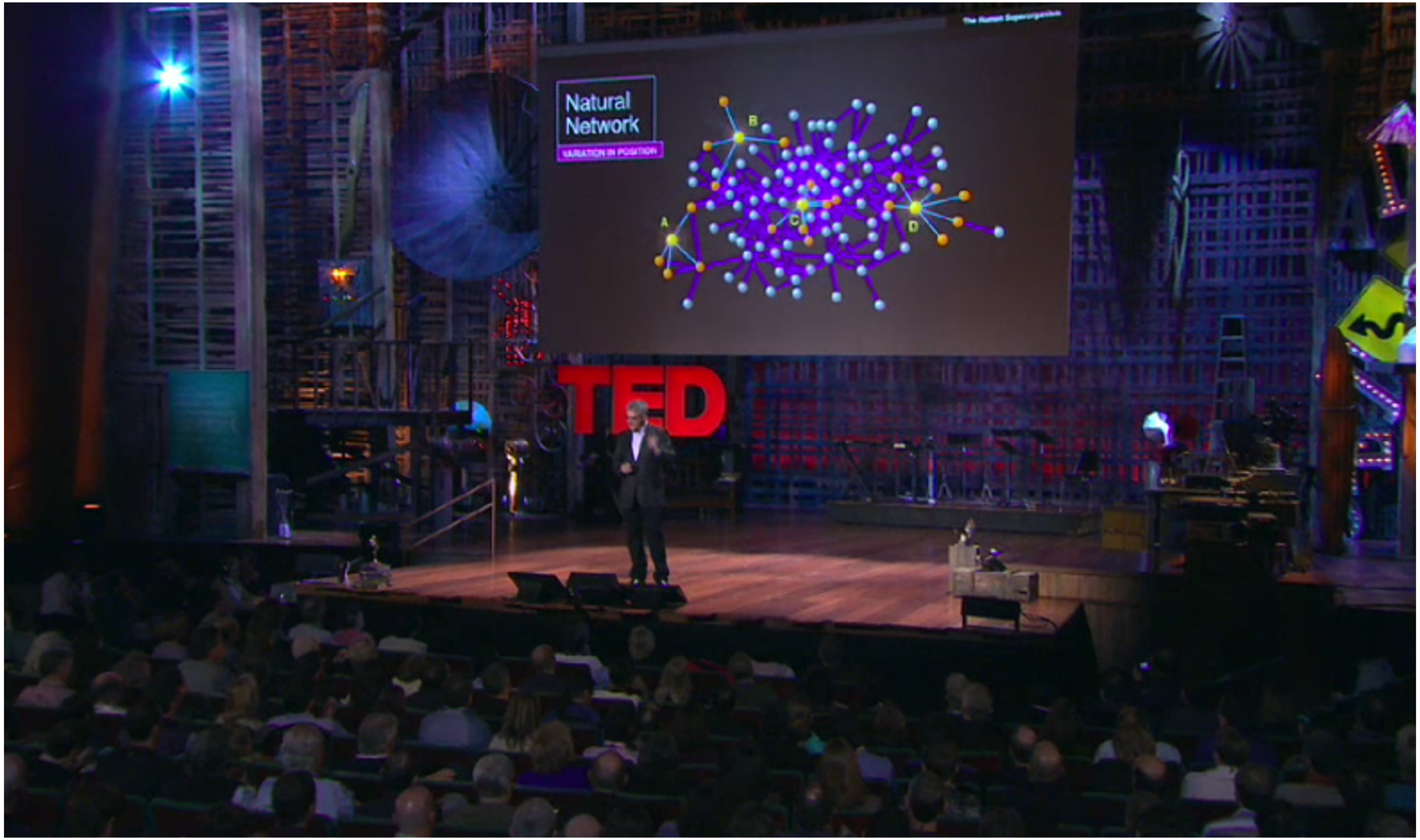


[Beyer 2014]

CITRATE
ATP
MALONATE, OXALO-ACETATE
SUCCINATE
FUMARATE
SUCCINATE DEHYDROGENASE
FADH2 · E
FAD · E
ACCEPTOR (UBIQUINONE)
ACC · H2 (UBIQUINOL)
CoA-SH
SUCCINYL-CoA-HYDROLASE
PPi, ATP
H2O
ADP
Pi
GTP (ITP)
GDP (IDP)
SUCCINYL-CoA
SUCCINYL-CoA-LIGASES
PROTEINS
GLYCYL-tRNA
GLYCINE-tRNA LIGASE
ACETYL-CoA
GLYCINE
5-AMINOLEVULINATE SYNTHASE
increased in acute intermittent porph
5-AMINO-LEVULINATE SYNTHASE
PyrP
HEME
hv (in plants and photo-bacteria)
CO2
NADH + H+
FORMATE DEHYDRO-GENASE (CYTOCHROME)
FORMATE DEHYDRO-GENASE
NAD+
MOLYBDO-PTERIN
2 Cytb-Fe++
FORMYL-CoA
FORMYL-CoA HYDRO-LASE
FORMATE
H2O
CoA-SH
OXALATE DECARBOXY-LASE
OXALYL-CoA DECARBOXY-LASE
OXALATE-CoA-TRANSFERASE
FERRE-DOXIN (red)
FERRE-DOXIN (ox.)
OXO-GLUTARATE SYNTHASE
DIHYDRO-LIPOAMIDE SUCCINYL TRANS-FERASE
DIHYDRO-LIPO-AMIDE DEHY-DRO-GENASE
FAD · E
FADH2 · E
NADH + H+
Deficient in hyper-oxaluria I
PYRUVATE
ALANINE-GLYOXYLATE TRANS-AMINASE
NADH

Michal 2000

Graph Analysis Case Study

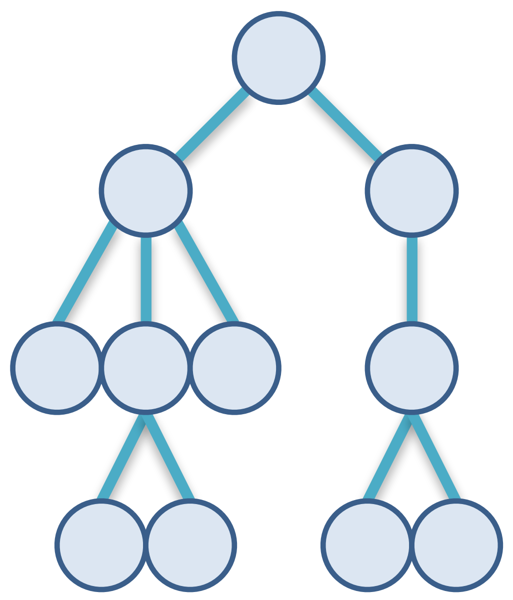


Graph Theory Fundamentals

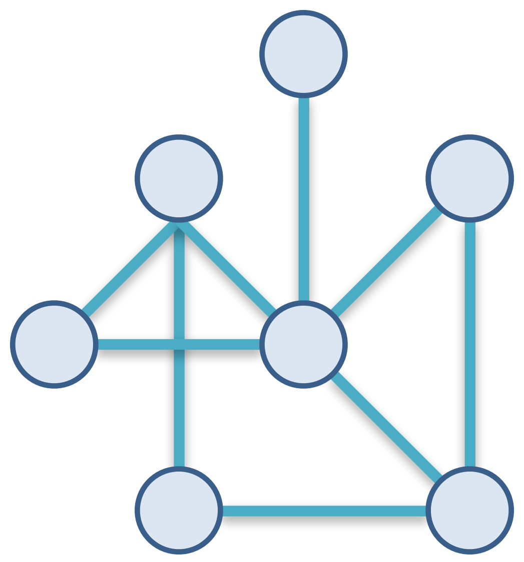
See also "Network Science", Barabasi

http://barabasi.com/networksciencebook/chapter/2

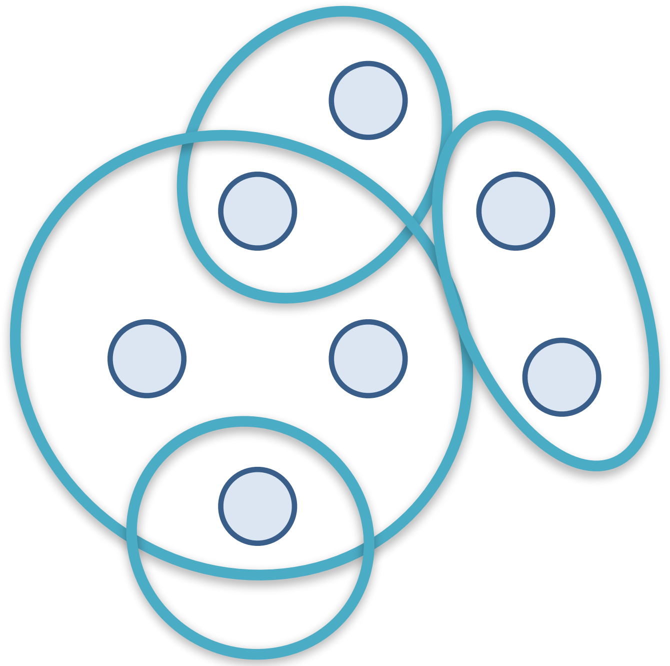
Tree



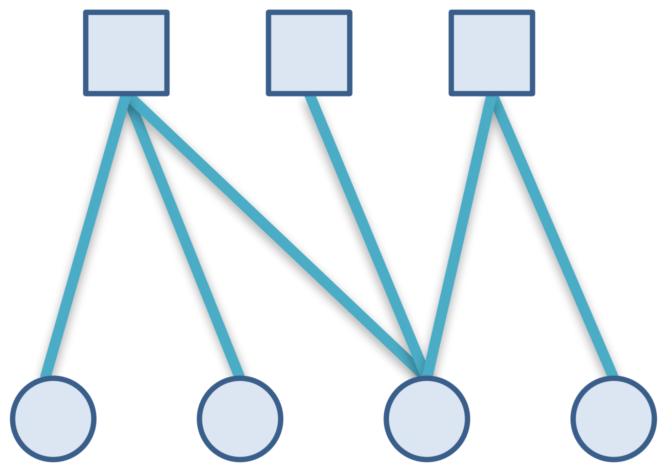
Network



Hypergraph

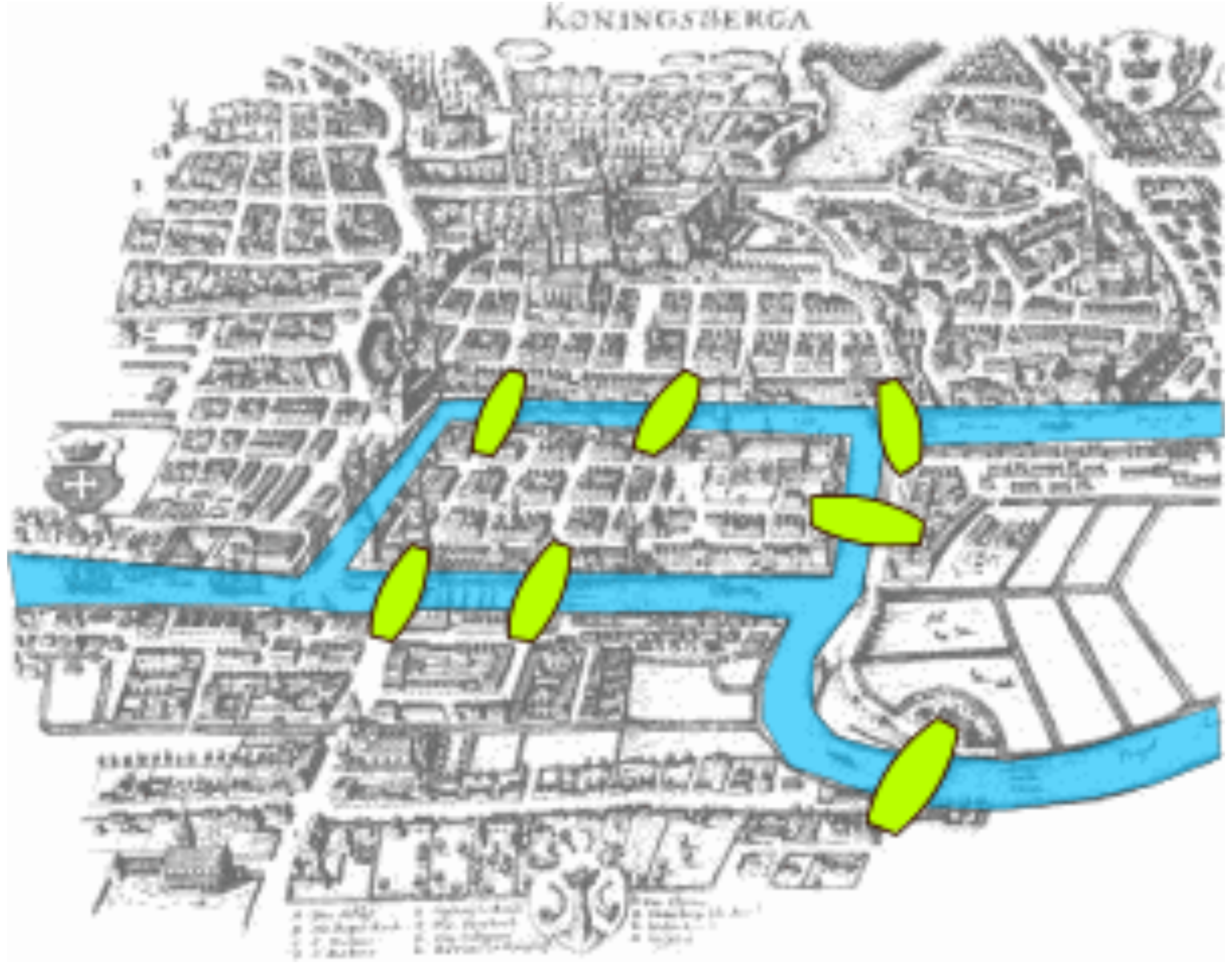


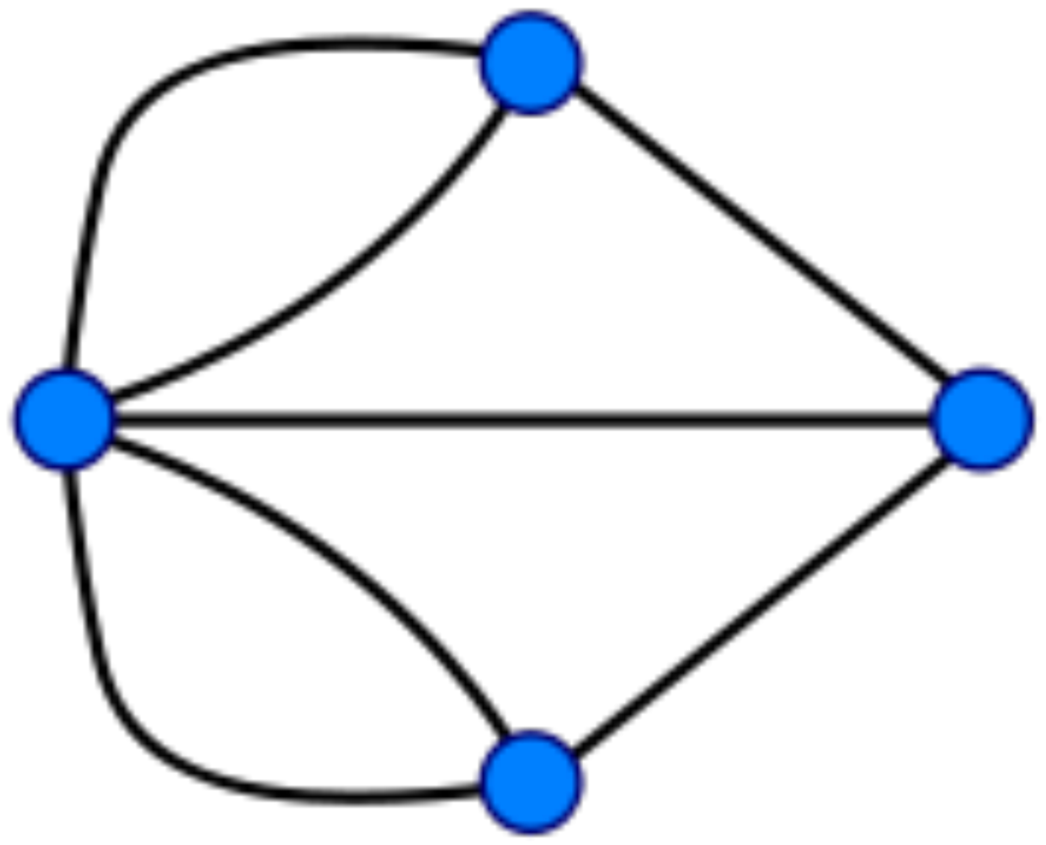
Bipartite Graph

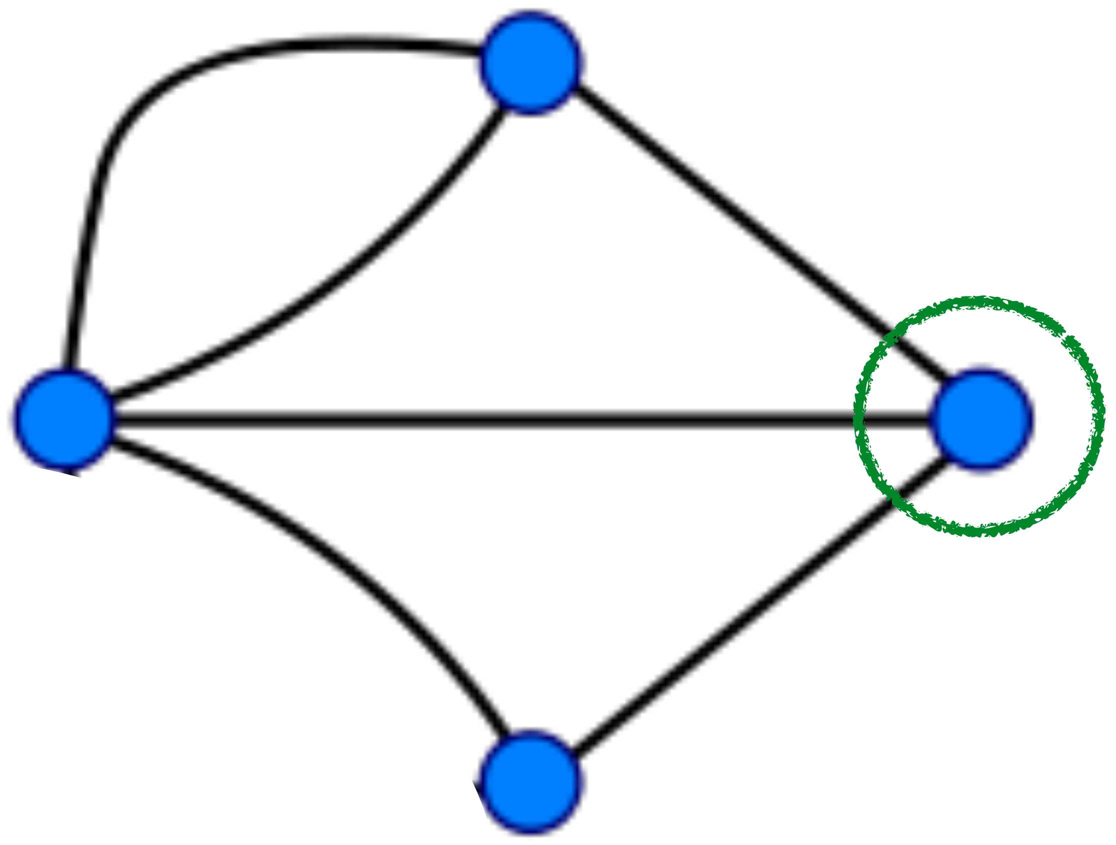


§

Now Kaliningrad: historically German, now a Russian exclave
Can you take a walk and visit every land mass without crossing a bridge twice?







Leonhard Euler:
Only possible with a graph with at most two nodes with an odd number of links.
This graph has four nodes (all) with odd number of links.

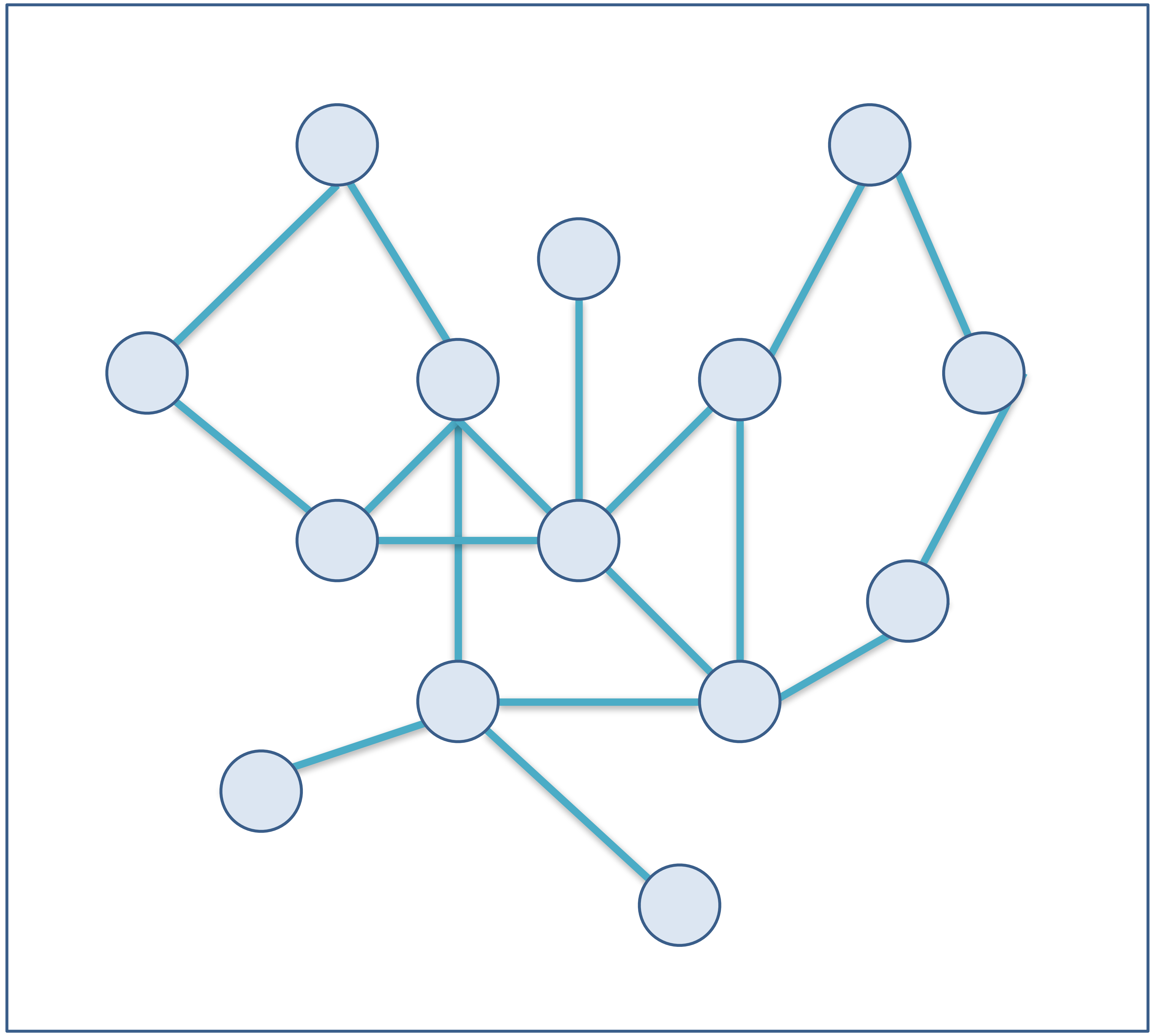
Related: a “Hamiltonian path”, i.e., a path that visits each vertex exactly once

http://barabasi.com/networksciencebook/chapter/2#bridges

Graph Terms

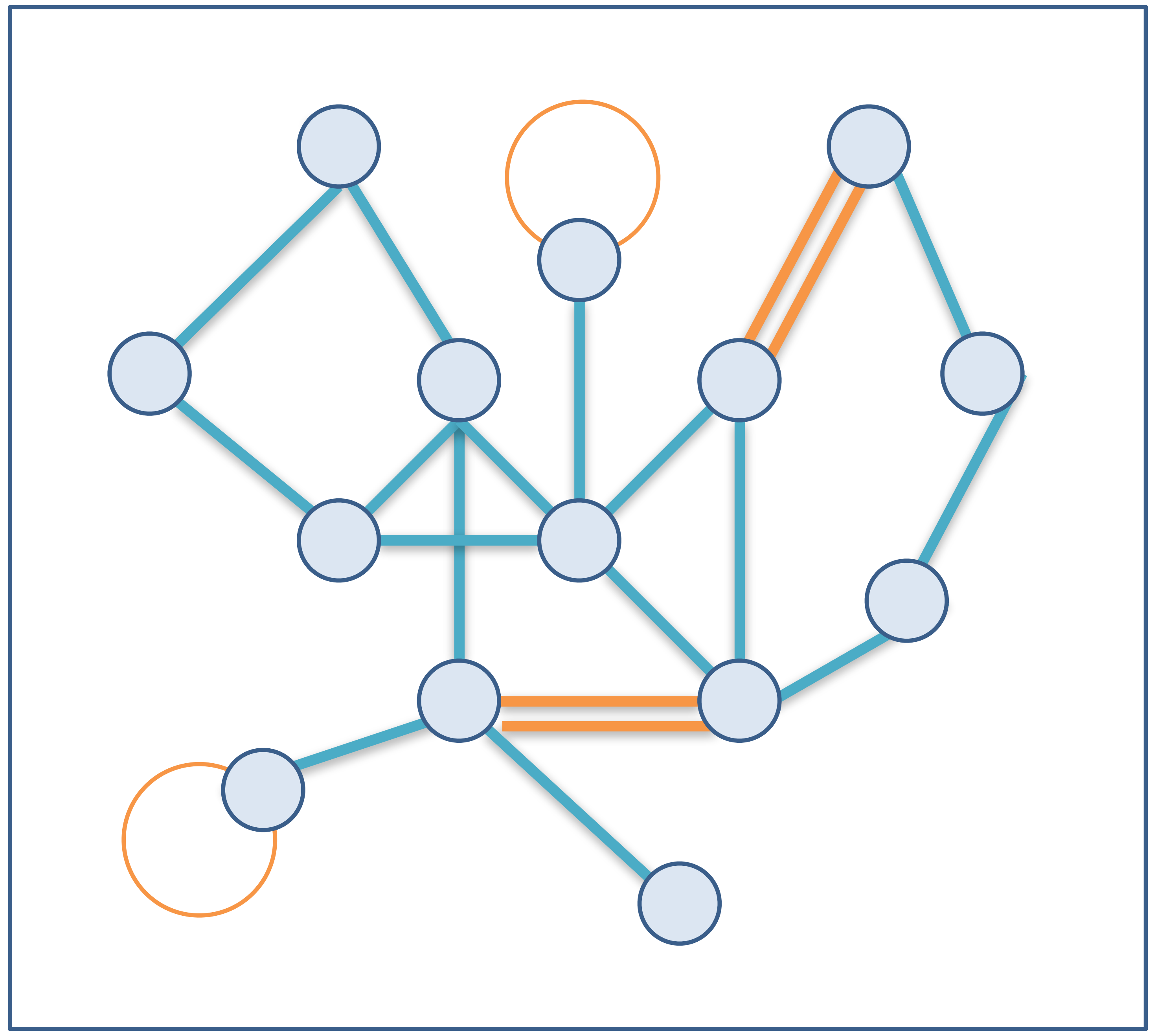
A graph **G(V,E)** consists of a set of **vertices V** (also called nodes) and a

set of **edges E** (also called links) connecting these vertices.



Graph Term: Simple Graph

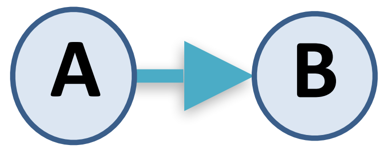
A simple graph G(V,E) is a graph which contains **no multi-edges** and **no loops**

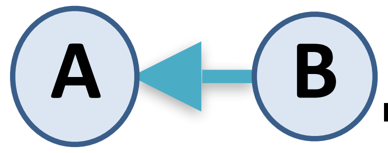


Not a simple graph!

→ A ***general graph***

Graph Term: Directed Graph

A directed graph (digraph) is a graph that discerns between the edges 

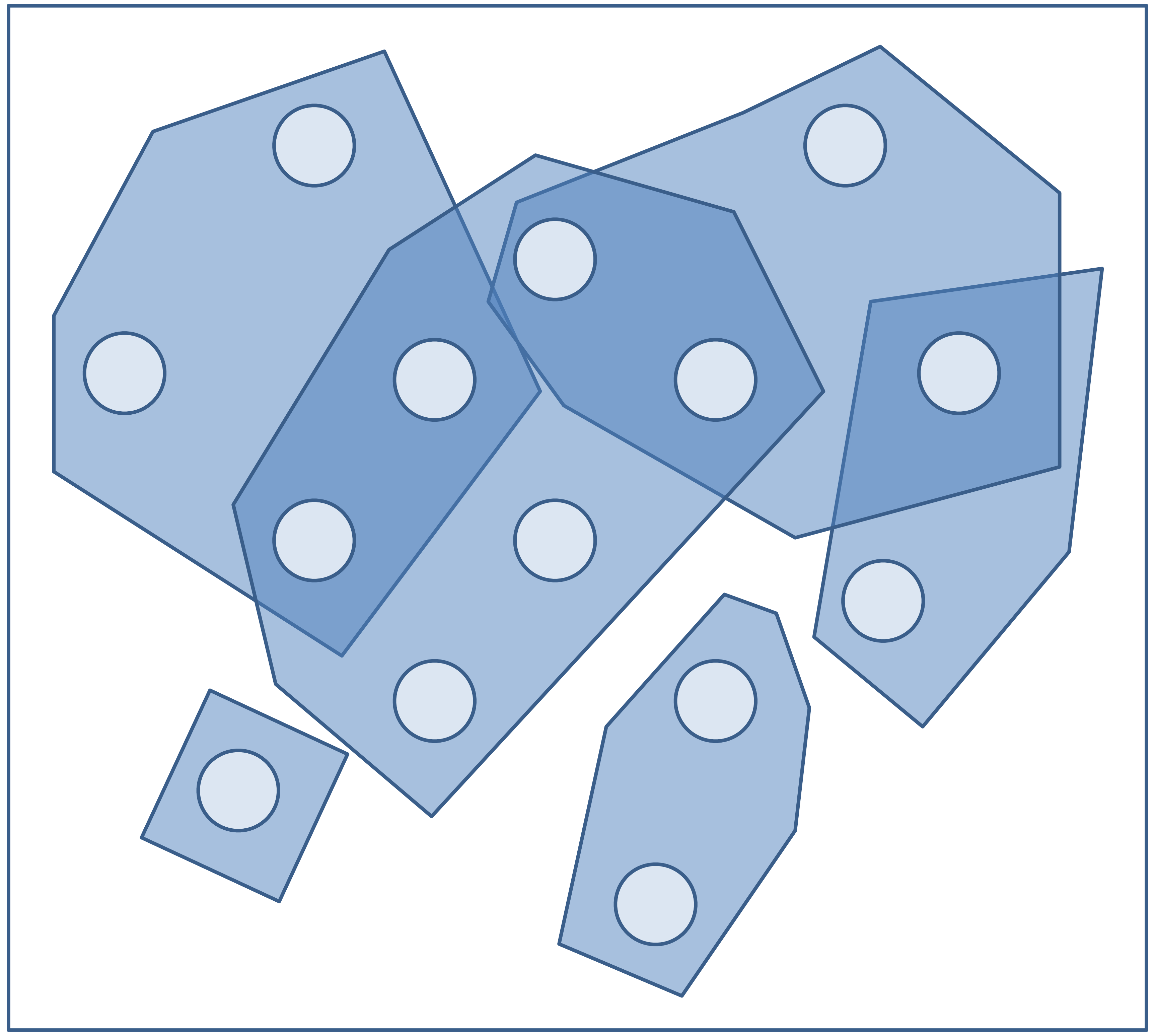
and 

.

Graph Terms: Hypergraph

A hypergraph is a graph with edges connecting any number of vertices.

Think of edges as sets.

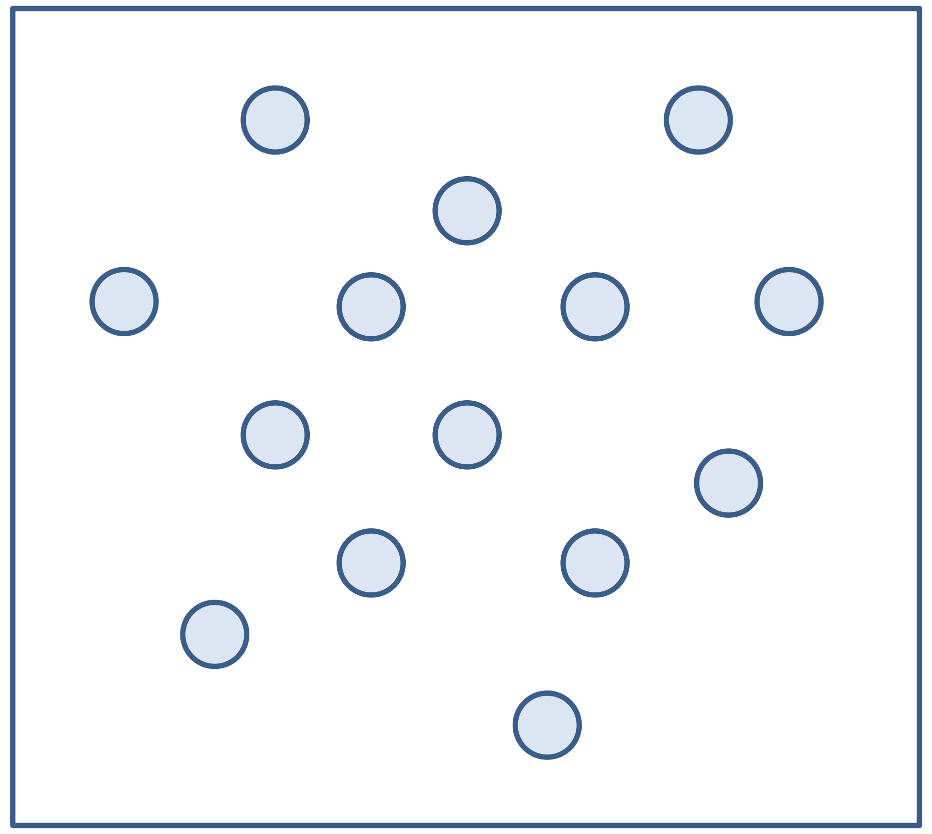


Hypergraph Example

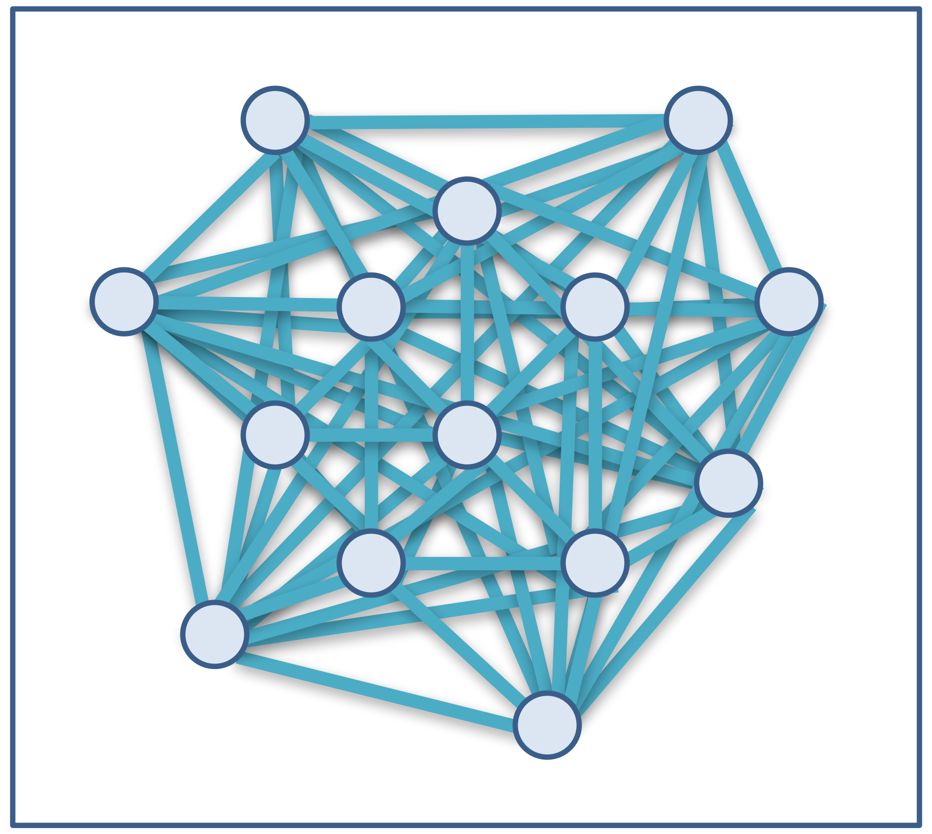
Graph Terms

Independent Set
G contains no edges

Clique
G contains all possible edges



Independent Set

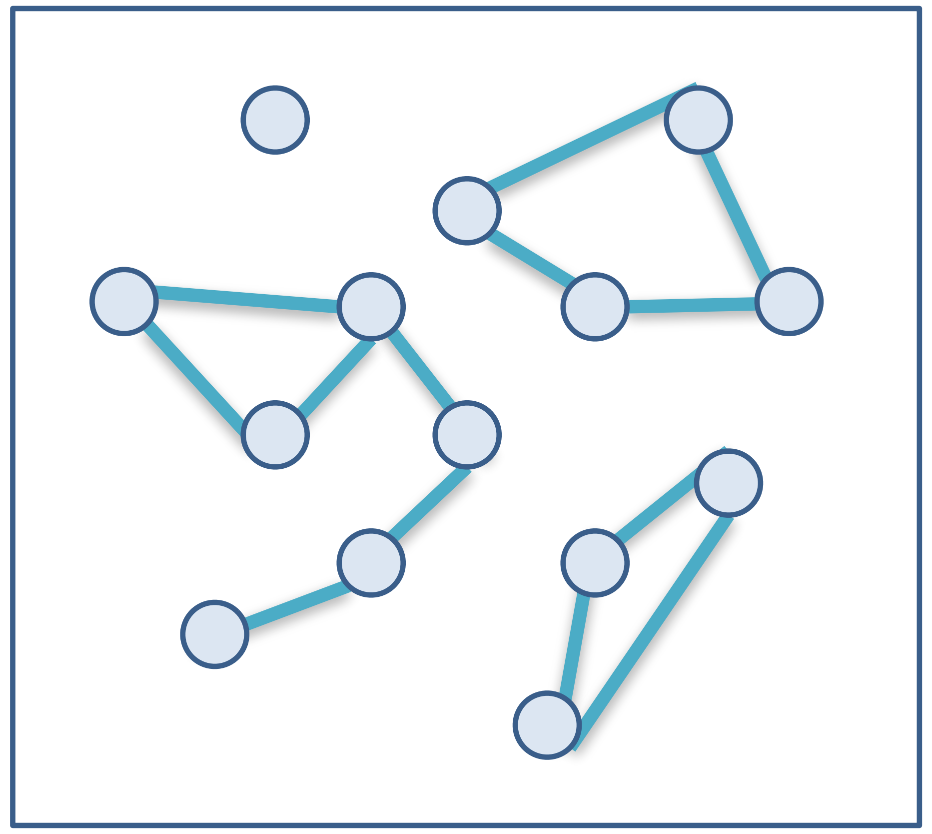


Clique

Unconnected Graphs, Articulation Points

Unconnected graph

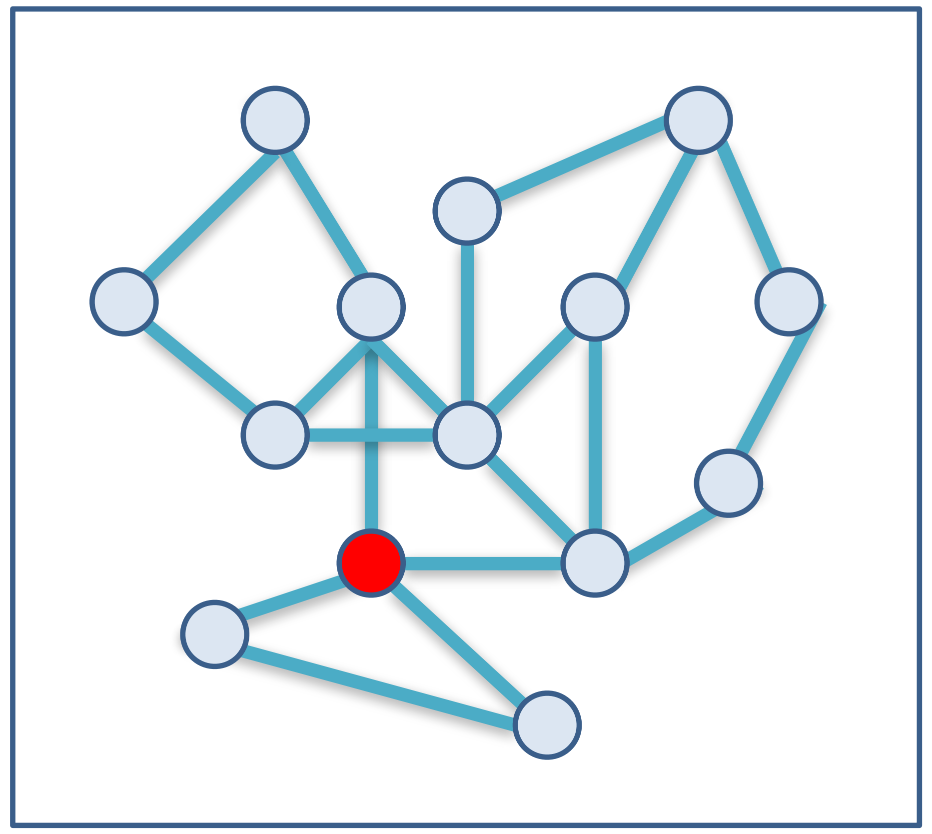
An edge traversal starting from a given vertex cannot reach any other vertex.



Unconnected Graph

Articulation point

Vertices, which if deleted from the graph, would break up the graph in multiple sub-graphs.

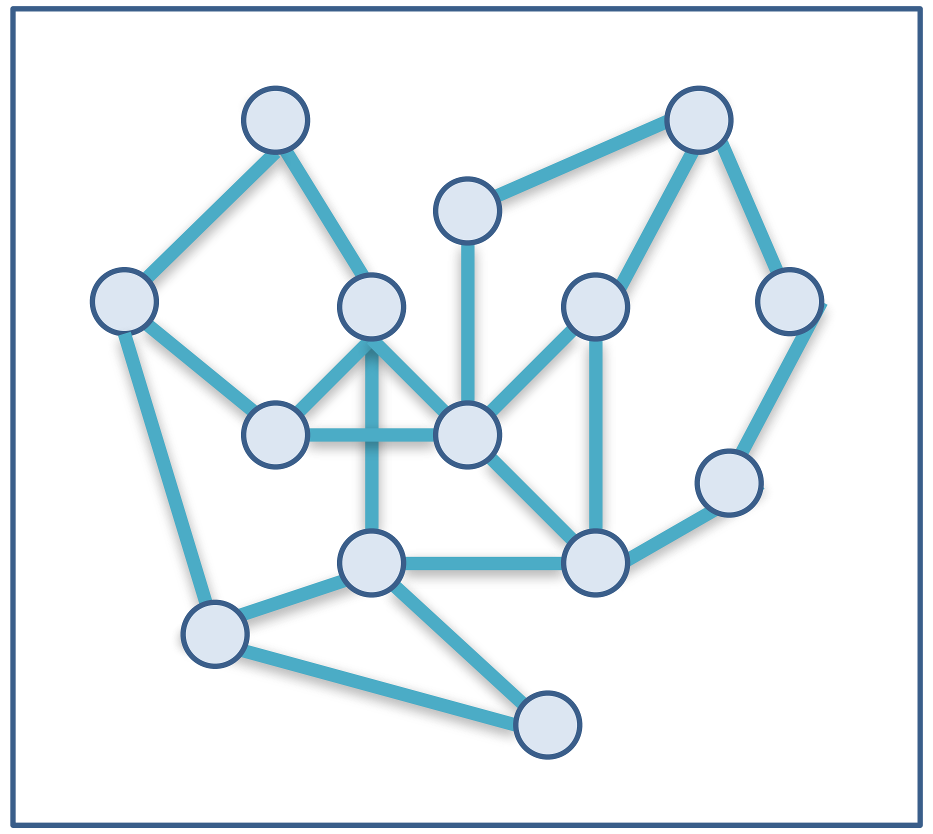


Articulation Point (red)

Biconnected, Bipartite Graphs

Biconnected graph

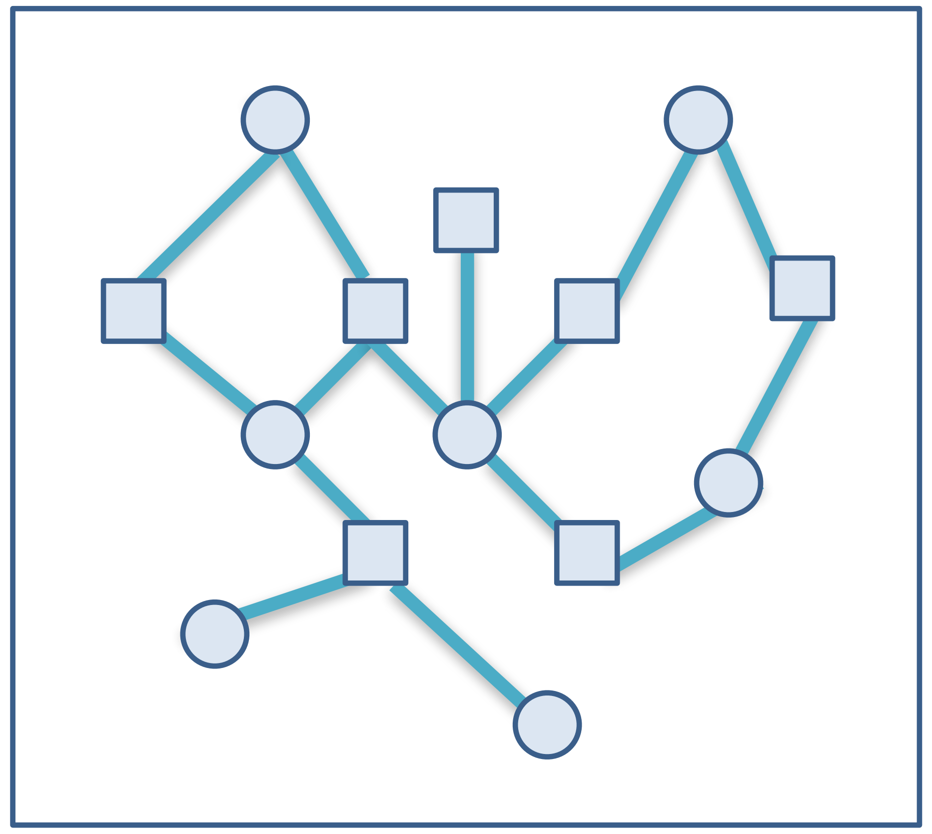
A graph without articulation points.



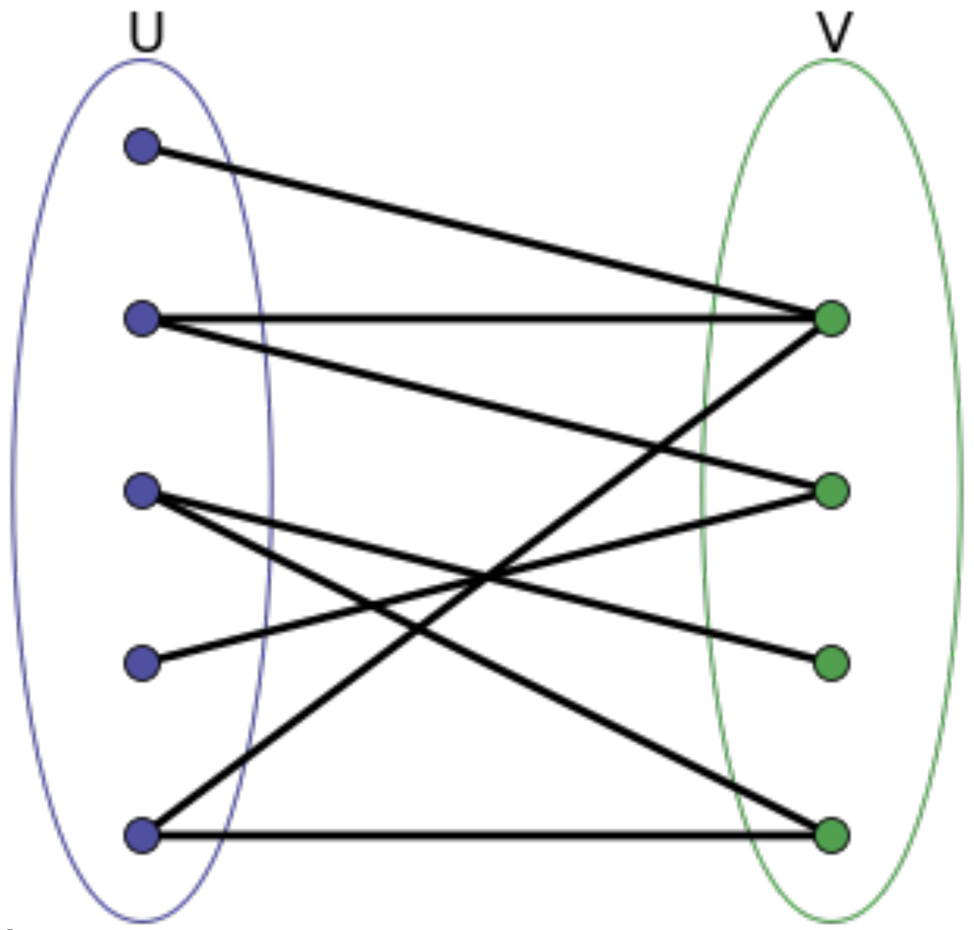
Biconnected Graph

Bipartite graph

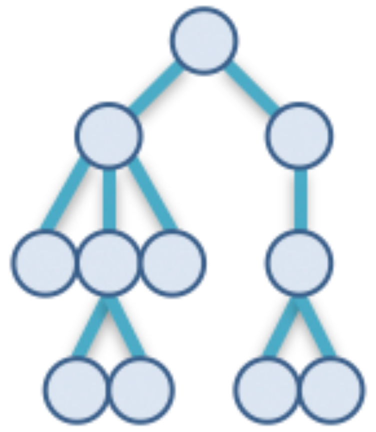
The vertices can be partitioned in two independent sets.

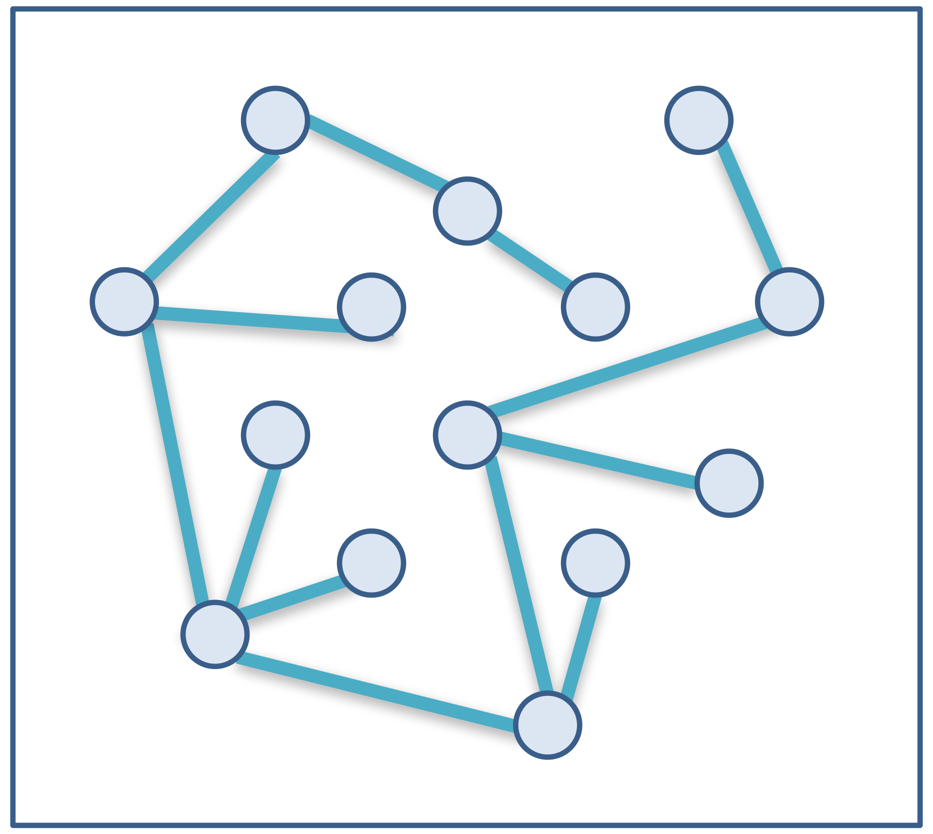


Bipartite Graph



Tree

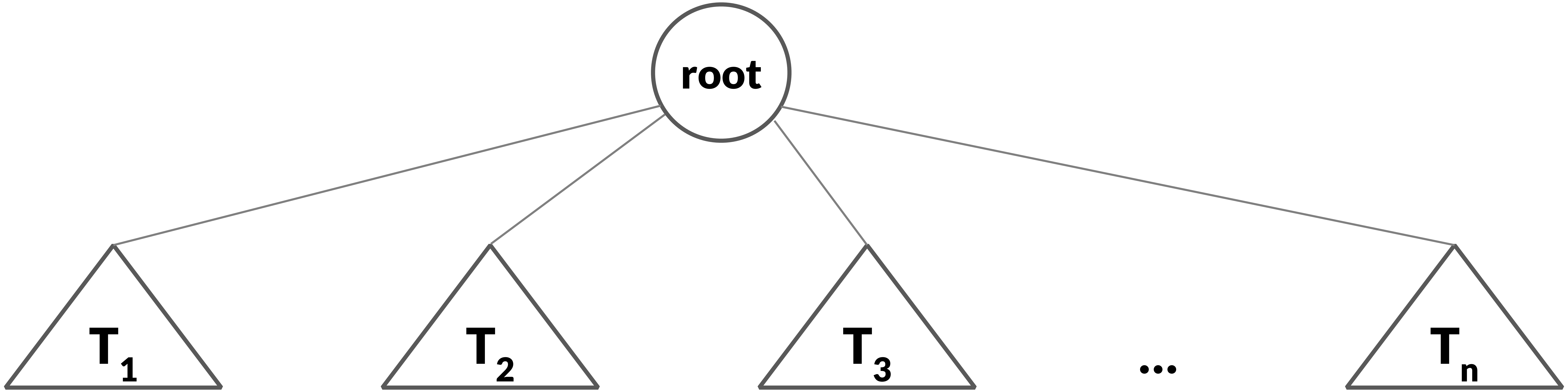




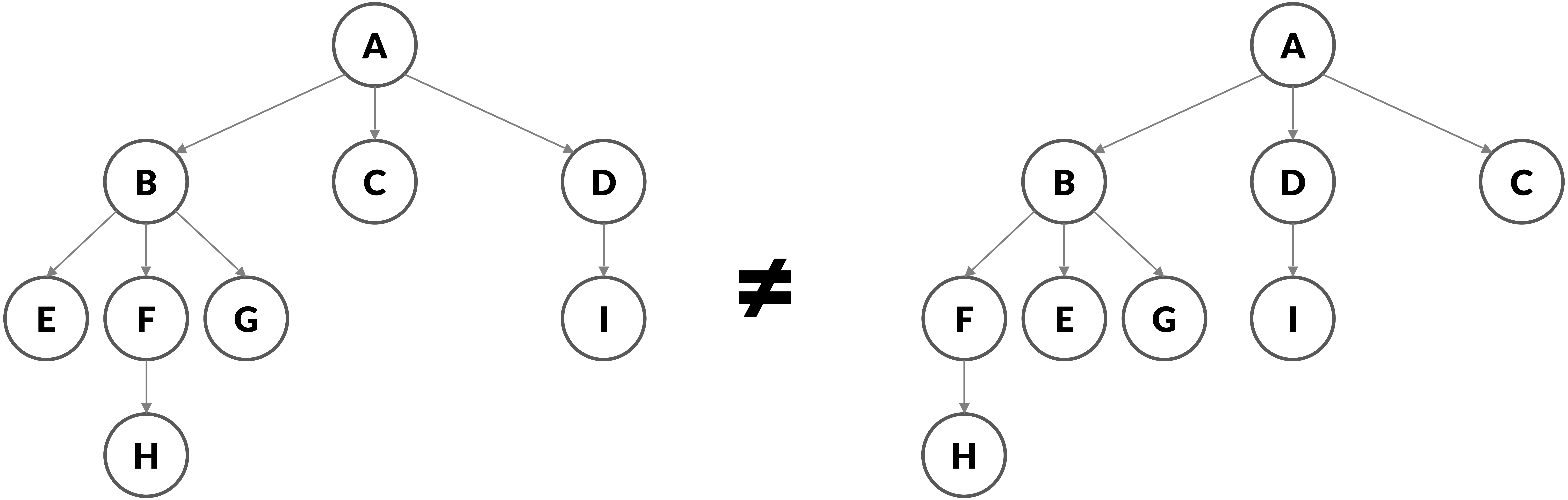
A graph with no cycles - or:

A collection of nodes

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



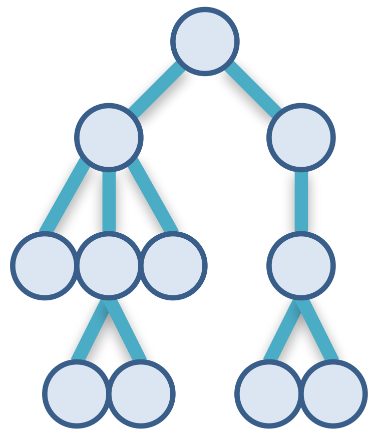
Ordered Tree



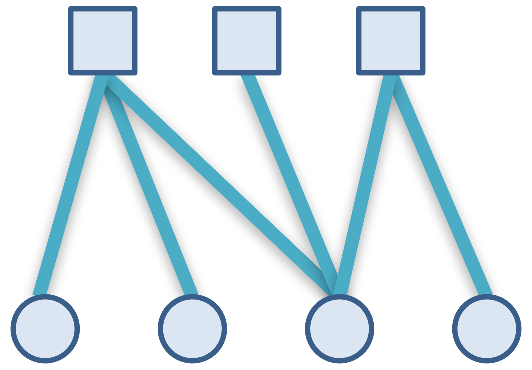
Different Kinds of Graphs

Over 1000 different graph classes

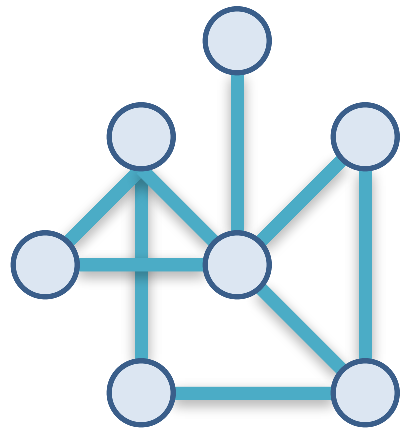
Tree



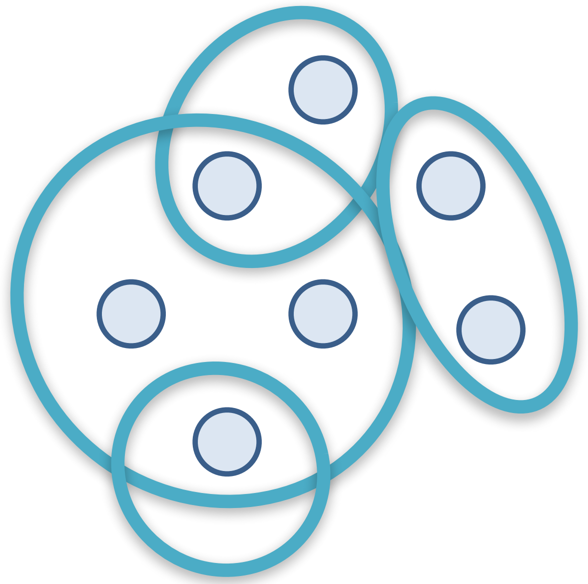
Bipartite Graph

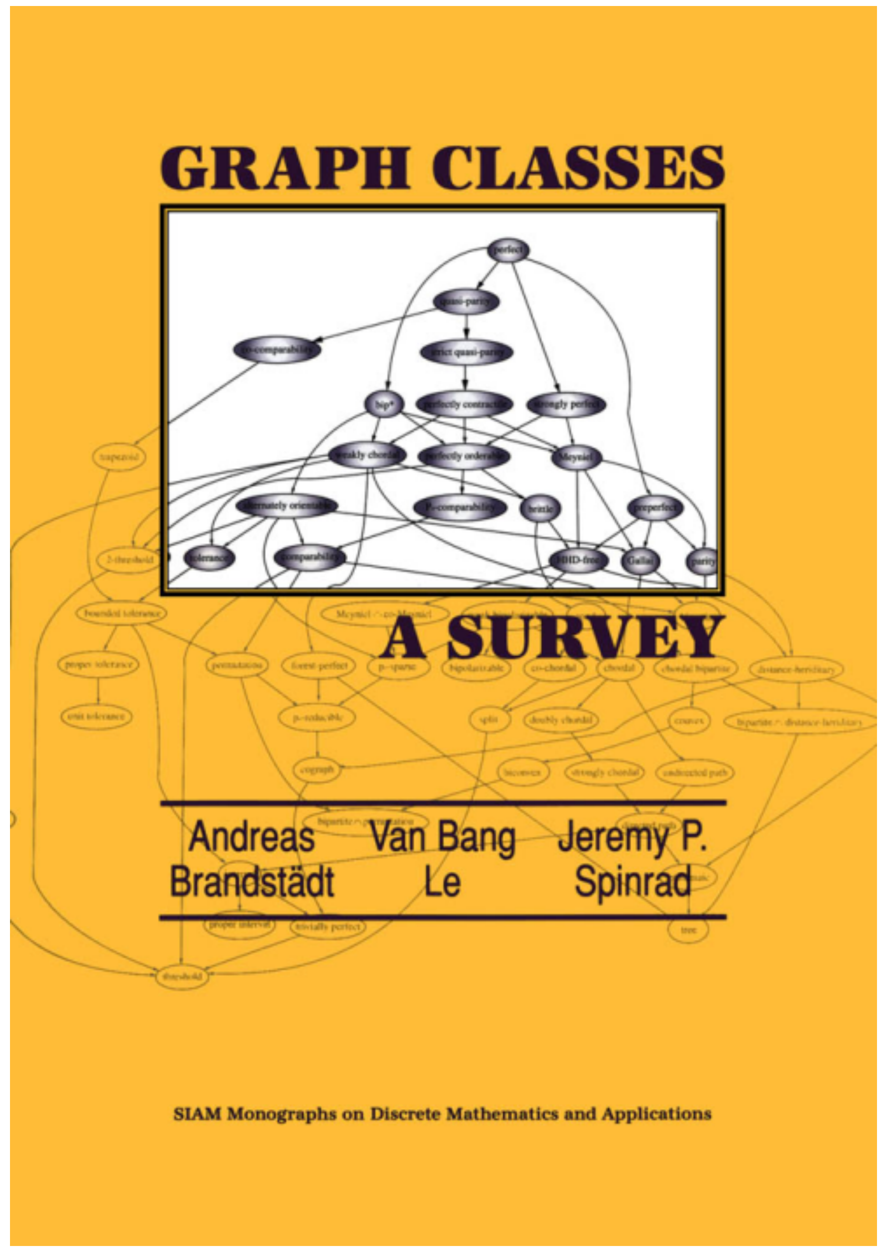


Network



Hypergraph





A. Brandstädt et al. 1999

Degree

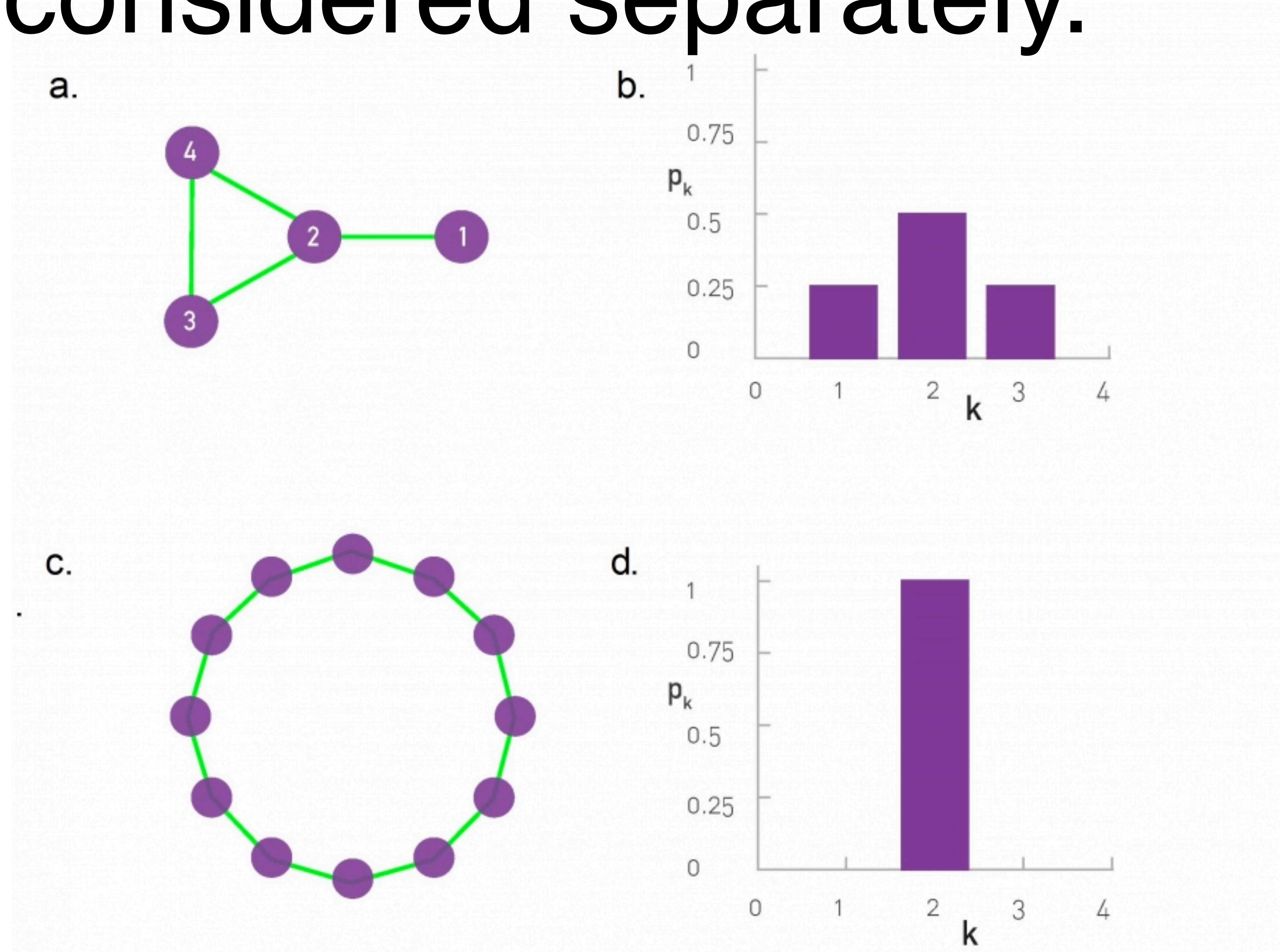
Node degree deg(x)

The number of edges connecting a node. For directed graphs in- and out-degree are considered separately.

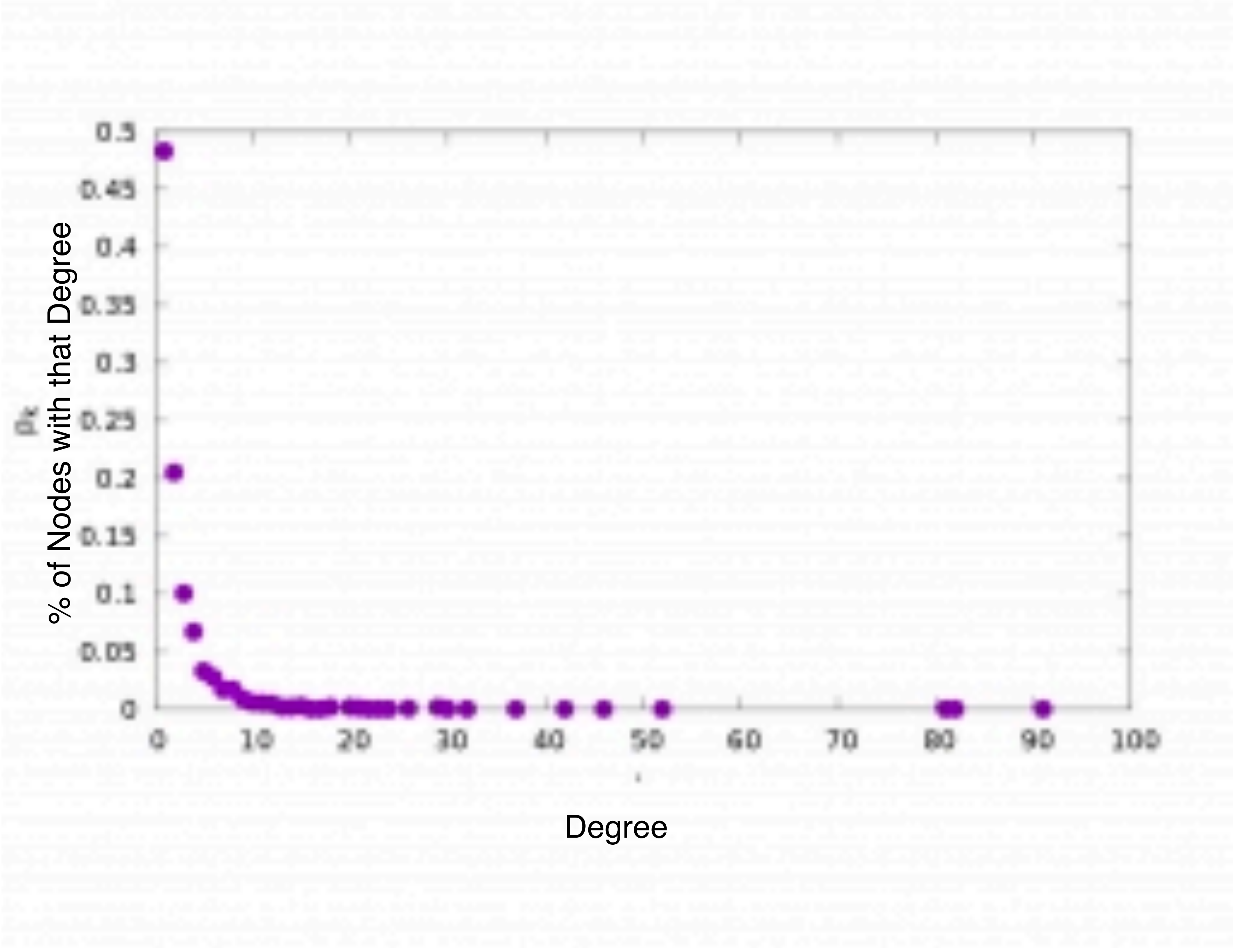
Average degree

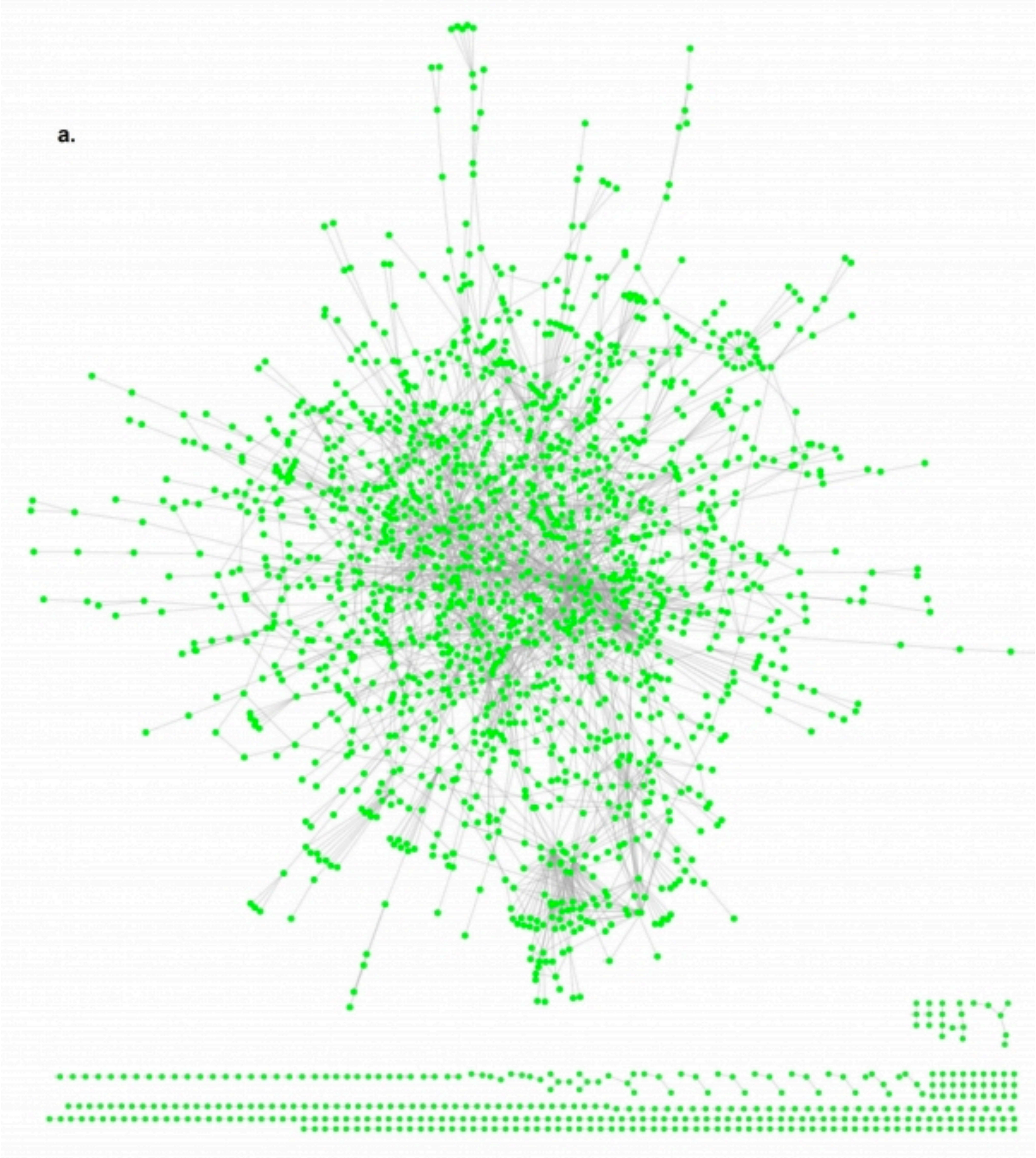
$$\langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2L}{N}$$

Degree distribution



Degree Distribution of a real Network

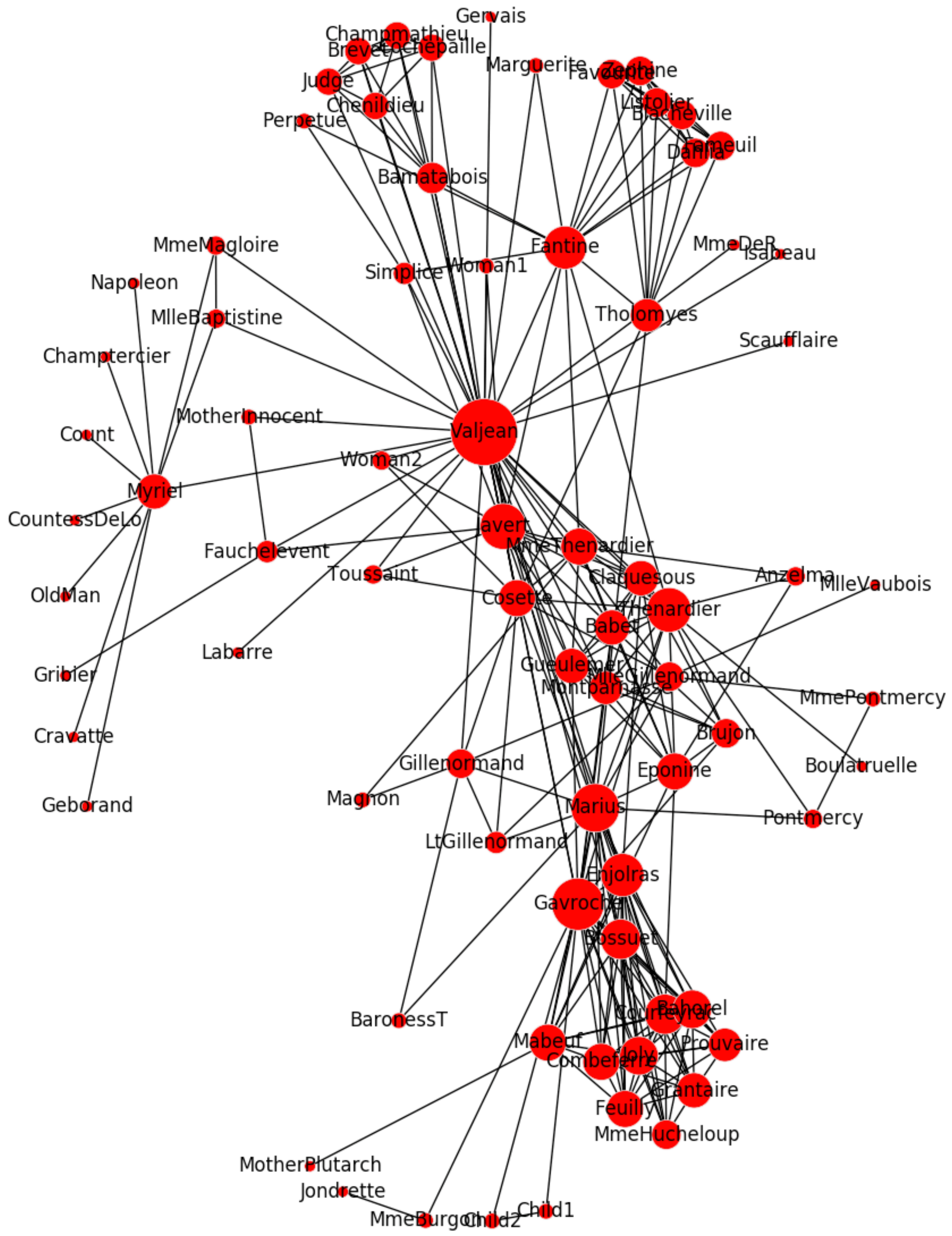




Protein Interaction Network, Barabasi

Degrees

Degree is a measure of local importance



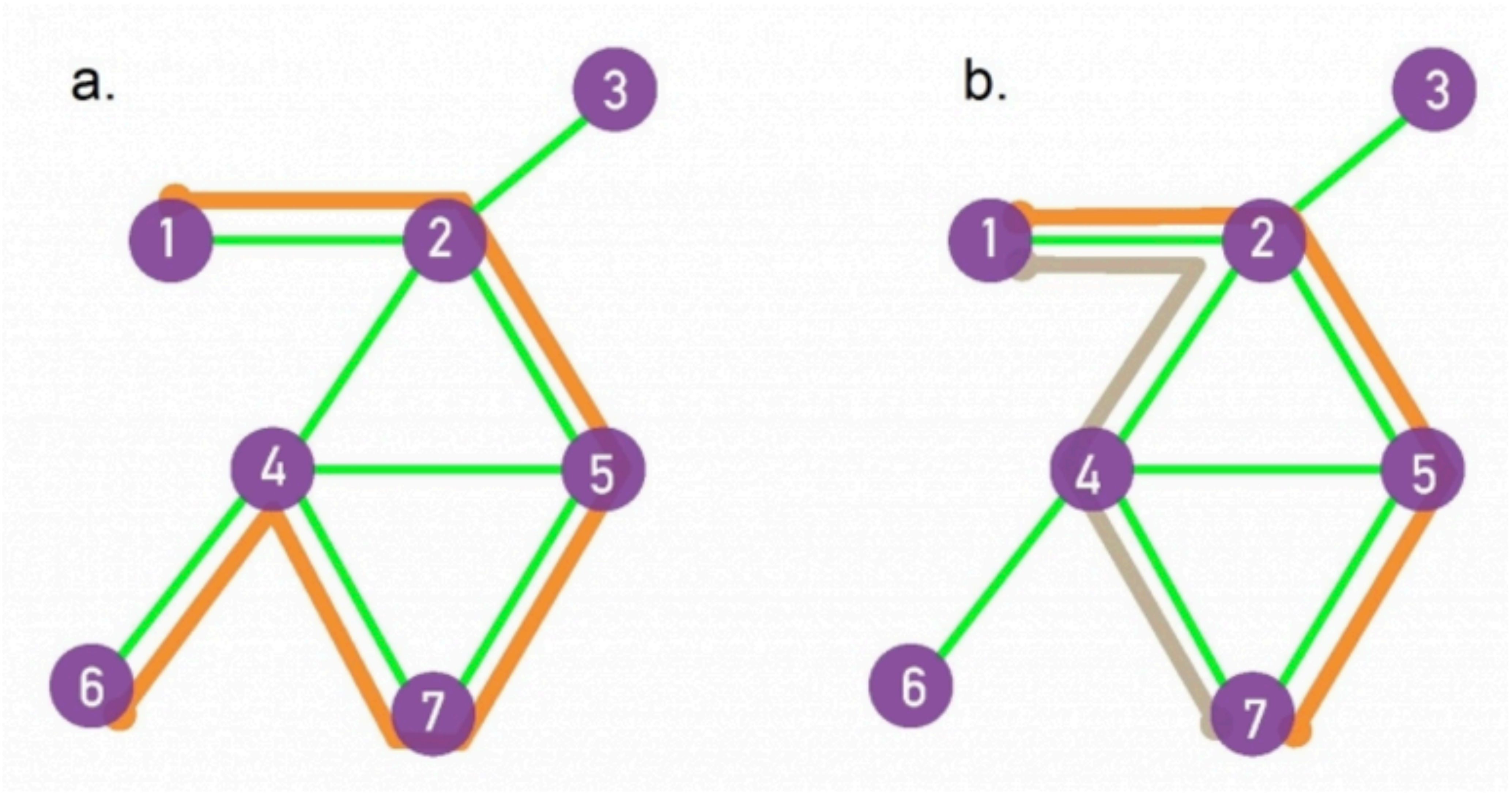
Paths & Distances

Path is **route along links**

Path length is the **number of links** contained

Shortest paths connects nodes i and j with the smallest number of links

Diameter of graph G
The longest shortest path within G.

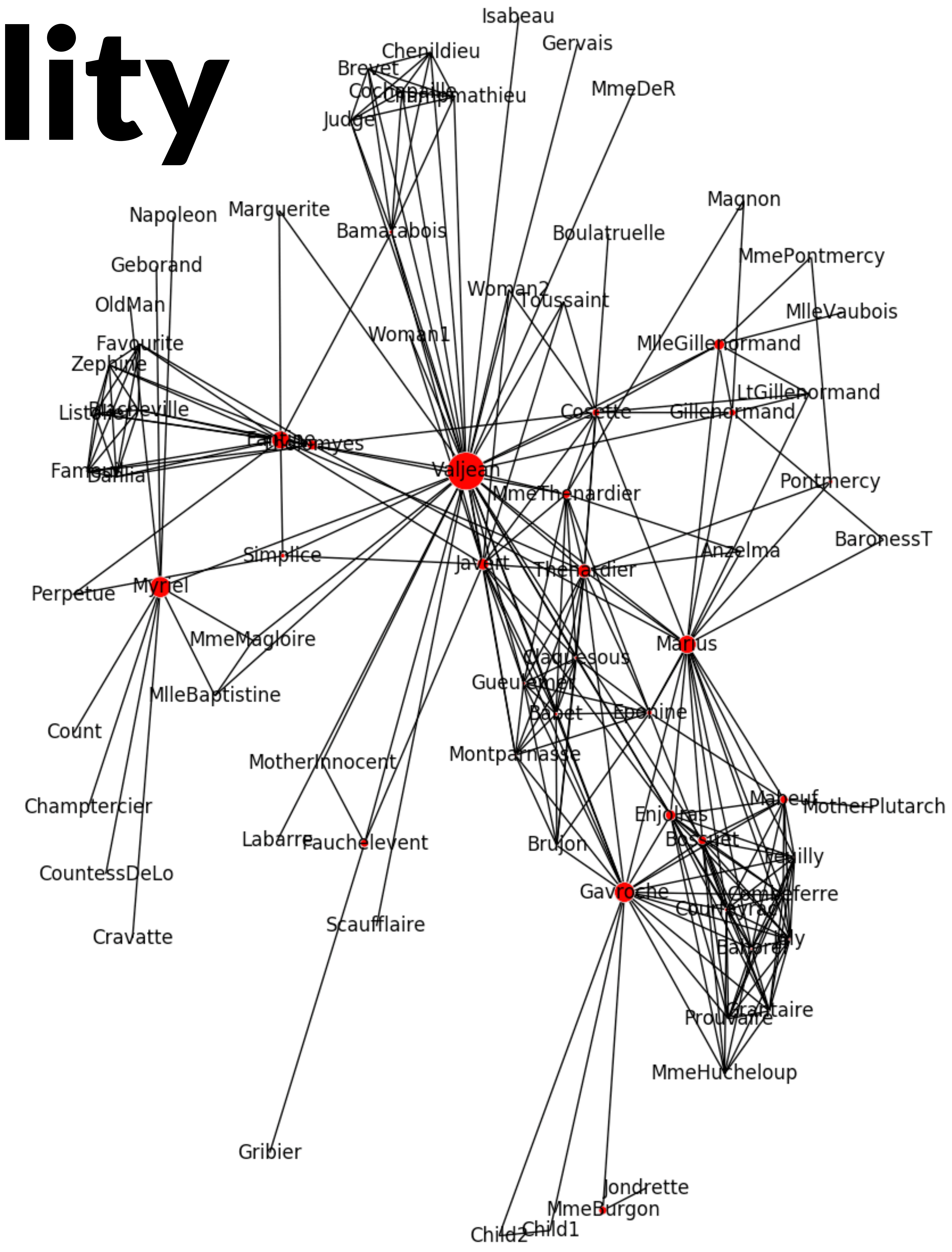


A path from 1 to 6 Shortest paths (two) from 1 to 7.

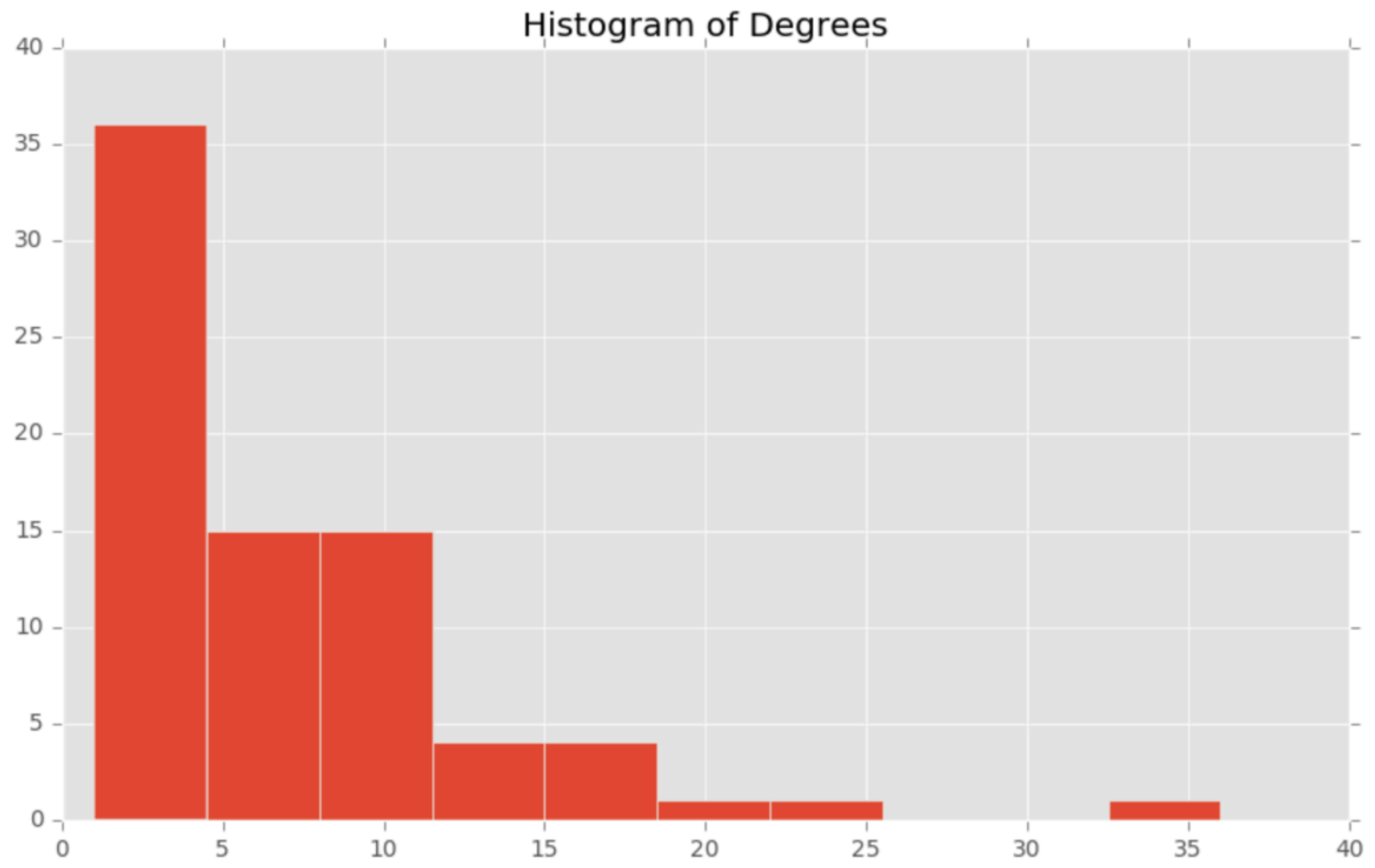
Betweenness Centrality

a measure of how many shortest paths pass through a node

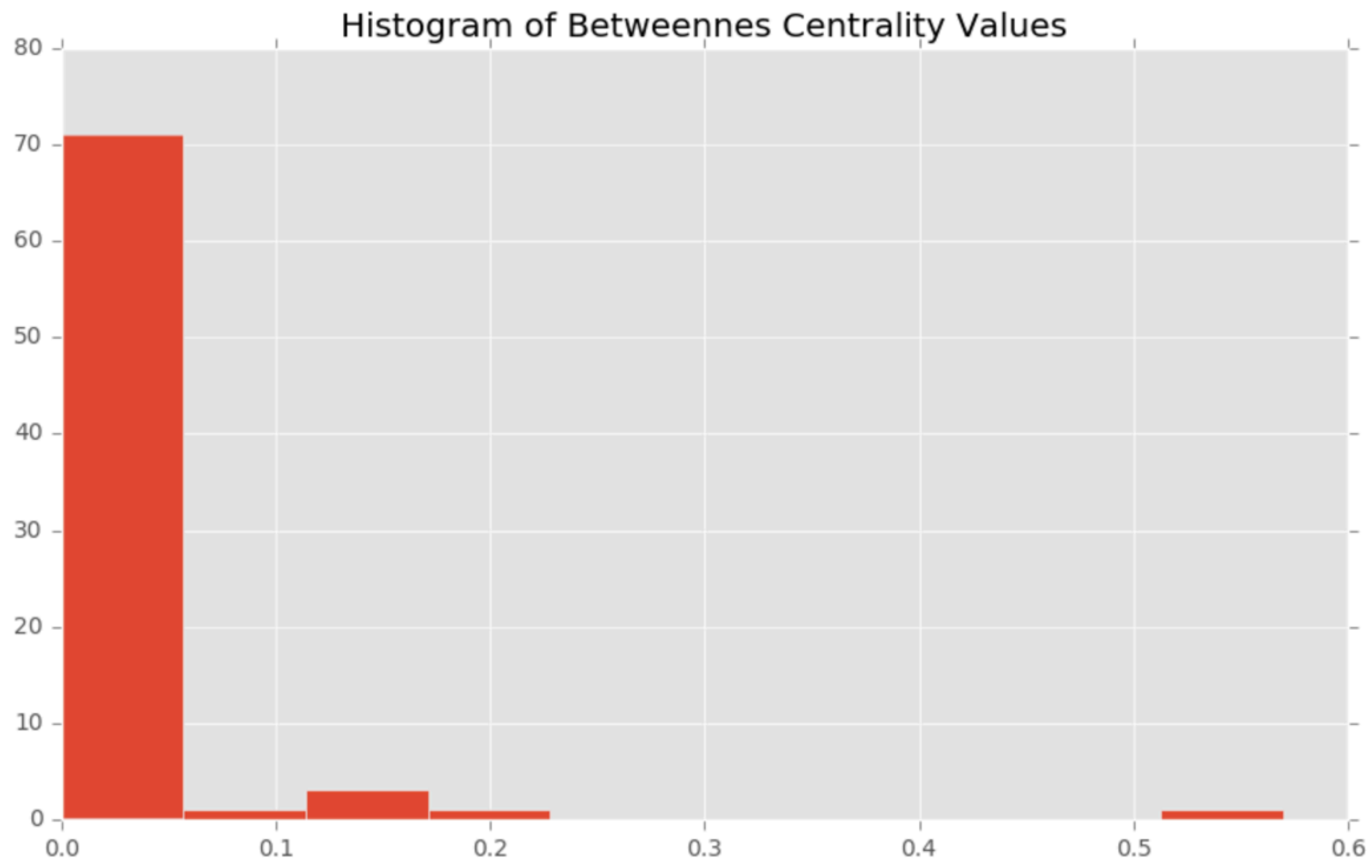
good measure for the overall relevance of a node in a graph



Degree vs BC



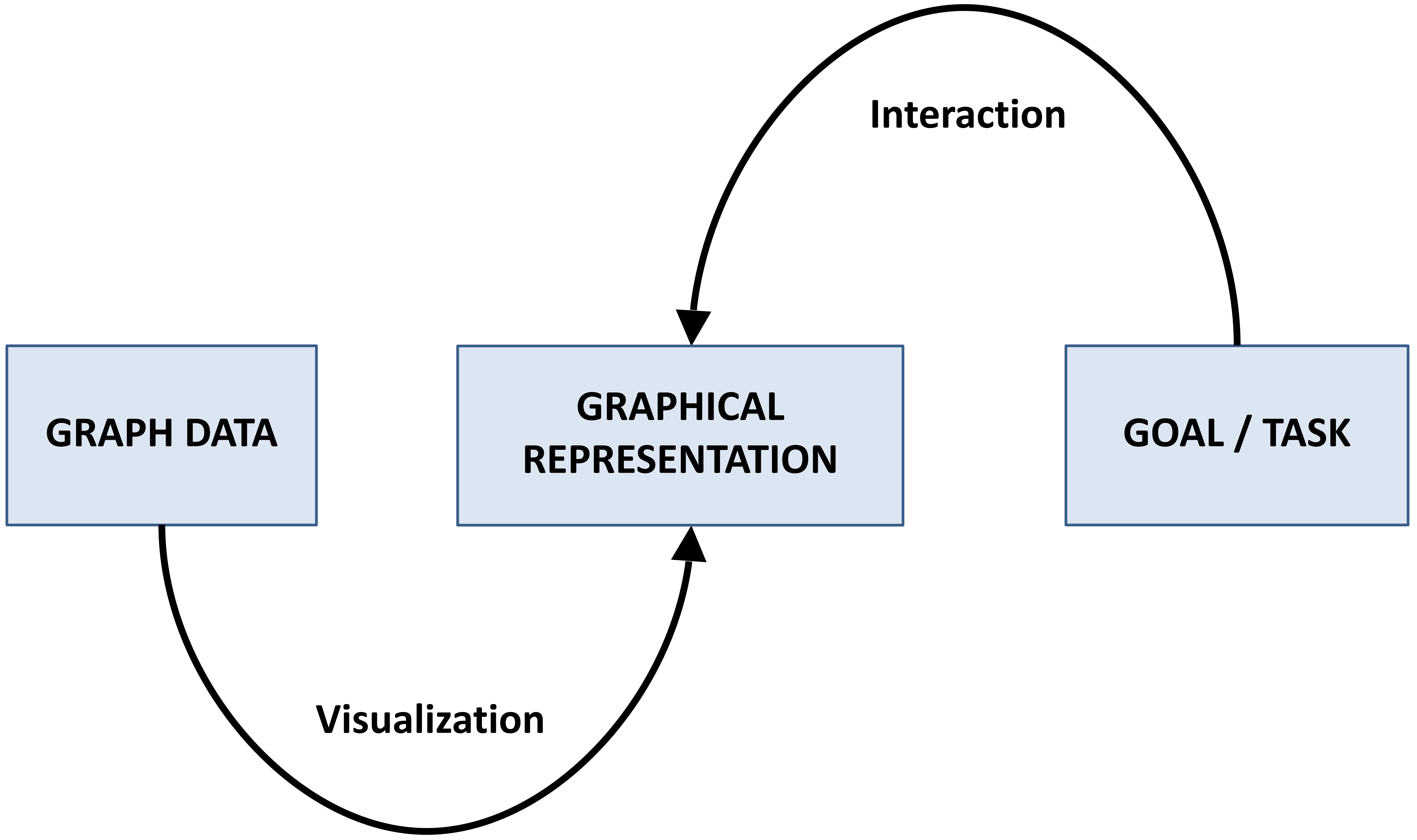
Histogram of Degrees
40
35
30
25
20
15
10
5
0
0
5
10
15
20
25
30
35
40



Histogram of Betweennes Centrality Values
80
70
60
50
40
30
20
10
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6

Network and Tree Visualization

Setting the Stage



How to decide which **representation** to use for which **type of graph** in order to achieve which kind of **goal**?

Task Taxonomy for Graph Visualization

Bongshin Lee, Catherine Plaisant,
Cynthia Sims Parr
Human-Computer Interaction Lab
University of Maryland,
College Park, MD 20742, USA
+1-301-405-7445
{bongshin, plaisant, csparr}@cs.umd.edu

Jean-Daniel Fekete,
Nathalie Henry
INRIA Futurs/LRI Bat. 490
Université Paris-Sud,
91405 ORSAY, France
+33-1-69153460
Jean-Daniel.Fekete@inria.fr, nhenry@lri.fr

ABSTRACT
Our goal is to define a list of tasks for graph visualization that has enough detail and specificity to be useful to: 1) designers who want to improve their system and 2) to evaluators who want to compare graph visualization systems. In this paper, we suggest a list of tasks we believe are commonly encountered while analyzing graph data. We define graph specific objects and demonstrate how all complex tasks could be seen as a series of low-level tasks performed on those objects. We believe that our taxonomy, associated with benchmark datasets and specific tasks, would help evaluators generalize results collected through a series of controlled experiments.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Graphical user interfaces (GUI)*

user studies of graph visualization techniques and extracted the tasks used in those studies.

After making those two lists, we considered the set of low-level Visual Analytics tasks proposed by Amar *et al*. [2]. These tasks were extracted from a corpus of questions about tabular data. We realized that our tasks all seem to be compound tasks made up of Amar *et al*'s primitive tasks applied to the graph objects. When some tasks could not be represented with those tasks and objects, we added either an object or a low-level task. In this paper, we demonstrate how all complex tasks could be seen as a series of low-level tasks performed on those objects.

2. GRAPH-SPECIFIC OBJECTS
A graph consists of two types of primitive elements, nodes and links. A subgraph of a graph G is a graph whose nodes and links are subsets of G. There are several meaningful subgraphs such as

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 with a given property

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

Find neighbors nodes

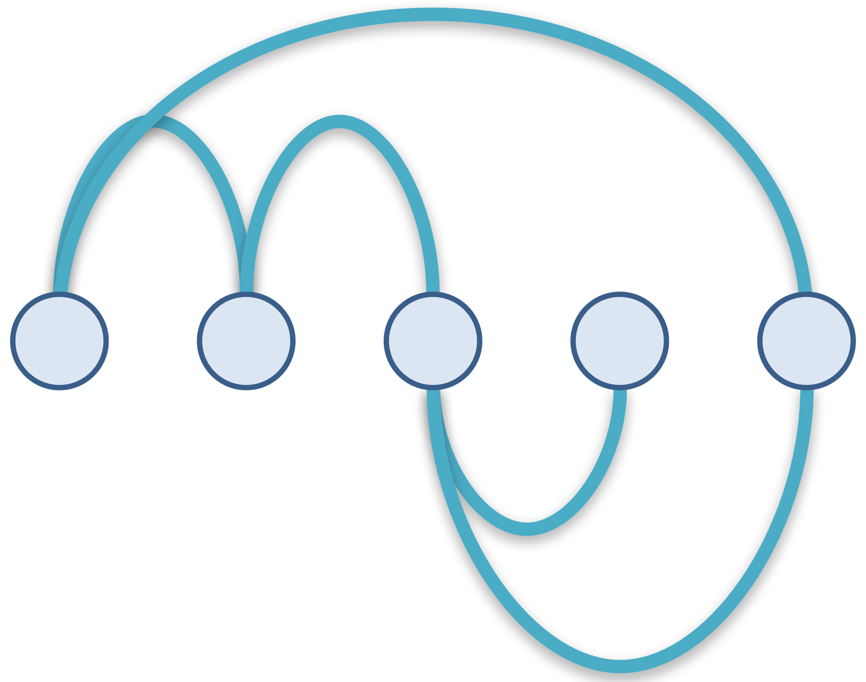
Identify Clusters / Communities

Find Paths

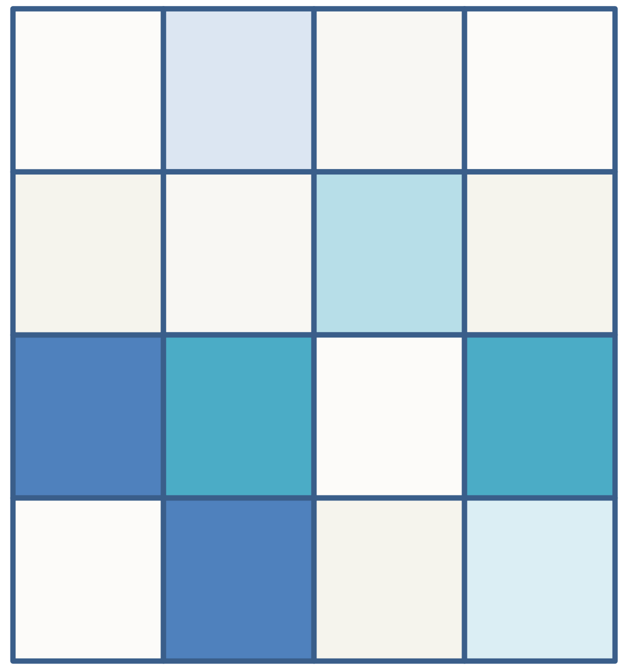
….

list adapted from Schulz 2010

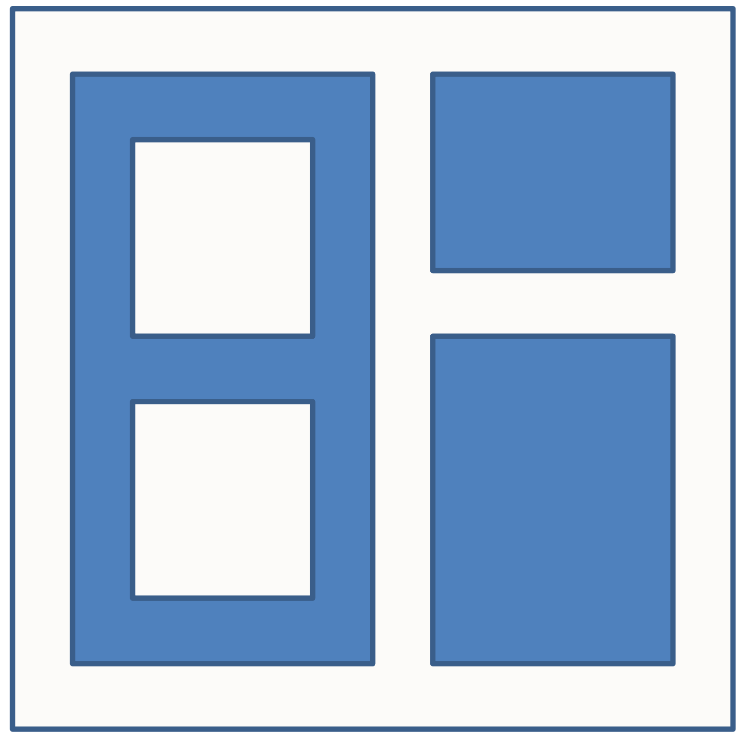
Three Types of Graph Representations



Explicit
(Node-Link)



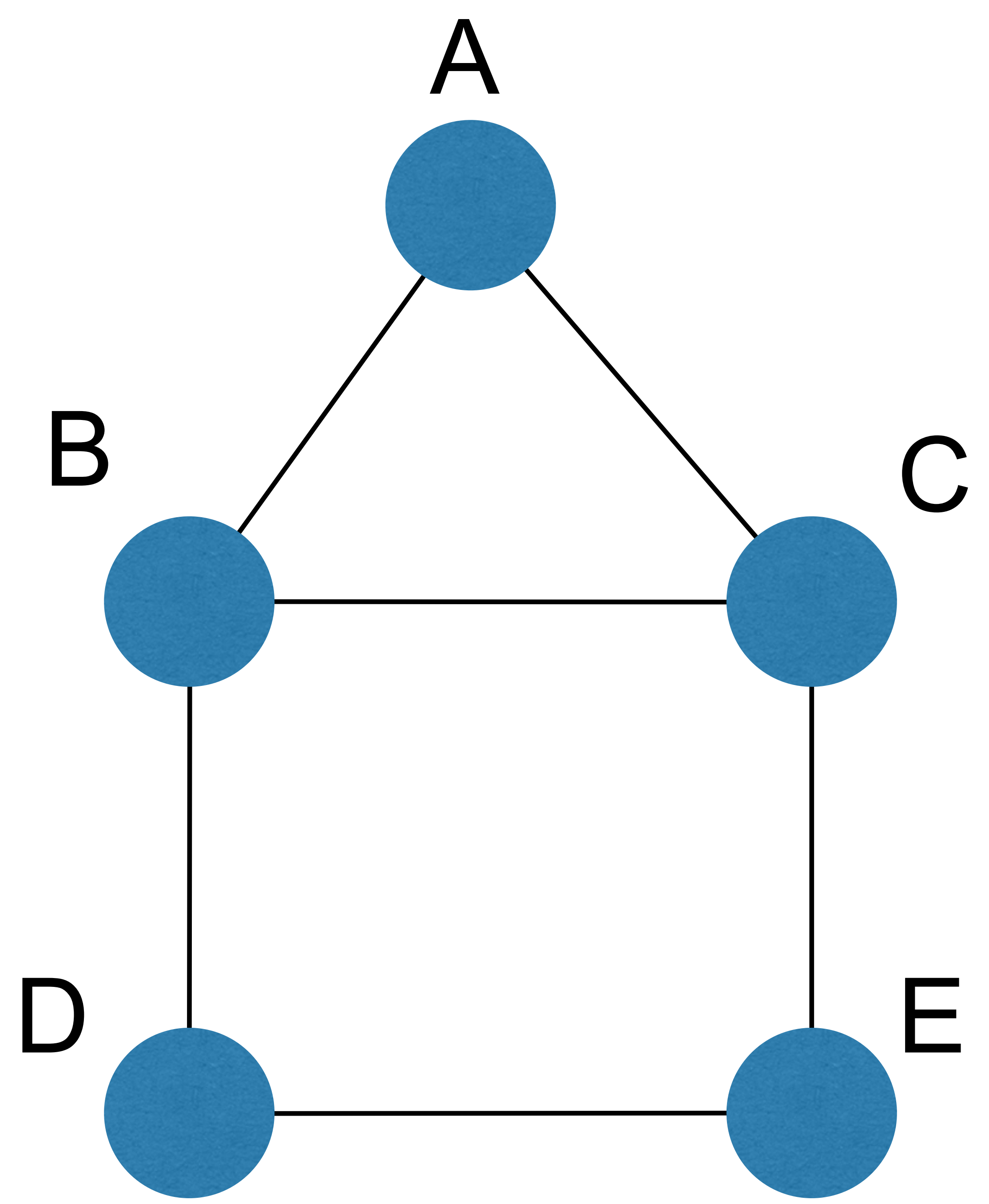
Matrix

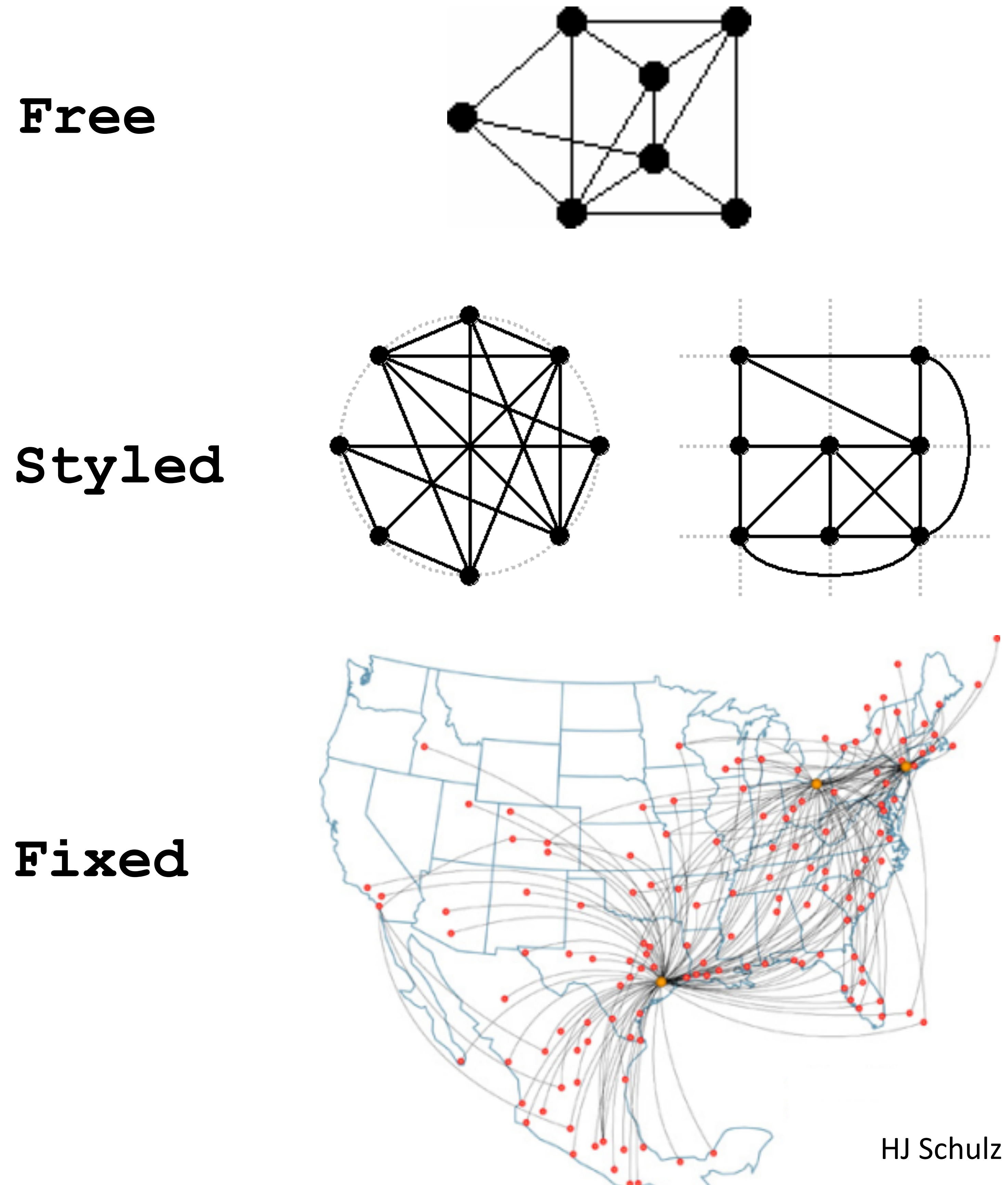


Implicit

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

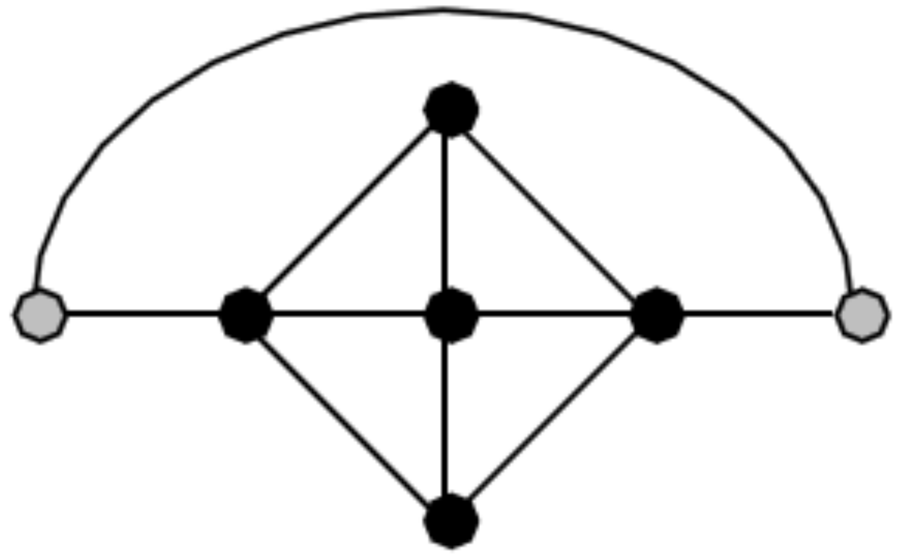
Aspect ratio about 1 (not too long and not too wide)

Symmetry: similar graph structures should look similar

list adapted from Battista et al. 1999

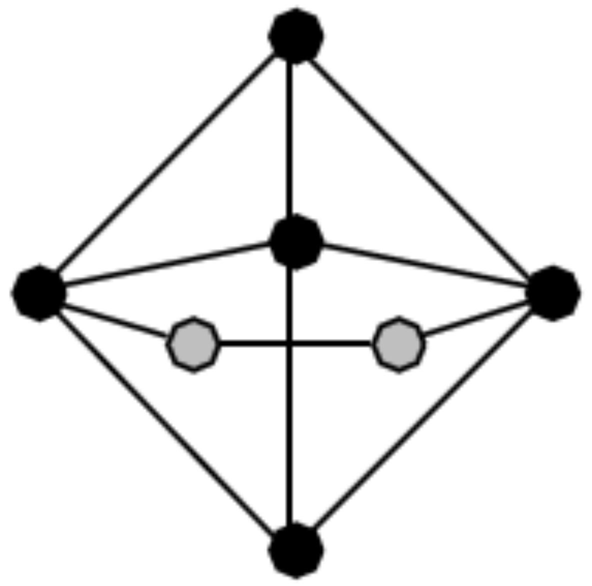
Conflicting Criteria

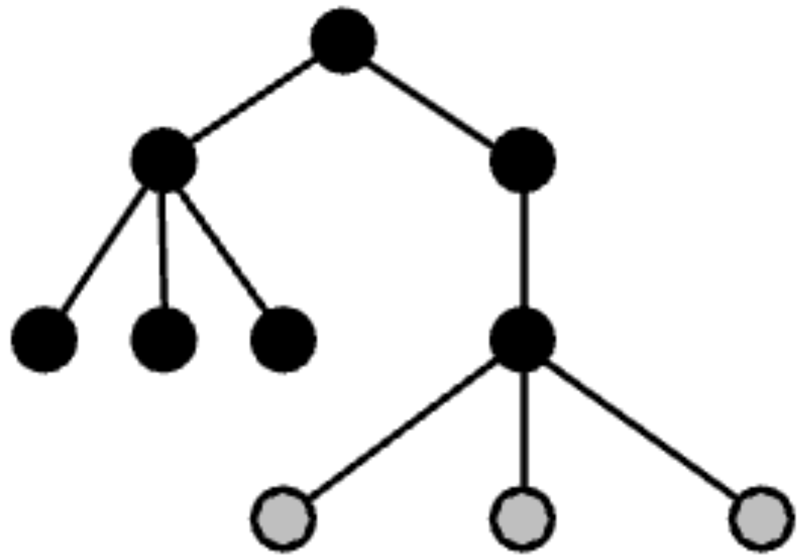
Minimum number
of edge crossings



vs.

Uniform edge
length

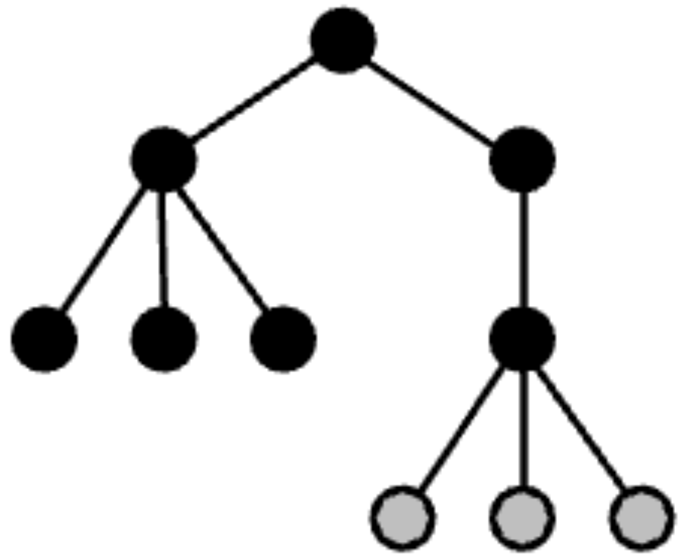




Space utilization

vs.

Symmetry



Schulz 2004

Explicit Layouts

Layout approach: formulate the layout problem as an optimization problem

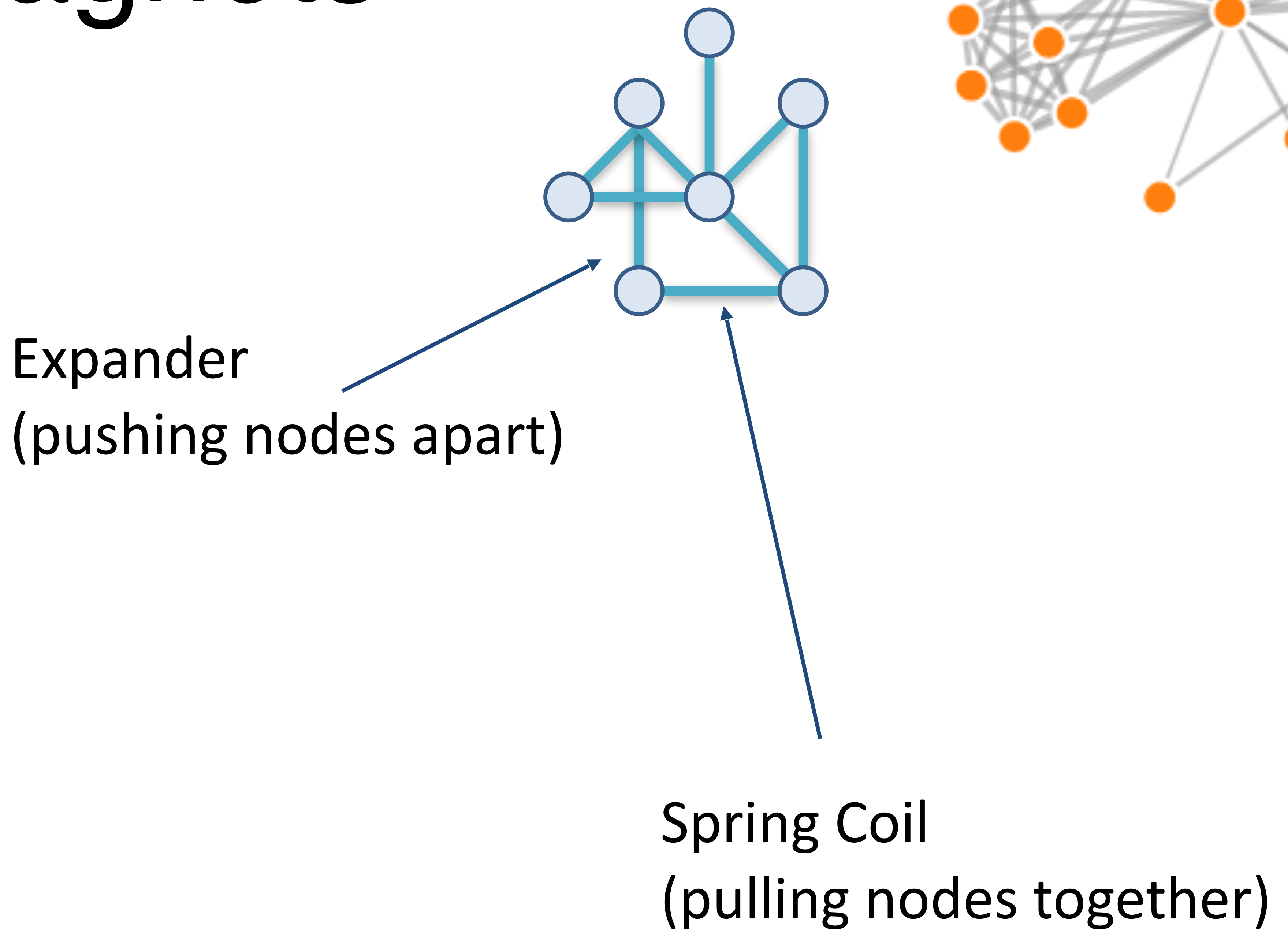
1. Conversion of the layout criteria into a weighted cost function:

    F(layout) = a*|edge crossings| + … + f *|used drawing space|

2. Use a standard optimization technique (e.g., simulated annealing) to find a layout that minimizes the cost function

Force Directed Layouts

Physics model:
edges = springs,
vertices = repulsive magnets

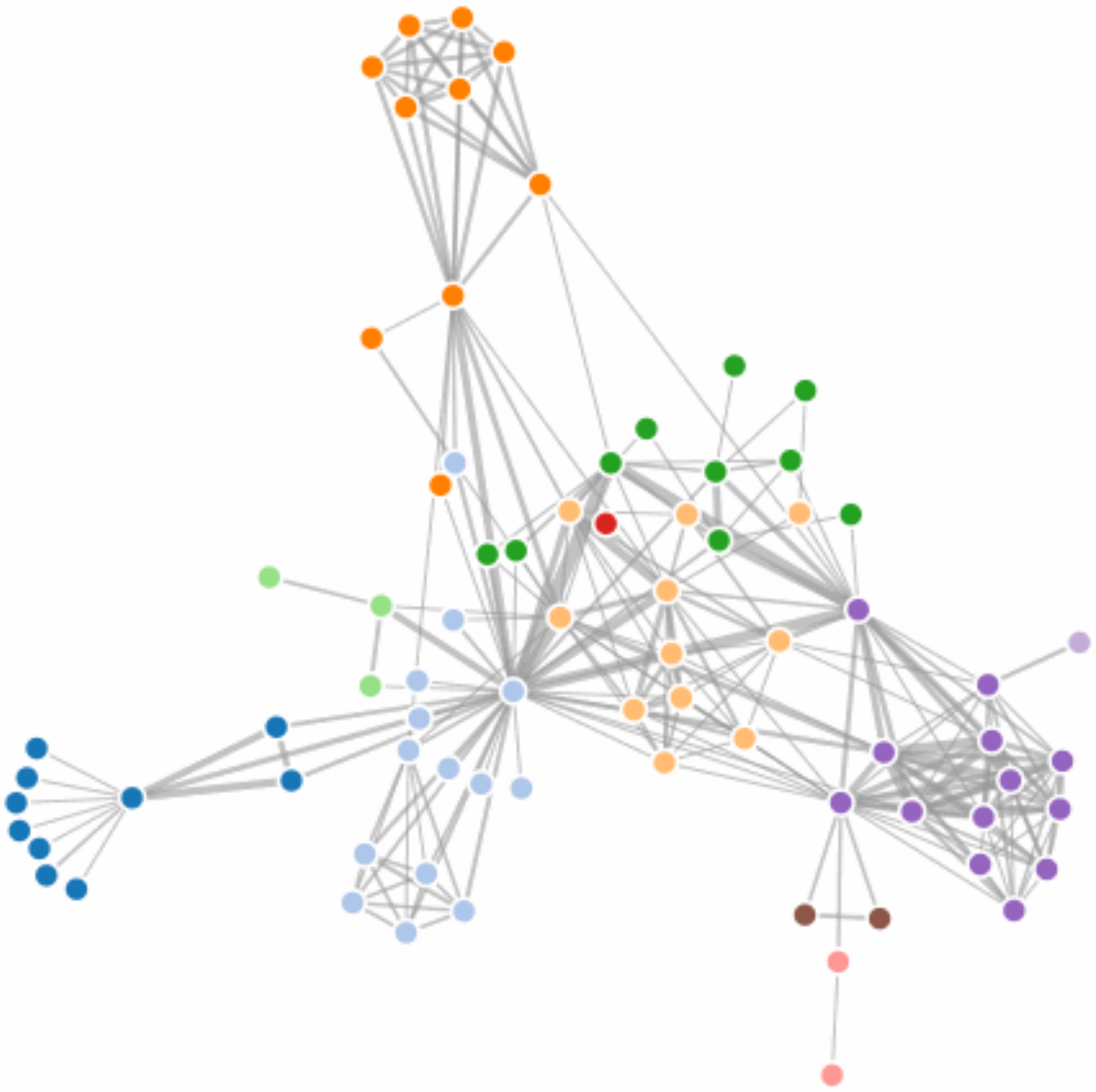


Algorithm

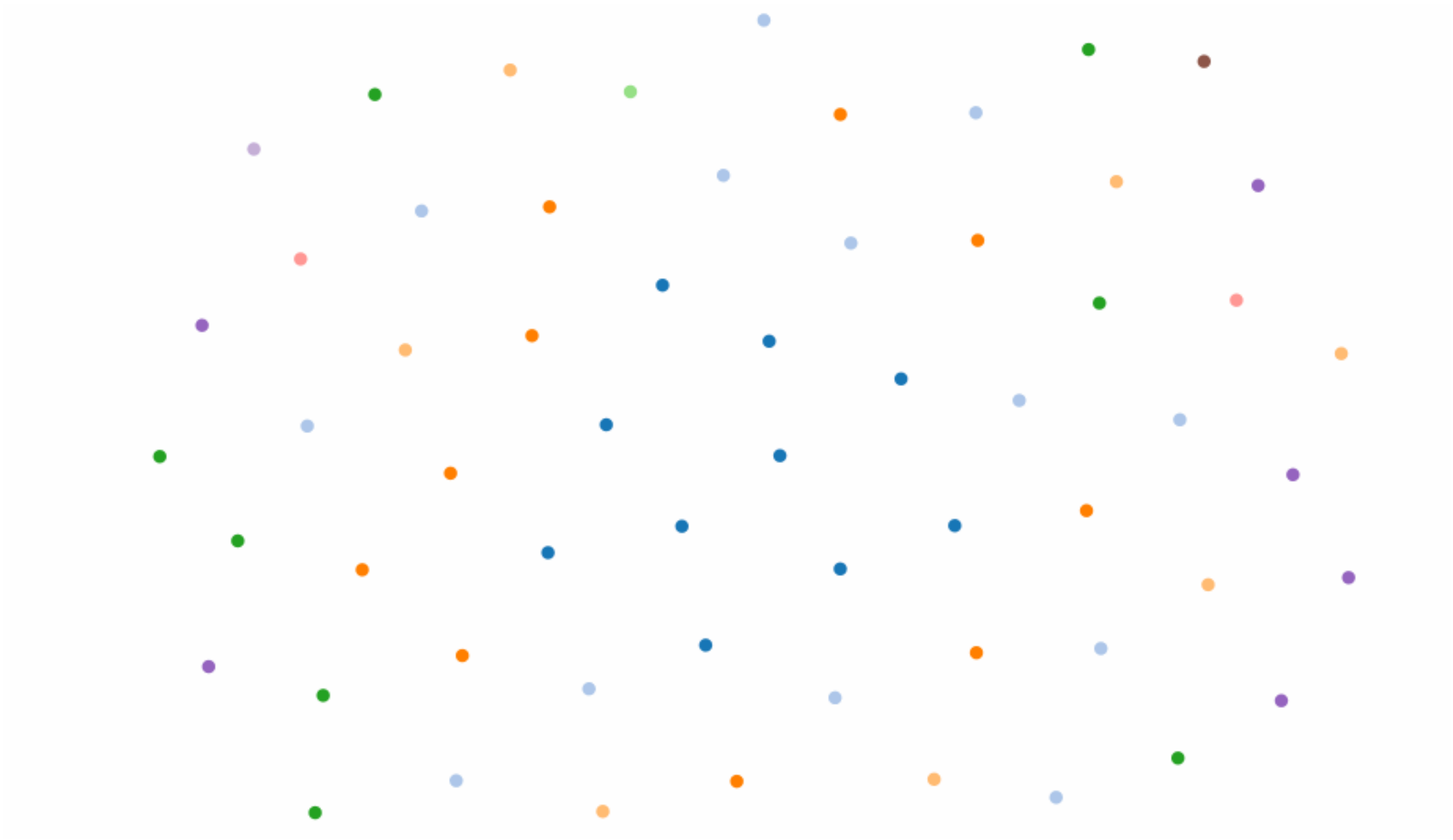
Place Vertices in random locations

While not equilibrium

- calculate force on vertex
  - sum of
    - pairwise repulsion of all nodes (n*n operations)
    - attraction between connected nodes
- move vertex by c * force on vertex



What happens when there are no links?



Properties

Generally good layout

Uniform edge length

Clusters commonly visible

Not deterministic

Computationally expensive: $O(n^3)$

n^2 in every step, it takes about n cycles to reach equilibrium

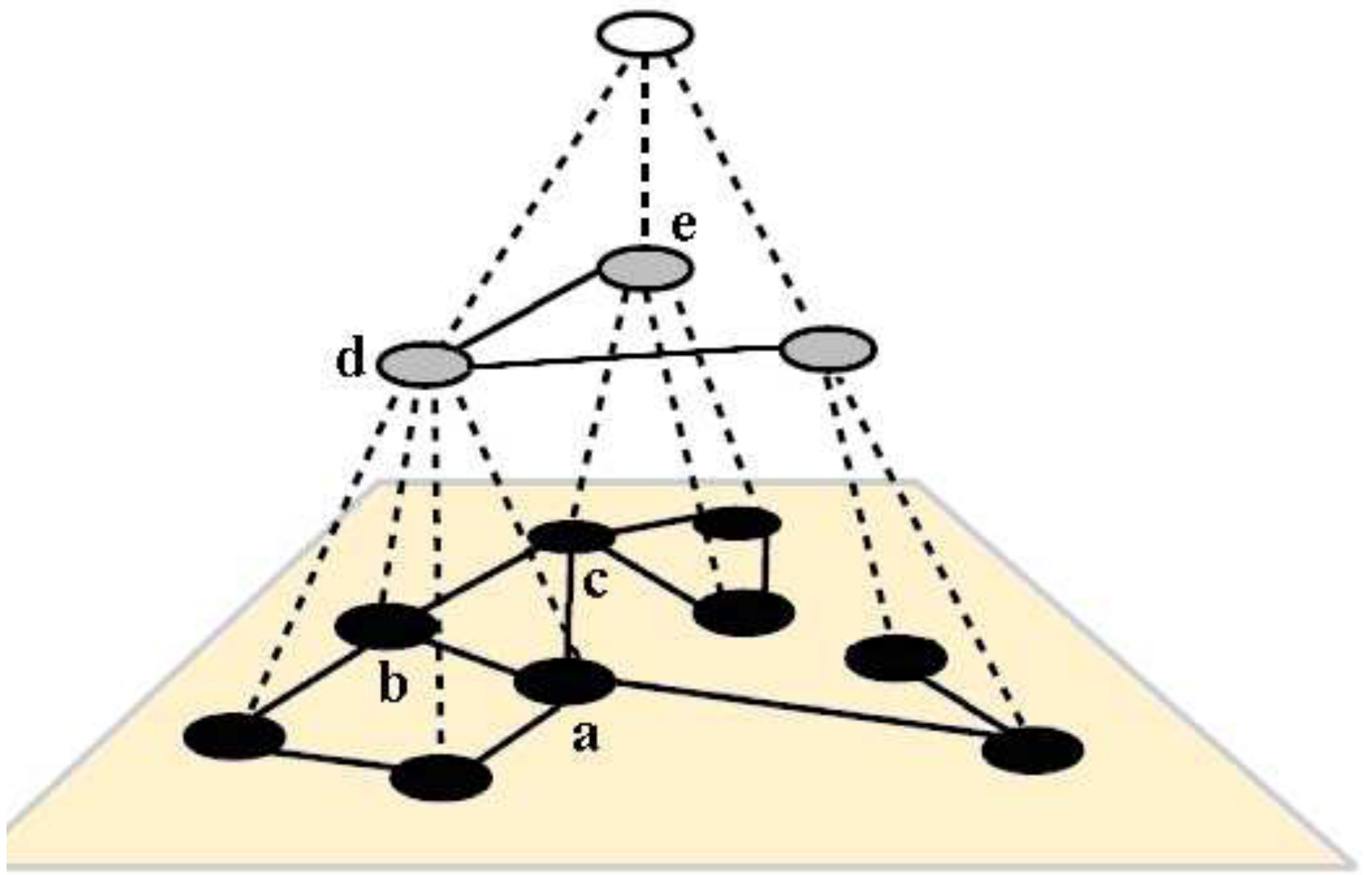
Limit (interactive): ~1000 nodes

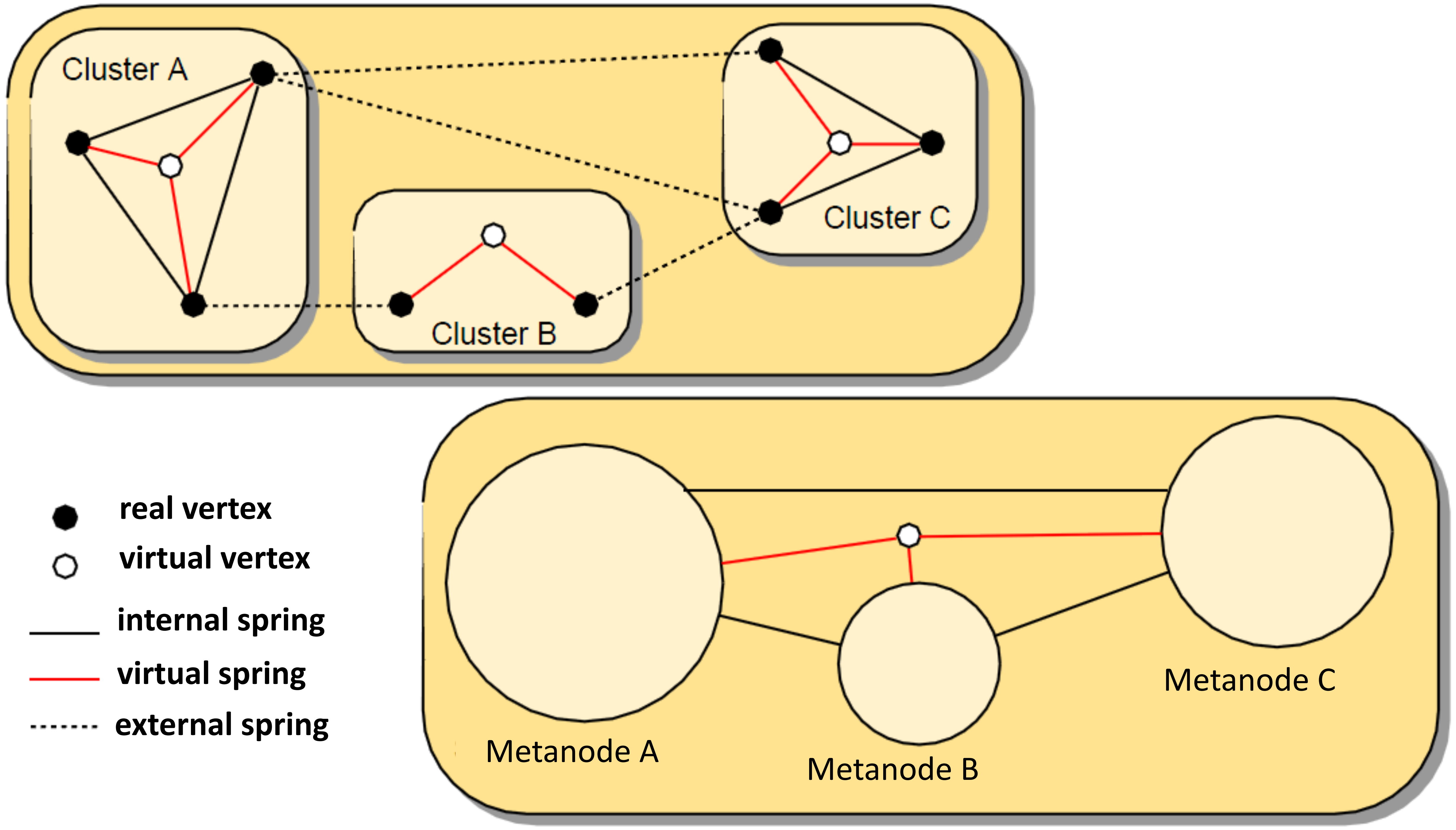
in practice: damping, center of gravity

http://bl.ocks.org/steveharoz/8c3e2524079a8c440df60c1ab72b5d03

Giant Hairball

Adress Computational Scalability: Multilevel Approaches



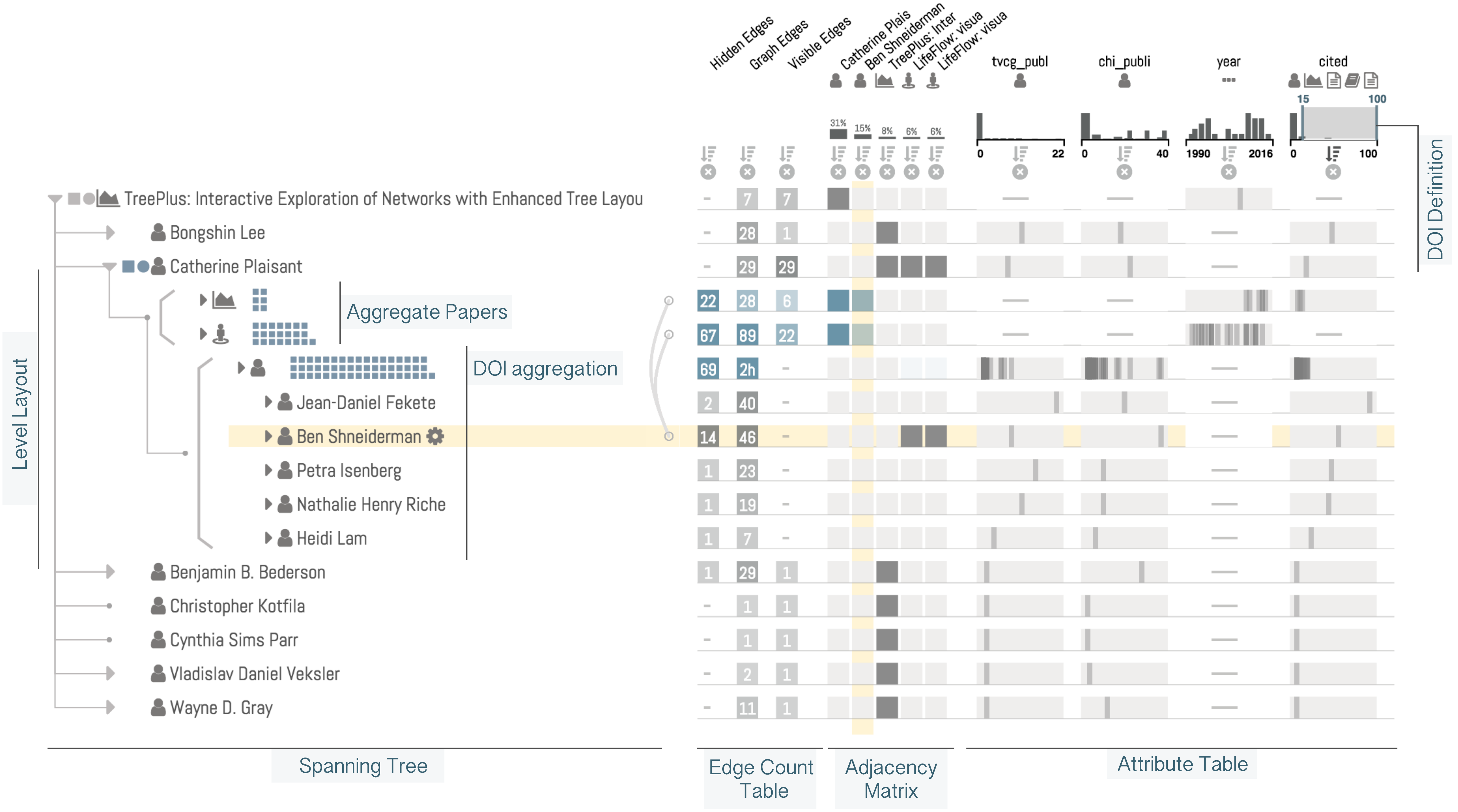


[Schulz 2004]

Alternative Approach: Query first, Expand on Demand

What do you want to know from a network?

Rarely is an overview helpful.



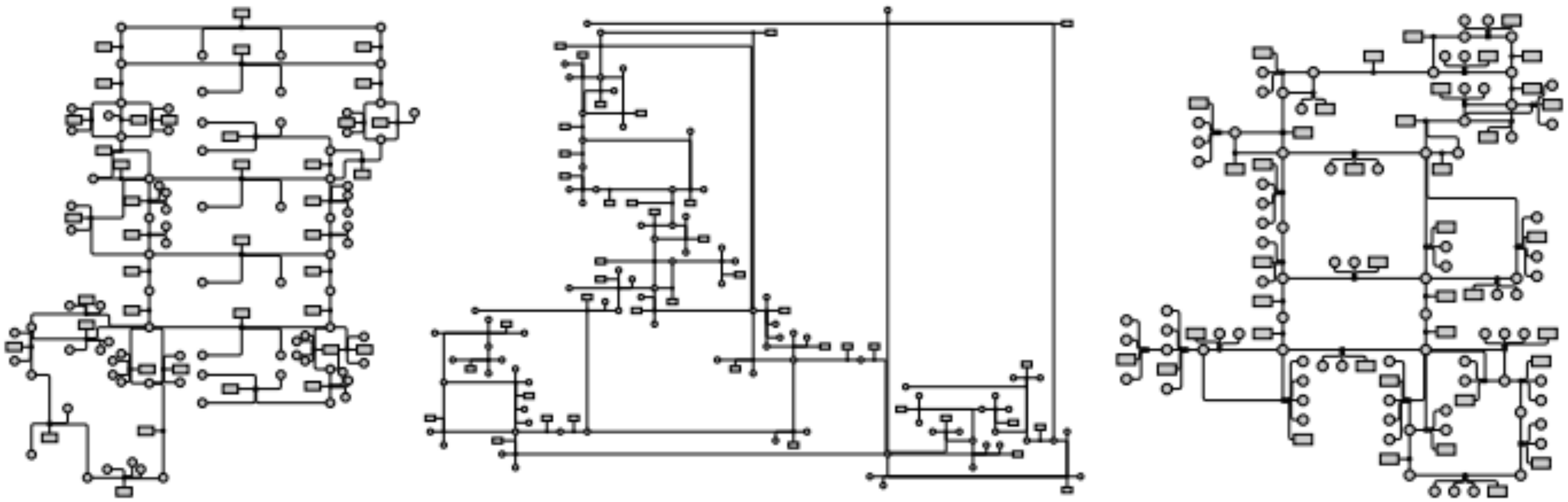
[Nobre et al, Juniper, TVCG 2018]

HOLA: Human-like Orthogonal Layout

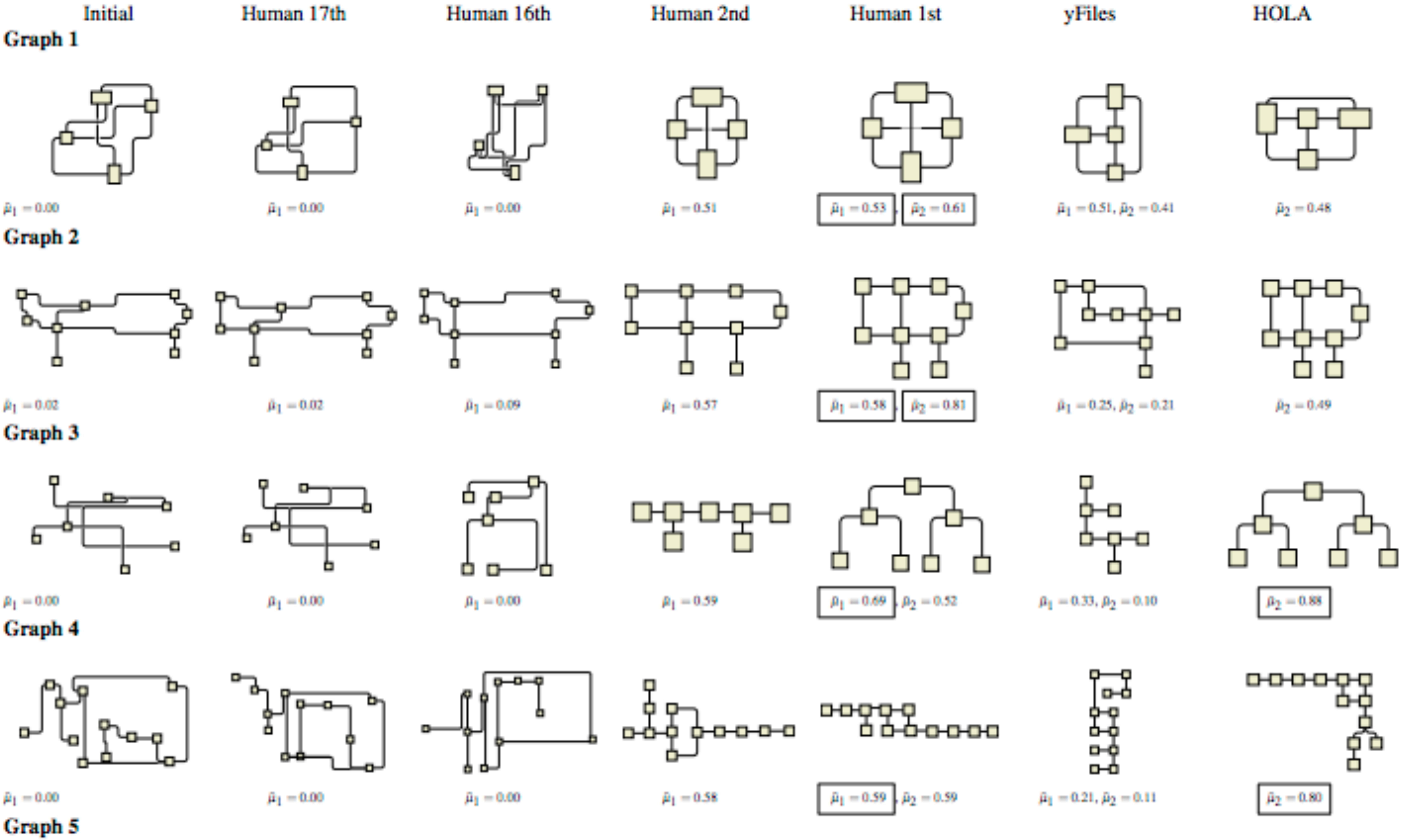
Study how humans lay-out a graph

Try to emulate layout

Left: human, middle: conventional algo, right new algo



[Kieffer et al, InfoVis 2015]


Initial
Human 17th
Human 16th
Human 2nd
Human 1st
yFiles
HOLA
Graph 1
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.51$
$\hat{\mu}_1 = 0.53$, $\hat{\mu}_2 = 0.61$
$\hat{\mu}_1 = 0.51$, $\hat{\mu}_2 = 0.41$
$\hat{\mu}_2 = 0.48$
Graph 2
$\hat{\mu}_1 = 0.02$
$\hat{\mu}_1 = 0.02$
$\hat{\mu}_1 = 0.09$
$\hat{\mu}_1 = 0.57$
$\hat{\mu}_1 = 0.58$, $\hat{\mu}_2 = 0.81$
$\hat{\mu}_1 = 0.25$, $\hat{\mu}_2 = 0.21$
$\hat{\mu}_2 = 0.49$
Graph 3
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.59$
$\hat{\mu}_1 = 0.69$, $\hat{\mu}_2 = 0.52$
$\hat{\mu}_1 = 0.33$, $\hat{\mu}_2 = 0.10$
$\hat{\mu}_2 = 0.88$
Graph 4
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.00$
$\hat{\mu}_1 = 0.58$
$\hat{\mu}_1 = 0.59$, $\hat{\mu}_2 = 0.59$
$\hat{\mu}_1 = 0.21$, $\hat{\mu}_2 = 0.11$
$\hat{\mu}_2 = 0.80$
Graph 5

Graphs in 3D

Why, why not visualize graphs in 3D?

Why, why not use AR/VR?

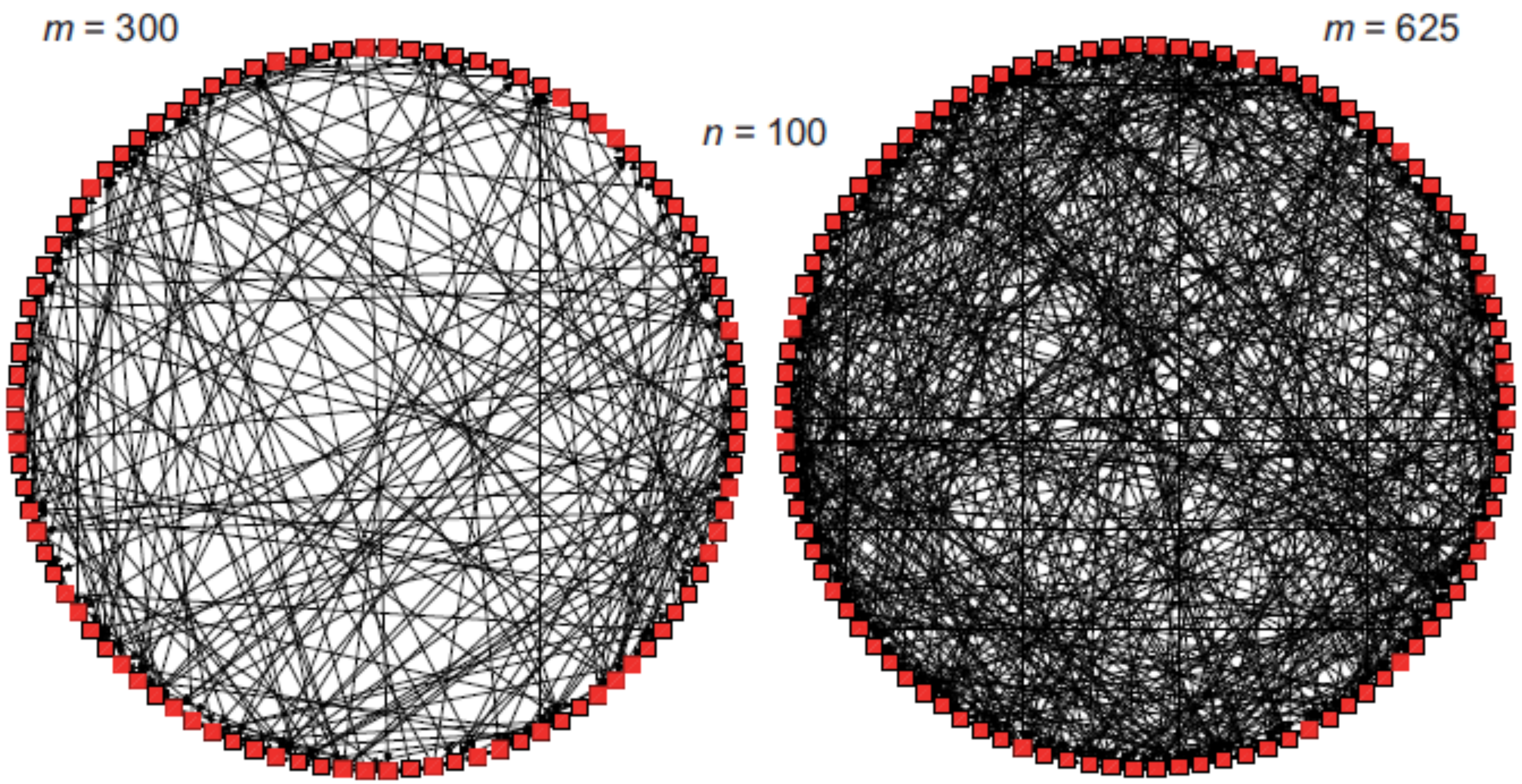
https://twitter.com/alexsigaras/status/860560655031685121

Styled / Restricted Layouts

Circular Layout

Node ordering

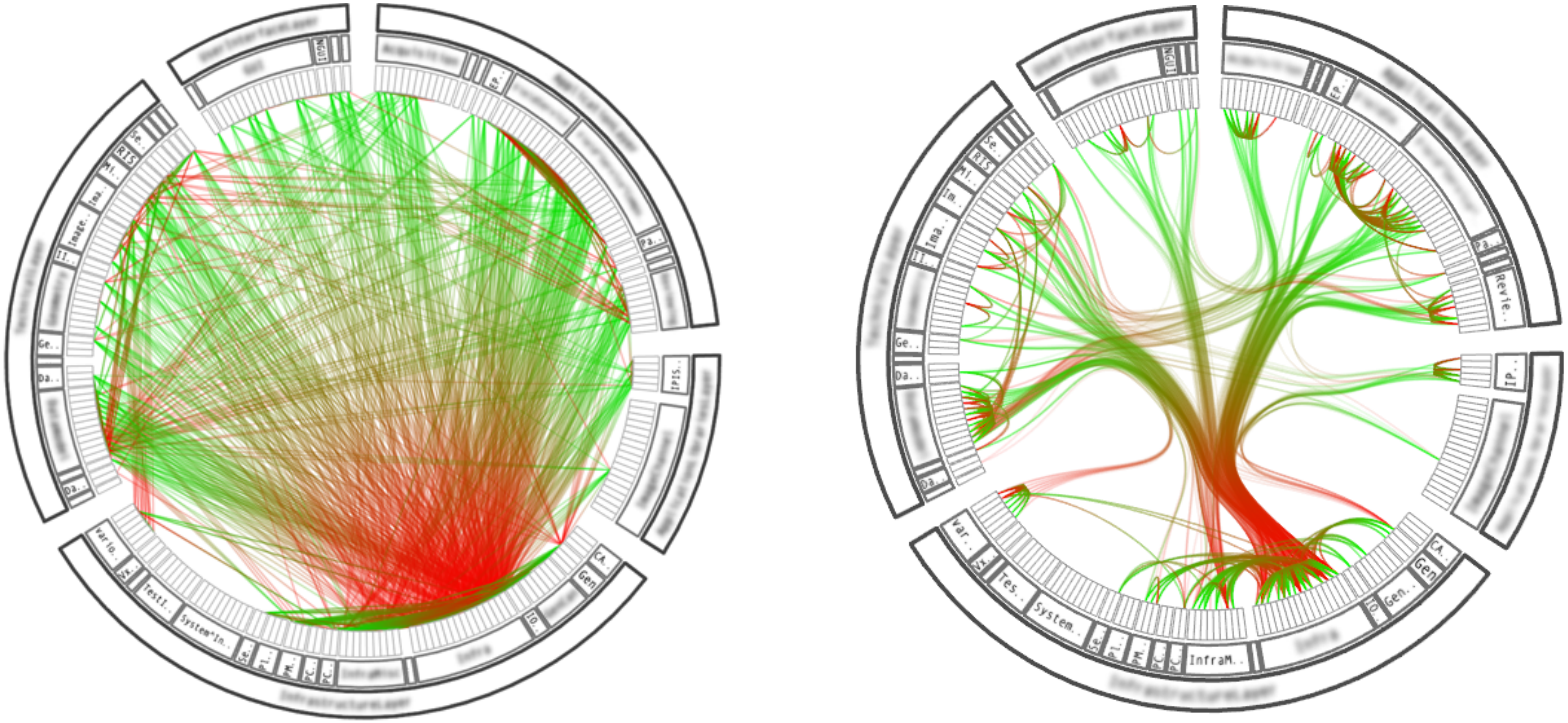
Edge Clutter



ca. 3% of all possible edges

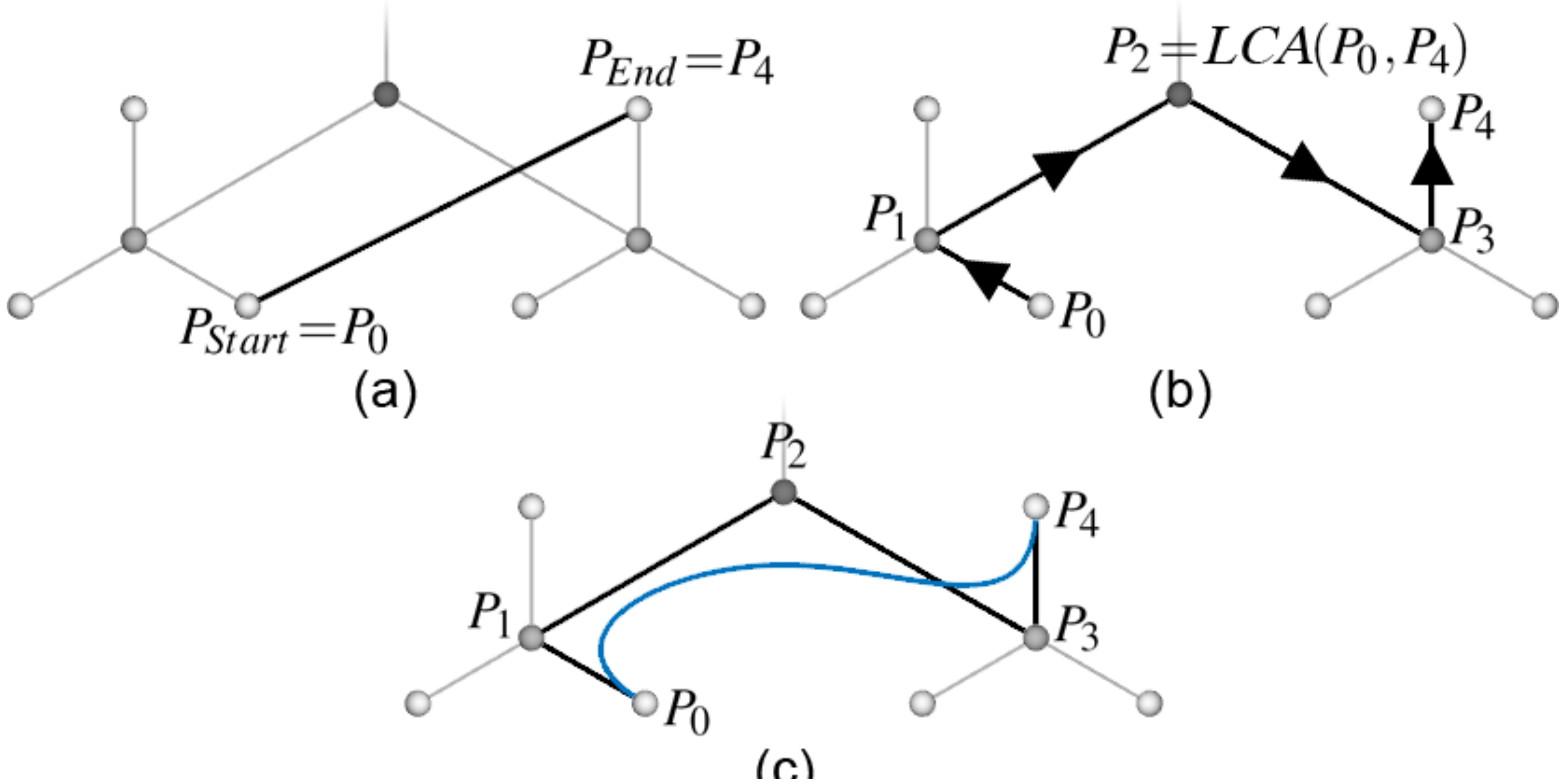
ca. 6,3% of all possible edges

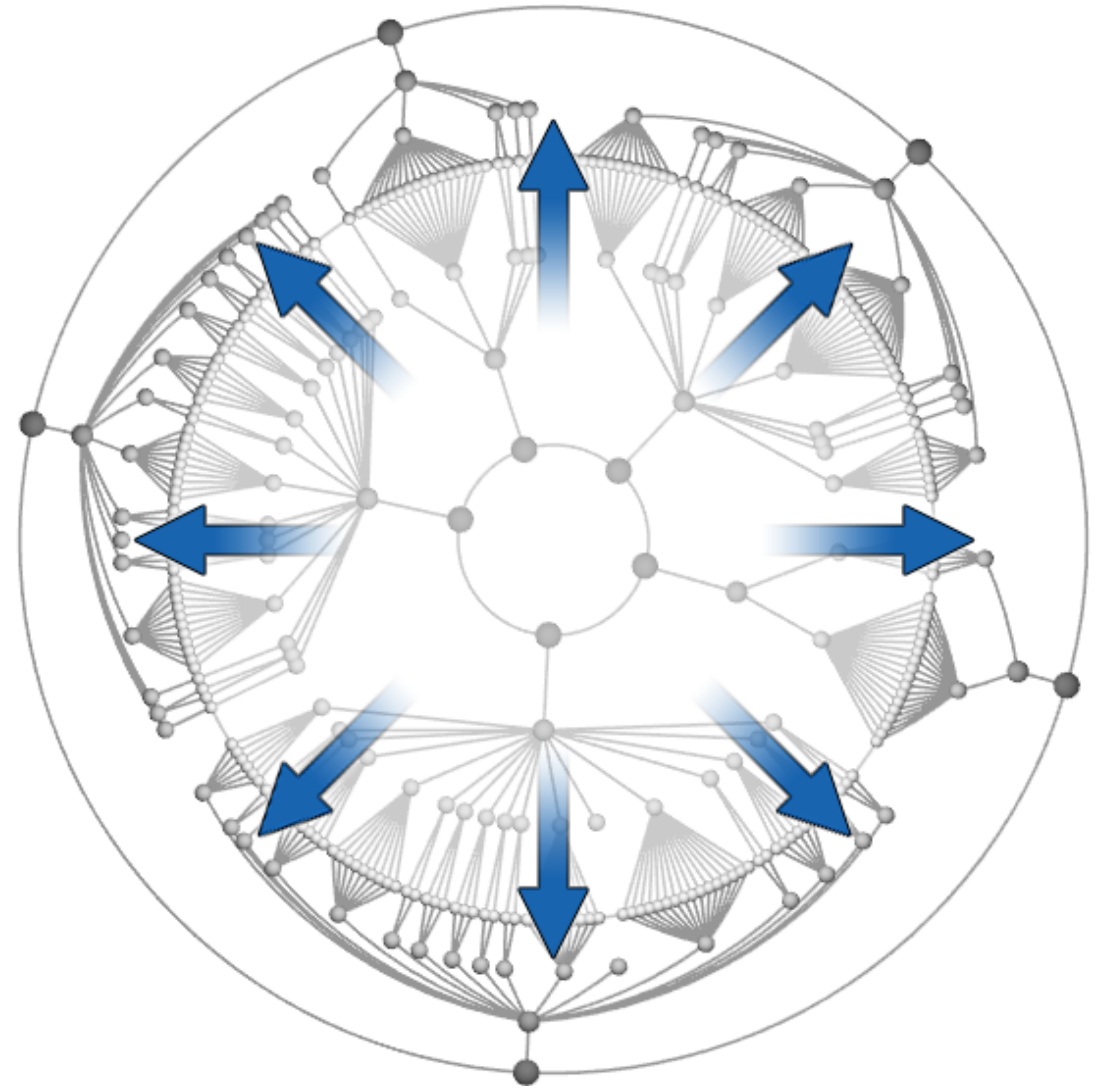
Reduce Clutter: Edge Bundling

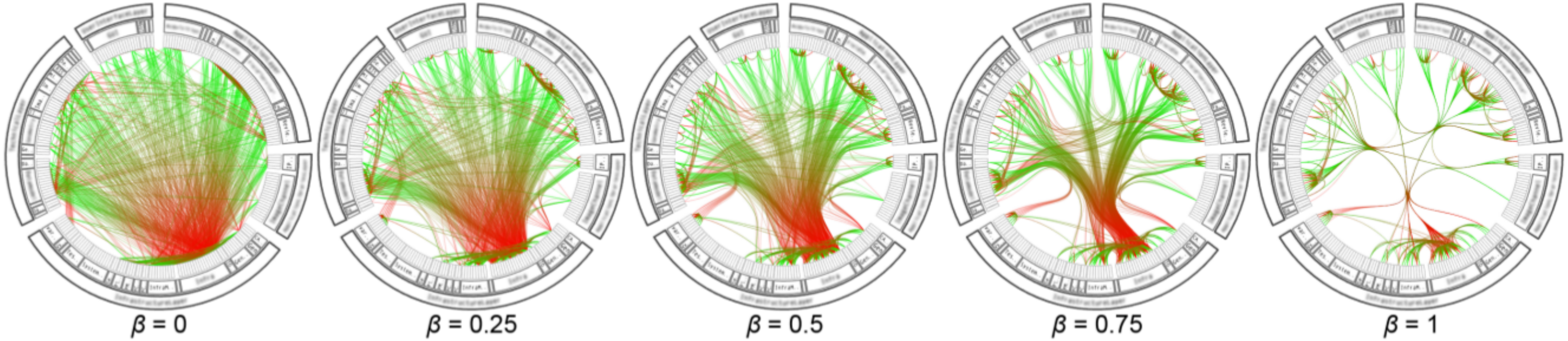


Holten et al. 2006

Hierarchical Edge Bundling



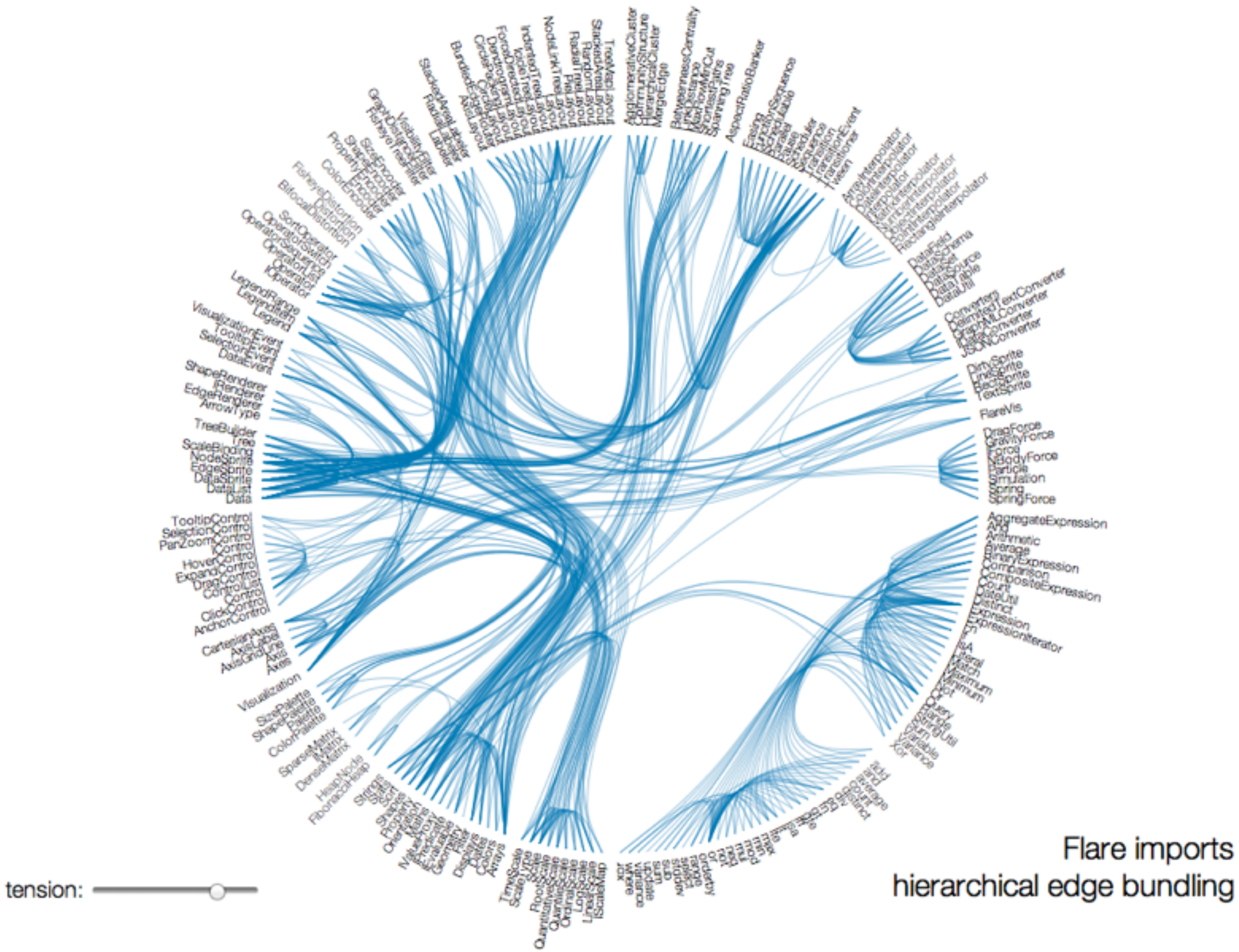




Bundling Strength

Holten et al. 2006

Bundling Strength



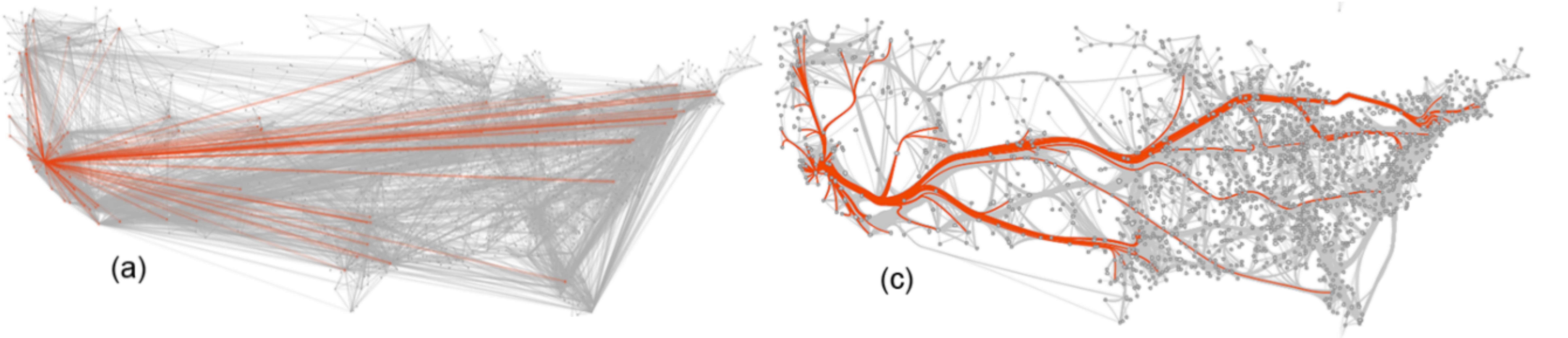
mbostock.github.com/d3/talk/20111116/bundle.html

Michael Bostock

Fixed Layouts

Can’t vary position of nodes

Edge routing important

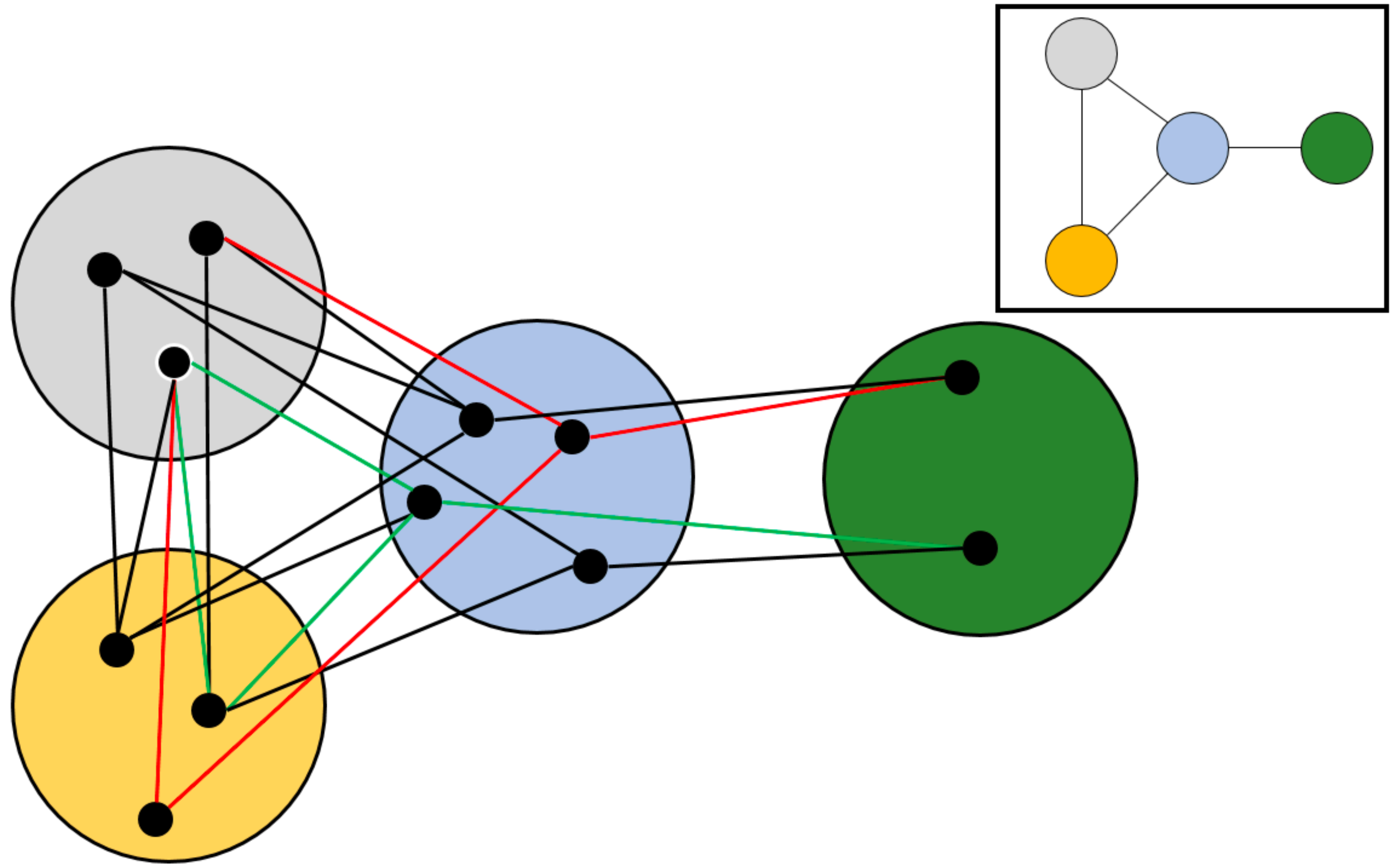


Supernodes / Aggregation

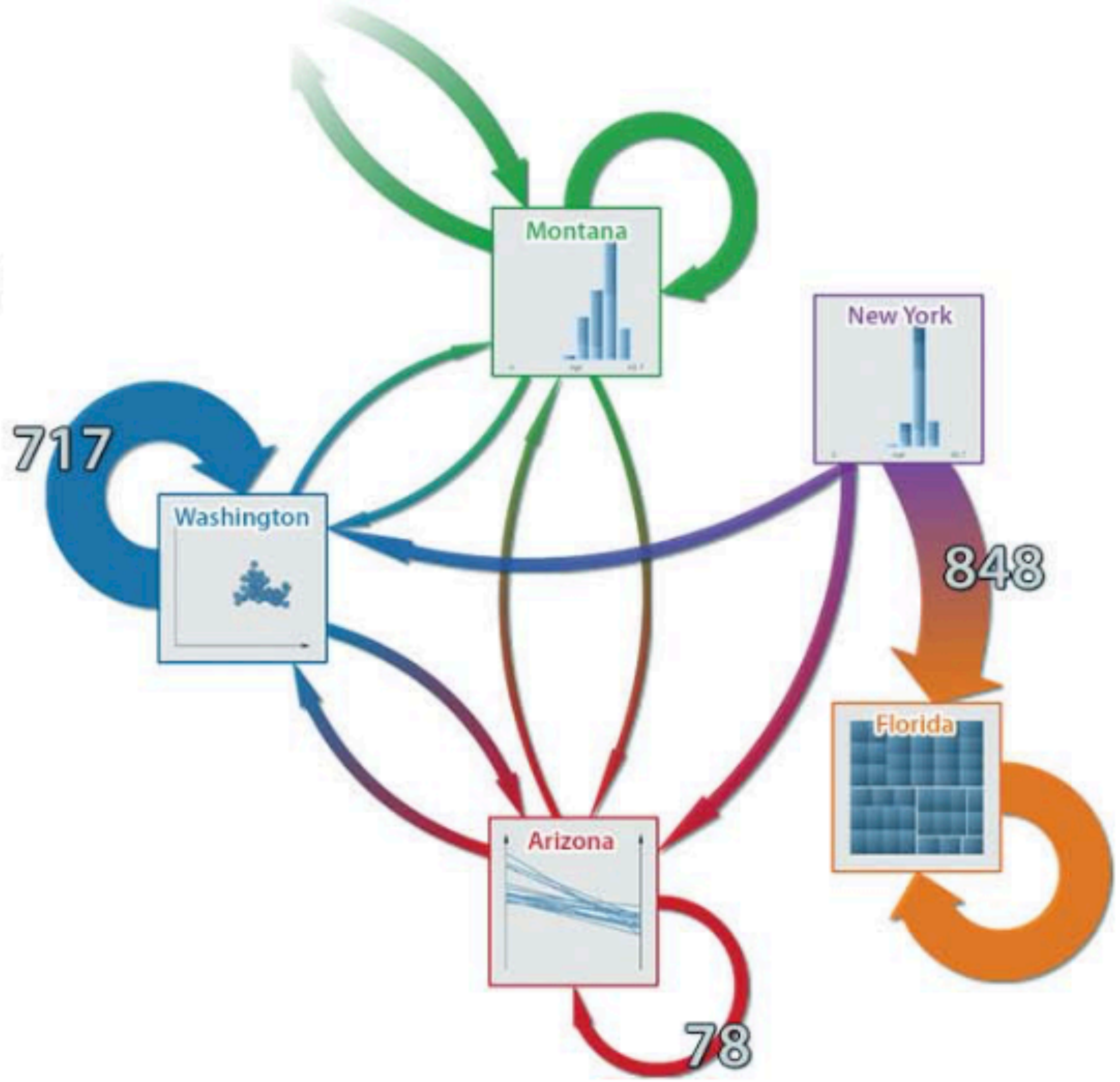
Supernodes: aggregate of nodes

manual or algorithmic

- clustering



Aggregation



https://youtu.be/E1PVTitj7h0?t=57

Explicit Representations

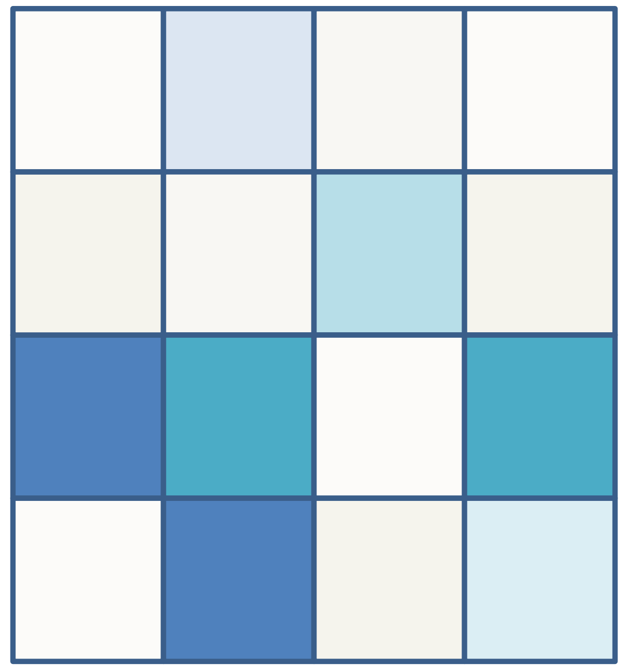
Pros:

- 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

Cons:

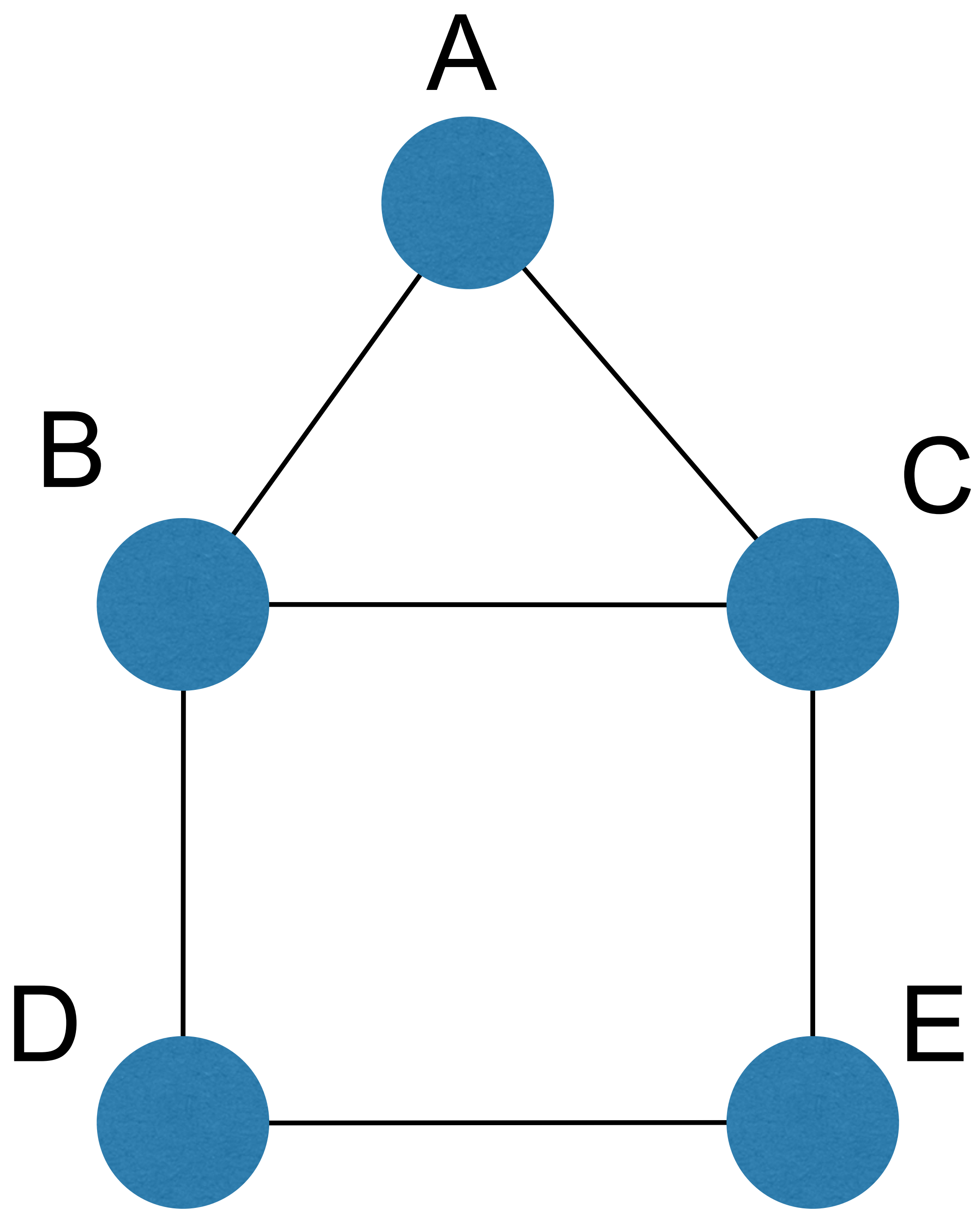
- computation of an optimal graph layout is in NP
  (even just achieving minimal edge crossings is already in NP)
- even heuristics are still slow/complex (e.g., naïve spring embedder is in O(n3))
- has a tendency to clutter (edge clutter, “hairball”)

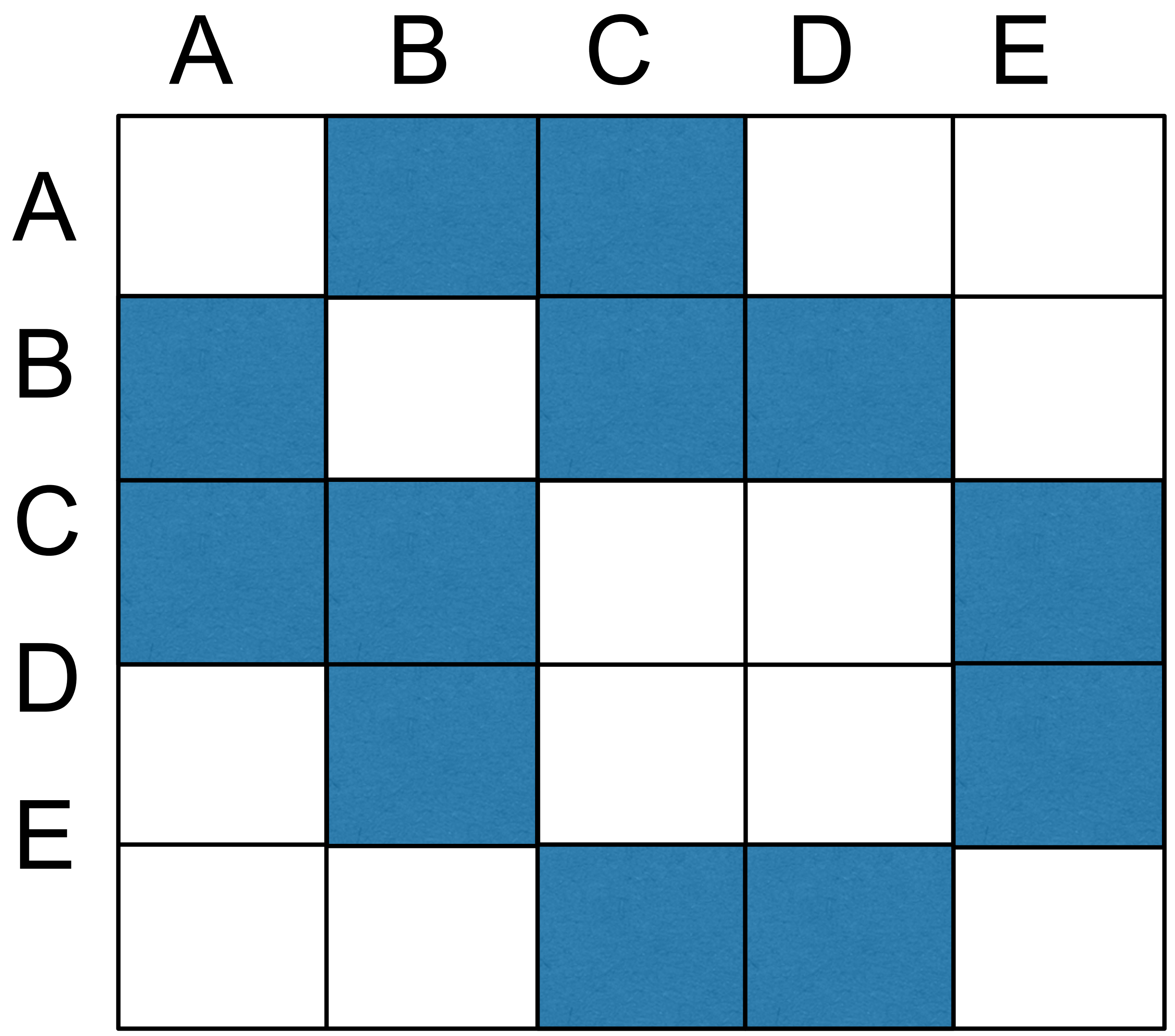
Matrix Representations



Matrix Representations

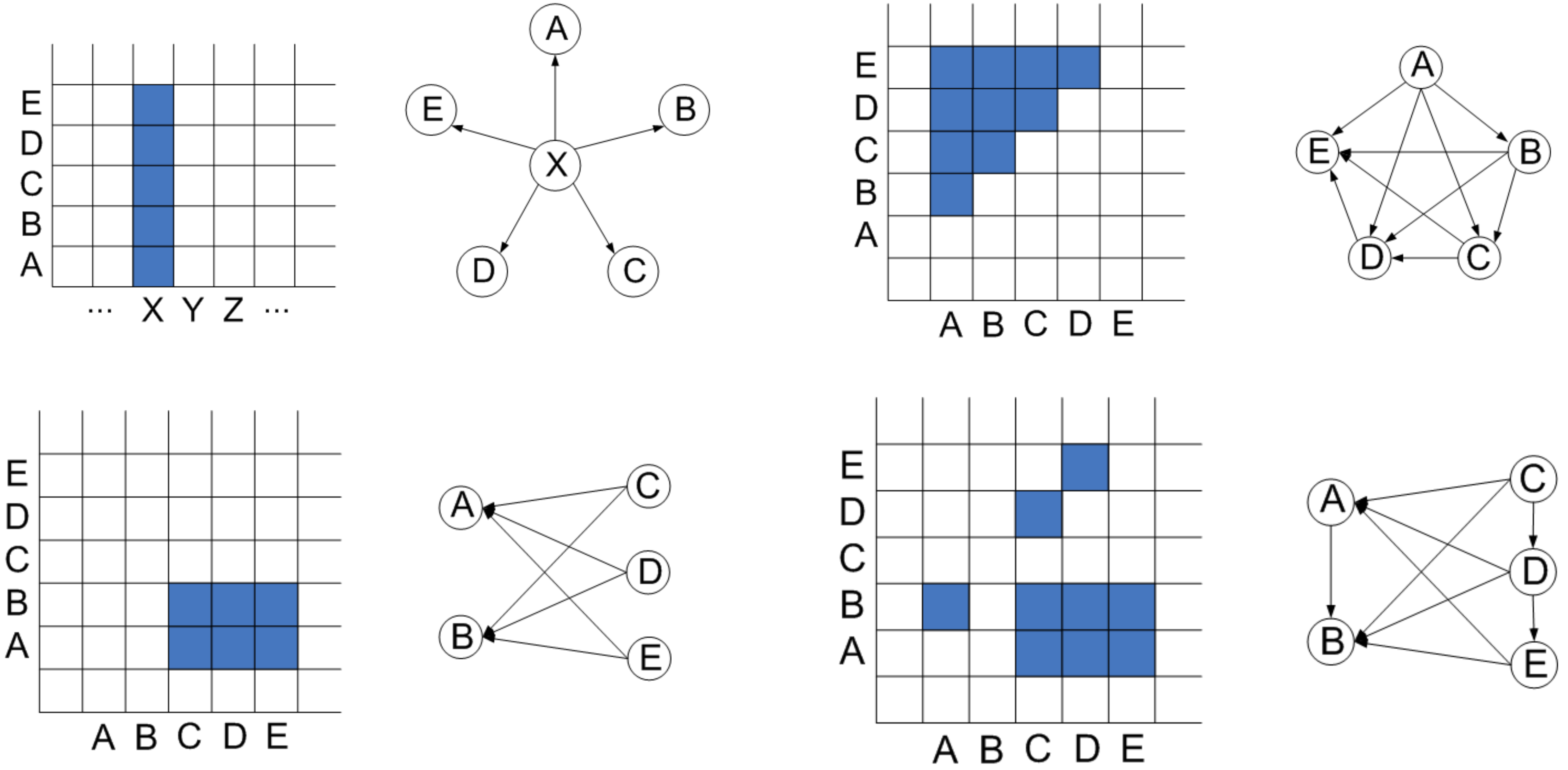
Instead of node link diagram, use adjacency matrix





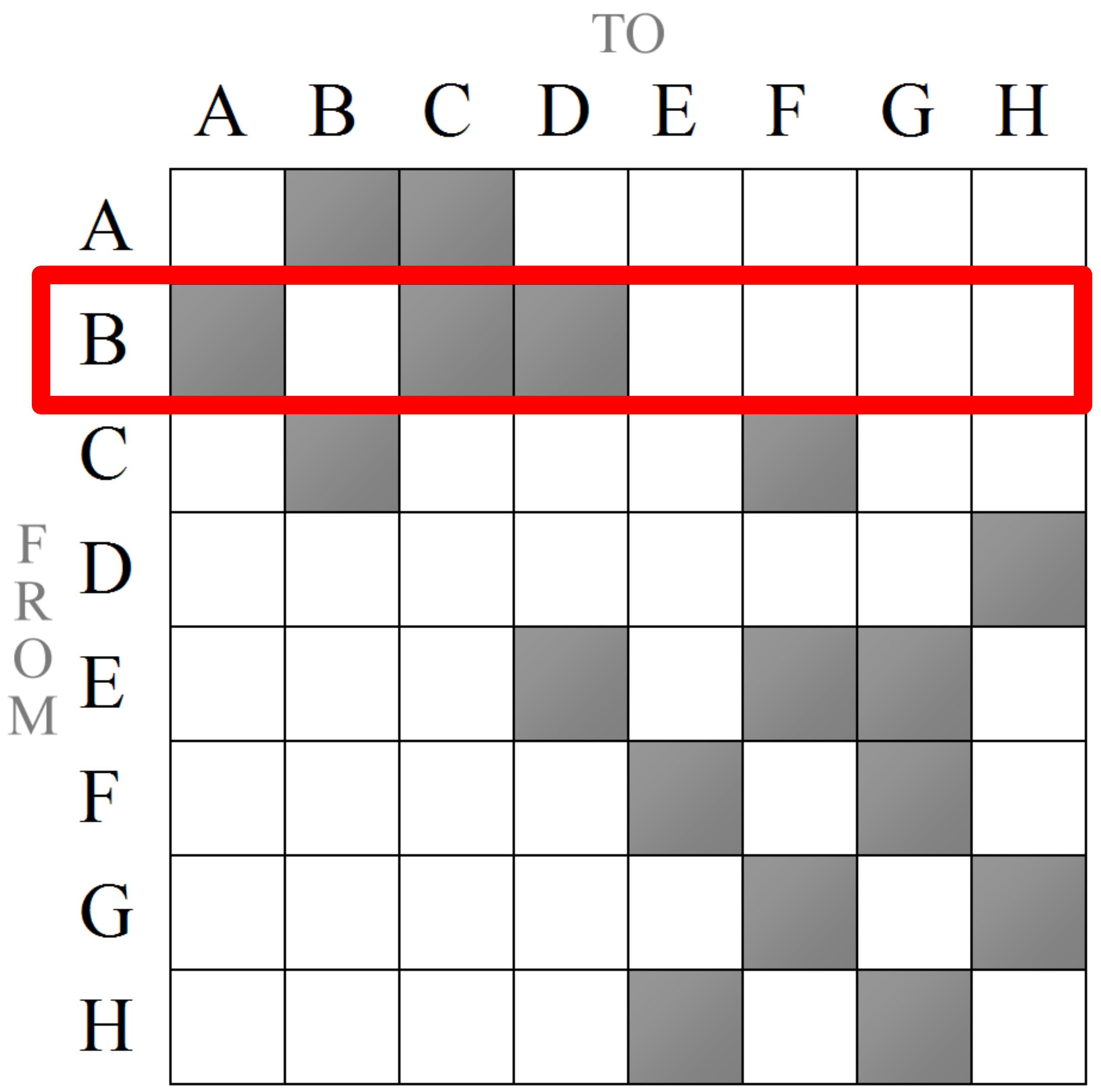
Matrix Representations

Examples:

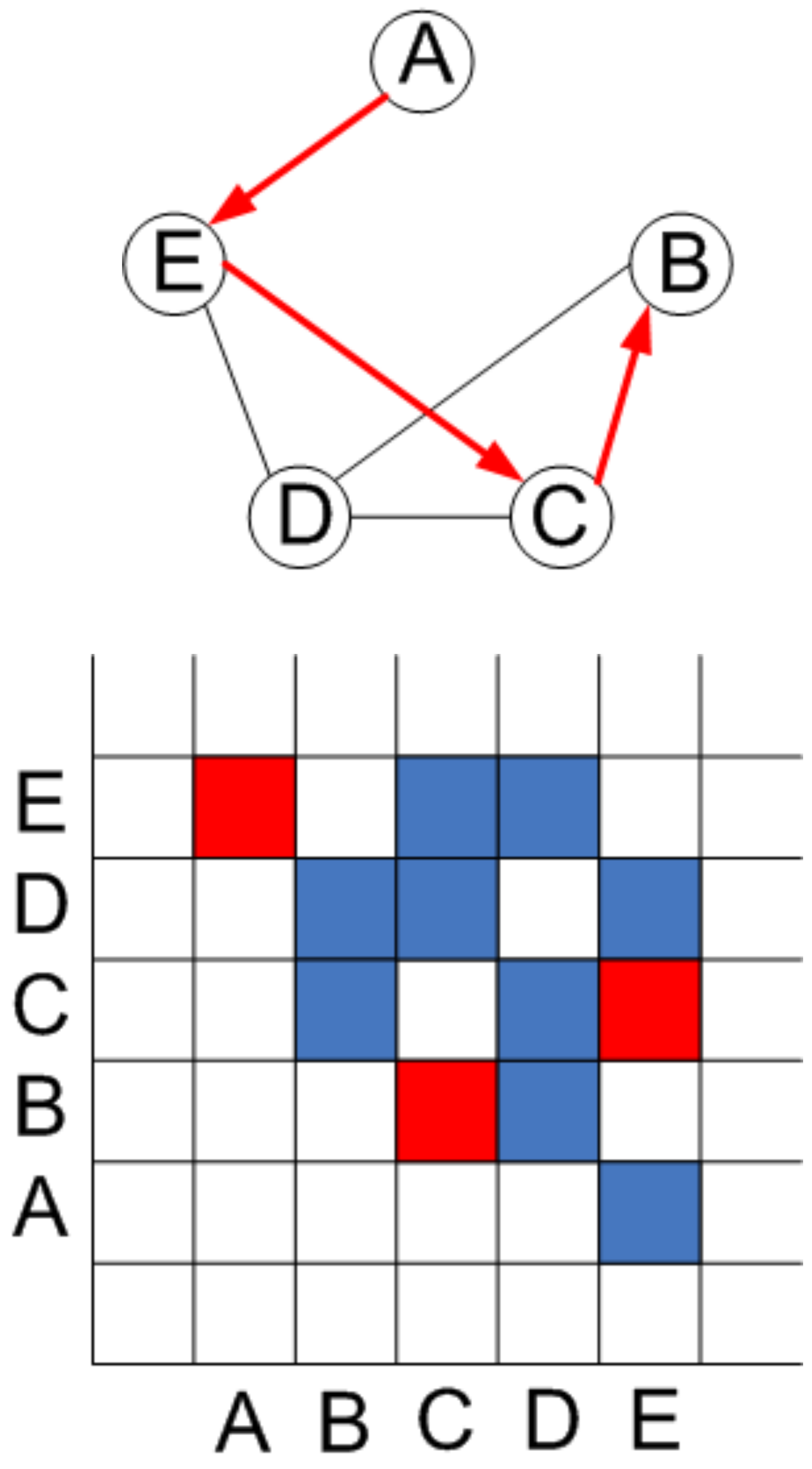


HJ Schulz 2007

Matrix Representations

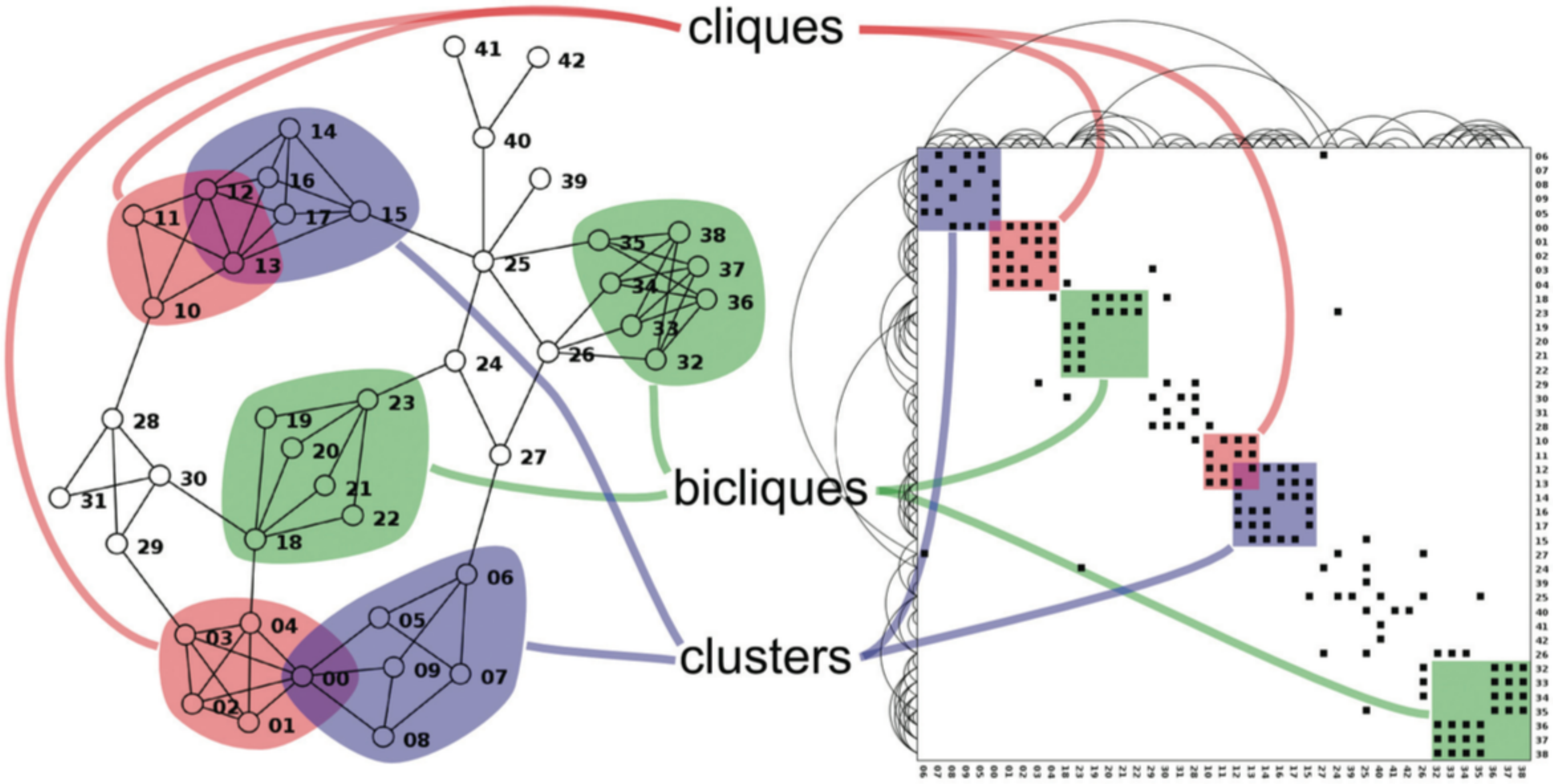


Well suited for
neighborhood-related TBTs



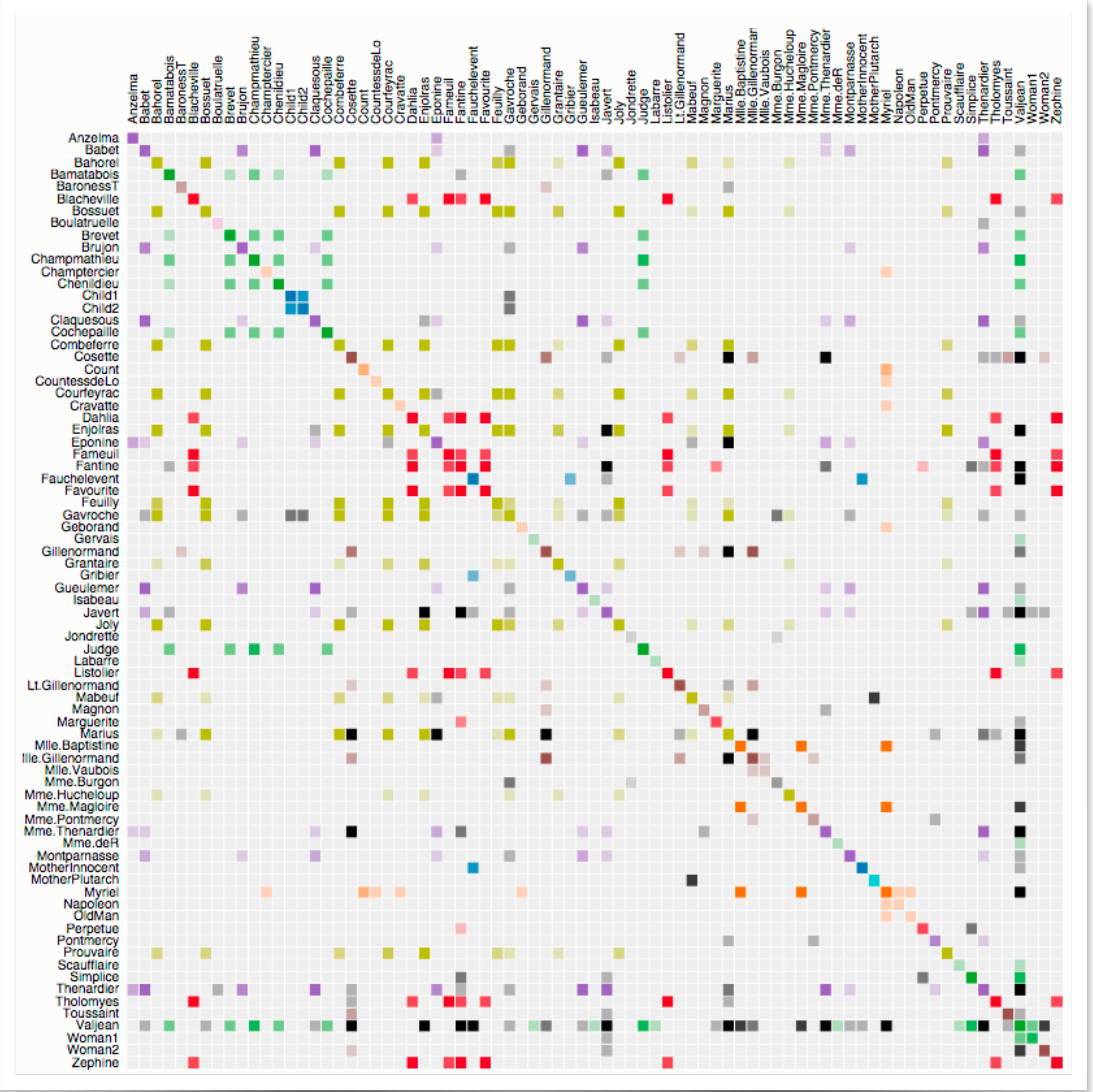
Not suited for
path-related TBTs

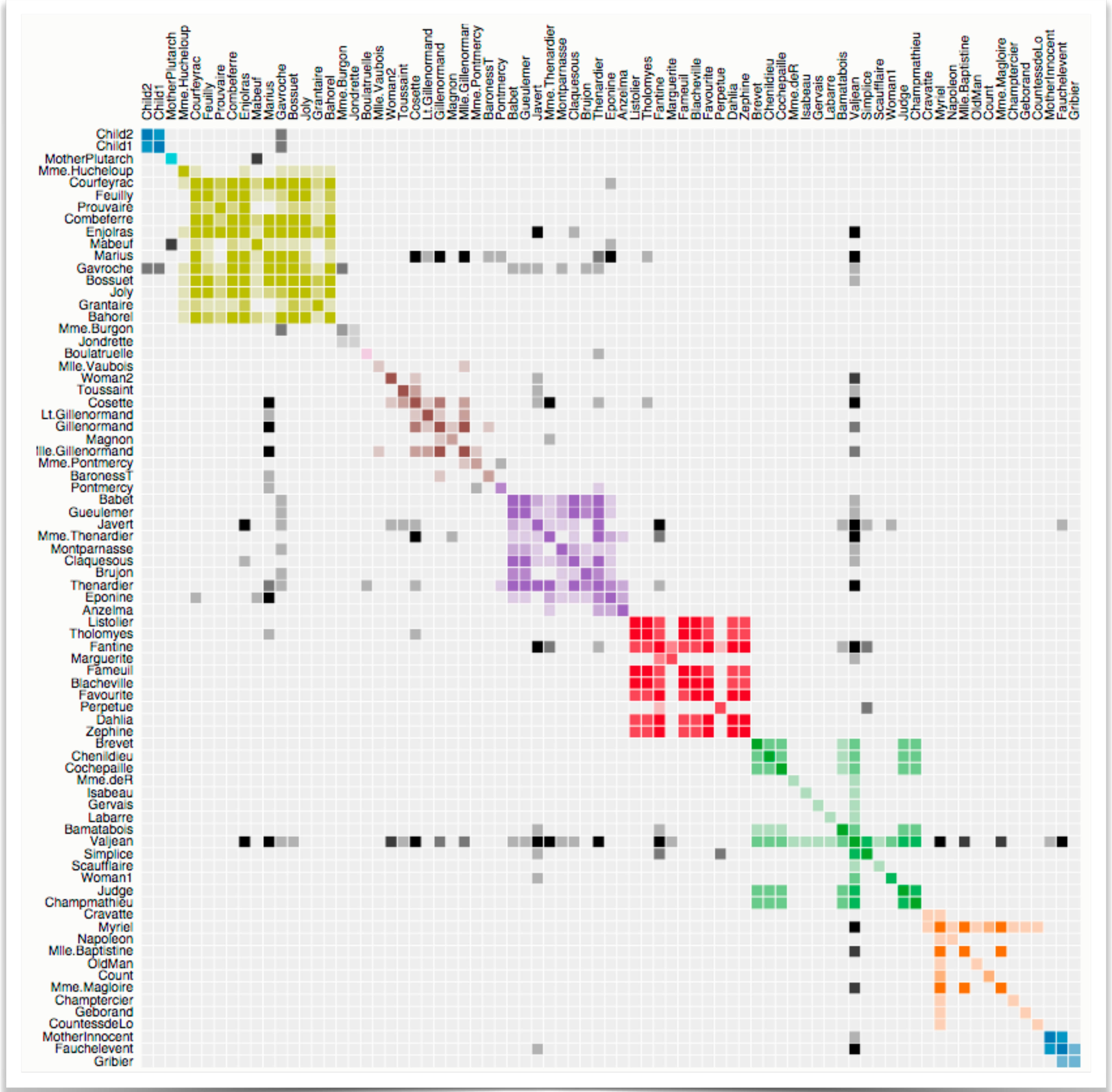
van Ham et al. 2009
Shen et al. 2007



McGuffin 2012

Order Critical!





Matrix Representations

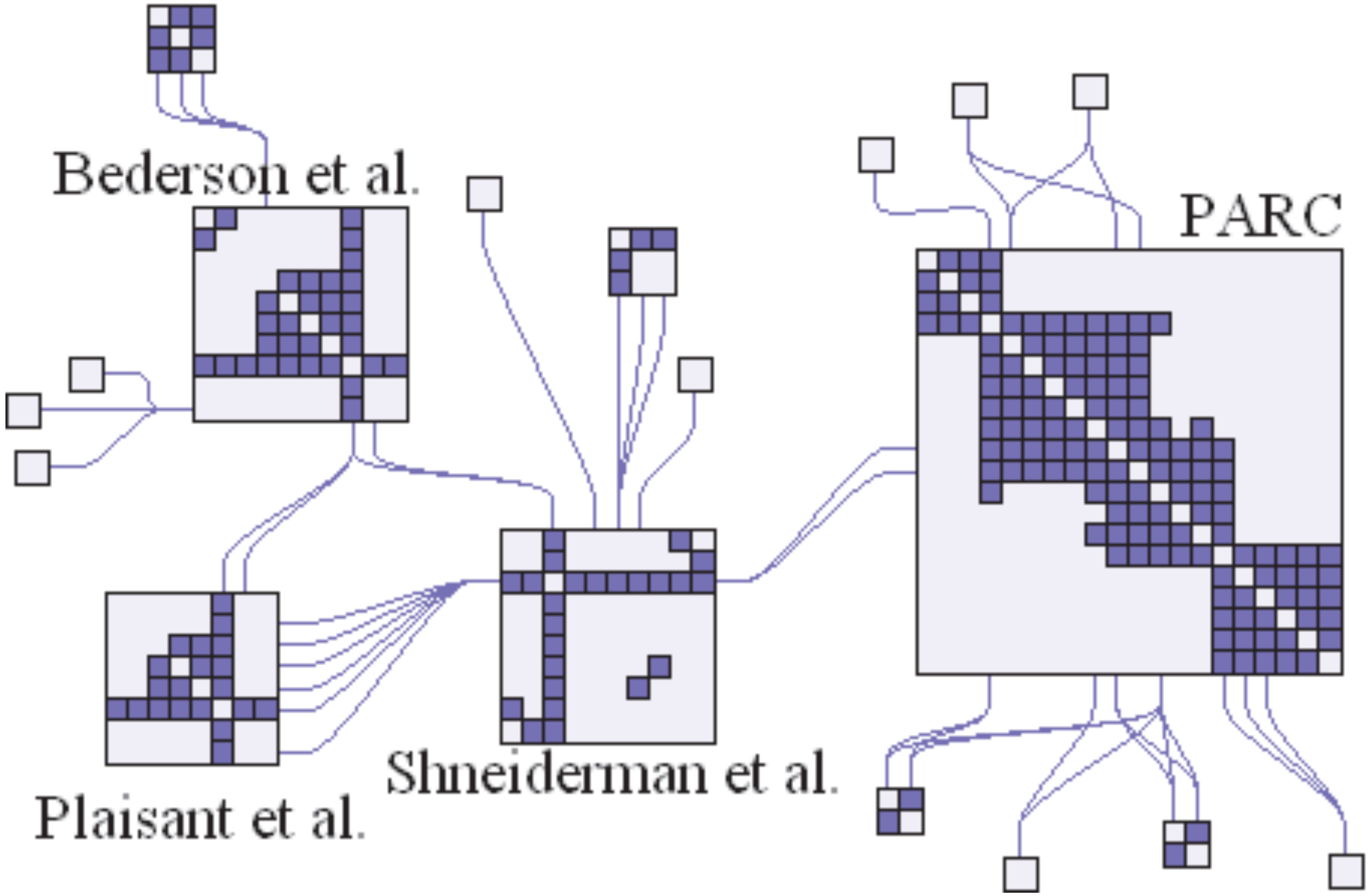
Pros:

- 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

Hybrid Explicit/Matrix

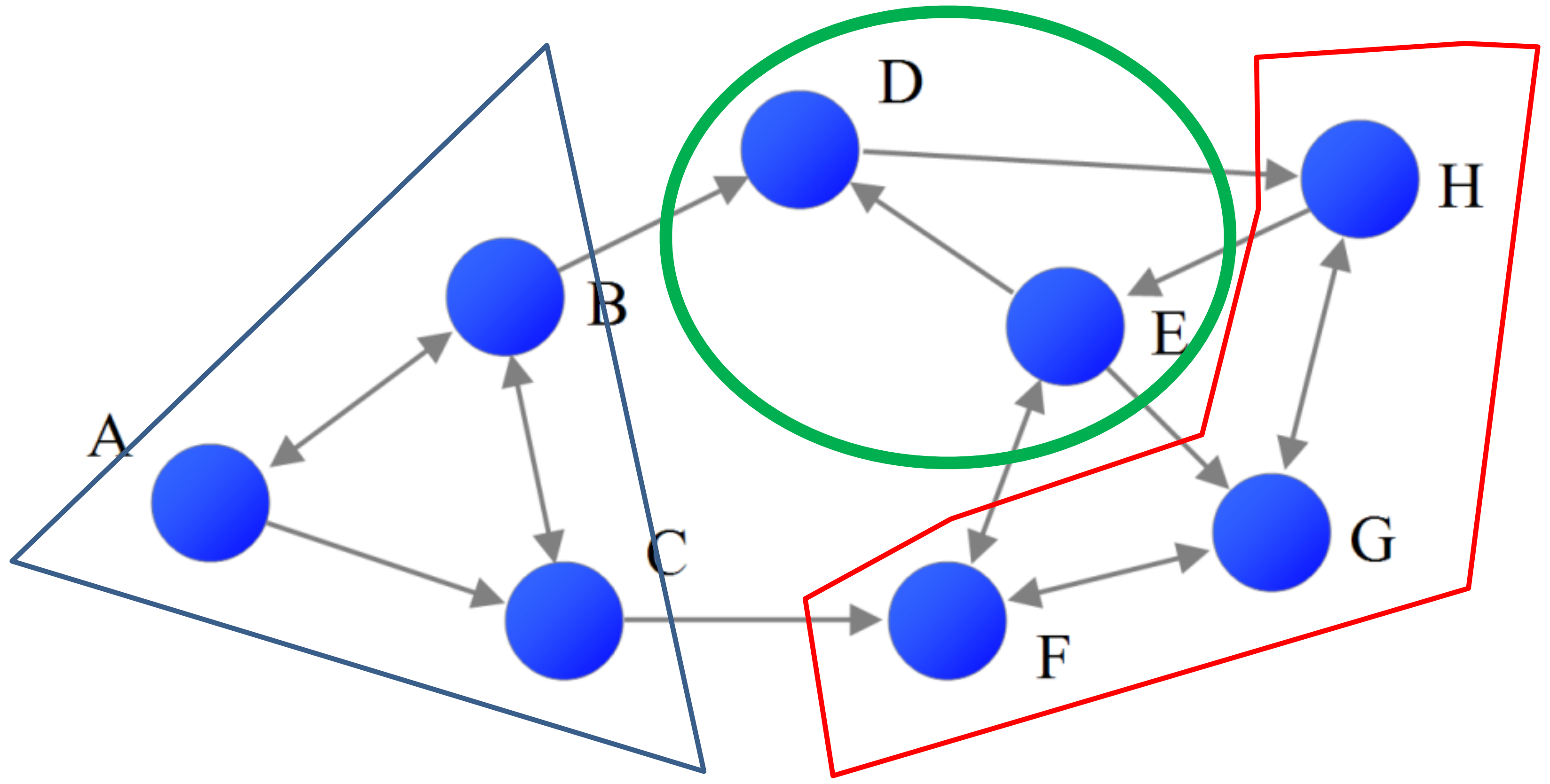


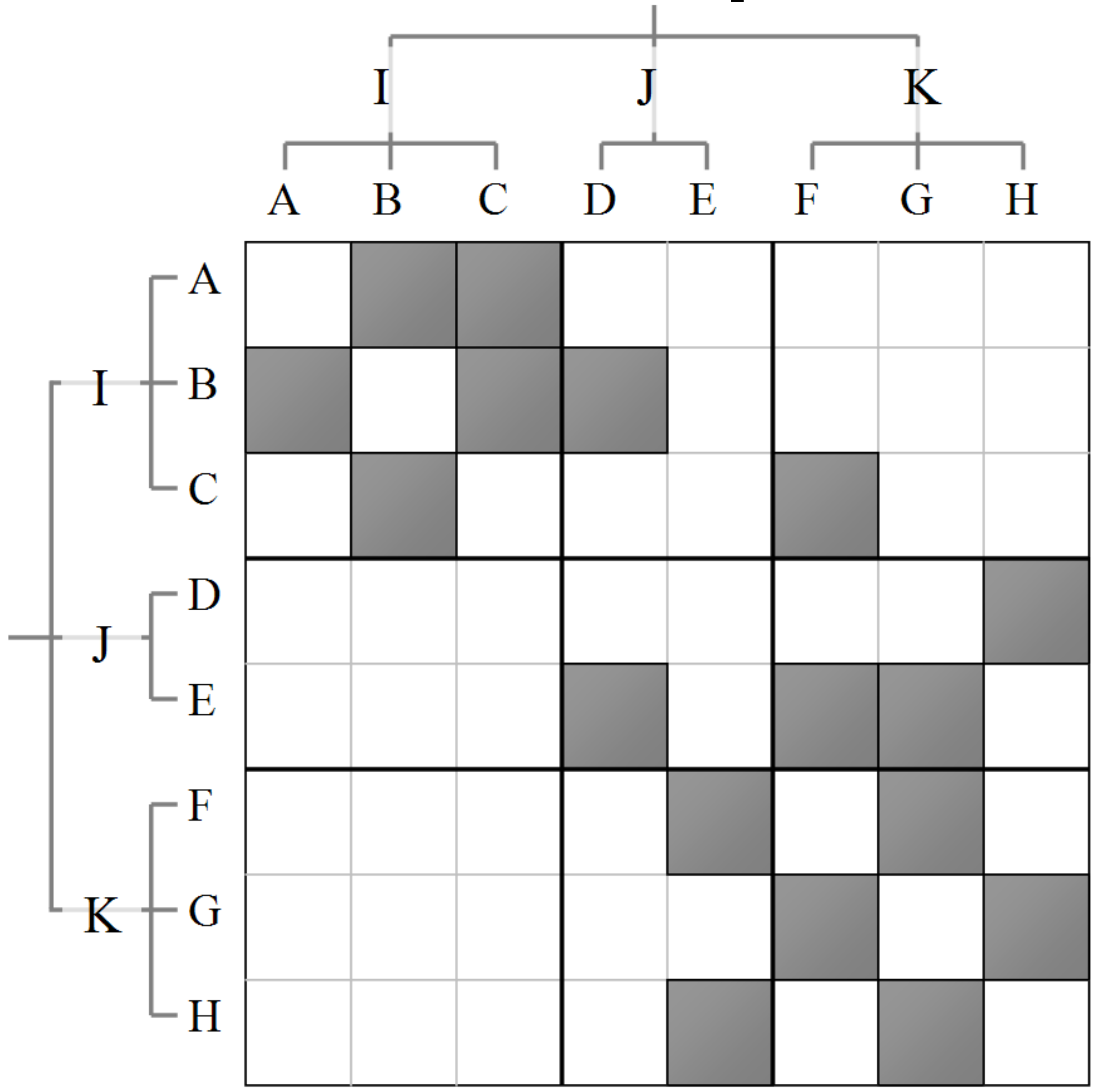
NodeTrix
[Henry et al. 2007]

Matrix Representations

Problem: used screen real estate is quadratic in the number of nodes

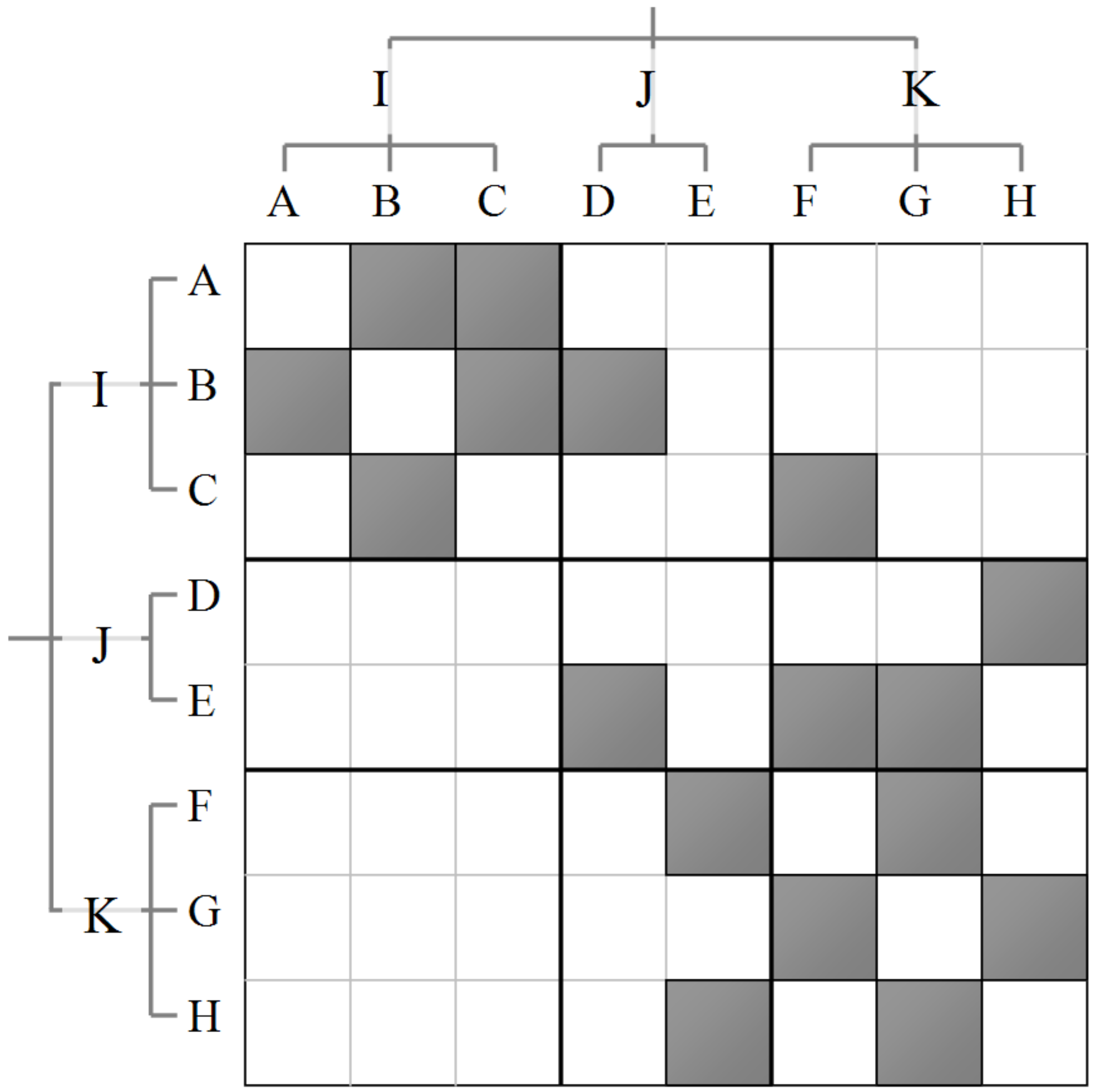
Solution approach: hierarchization of the representation

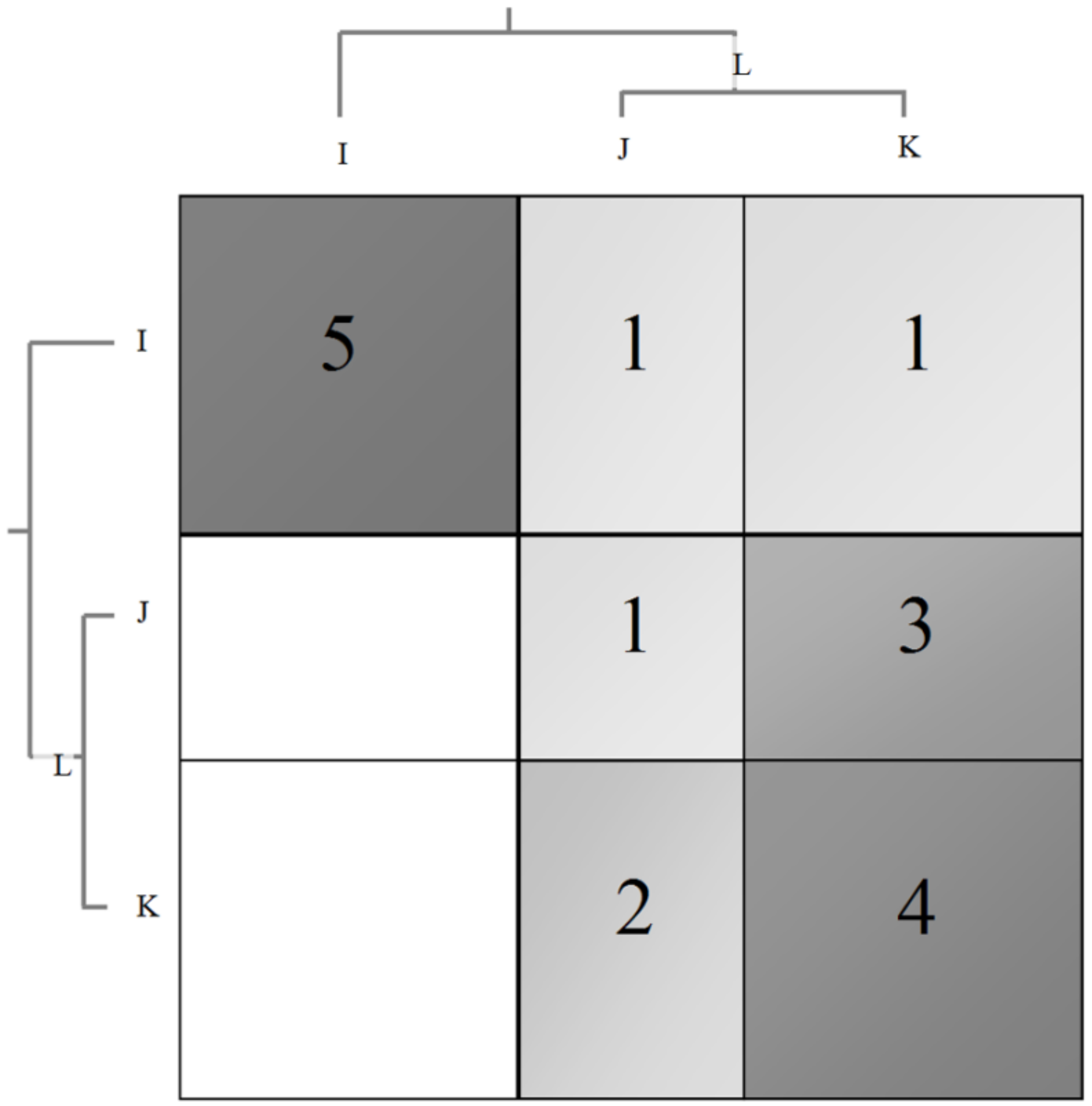




[van Ham et al. 2009]

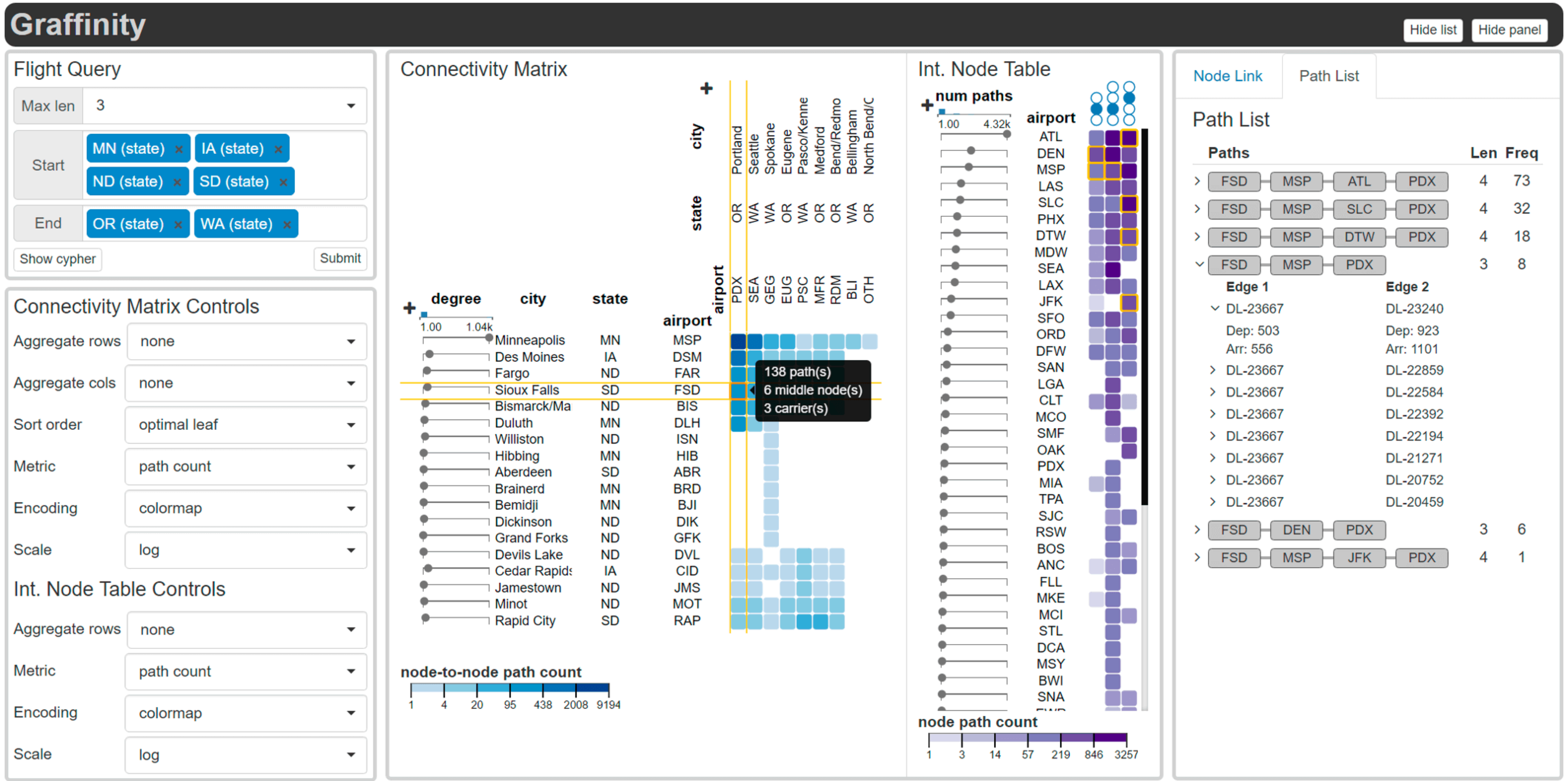
Matrix Representations





[van Ham et al. 2009]

Higher-Order Connectivity



[Kerzner et al., Graffinity, 2017]

Trees

Tree-Exercise

Tree Exercise

Here is part of a directory structure used for the material for this class and the relative file size.

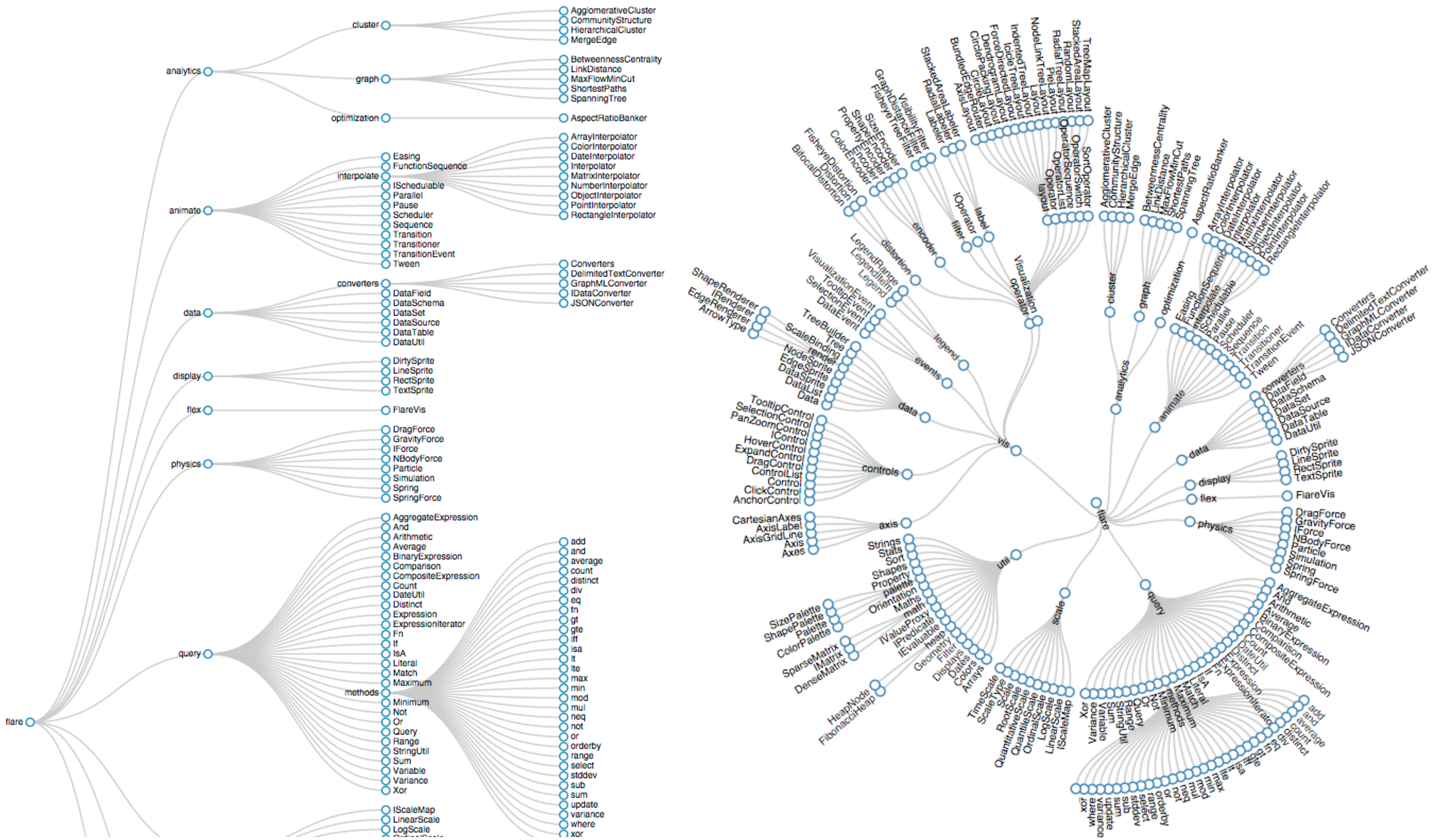
- datavis-17/
  - lectures/
    - Intro.key (110 MB)
    - perception/
      - Perception.key (113 MB)
      - Blindness.mov (15MB)
    - Data.key (12 MB)
    - Graphs.key (180 MB)
  - exams/
    - Exam1-solution.doc (5MB)
    - Exam1.doc (1MB)
  - exercise/
    - Graph.doc (3MB)
    - Graph-video.doc (210MB)

Sketch two different visualizations that show both, the directory structure and the size of the directories and the contained files.

Explicit Tree Visualization

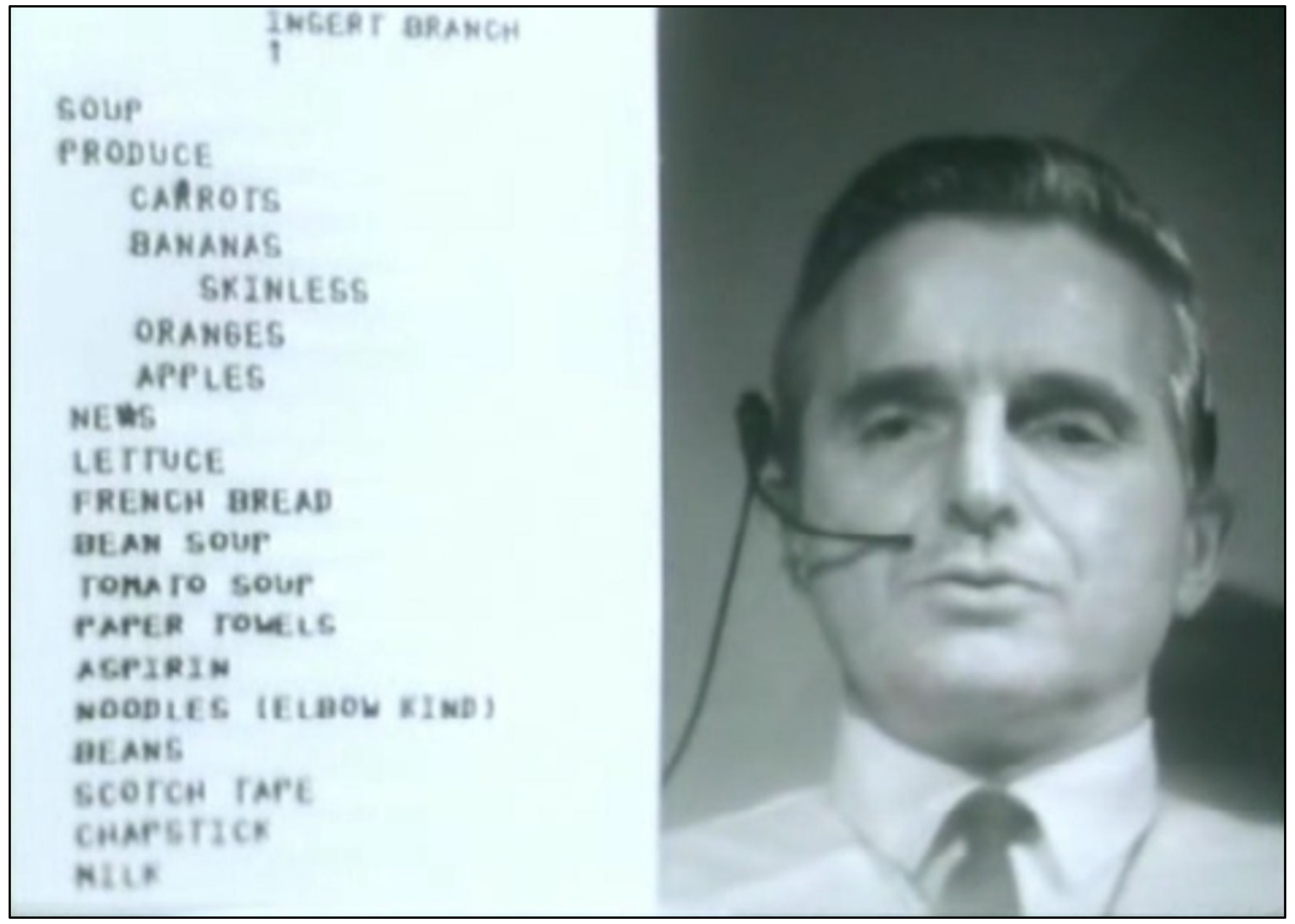
Reingold–Tilford layout

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

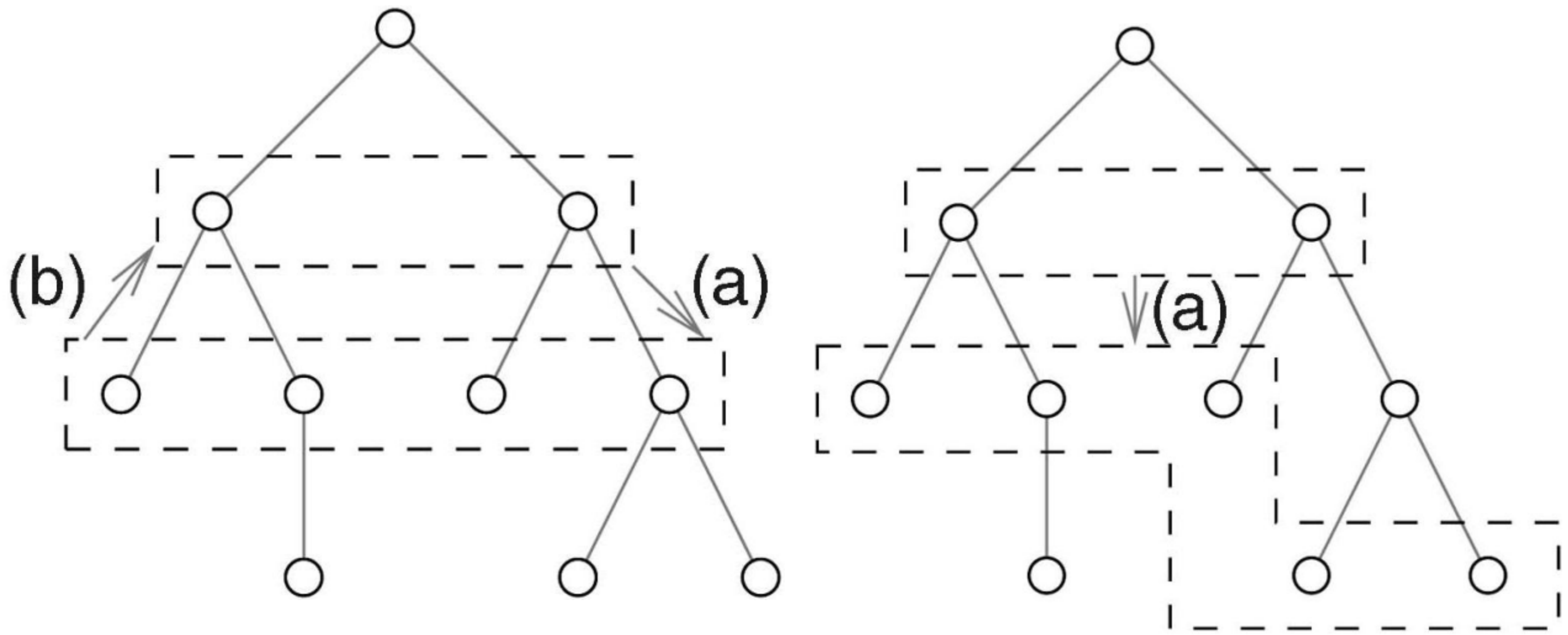


Manipulating Aggregation Levels

First interactive tree manipulation



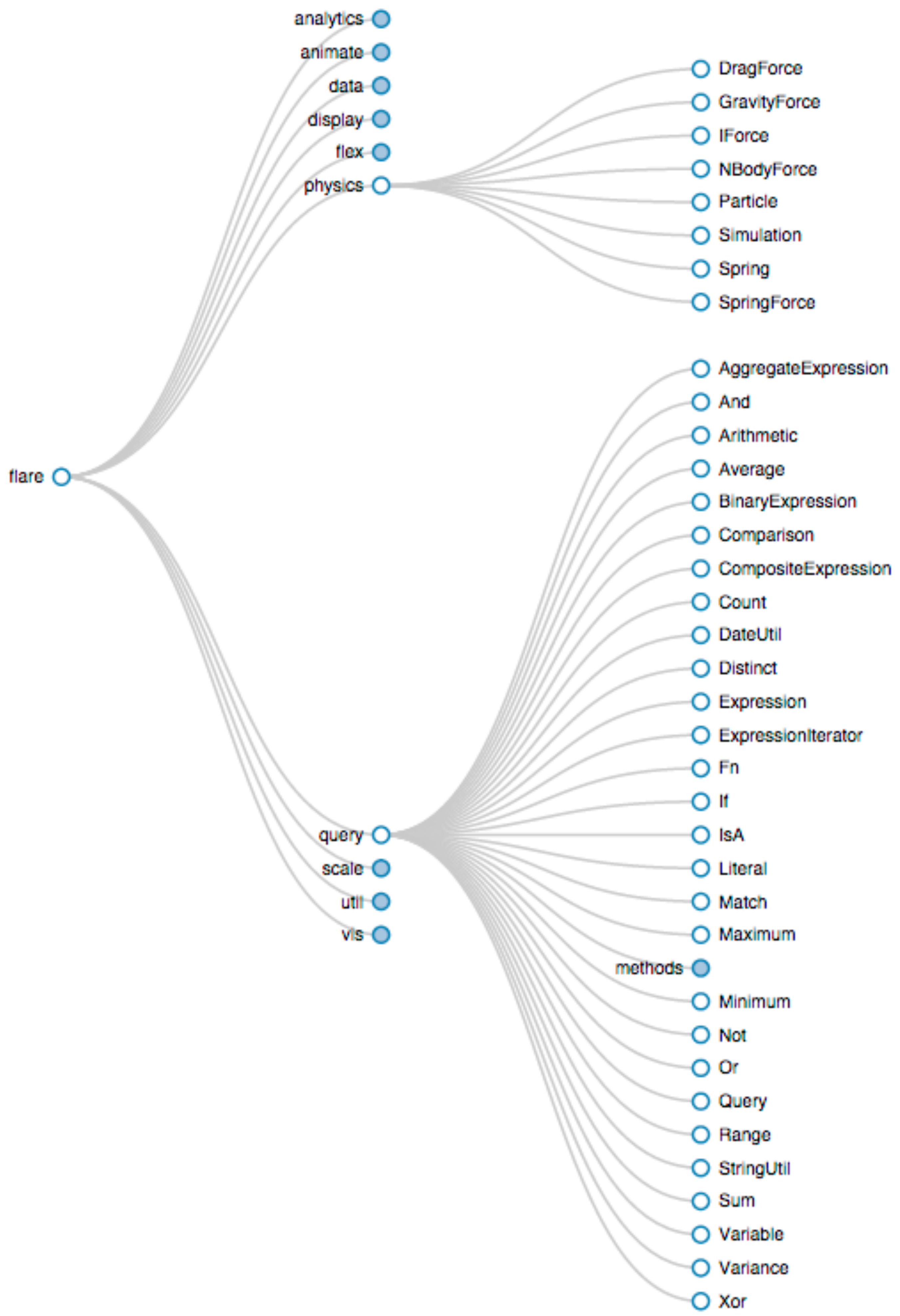
Douglas Engelbart 1968 - http://www.1968demo.org

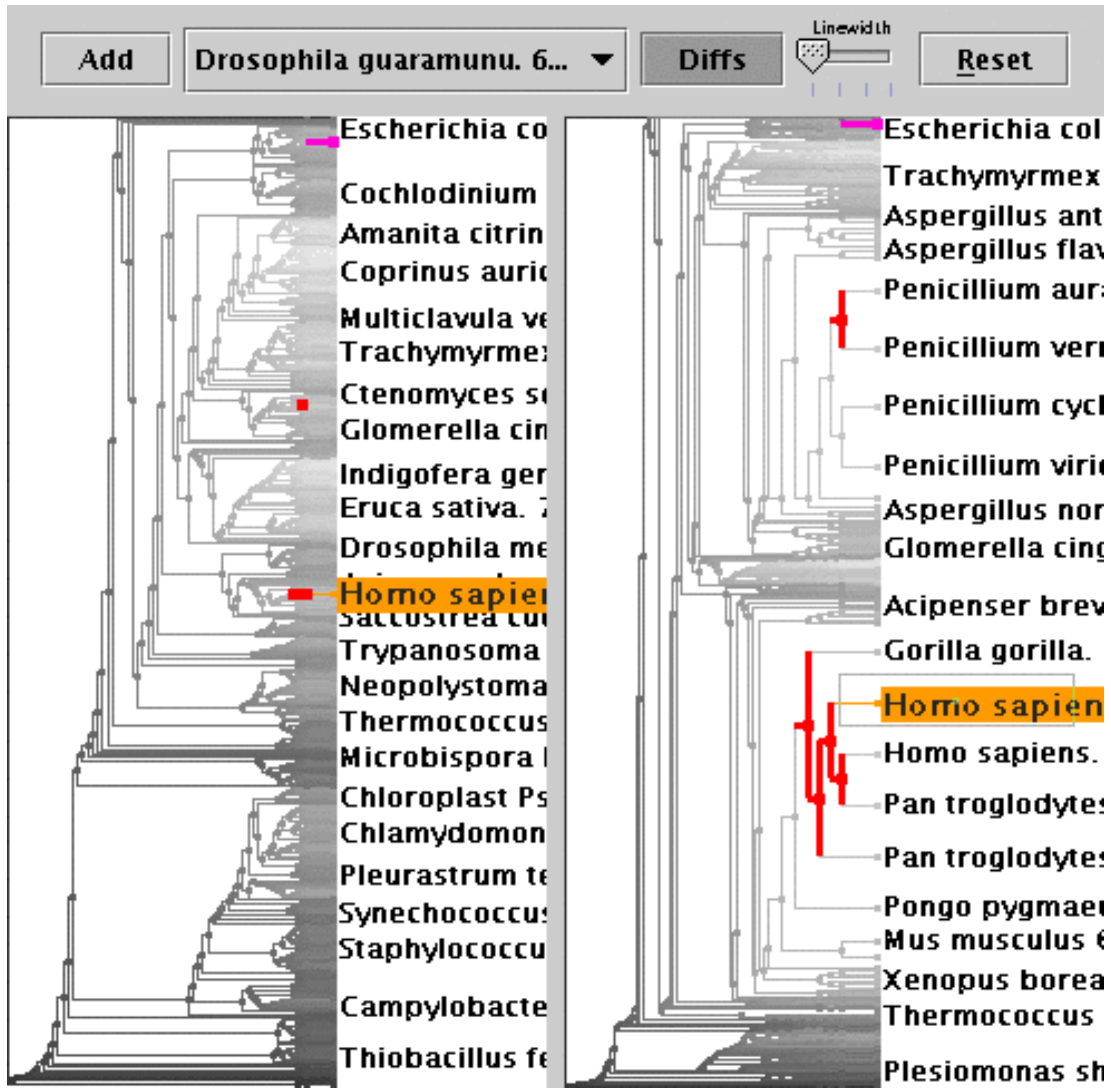


(a) Drill-Down (b) Roll-Up (a) Unbalanced Drill-Down

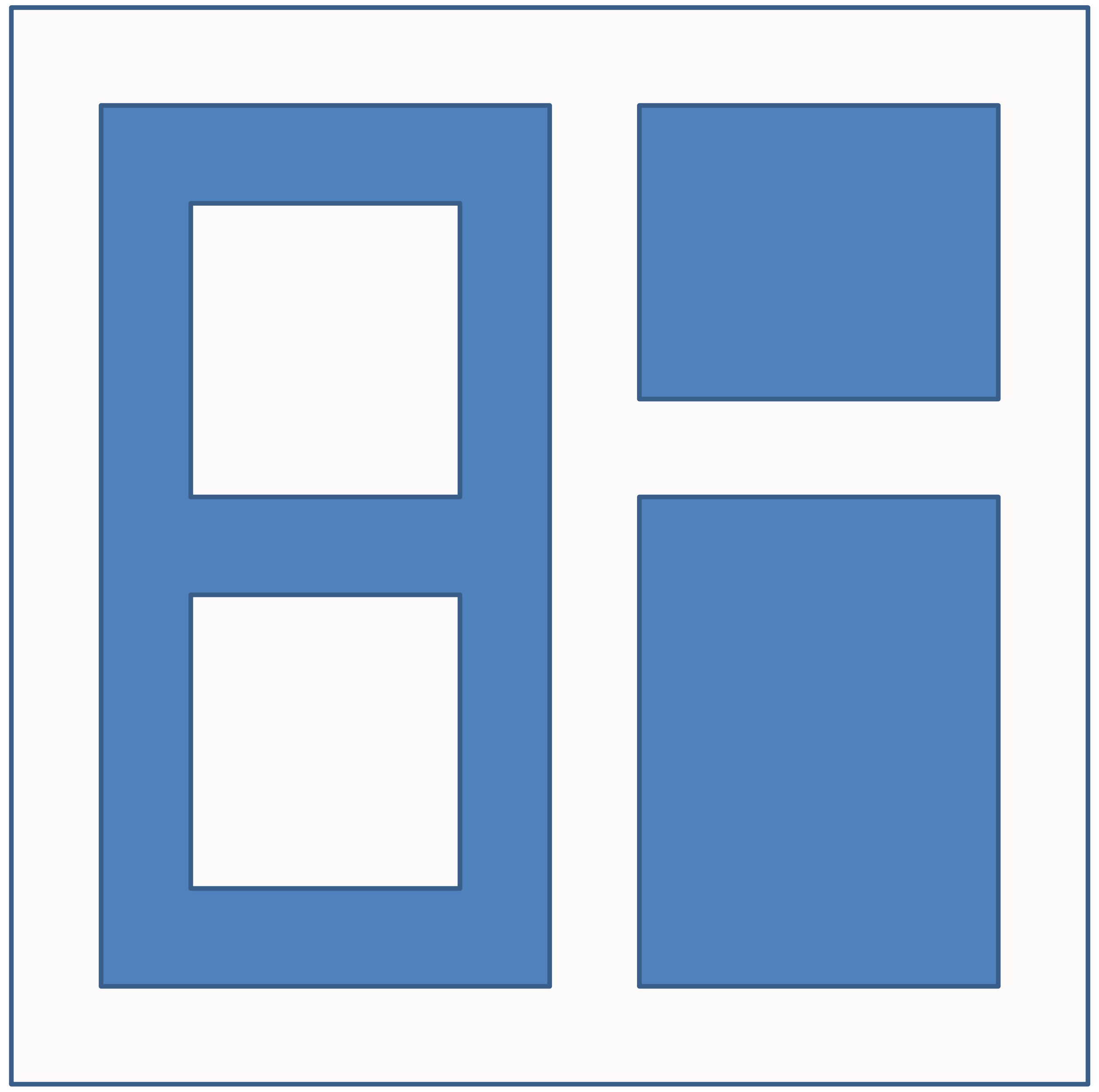
“The mother of all demos“
https://www.youtube.com/watch?v=yJDv-zdhzMY

Tree Interaction, Tree Comparison



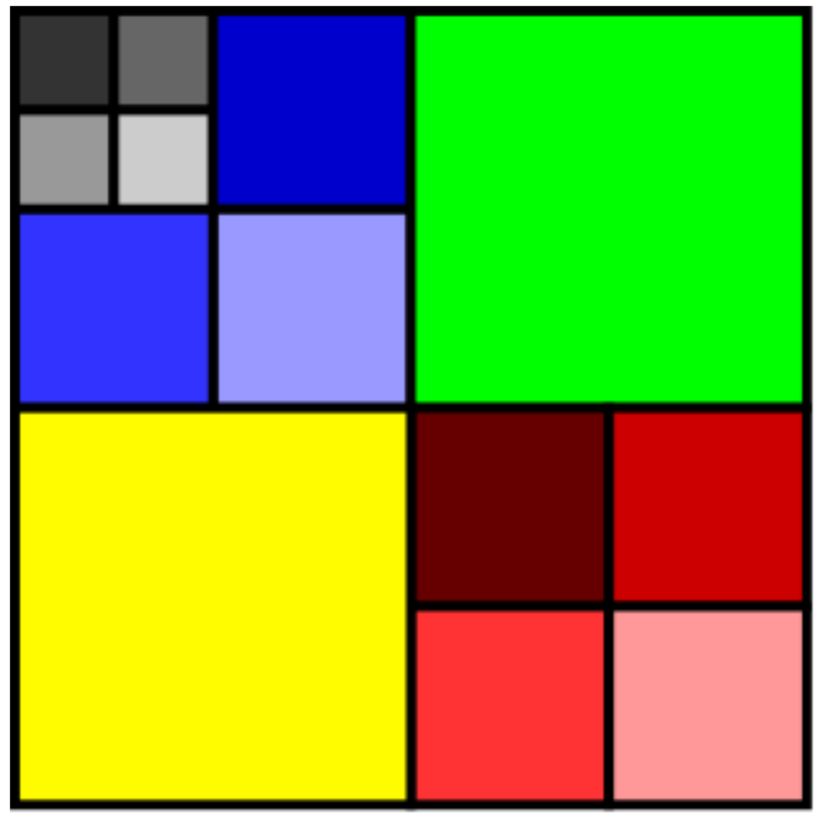


Implicit Layouts for Trees

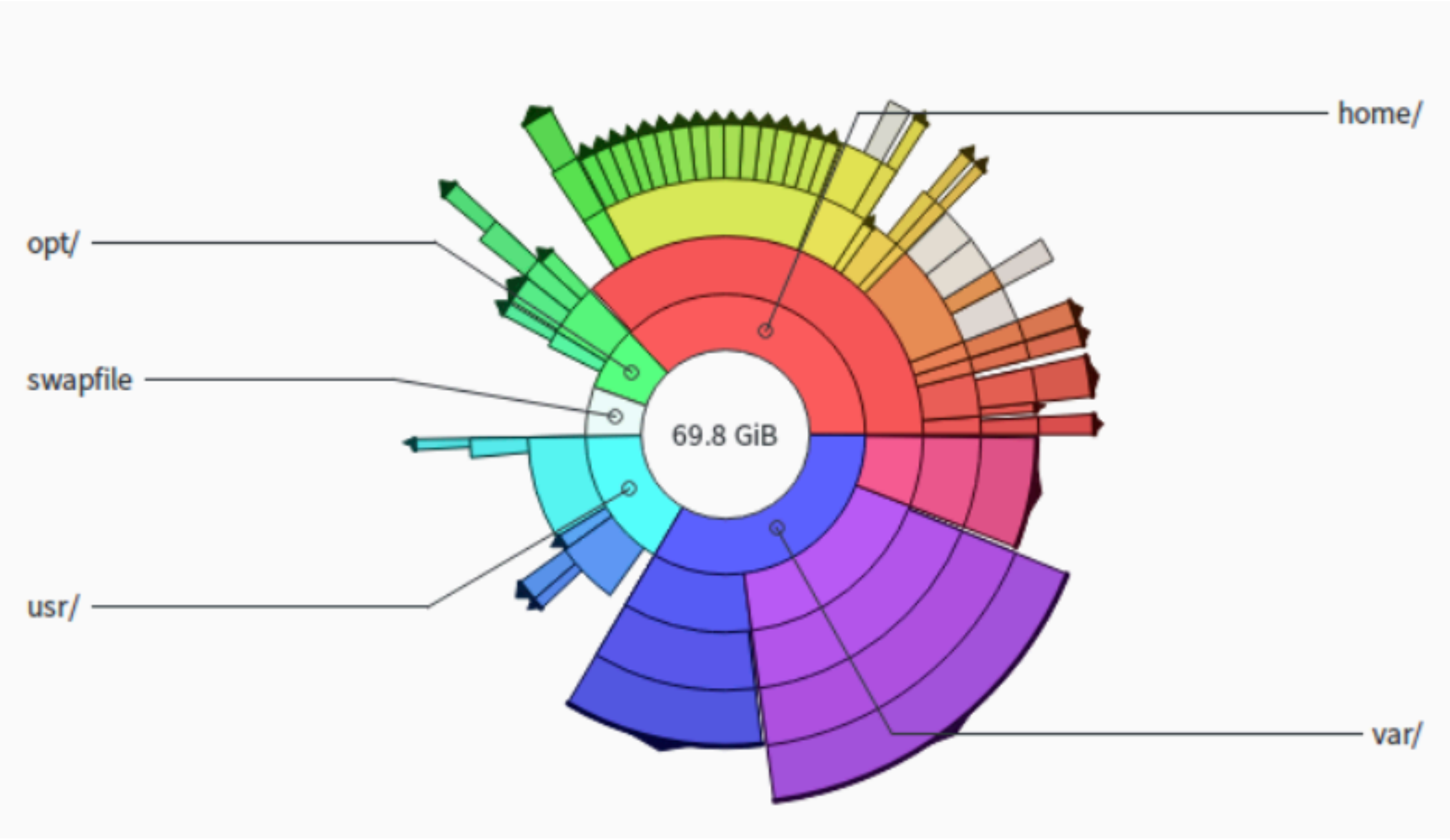


Implicit Layout Options

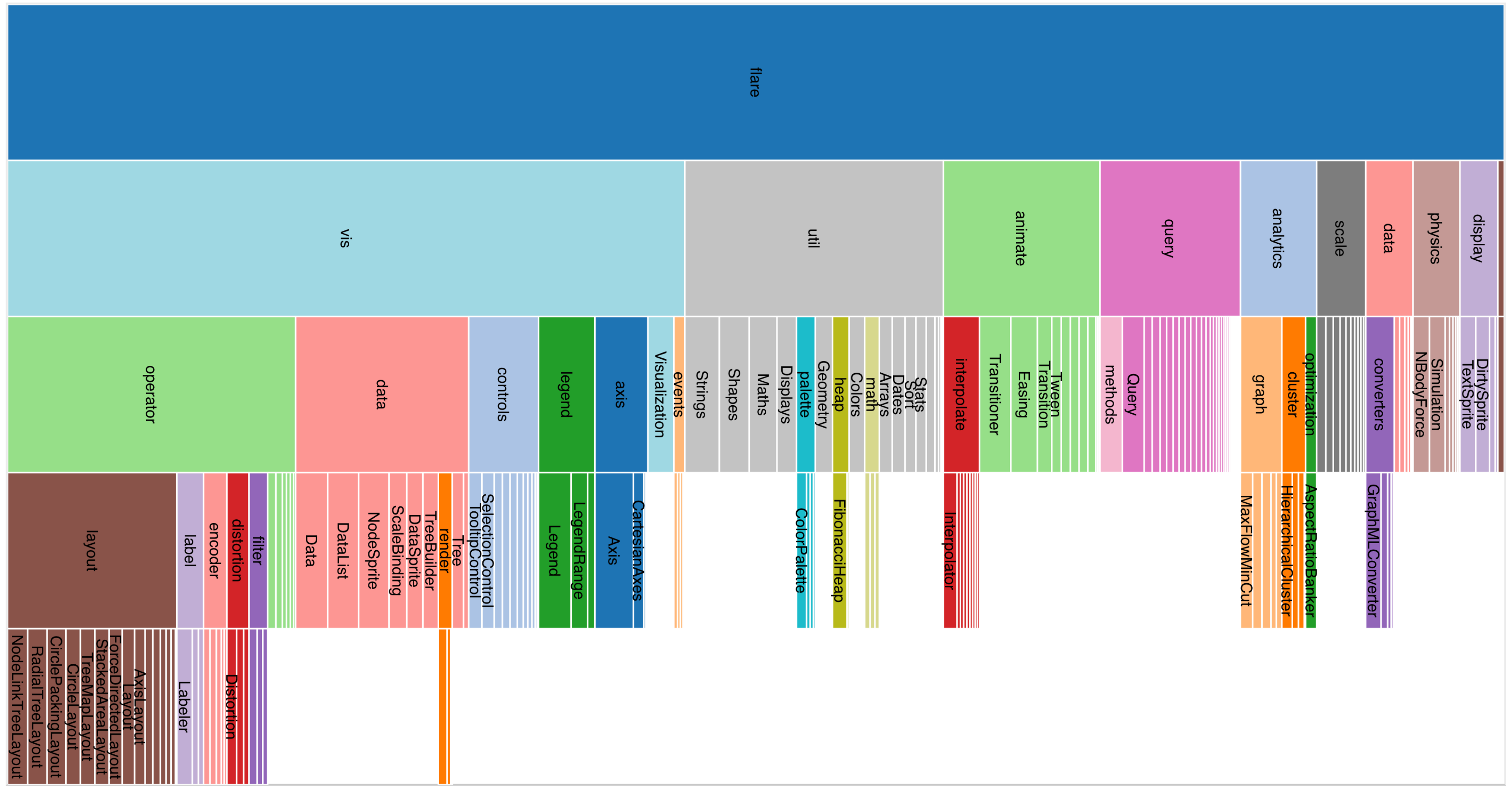
Treemap



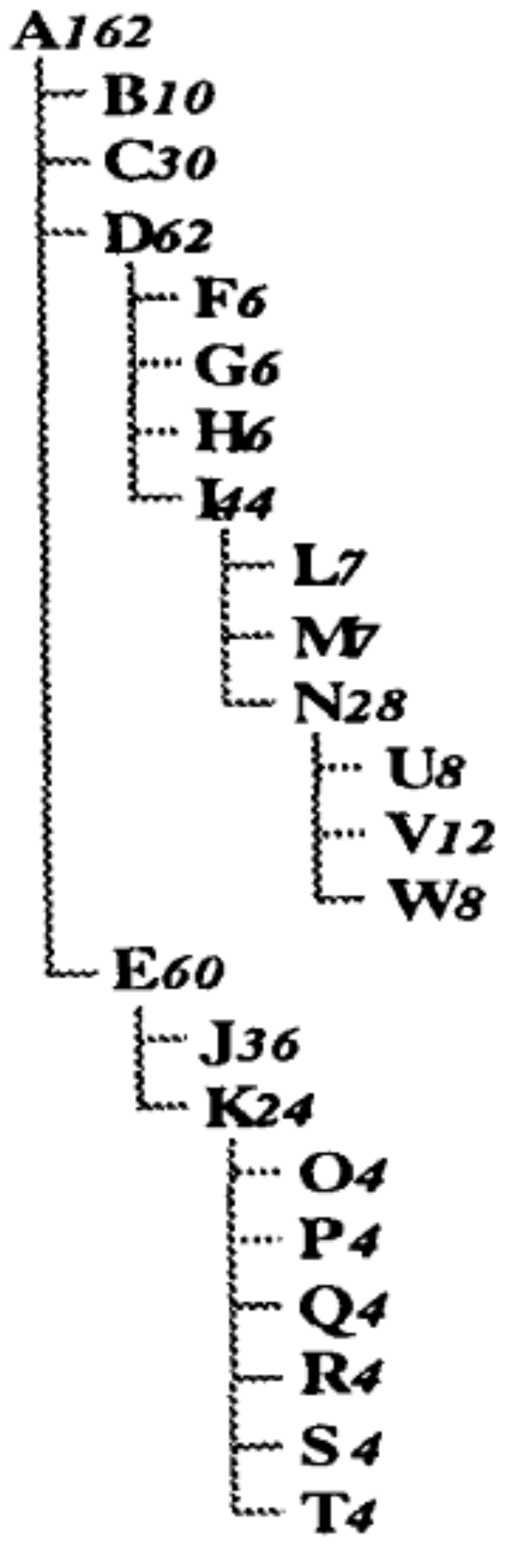
Sunburst

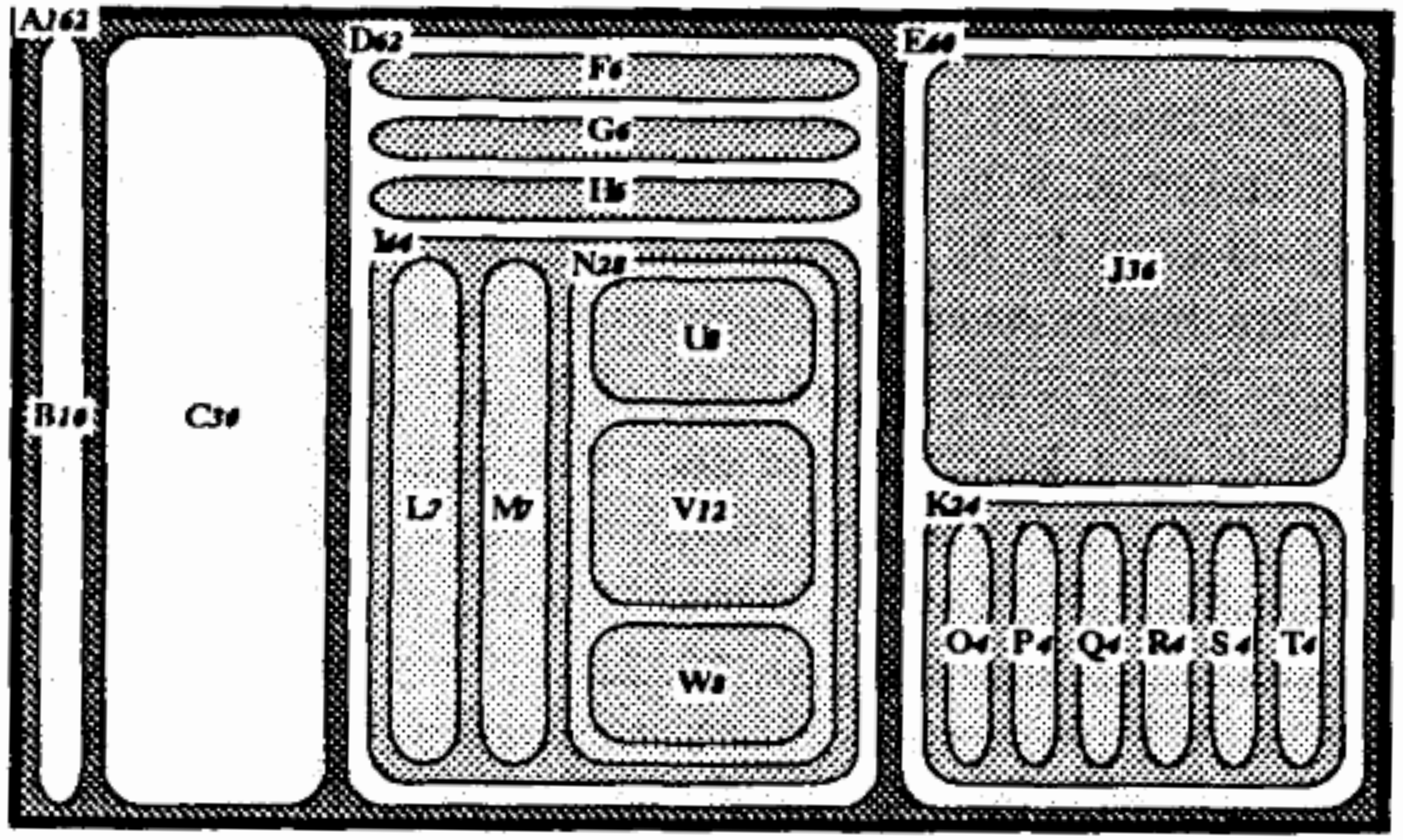


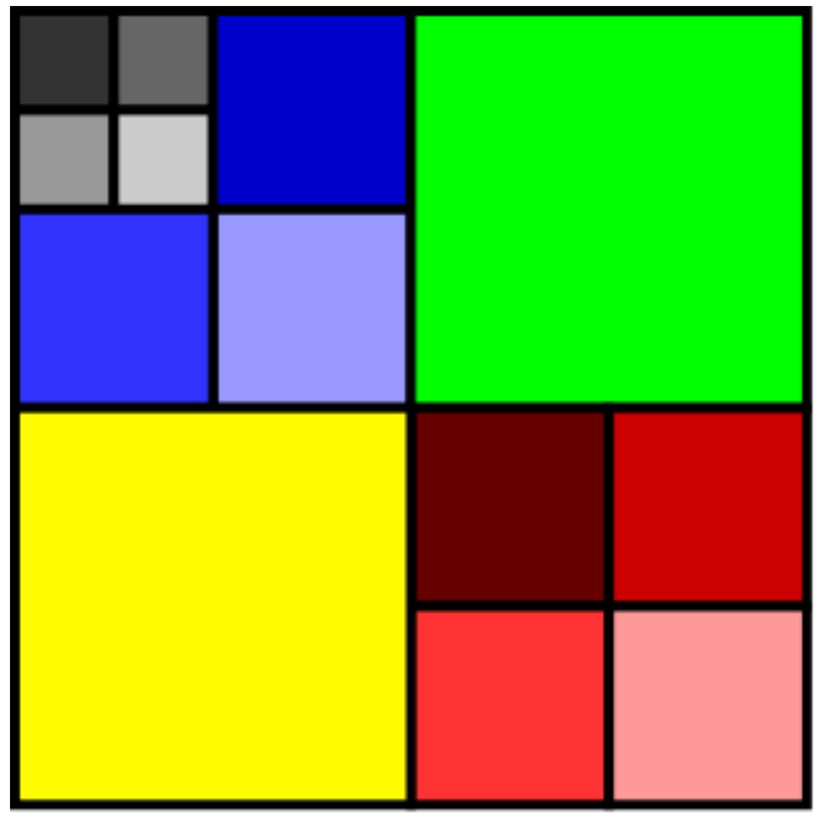
Icicle Plot

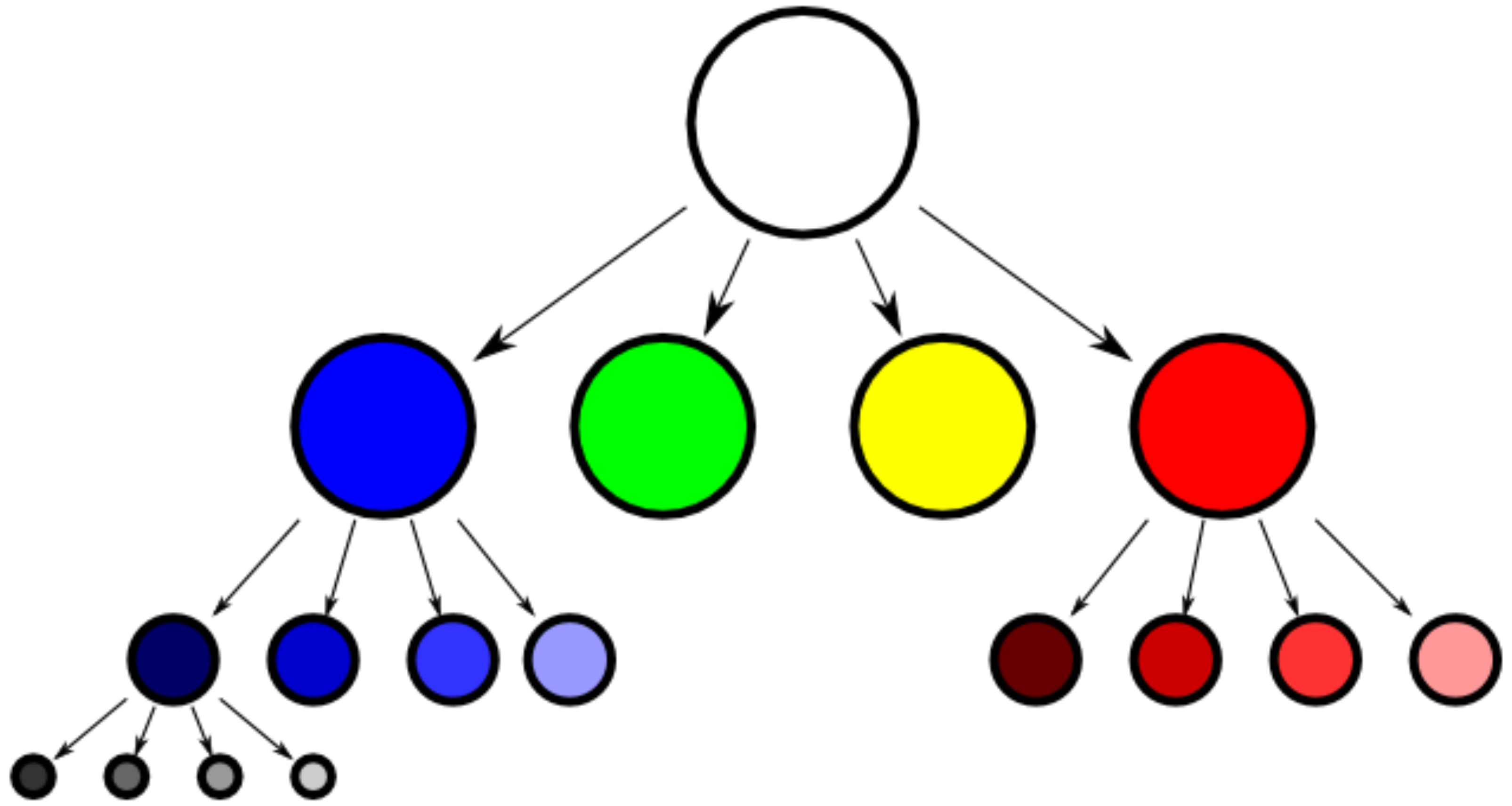


Tree Maps





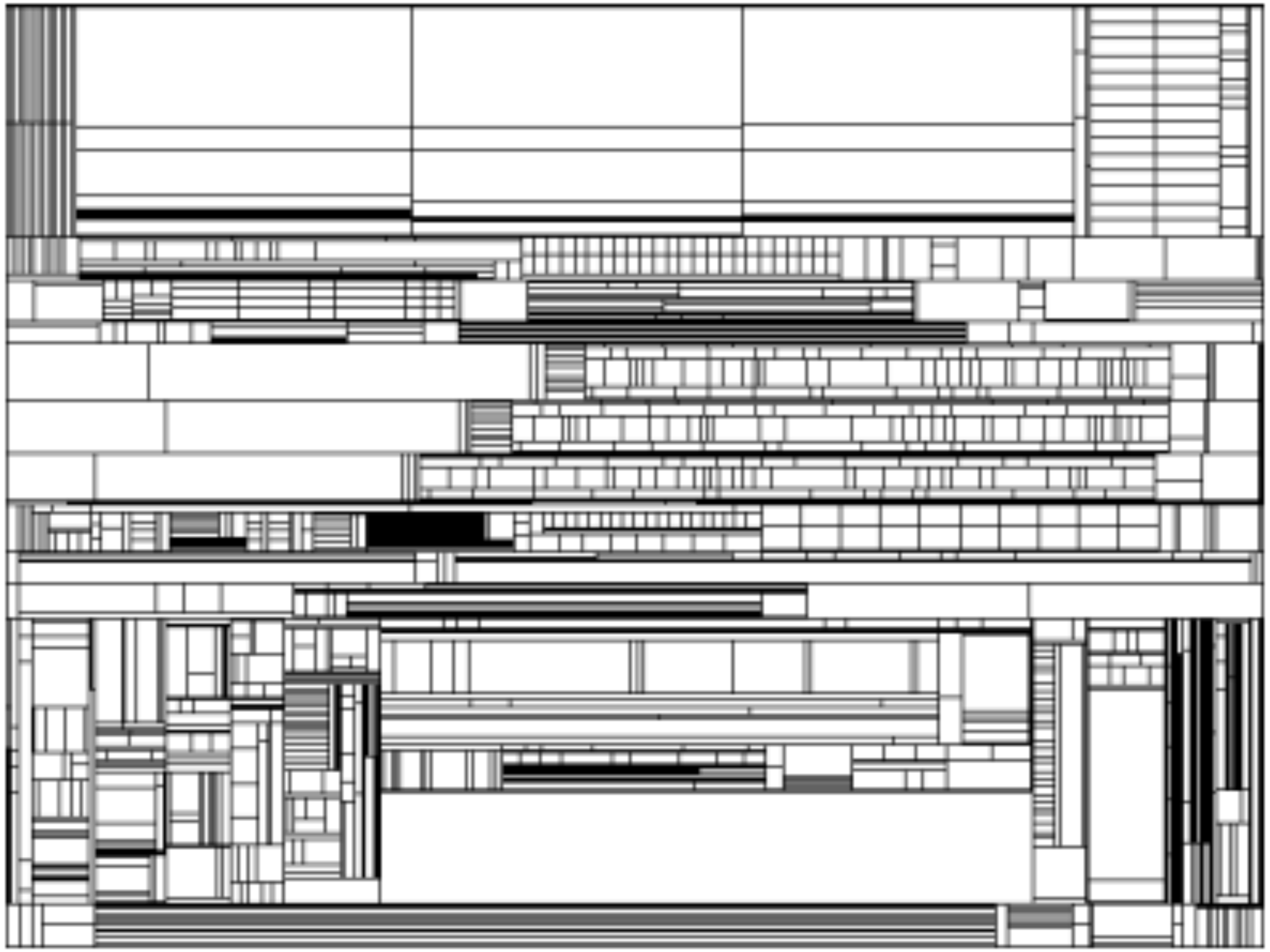




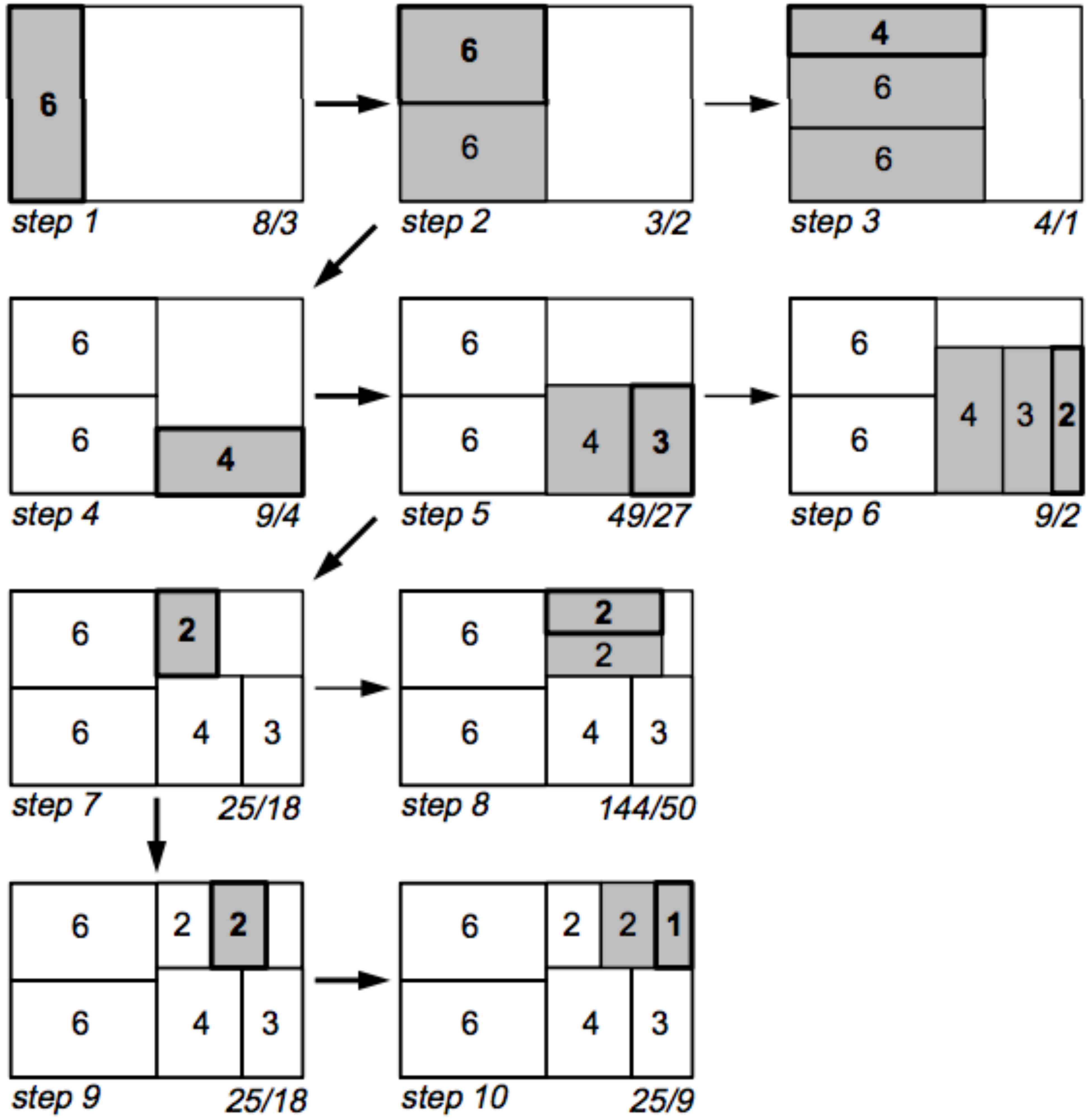
Johnson and Shneiderman 1991

Squarified Treemaps

Original Algorithm lead to thin slices



Squarified Treemaps



Algo by Bruls, Huizing, Van Wijk 2000]

1: Horizontal subdivision to optimize aspect ratio

2: adding rect improves aspect ration

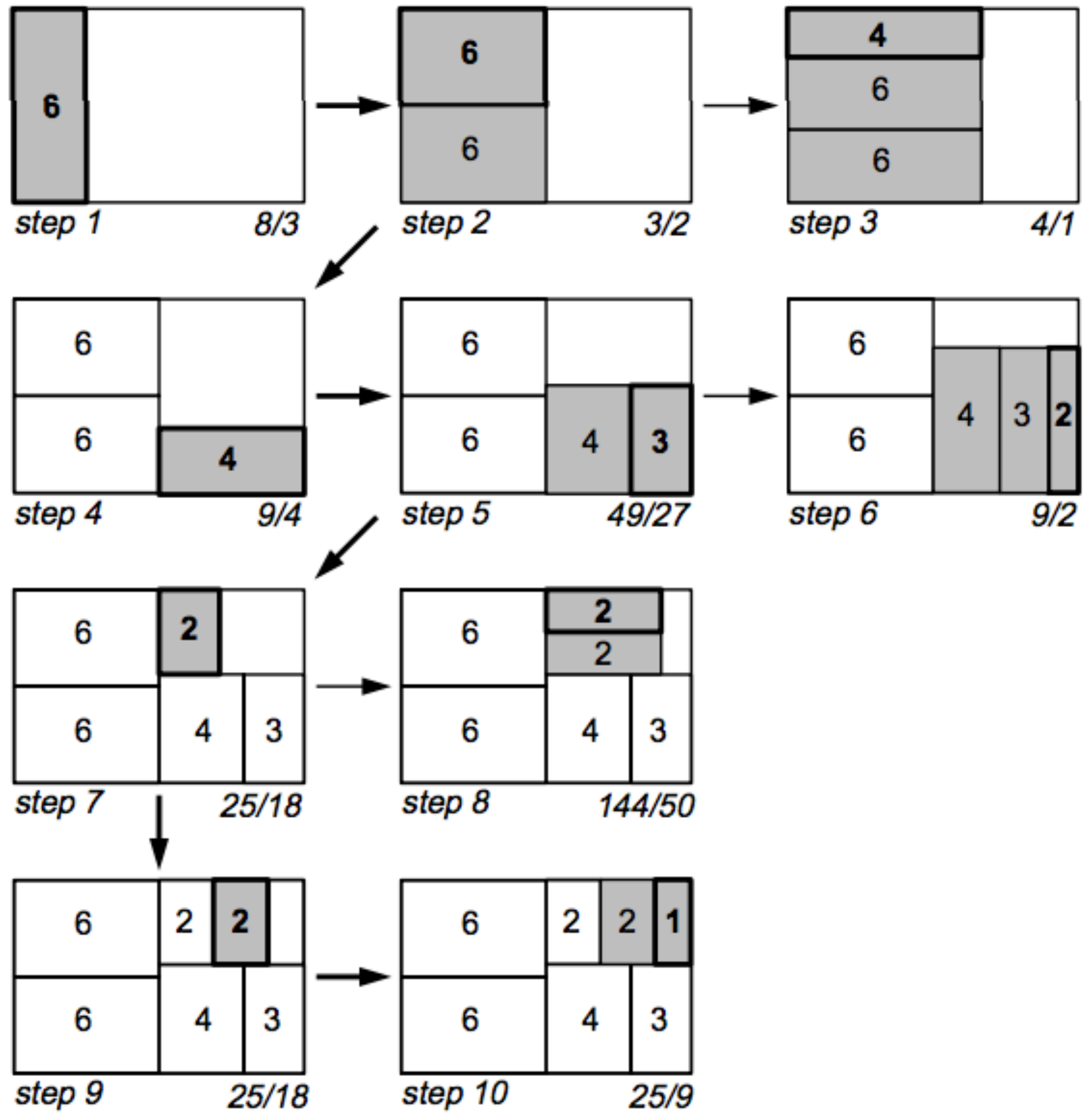
3: adding another deteriorates aspect ratio, back-track

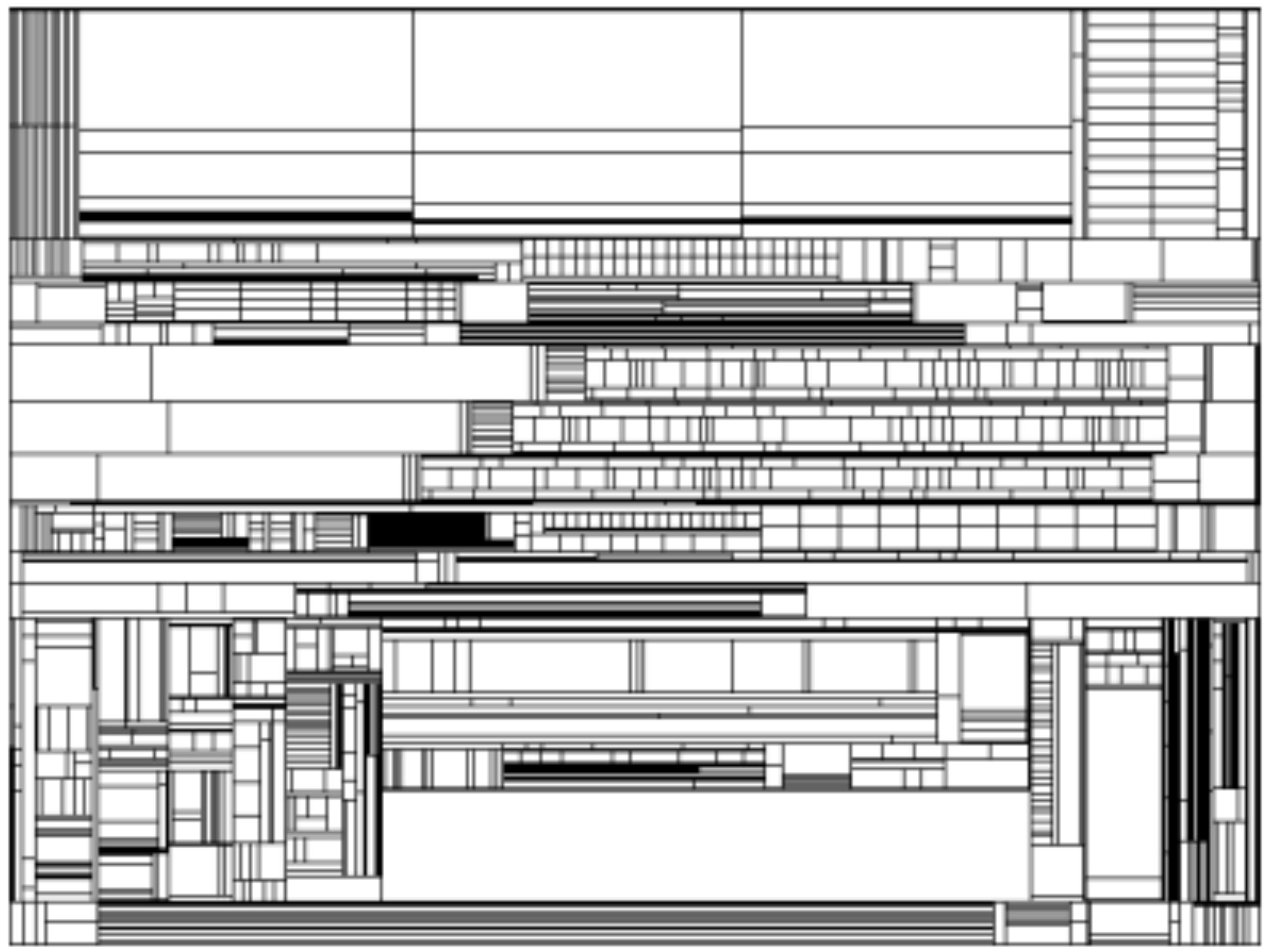
4: add rect to unused area

5: …

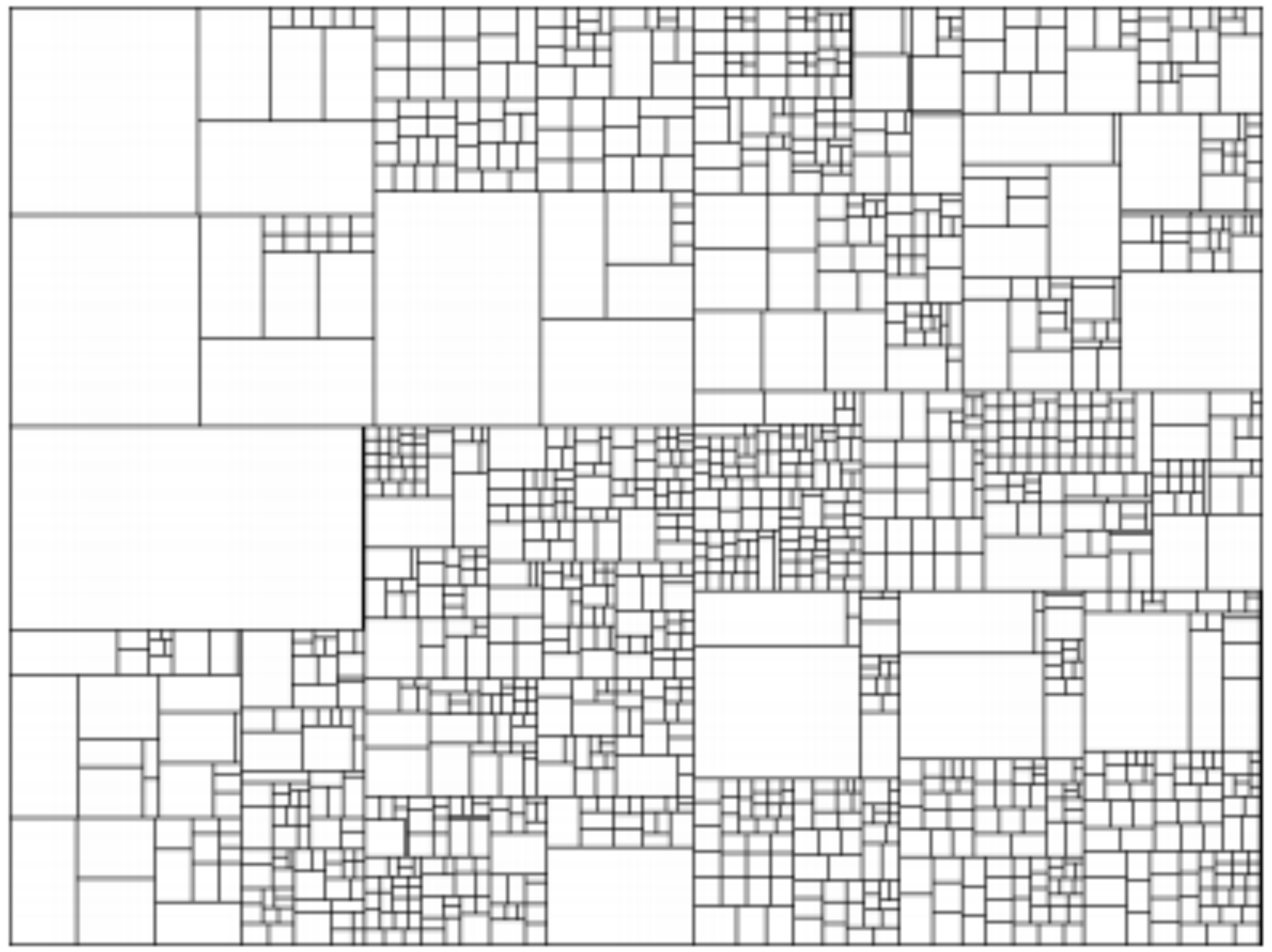
Squarified Treemaps

Squarified treemaps [Bruls, Huizing, Van Wijk 2000]



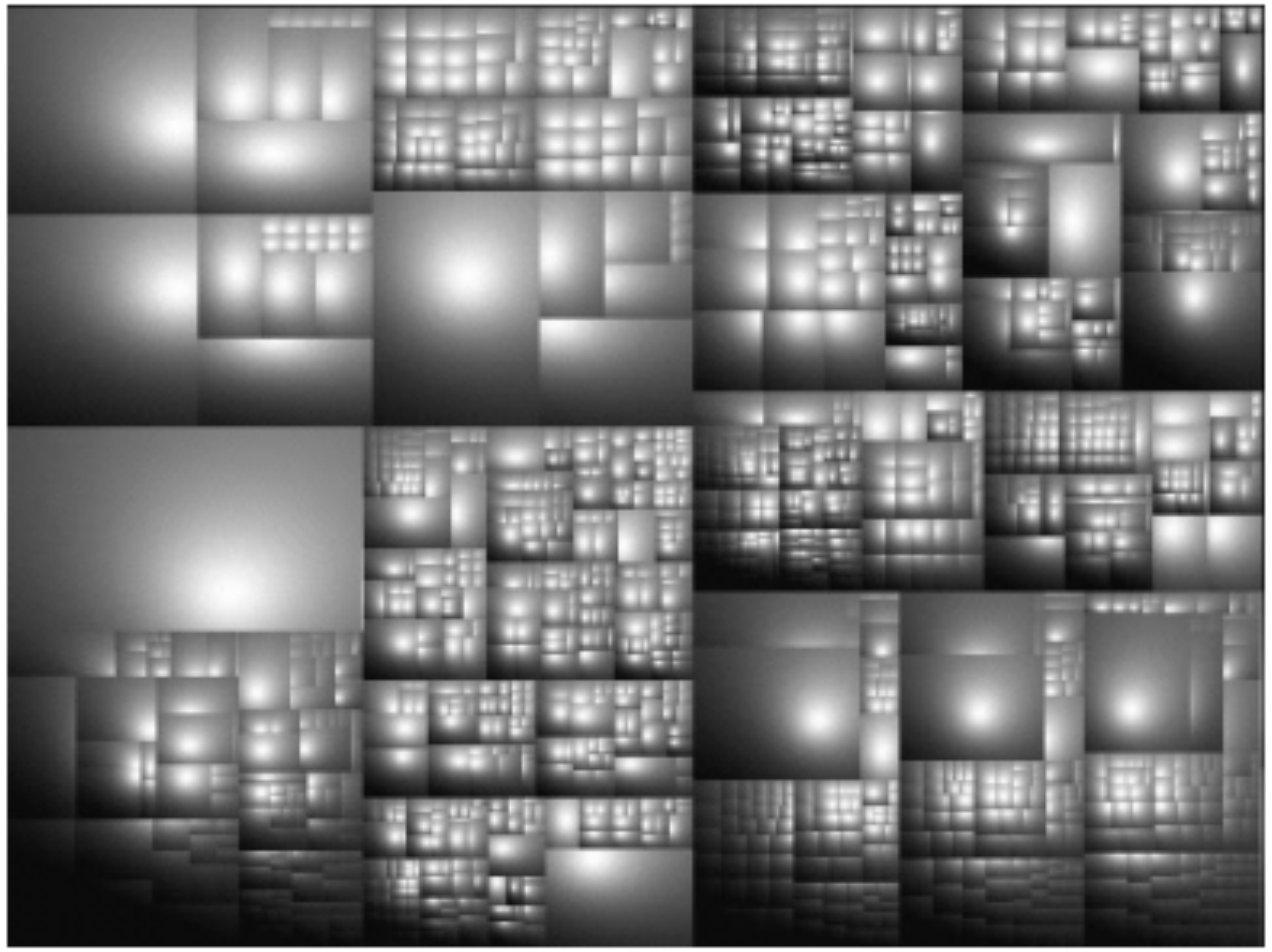


Before

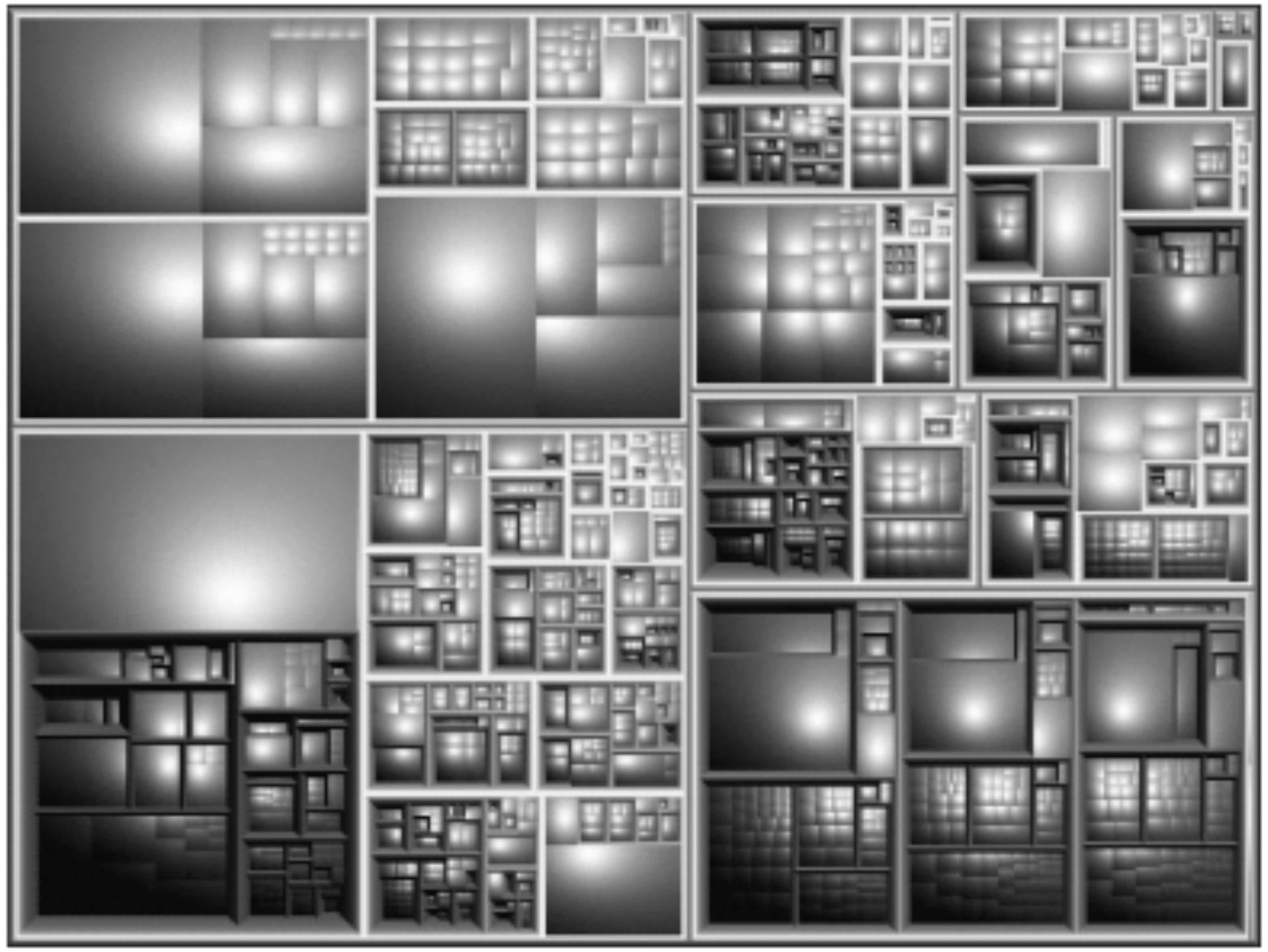


After

Seeing Tree Structure



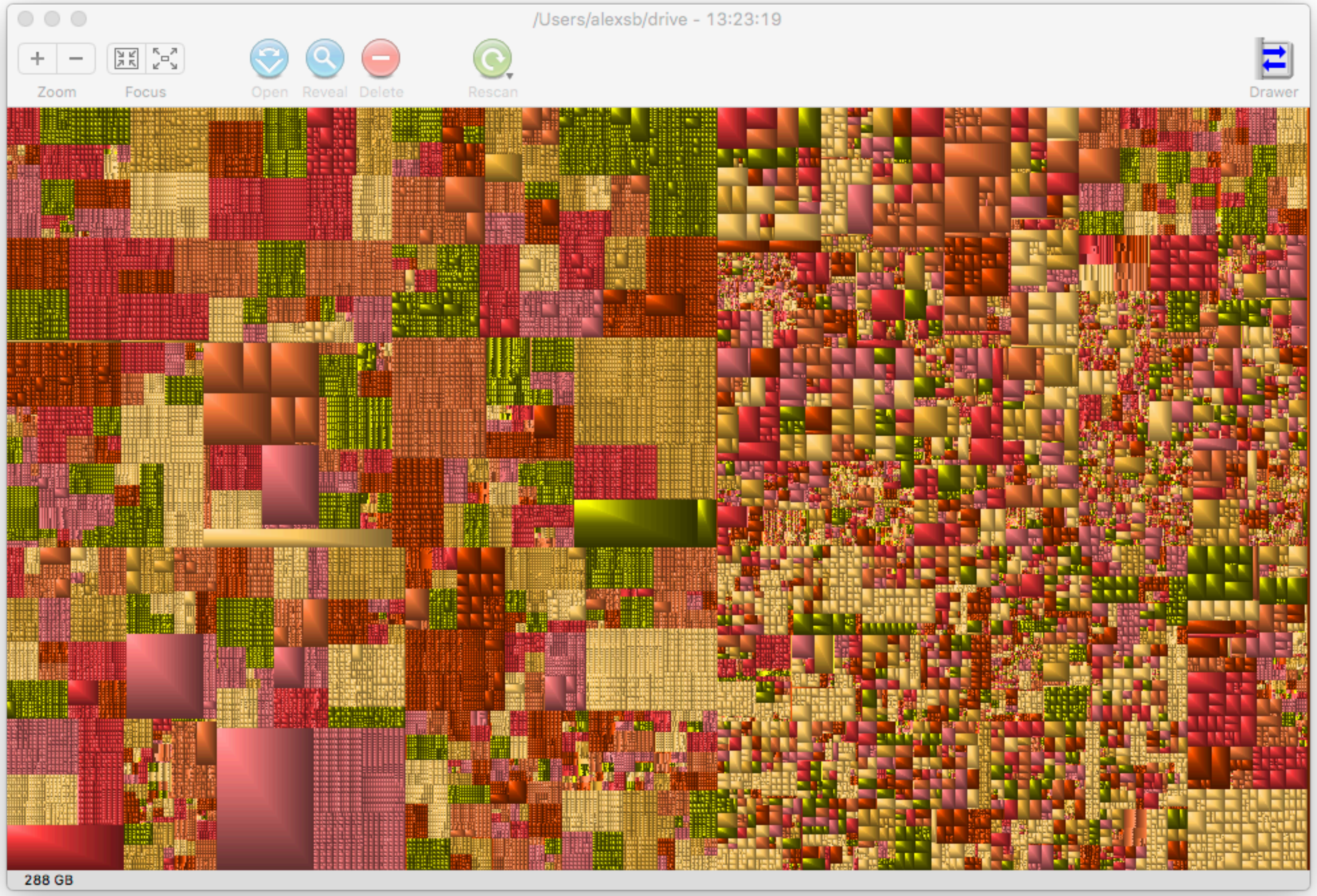
Unframed

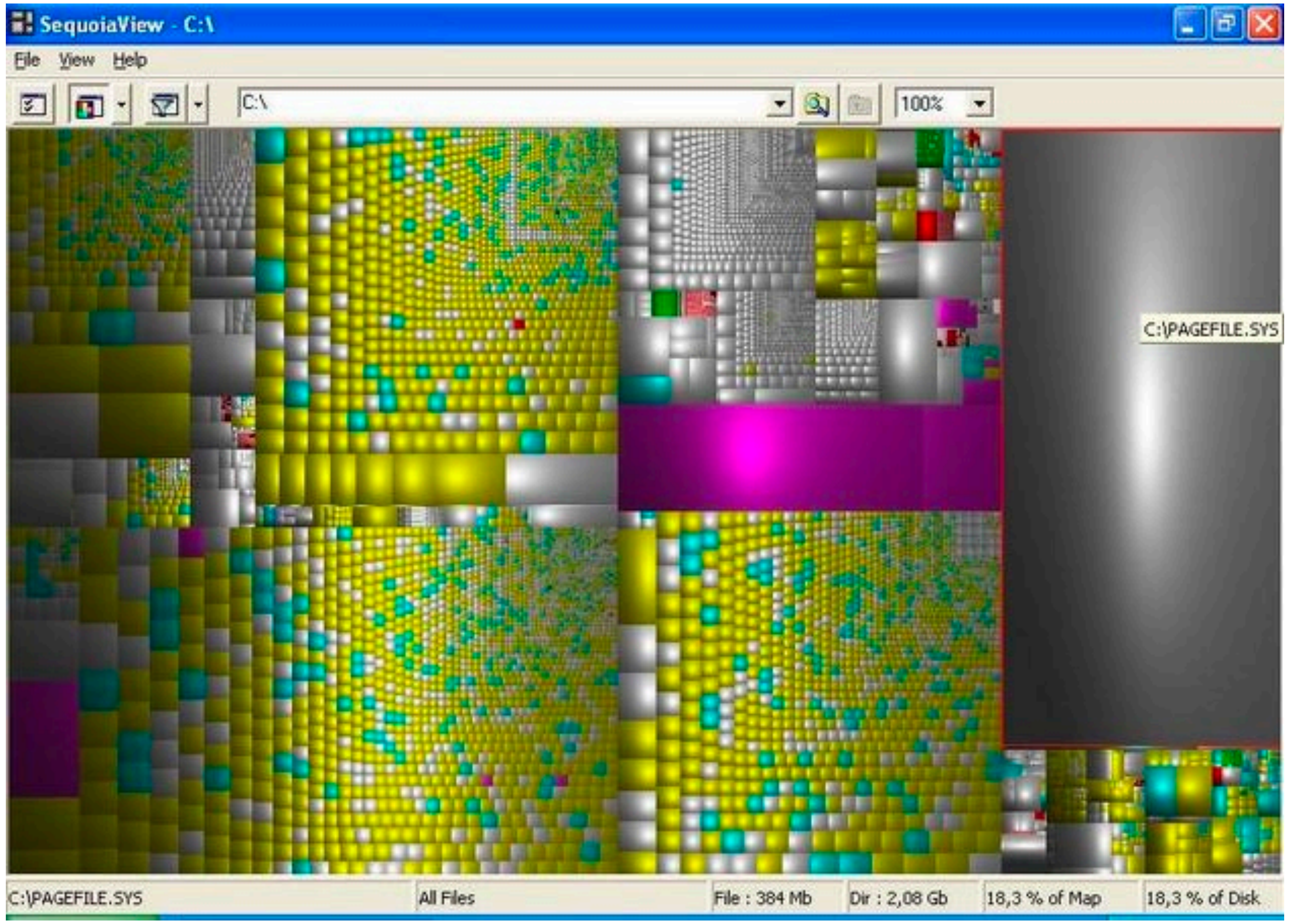


Framed

Software

Mac: GrandPerspective Windows: Sequoia View

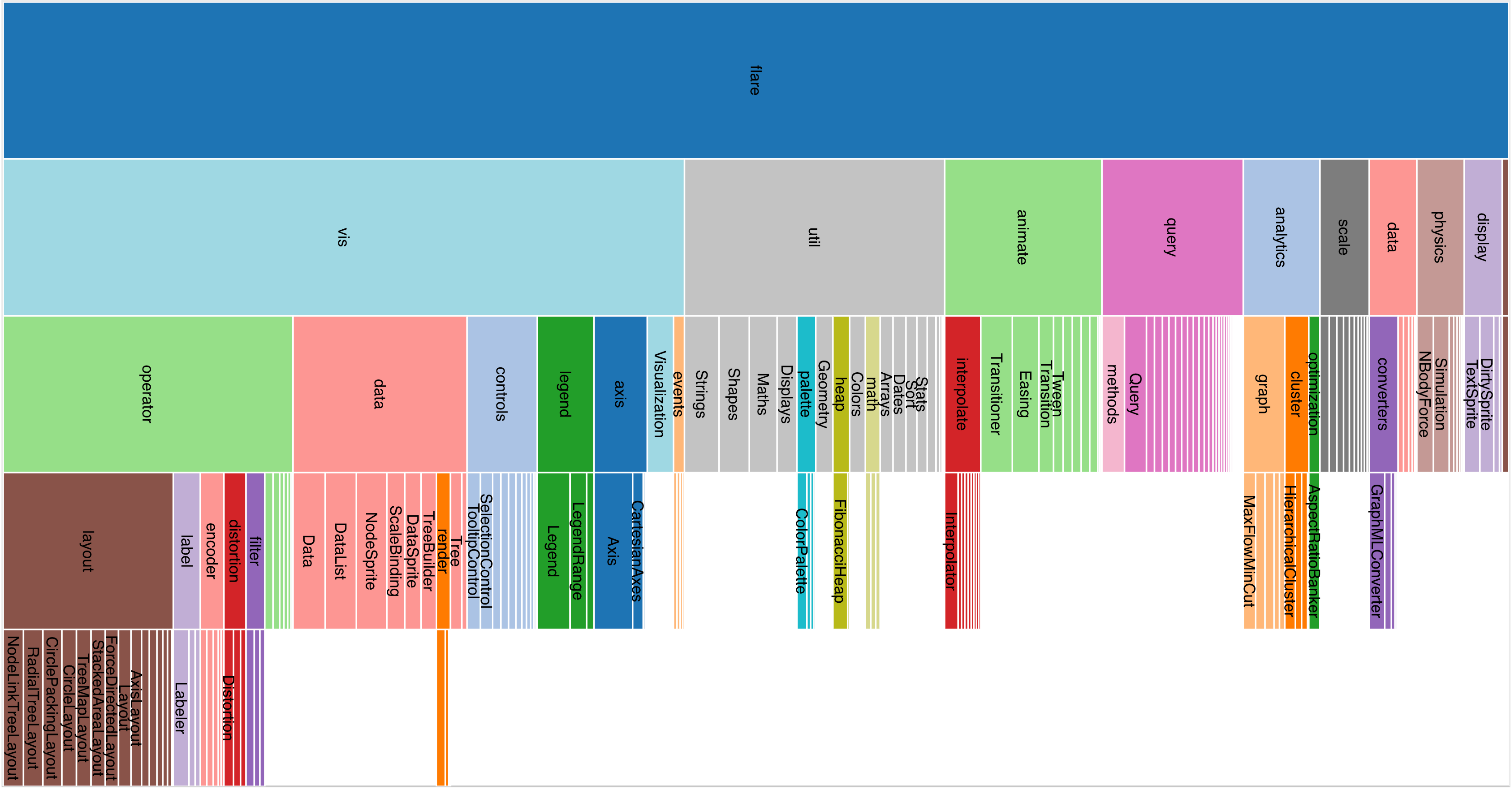




Sunburst: Radial Layout

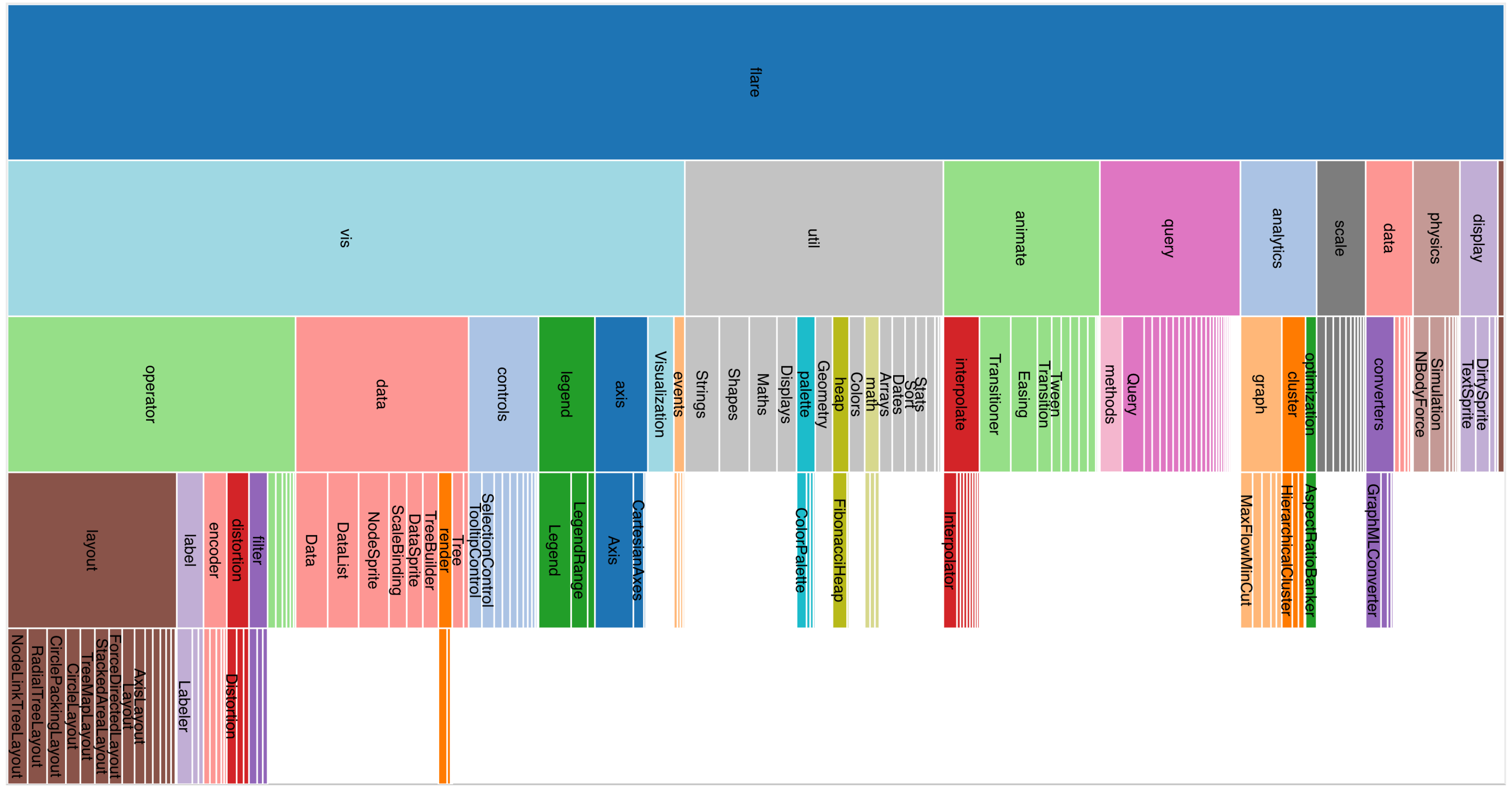
[Sunburst by John Stasko, Implementation in Caleydo by Christian Partl]

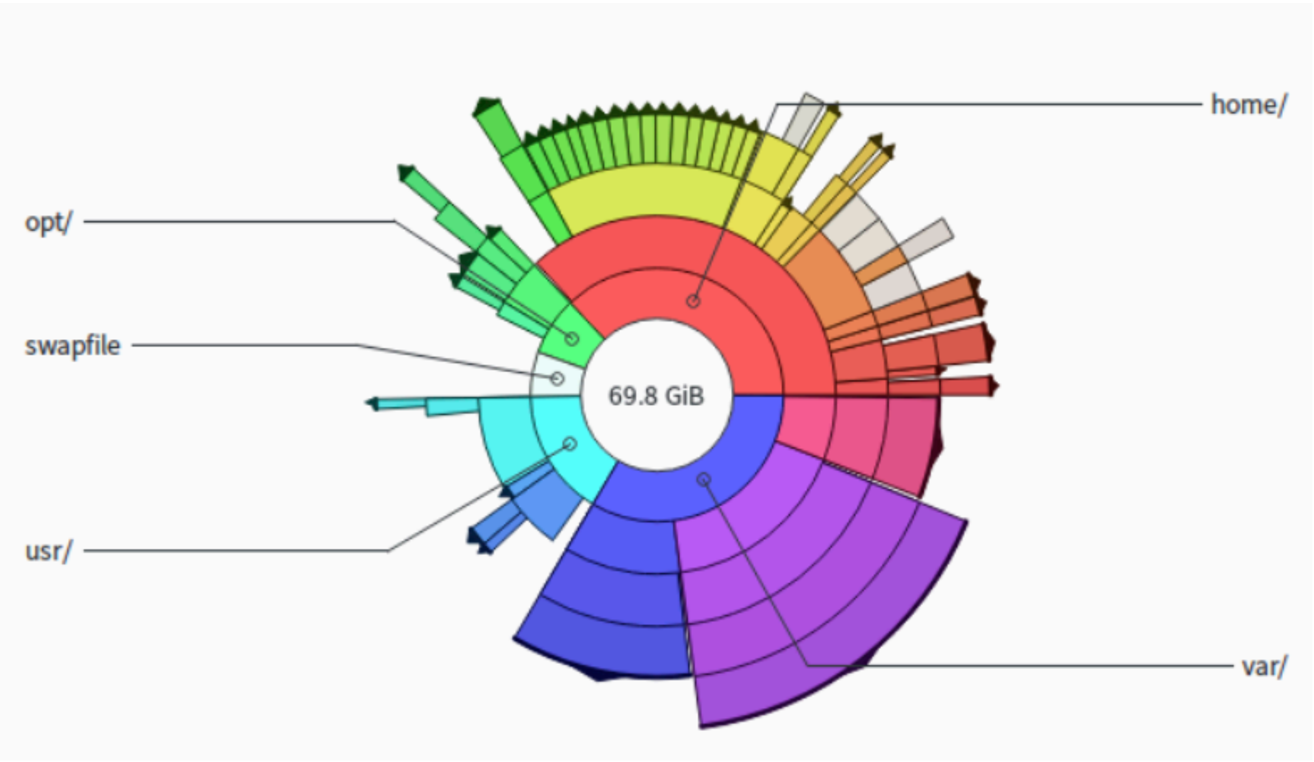
Icicle Plot

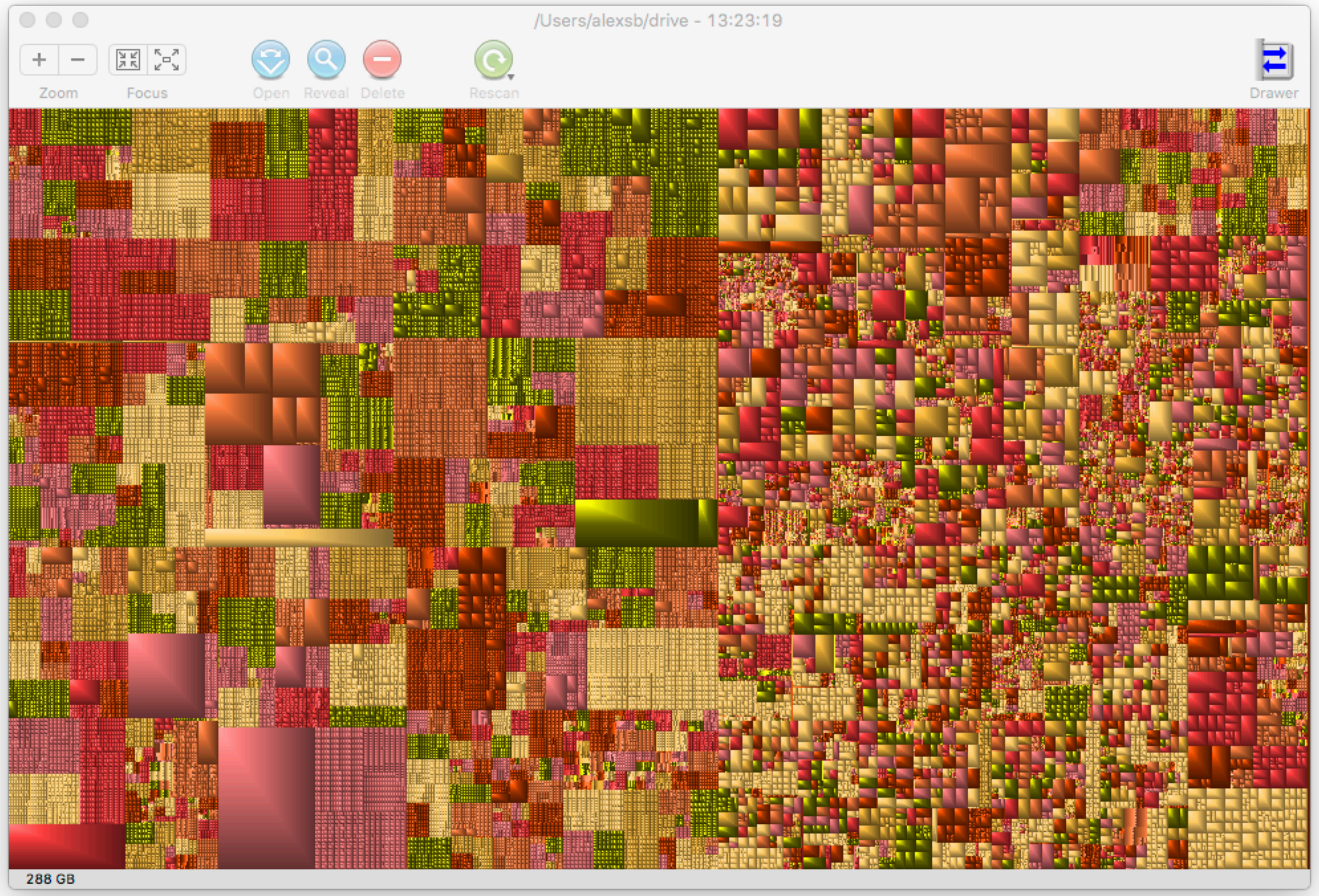


https://bl.ocks.org/mbostock/1005873

Differences? Pros, Cons?







Inner Nodes and Leaves Visible

Only Leaves Visible

Implicit Representations

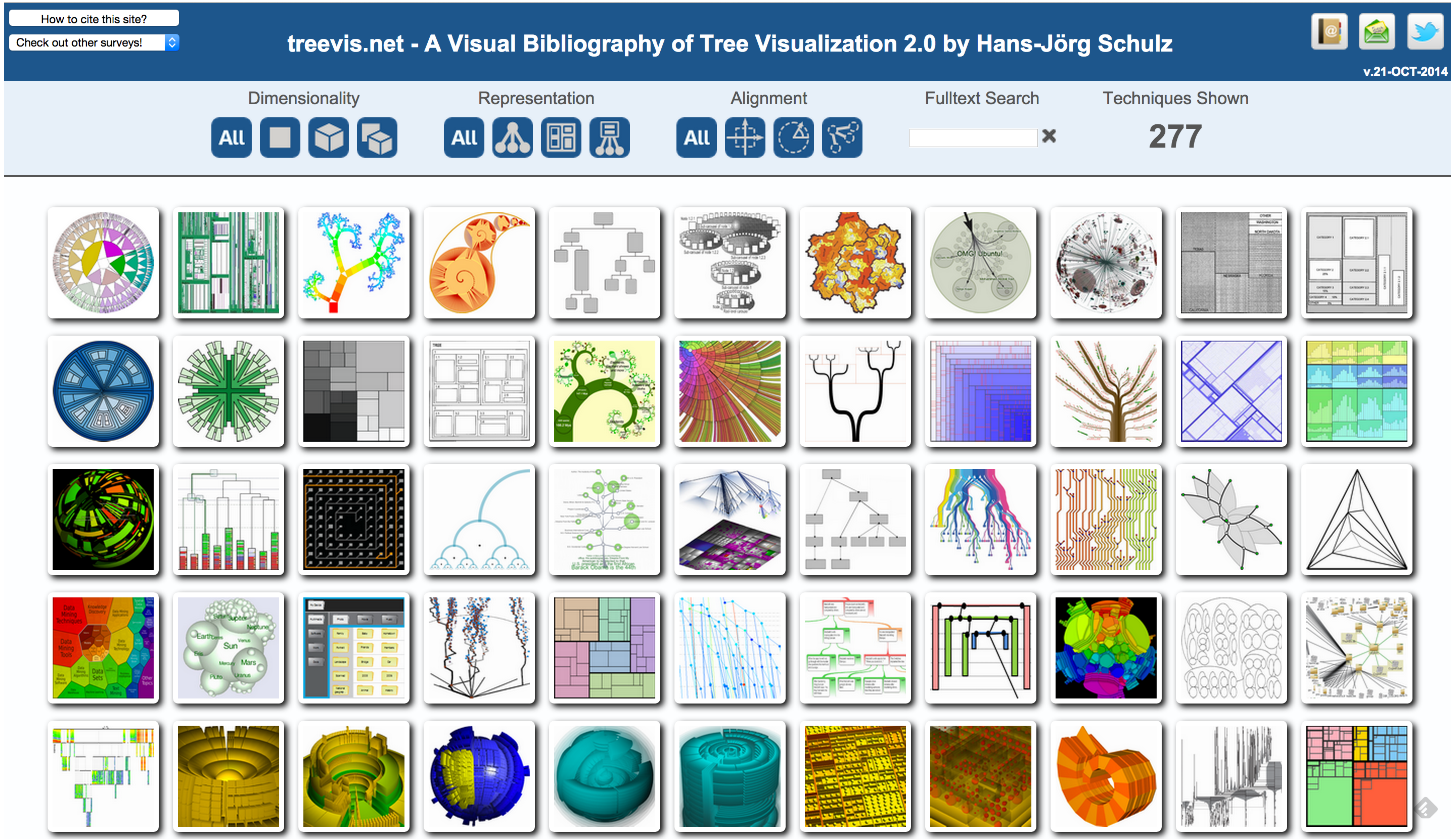
Pros:

- space-efficient because of the lack of explicitly drawn edges: scale well up to very large graphs
- in most cases well suited for ABTs on the node set
- depending on the spatial encoding also useful for TBTs

Cons:

- can only represent trees
- since the node positions are used to represent edges, they can no longer be freely arranged (e.g., to reflect geographical positions)
- useless to pursue any task on the edges

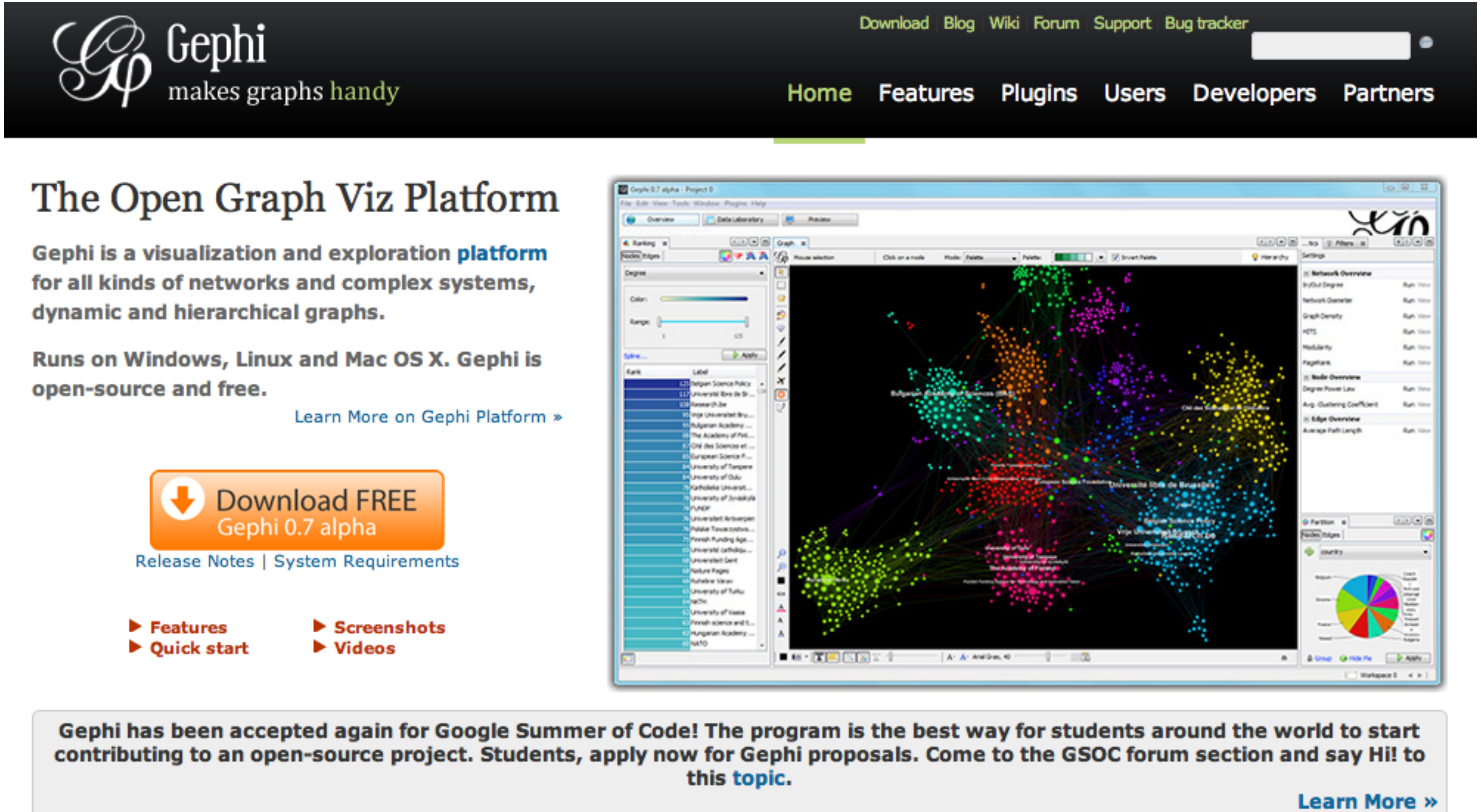
Tree Visualization Reference



Graph Tools & Applications

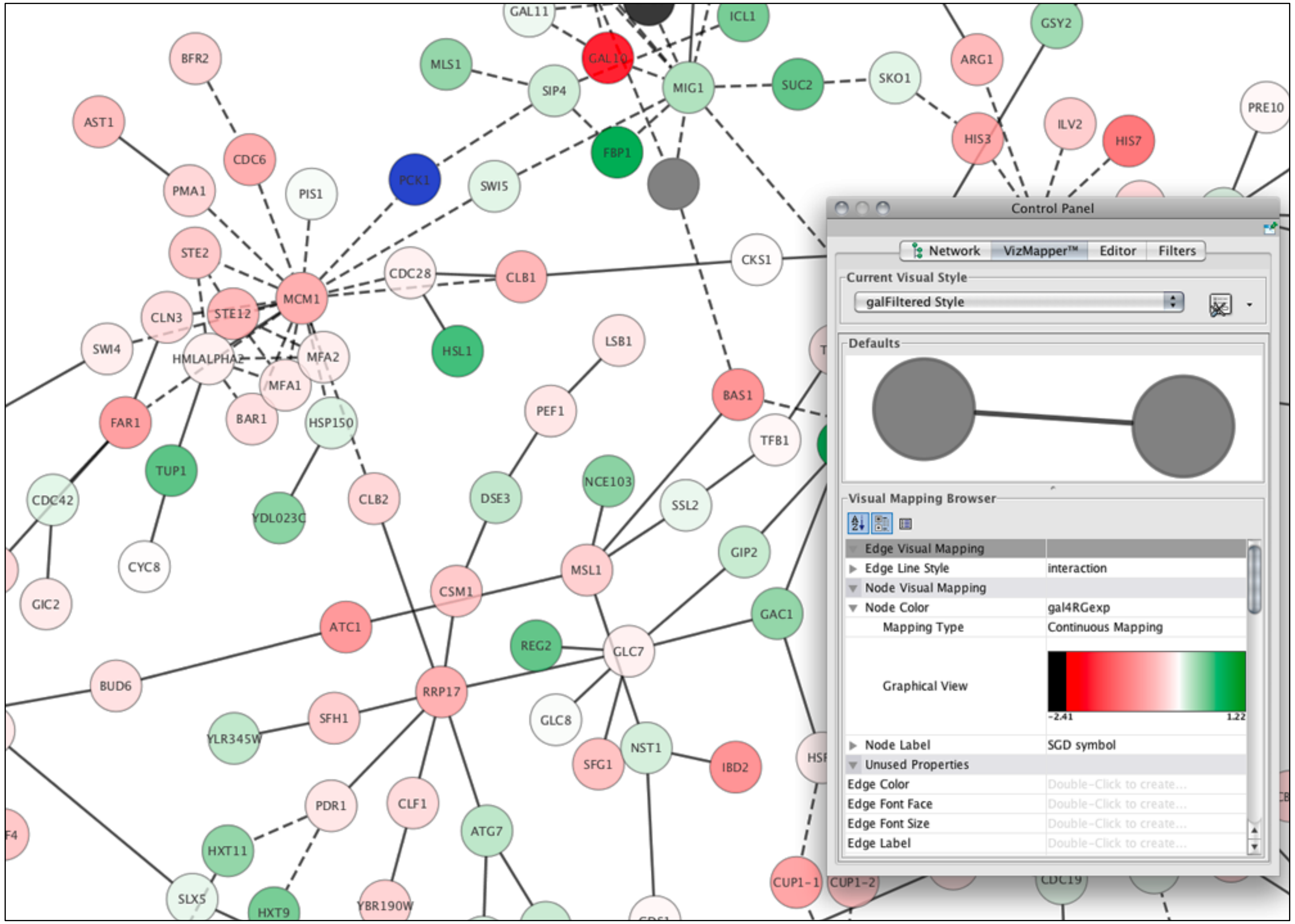
Gephi

http://gephi.org



Cytoscape

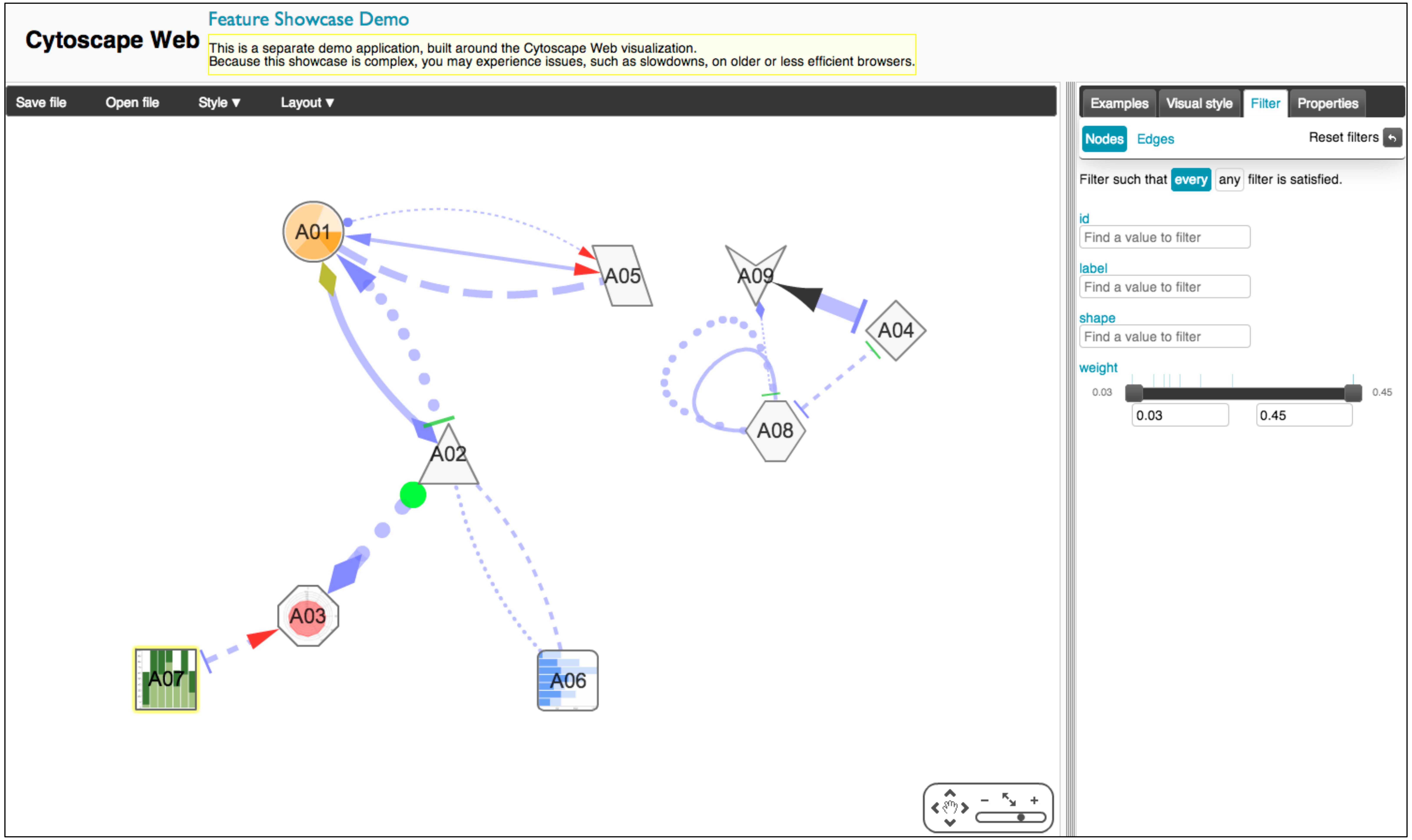
Open source platform for complex network analysis



http://www.cytoscape.org/

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

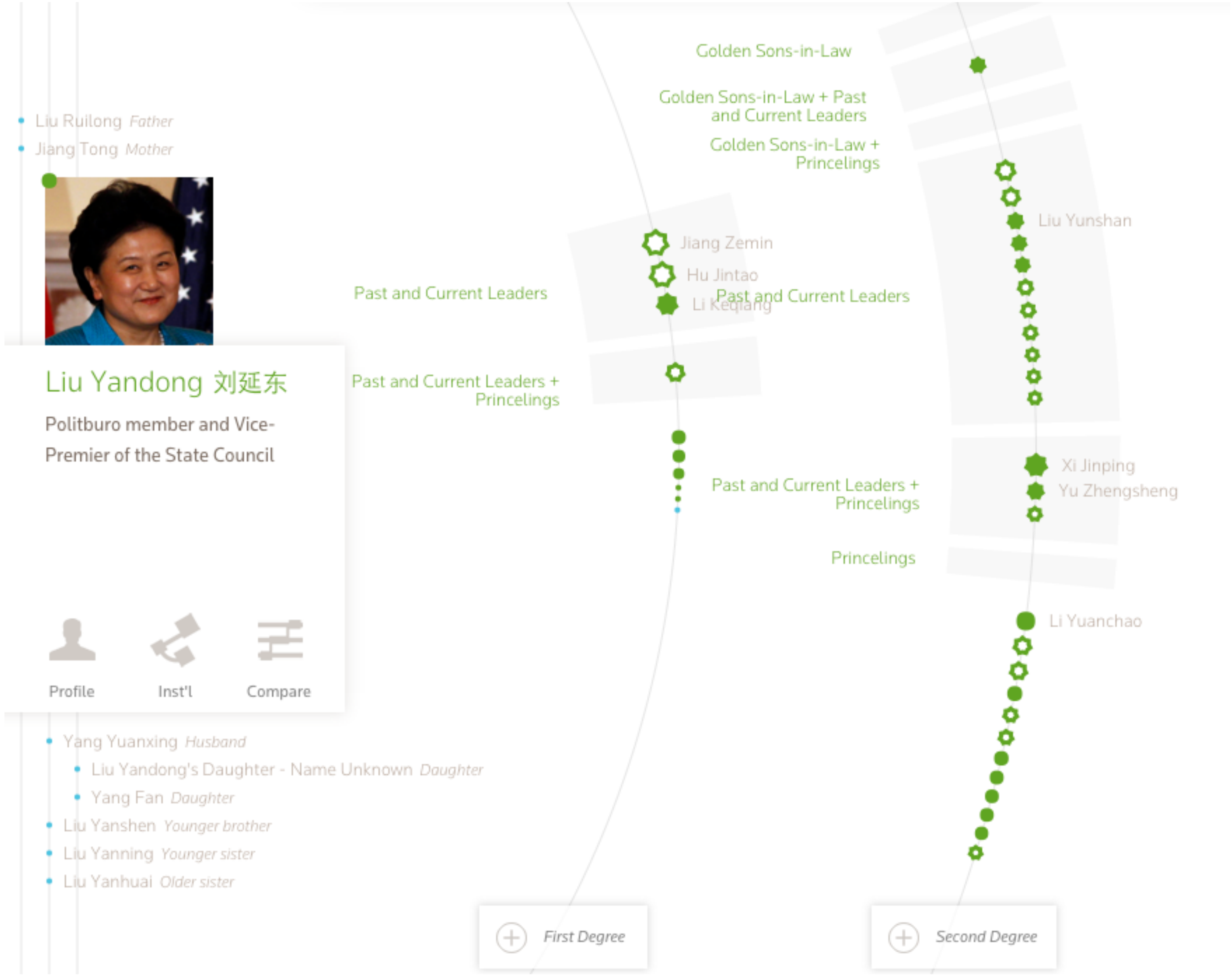
1.9dev
github | docs | pdf
build passing
coverage 83%

Contact

Mailing list
Issue tracker
Developer guide

Design Critique

Connected China



https://goo.gl/YXkWYX

http://china.fathom.info/