

CS 571 - Data Visualization & Exploration

Visualizing Networks and Trees

Instructor: Hamza Elhamdadi

UMassAmherst

Upcoming Dates

May 2:

- Homework 5 Due
- Project Screencast Submission Due
- All Quizzes Due

May 12: Final Project Submission Due

Request an Extension to May 9 for Project
Screencast and Homework 5

5 Dataset Types

Tables

Items

Attributes

Networks & Trees

Items (nodes)

Links

Attributes

Fields

Grids

Positions

Attributes

Geometry

Items

Positions

Clusters, Sets, Lists

Items

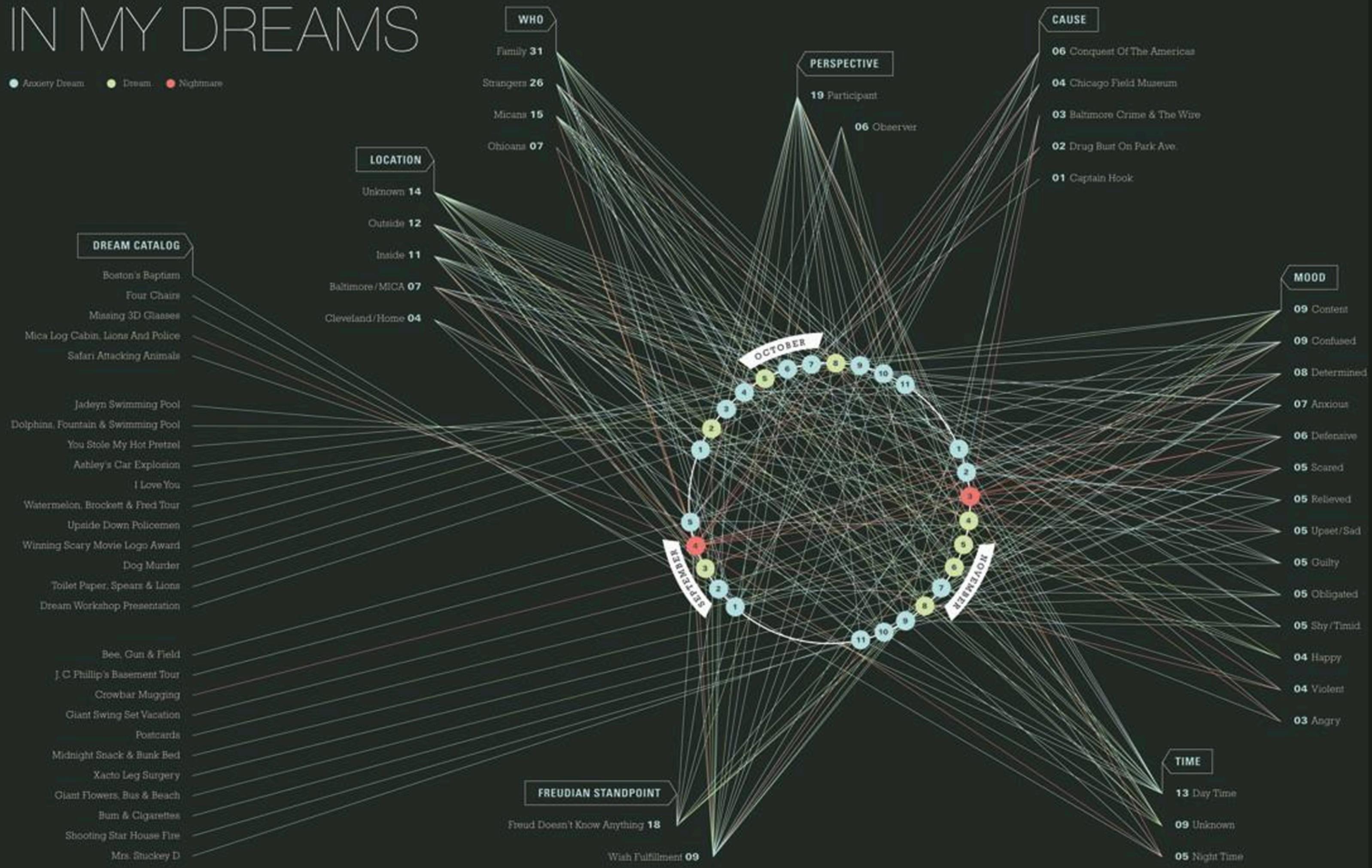


facebook

December 2010

[Paul Butler](#)

IN MY DREAMS



Kailie Parrish

Definitions

Graph

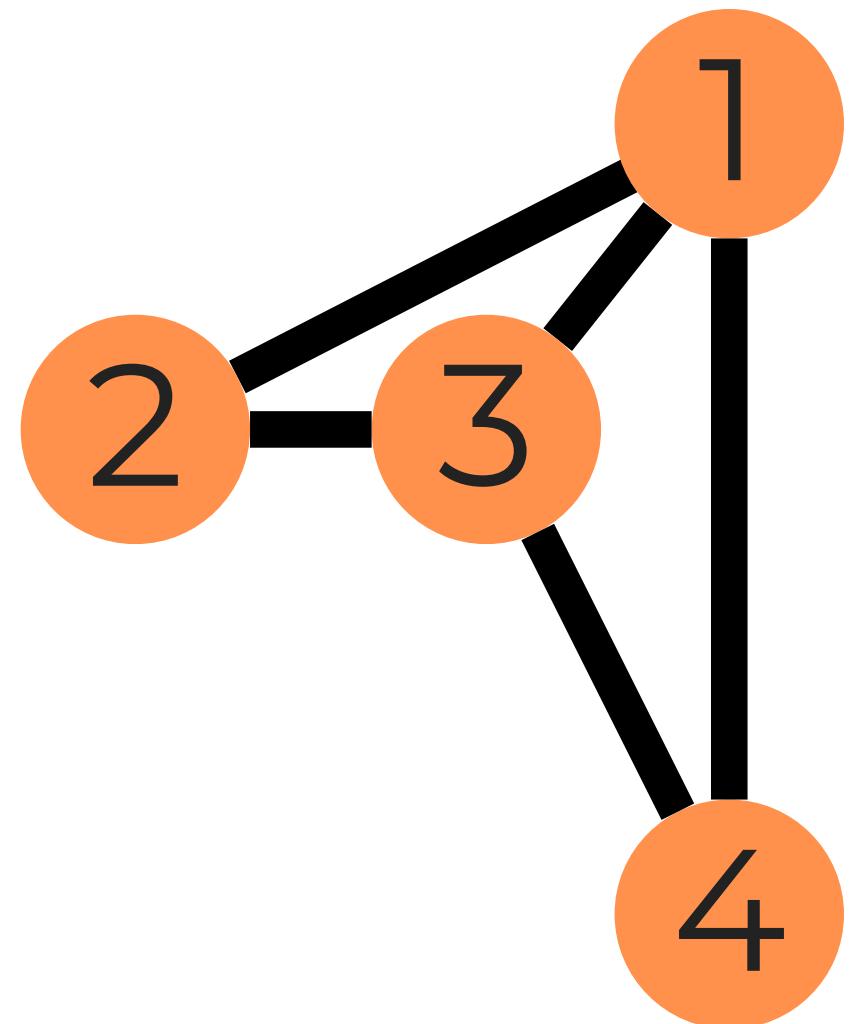
A graph **G** consists of

- **V** - a set of vertices (nodes)
- **E** - a set of edges

An edge **e(x,y)** connects vertex **x** and vertex **y**

Example:

- $V = \{1,2,3,4\}$
- $E = \{(1,2), (1,3), (2,3), (3,4), (4,1)\}$



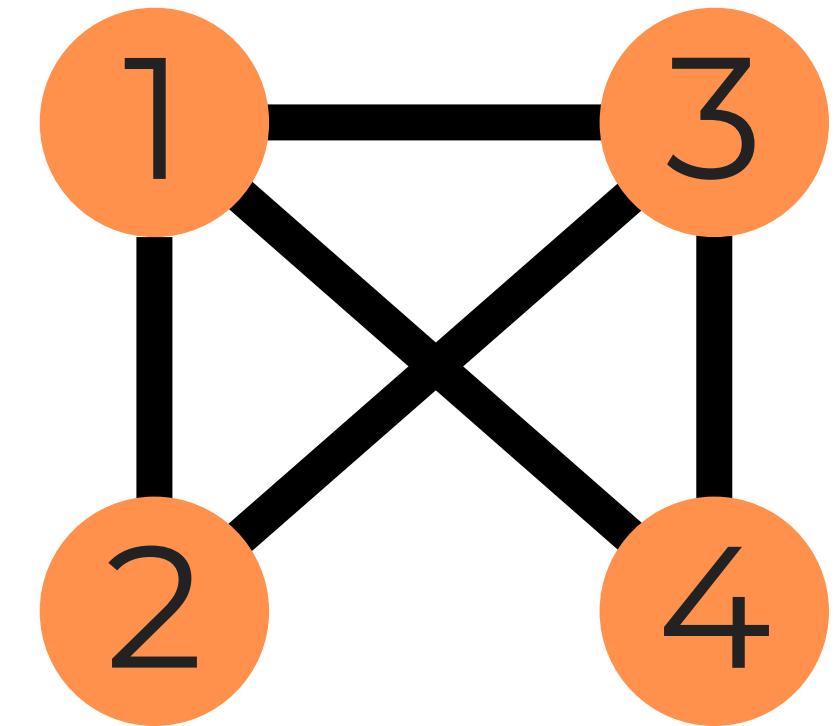
Adjacency Matrix

A matrix where:

- each **row/column** represents a **node**
- **non-zero values** represent **edges**

Example:

0	1	1	1
1	0	1	0
1	1	0	1
1	0	1	0



Graphs & Trees

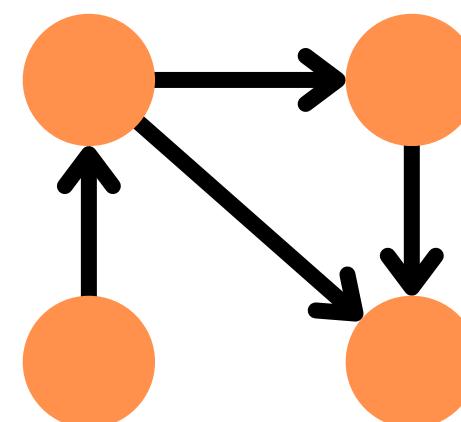
Graphs:

- model **relationships** in data

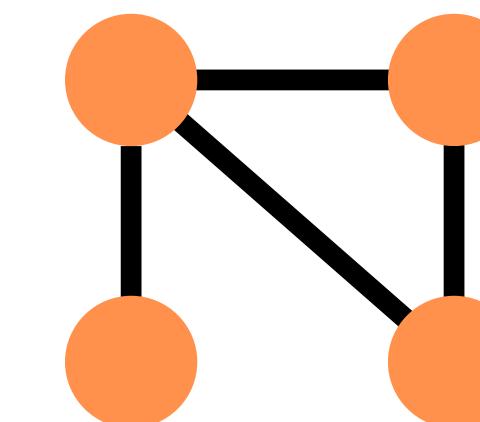
Trees:

- graphs with **hierarchical** structure
- nodes as **parents** and **children**

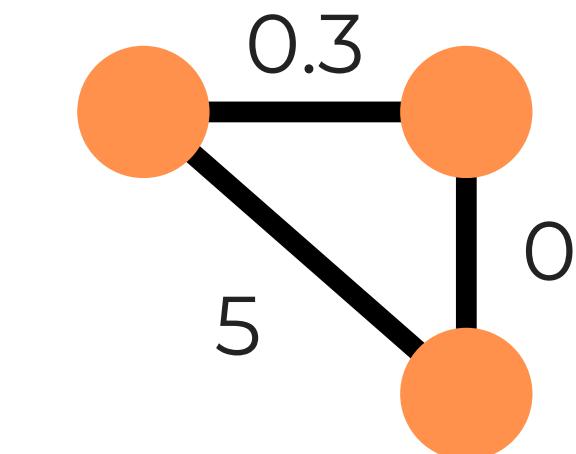
Other Important Definitions



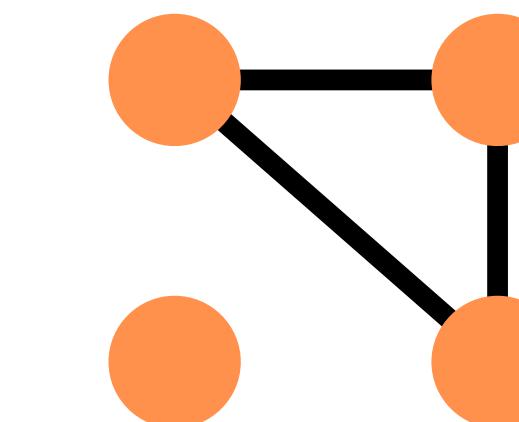
Directed



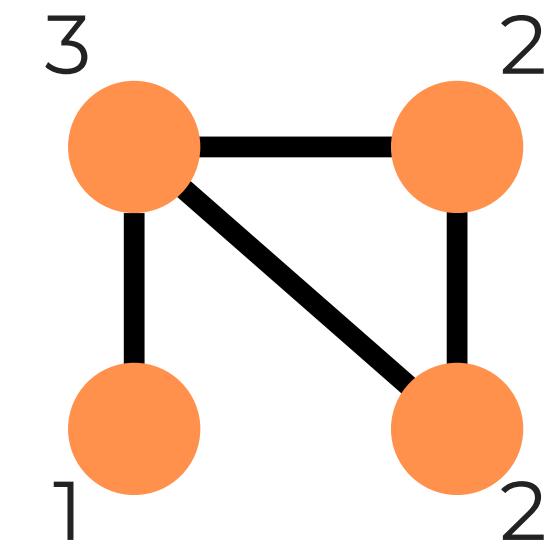
Undirected



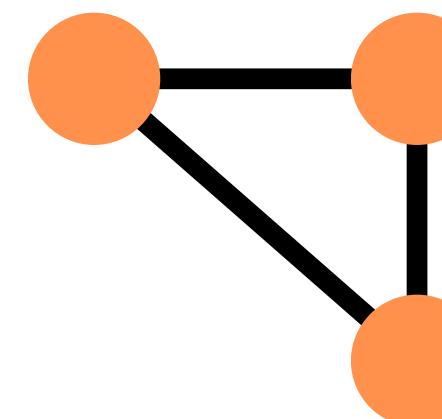
Edge Weights



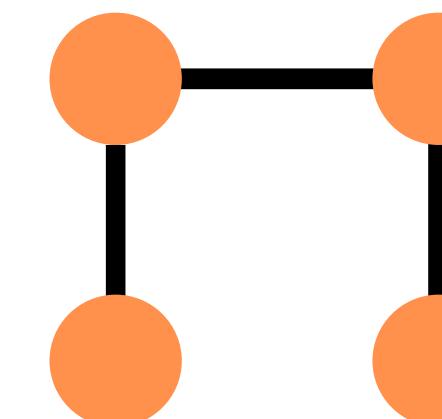
Unconnected



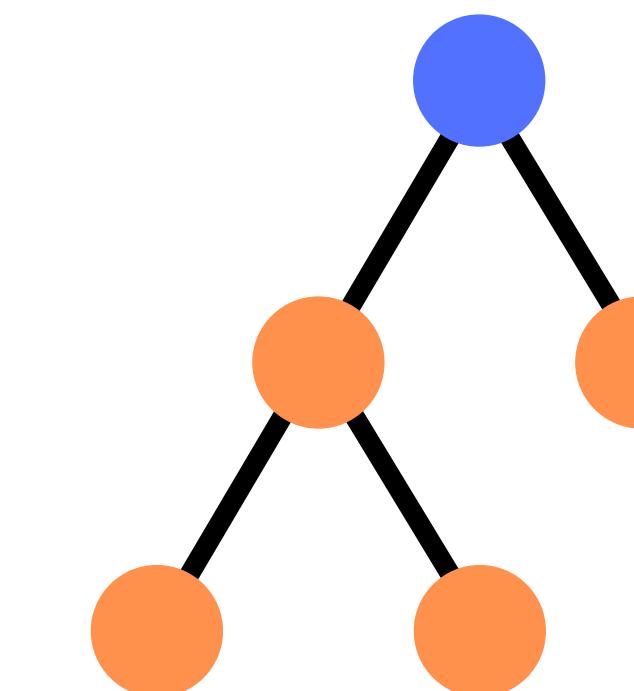
Node Degree



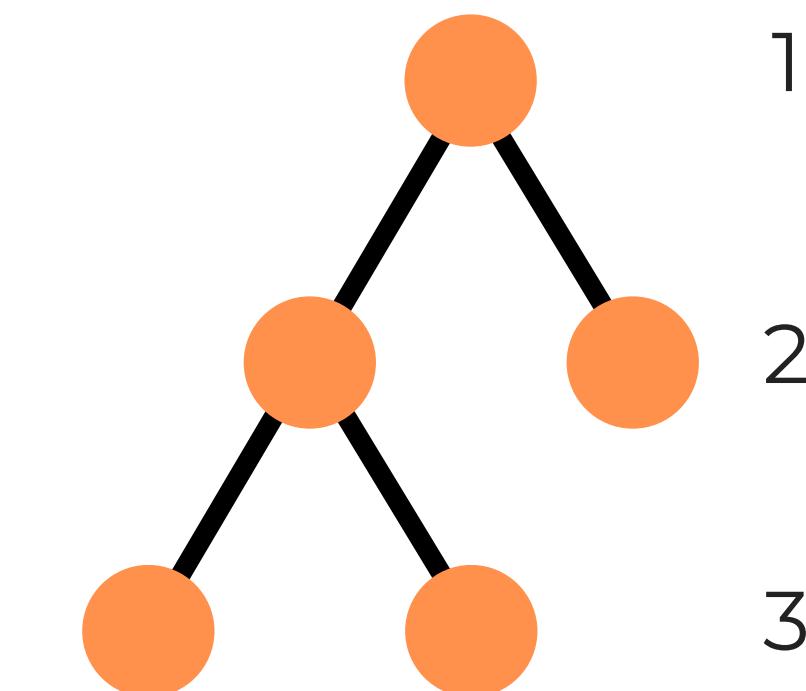
Cycle



Acyclic



Rooted Tree
(Hierarchy)



Node Depth

Visualizing Trees

Fast and Elegant to Draw Trees with Recursion

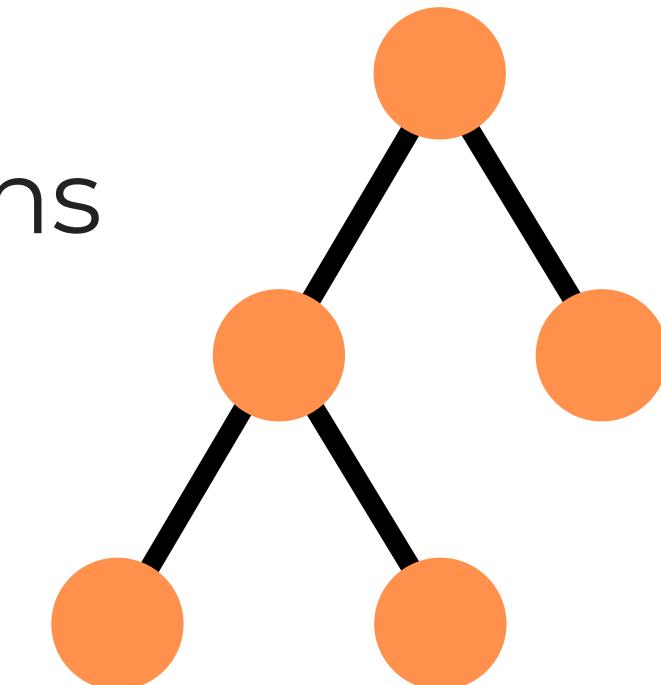
Node-Link Diagrams

Nodes distributed spatially, connected with straight or curved lines

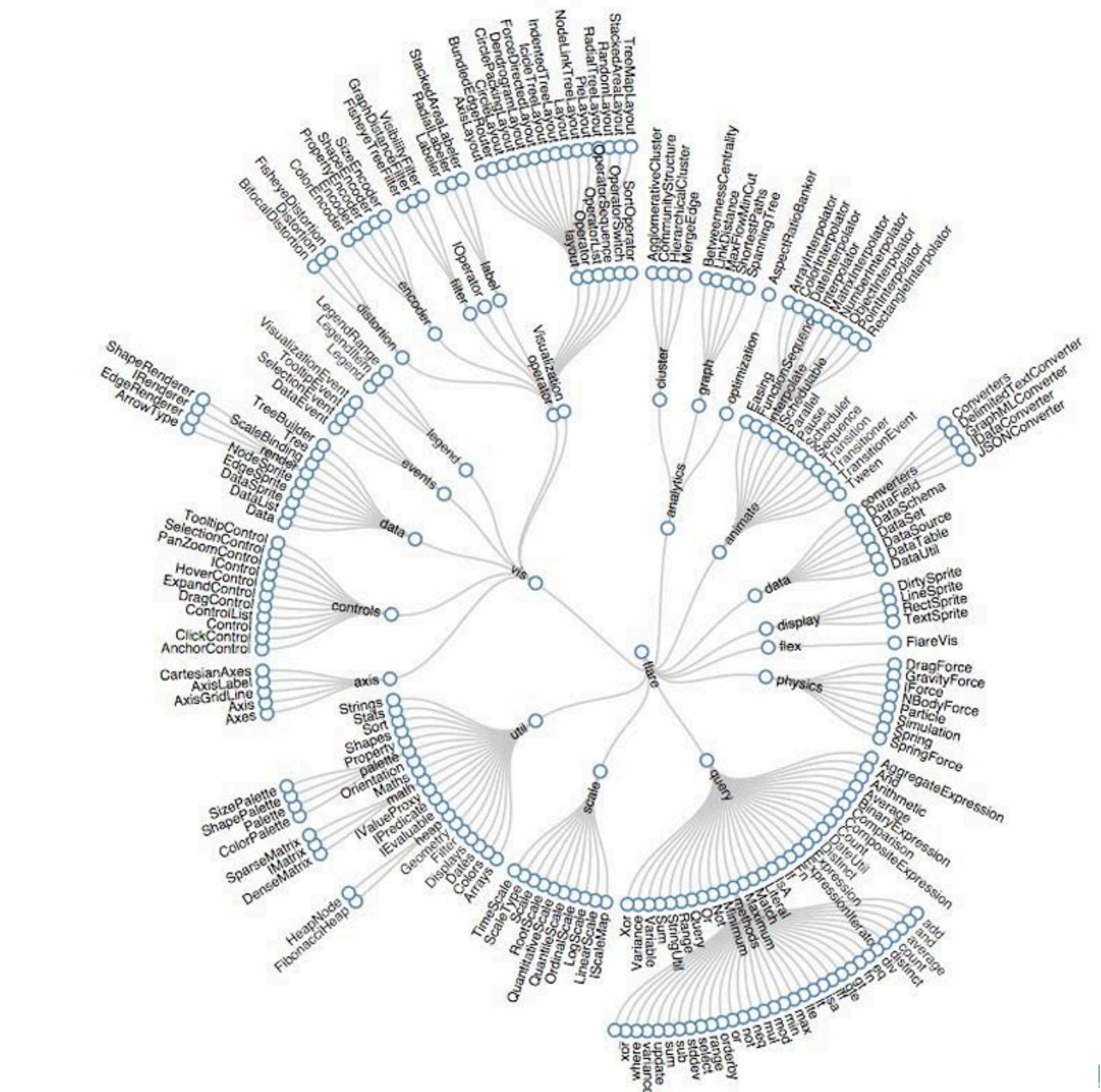
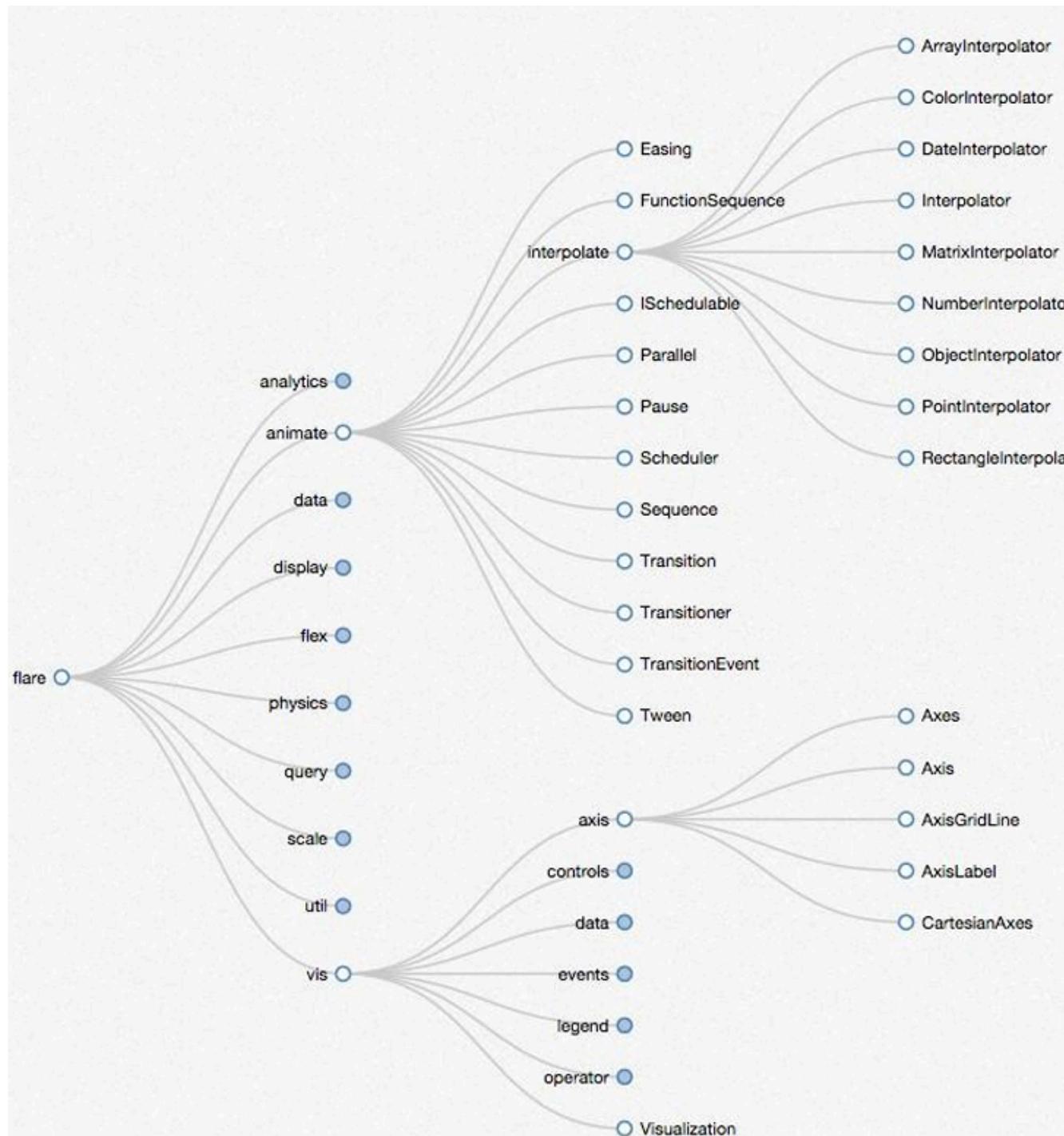
2D space typically used to break apart breadth and depth

- ex:
 - depth = y axis
 - breadth = x axis

Often space is used to convey hierarchical orientations



Node-Link Diagrams



Node-Link Diagrams: Reingold-Tilford

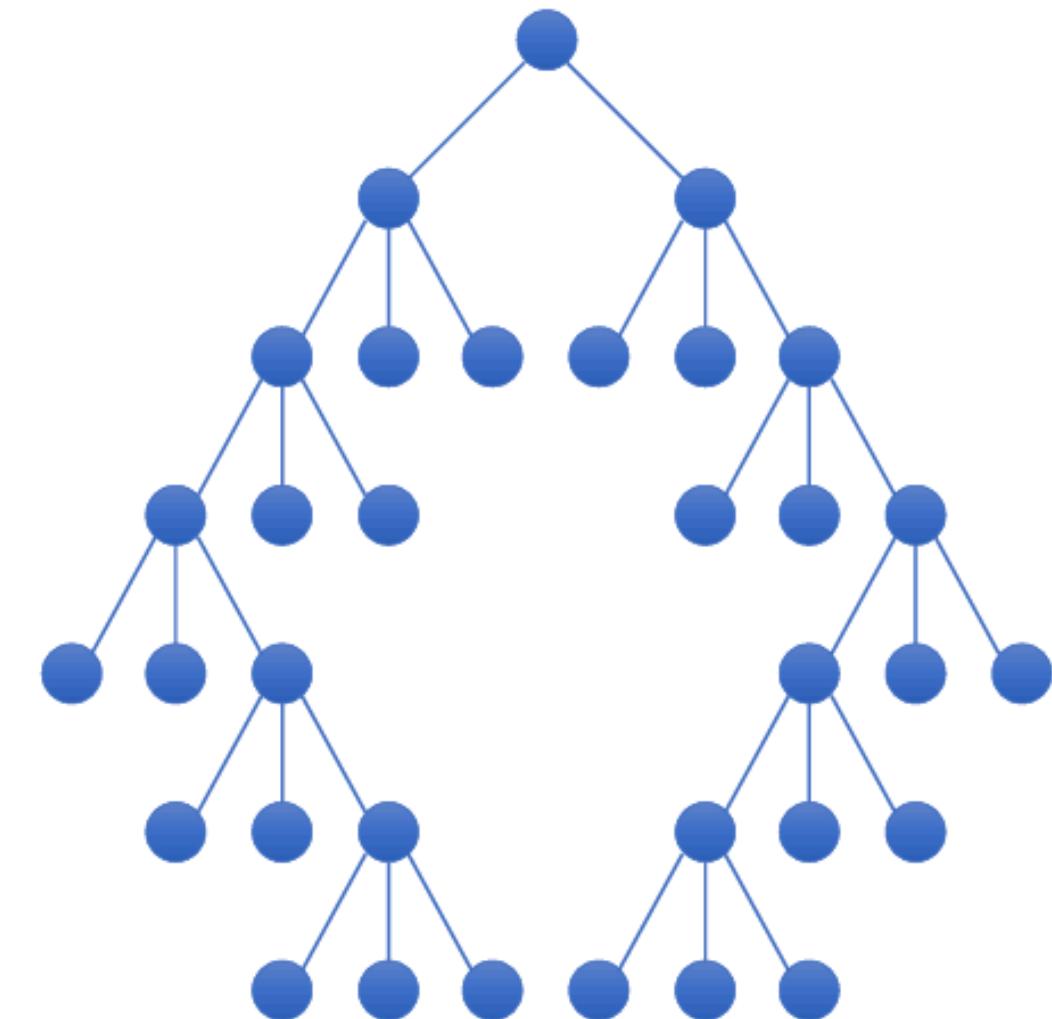
Repeatedly divide space based on leaf count

- breadth and depth on separate dimensions

Goal: maximize density and symmetry

Design Concerns:

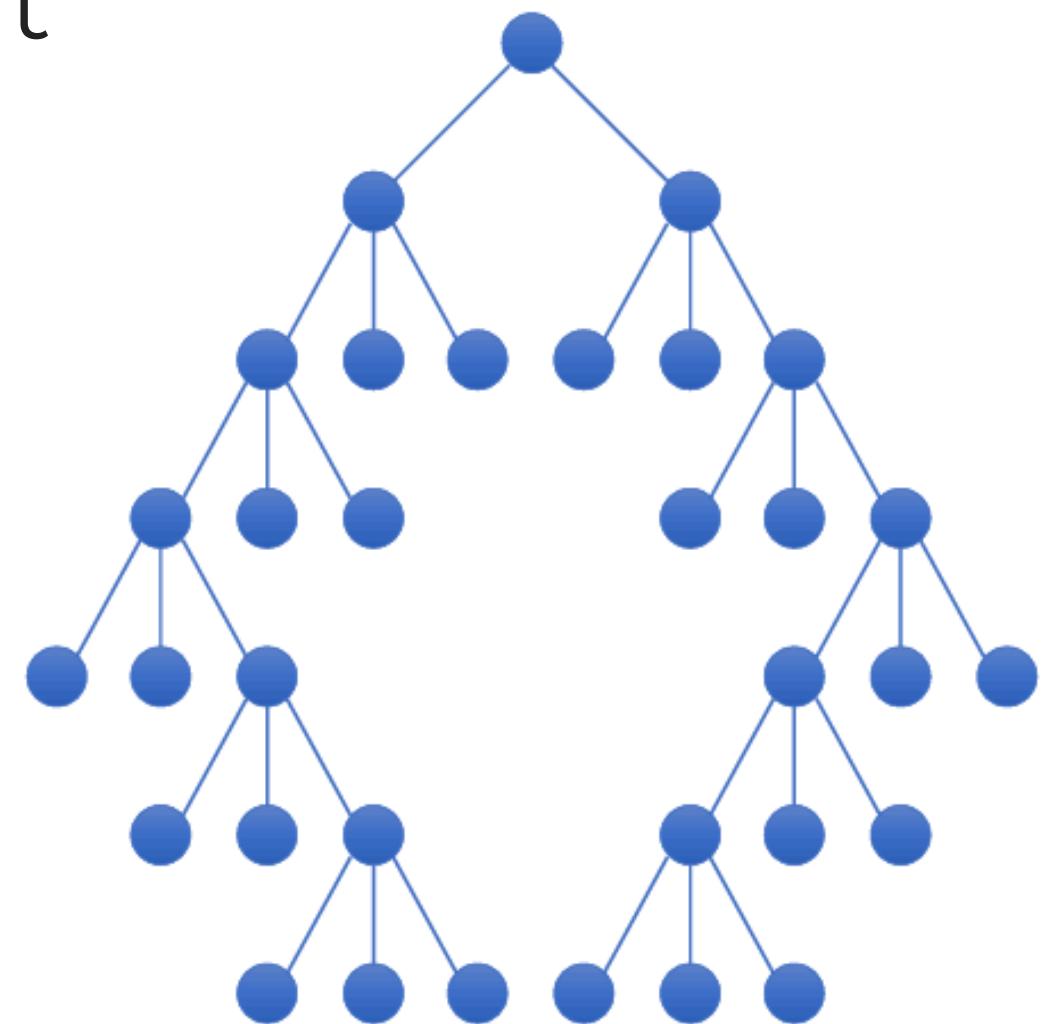
- clearly encode node depth
- no edge crossings
- isomorphic subtrees drawn identically
- compact



Node-Link Diagrams: Reingold-Tilford

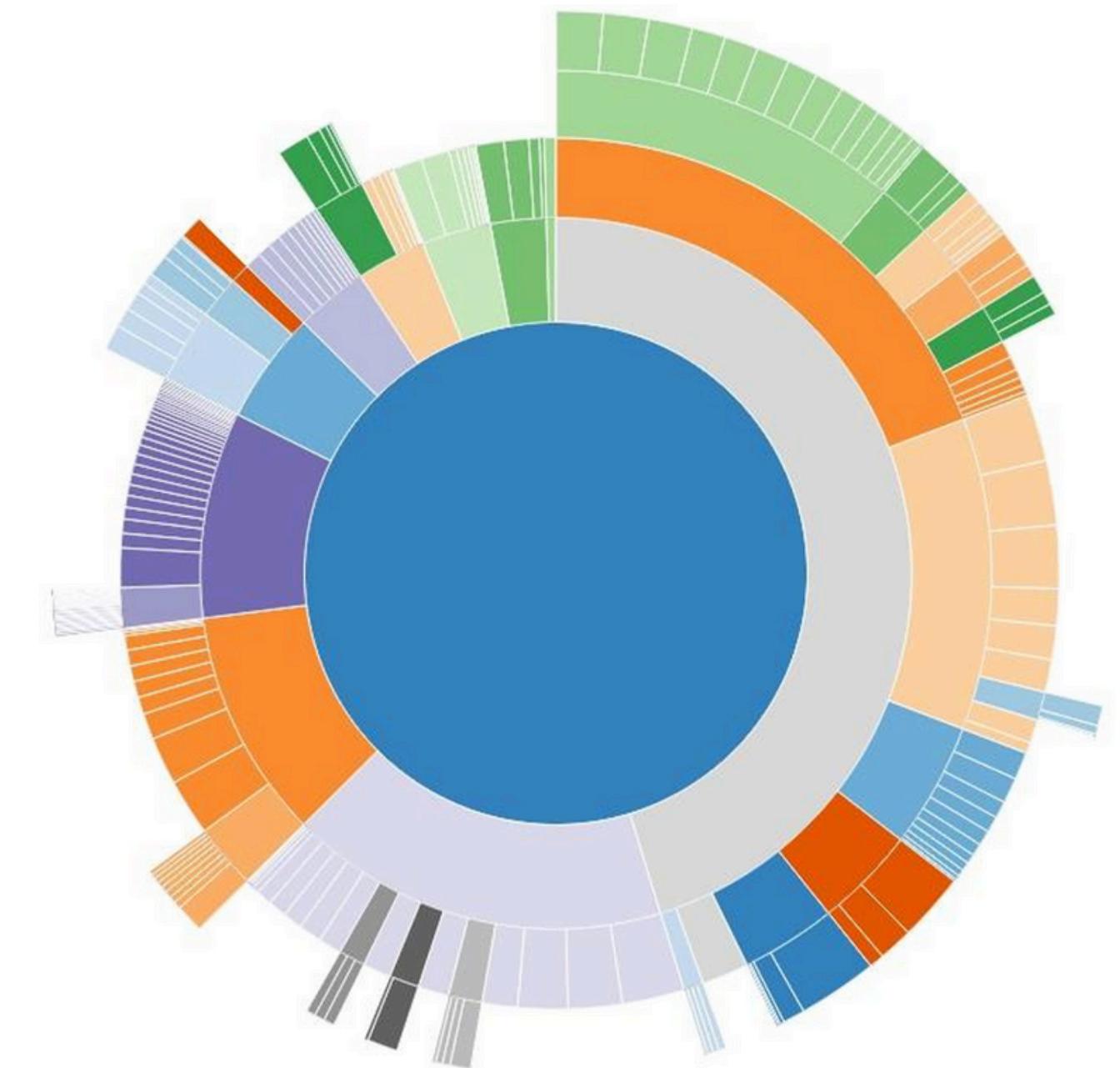
Approach:

- bottom-up recursive
- make sure subtree is drawn before parent
- pack subtrees as close as possible
- center parent over subtrees



Layered Diagrams

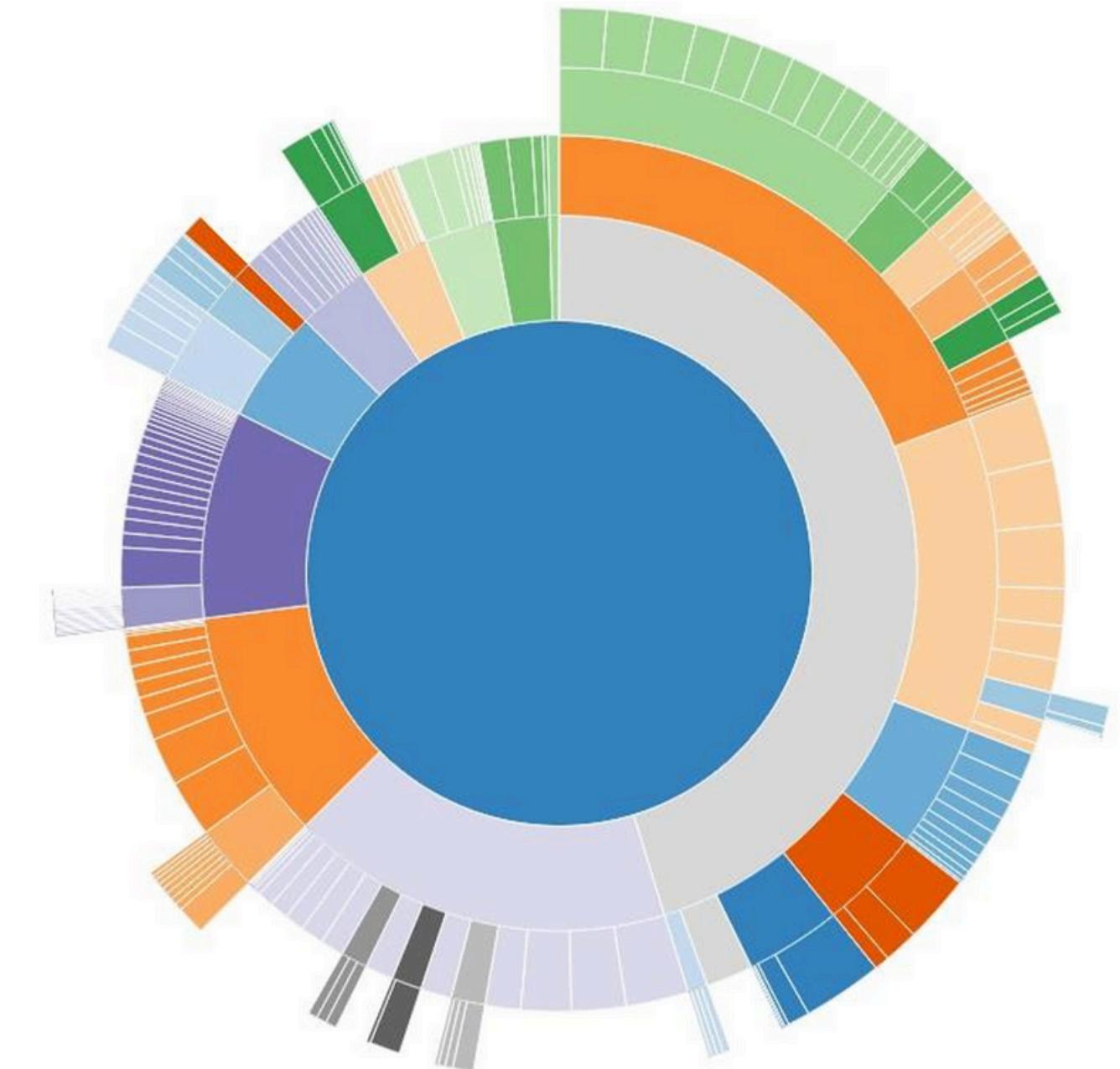
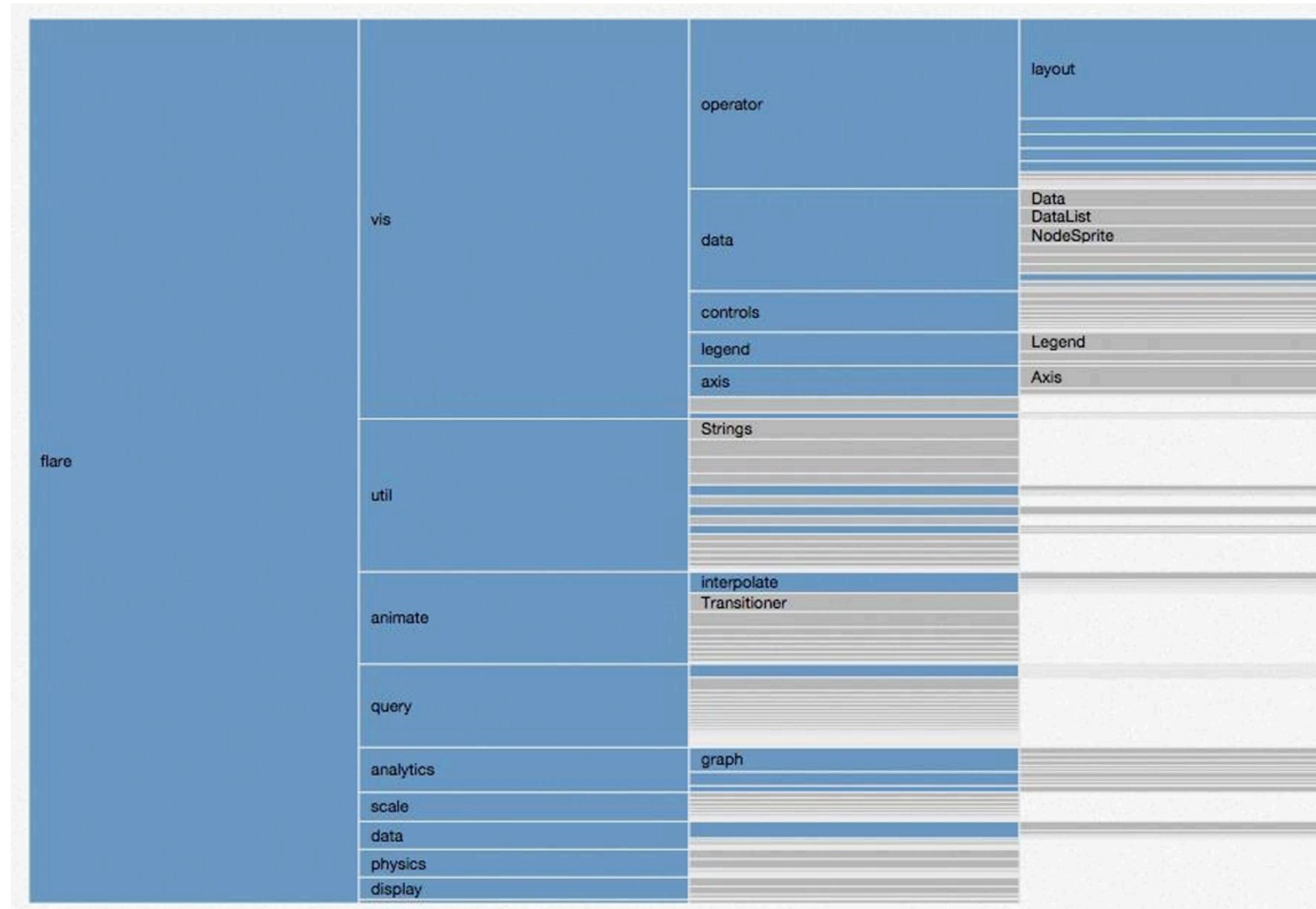
- Recursive subdivision of space
- Use layering, adjacency, and/or alignment to encode structure



Layered Diagrams

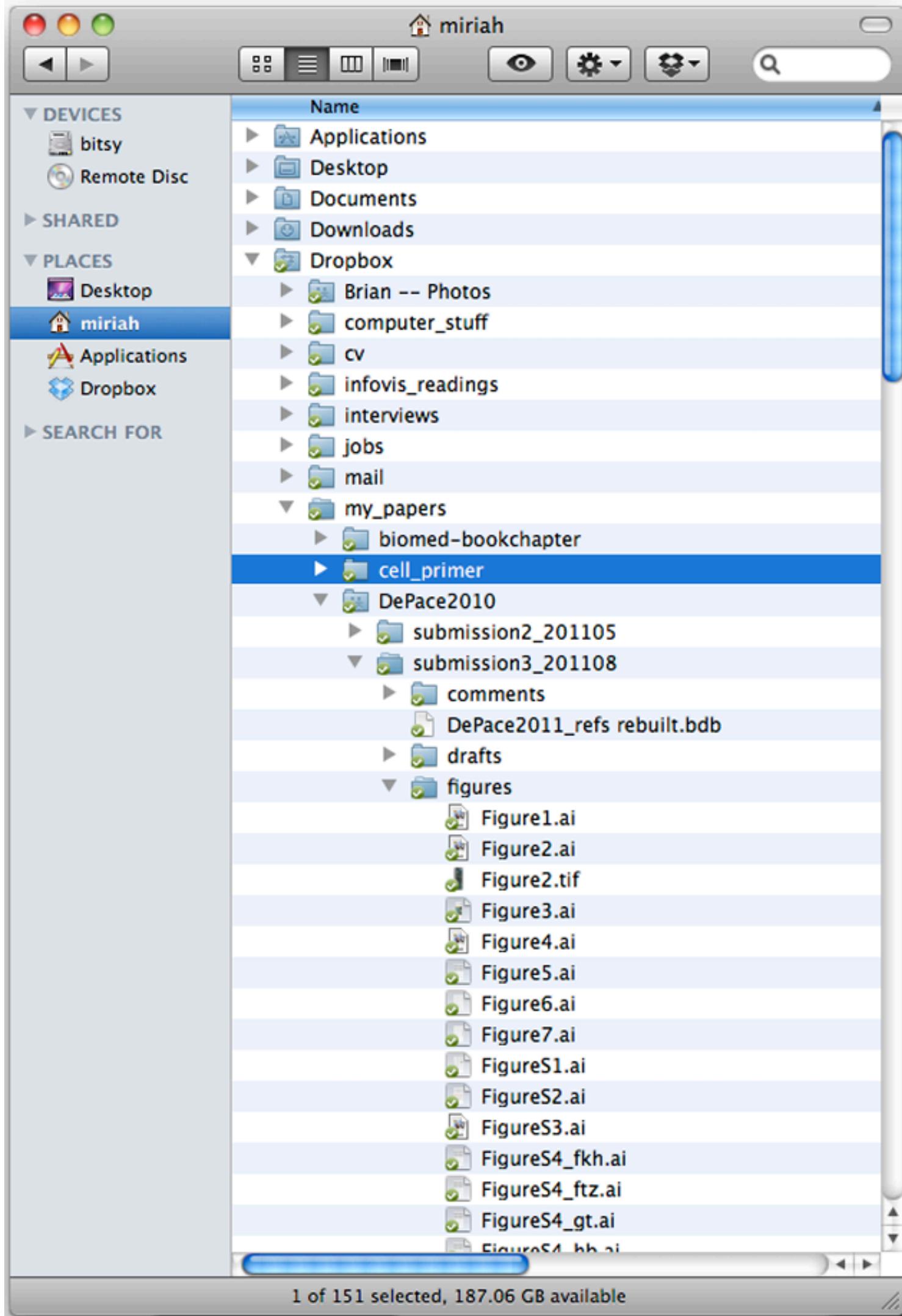
Tree breadth often grows exponentially (quickly run out of room)

Solutions: Scroll/Pan; Zoom/Filter; Hyperbolic Layout



Indentation

- used to show parent/child relationship
- breadth and depth contend for space
- Often requires a lot of scrolling



Enclosure Diagrams

Encode structure using spatial enclosure

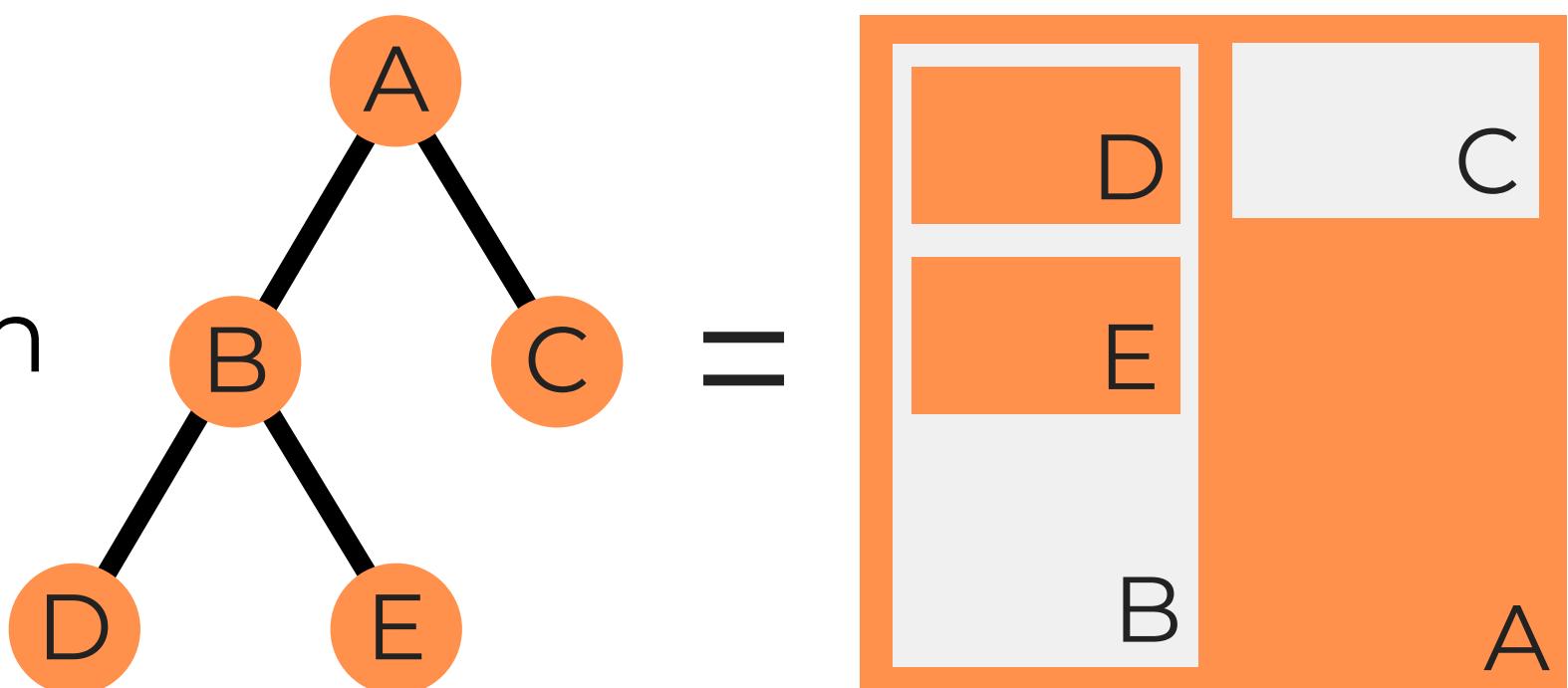
- often referred to as **treemaps**

Pros:

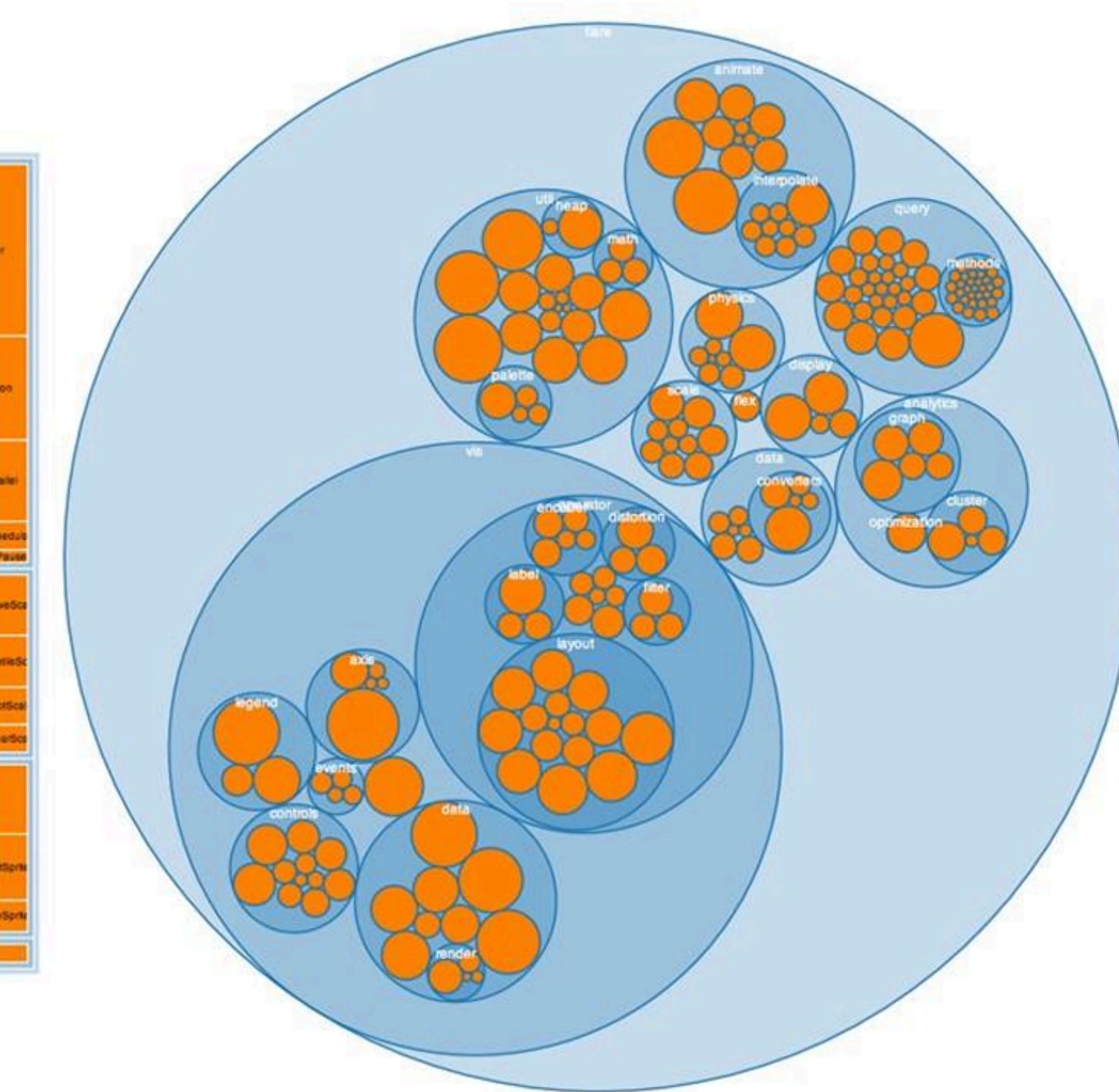
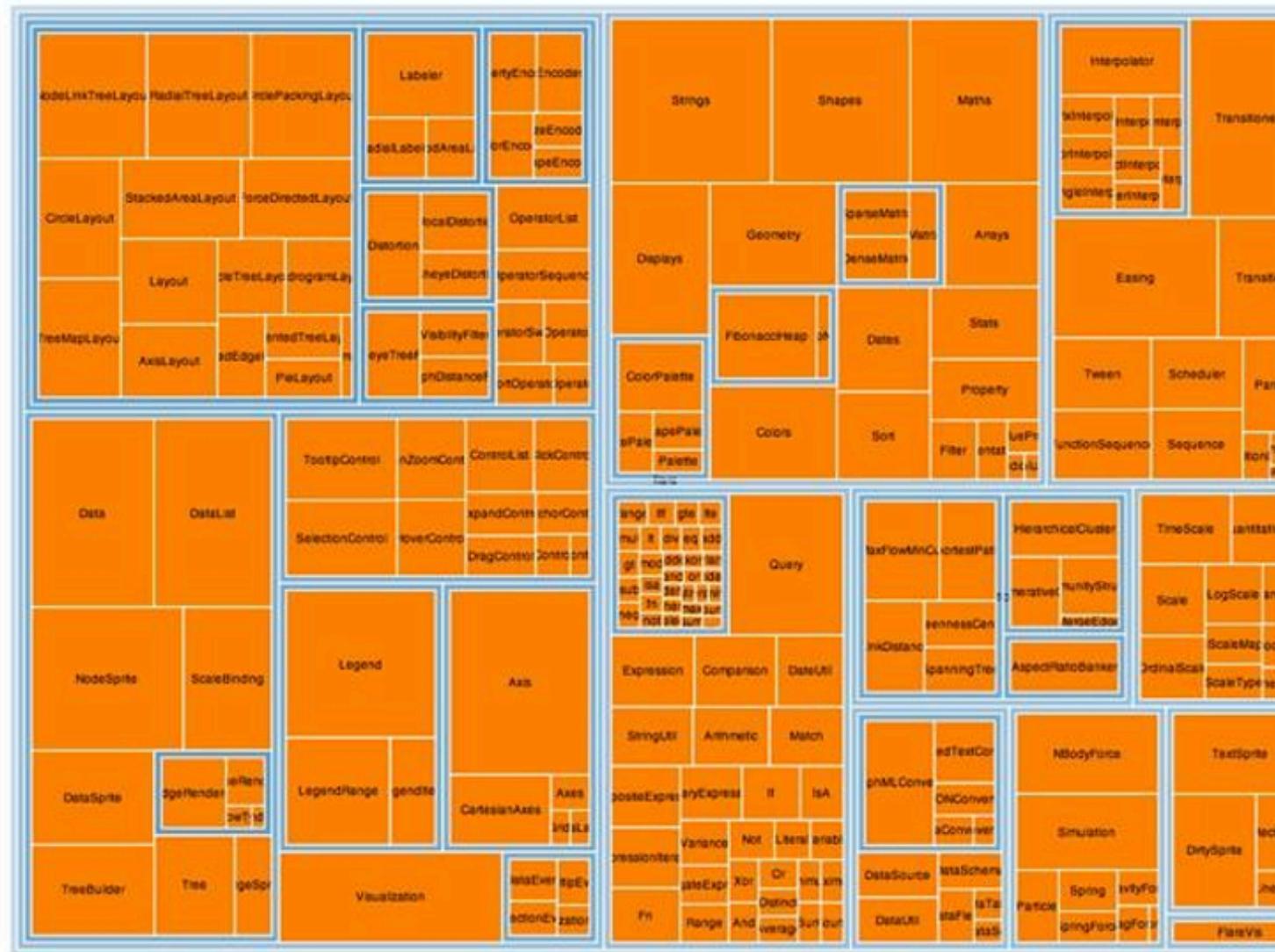
- whole tree visualized in a single view
- easy to spot nodes with larger/smaller subtrees

Cons:

- difficult to accurately read depth



Enclosure Diagrams



Visualizing Graphs

Visualizing Graphs: Node-Link Diagrams

A Graph-Drawing Exercise

Create an aesthetically-pleasing node-link diagram representation

	1	2	3	4	5	6	7	8	9	10
1	0	0	1	0	0	1	1	0	0	0
2	0	0	1	0	0	1	0	1	1	0
3	1	1	0	0	0	0	0	0	0	1
4	0	0	0	0	1	0	1	0	1	0
5	0	0	0	1	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	1	1
7	1	0	0	1	0	0	0	1	0	0
8	0	1	0	0	1	0	1	0	0	0
9	0	1	0	1	0	1	0	0	0	0
10	0	0	1	0	0	1	0	0	0	0

Spatial Layout

Primary concern of graph drawing is the **spatial layout of edges and nodes**

Often (but not always) the goal is to effectively depict the **graph structure**:

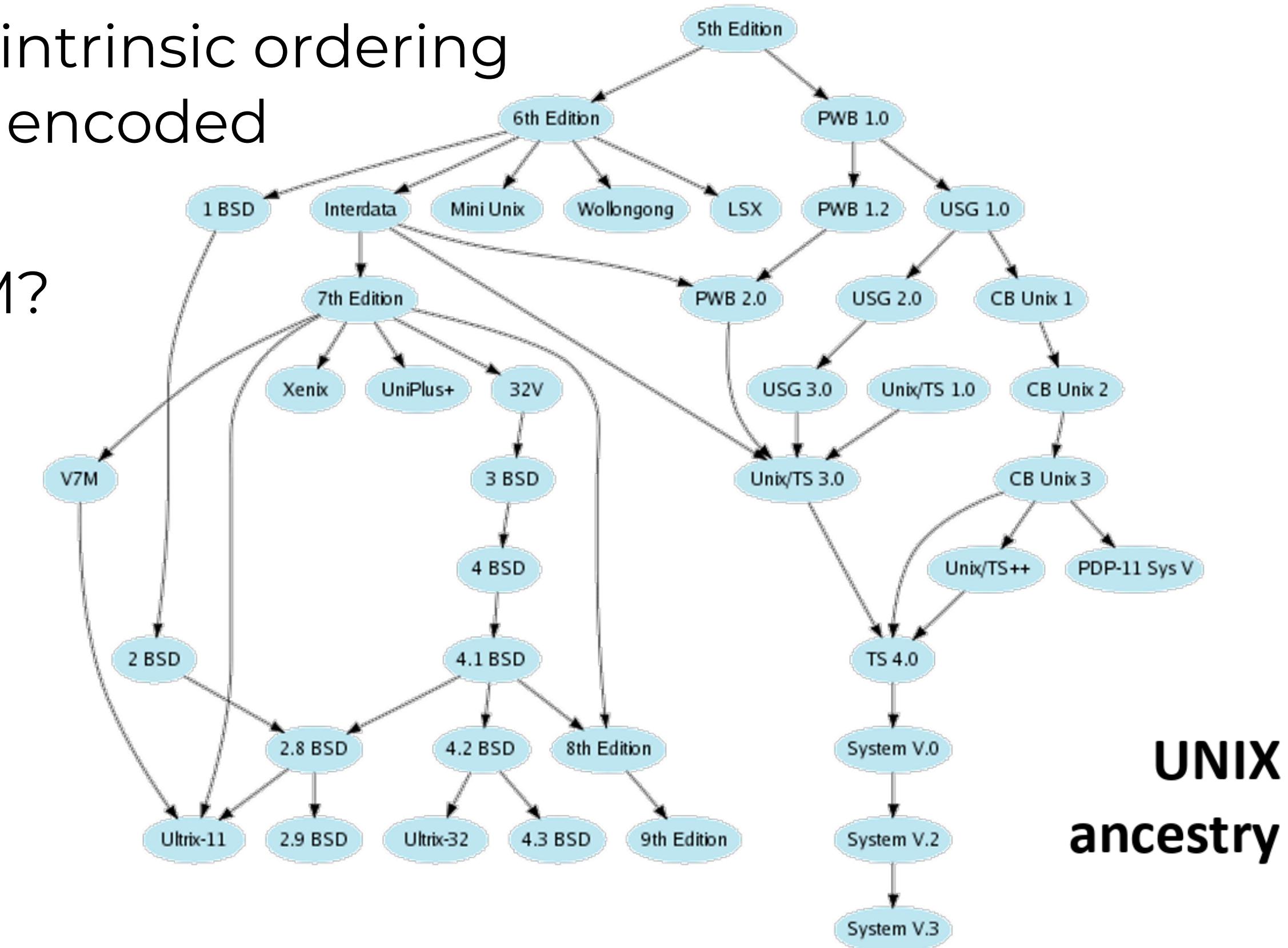
- connectivity, path-following
- network distance
- clustering
- ordering (e.g., hierarchical level)

Sugiyama Algorithm (DAGs)

Great for graphs with an intrinsic ordering

- but depth not strictly encoded

What is the depth of V7M?
3 BSD?



Sugiyama Algorithm (DAGs)

Pros:

- nice, readable top-down flow
- relatively fast

Cons:

- not really suitable for graphs that don't have a hierarchy
- difficult to implement

Use GraphViz: <http://www.graphviz.org/>

Force-Directed Layout

What if your graph has no intrinsic layering?

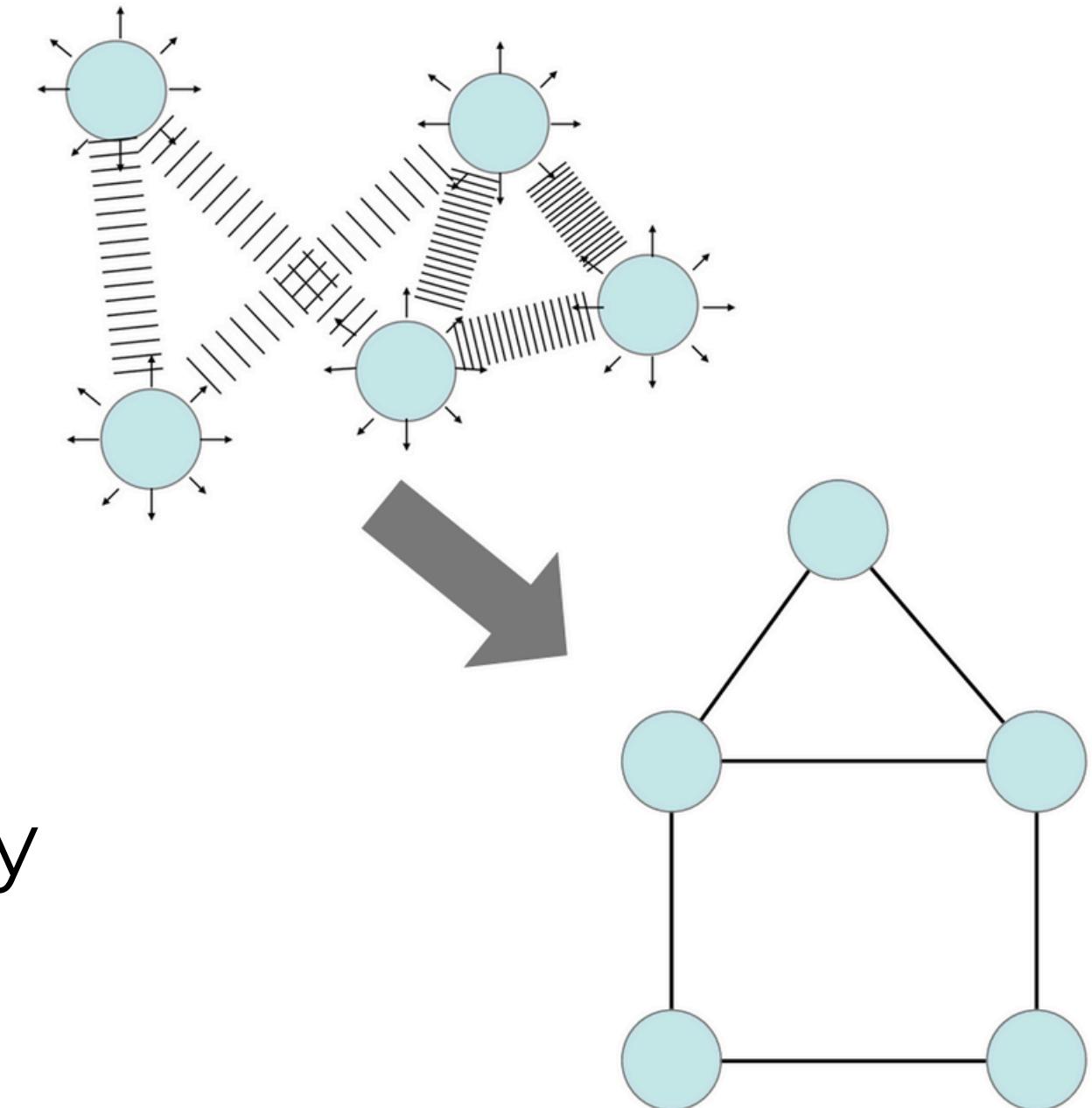
- use a physics-based model

Many variations, but generally:

- edges = springs
- nodes = repulsive particles

Requires multiple iterations

- location of nodes is updated iteratively



Force-Directed Layout: Physics Review

$$F = ma$$

- Force = mass * acceleration

$$\Delta v = a \Delta t$$

- Change in velocity = acceleration * time step

$$p' = p + v \Delta t + \frac{1}{2} a \Delta t^2$$

- new position = old position + ...

Force-Directed Layout: Force Model

$$\vec{a}_i = \frac{\vec{F}_i}{m_i}$$

- acceleration = Force / mass

$$\vec{v}'_i = \vec{v}_i + \vec{a}_i \Delta t$$

- new velocity = old velocity + change in velocity

$$p'_i = p_i + \vec{v}'_i \Delta t + \frac{1}{2} \vec{a}_i \Delta t^2$$

- new position = old position + new velocity * time step + ...

Force-Directed Layout: Force Model

- Use force to update acceleration
- Use acceleration to update velocity
- Use velocity and acceleration to update position
- Each node is repulsed by every other node
- Only **connected** nodes are attracted

Force-Directed Layout: Force Model

Repulsive Force $f_R(d) = \frac{C_R m_1 m_2}{d^2}$

- C_R is a strength constant
- m_1, m_2 are node masses
- d is a distance between nodes

Attractive Force $f_A(d) = C_A \max\{0, (d - L)\}$

- C_A is a strength constant
- L is the rest length of the spring
- d is a distance between nodes

Force-Directed Layout: Force Model

Repulsive Force

$$F_R(P) = \sum_{\text{all neighbors } Q} (Q - P) f_R(||P - Q||)$$

Attractive Force

$$F_A(P) = \sum_{\text{all neighbors } Q} (P - Q) f_R(||Q - P||)$$

Force-Directed Layout: Algorithm

- start from a random layout
- Loop:
 - for every pair of nodes, compute repulsive force
 - for every edge, compute attractive force
 - accumulate forces per node and update velocity
 - update each node position in direction of velocity
- Stop when layout is “good enough”

Force-Directed Layout: Improvements

What values do we set for:

- time step Δt ?
 - fixed time step (time since last frame was drawn)
- initial position p_i ?
 - random position
- initial velocity v_i ?
 - zero
- mass m_i ?
 - depends (the heavier a node is, the slower it moves)

Force-Directed Layout: Improvements

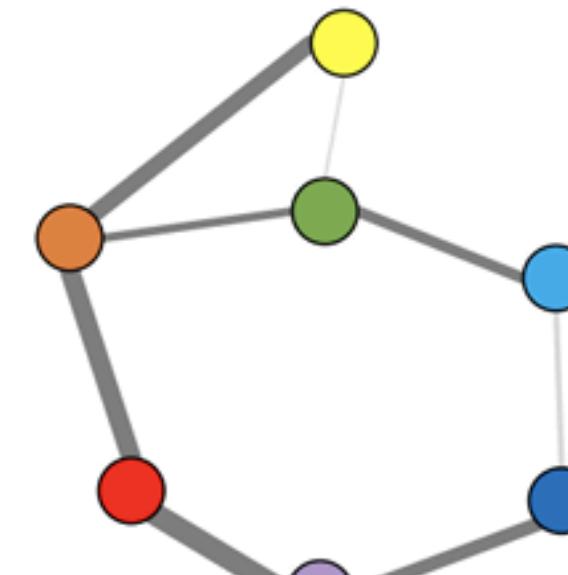
What values do we set for:

- force constants C_R, C_A ?
 - start with something small (weaker force)
- rest length of the spring L ?
 - closest distance you would like between nodes
 - They will end up being closer
 - 10 to 20 pixels is a good start

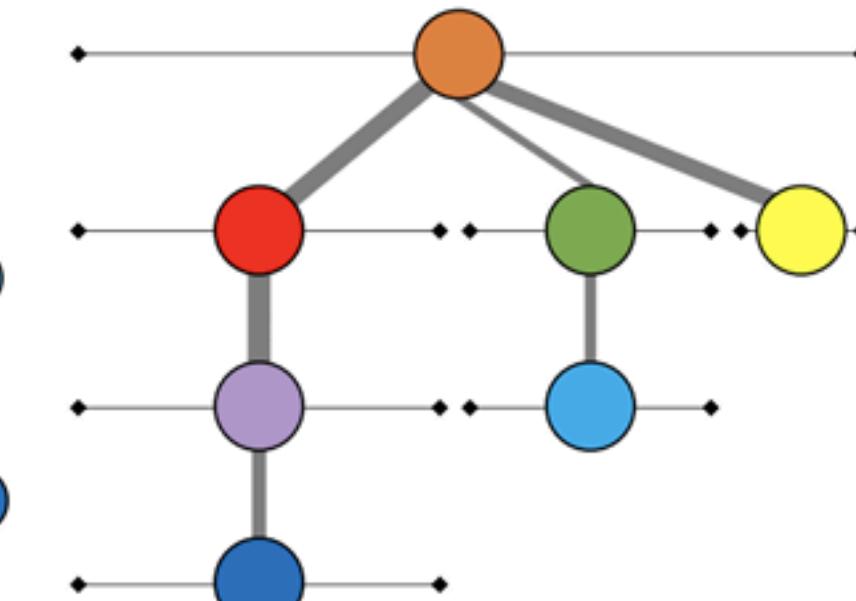
Force-Directed Layout: Improvements

Improving the initial starting position:

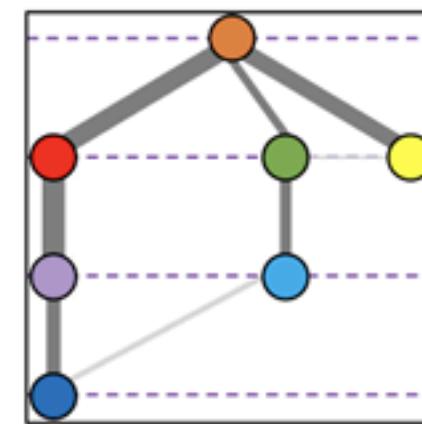
- consider the topology of the graph when initializing
- layout the **maximal spanning tree** in a **layered** or **radial** configuration



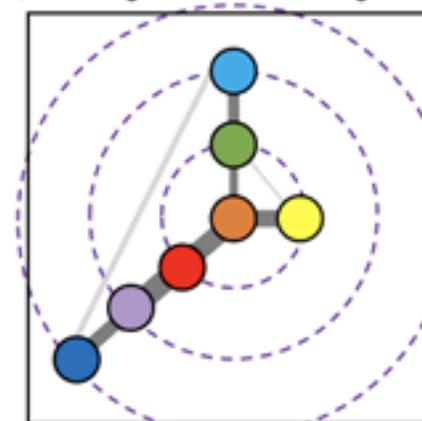
(a) Graph with maximal spanning tree edges in dark grey



(b) Abstract layout calculates node depth and distributes horizontal space by subtree sizes

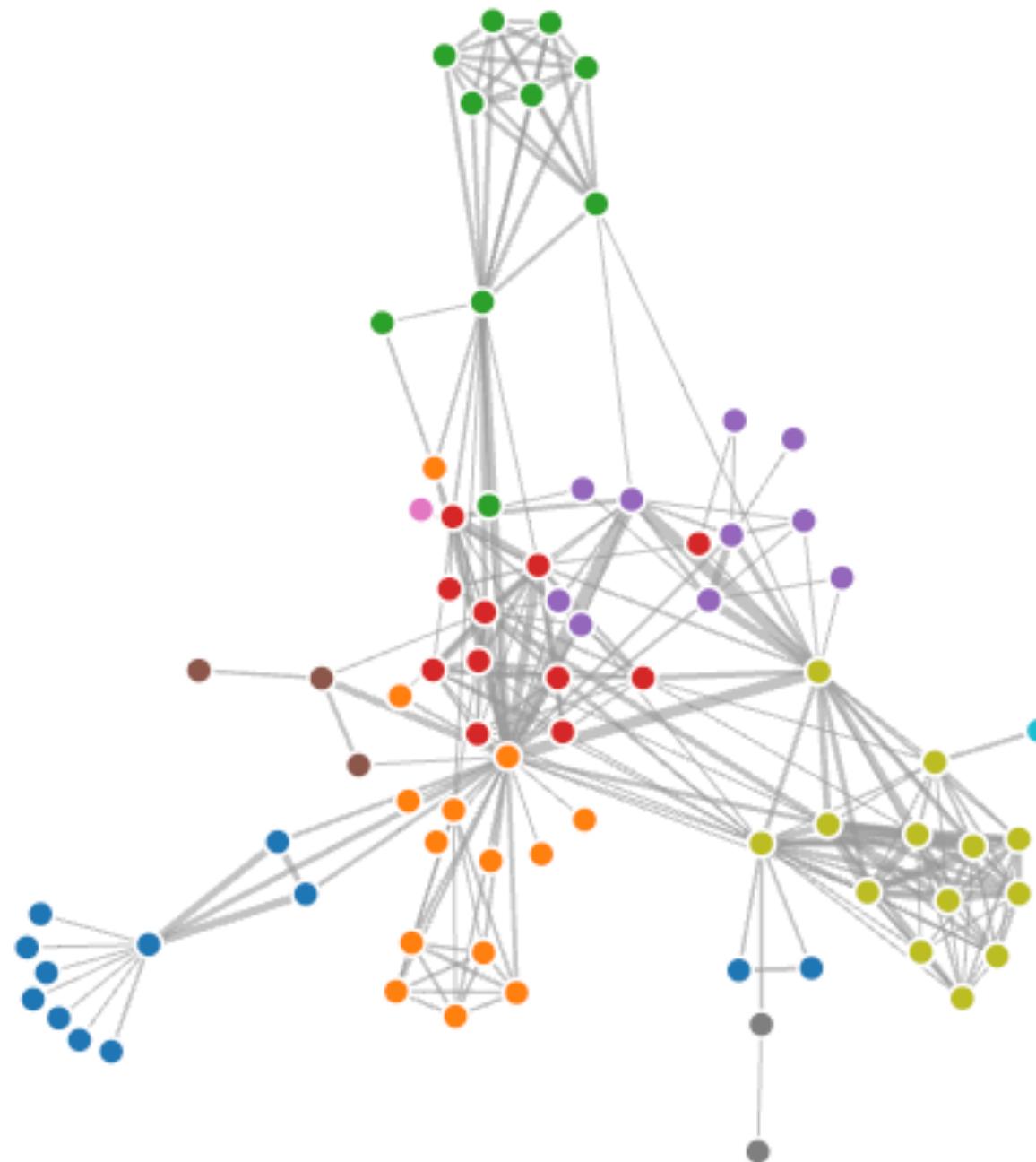


(c) Layered Layout



(d) Radial Layout

Force-Directed Layout



<https://observablehq.com/@d3/force-directed-graph/2>

Force-Directed Layout

Pros:

- very flexible, aesthetic layouts for many graph types
- can add custom forces
 - e.g., repulsion from edge of screen, attraction to center of screen
- relatively easy to implement

Cons:

- repulsion loop is $O(n^2)$ for each iteration
- doesn't work as well for highly-connected graphs

Other Node-Link Layouts

Orthogonal:

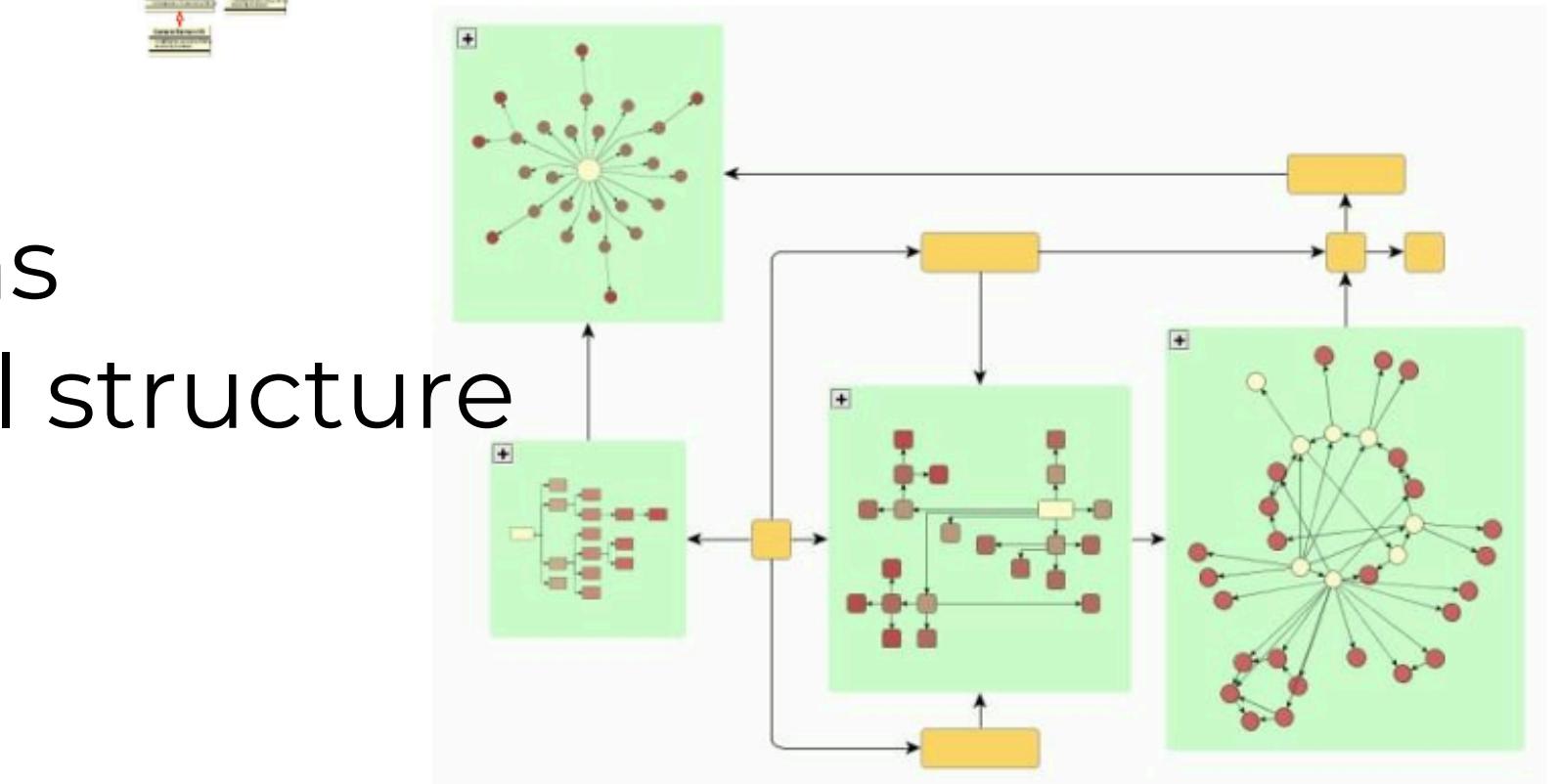
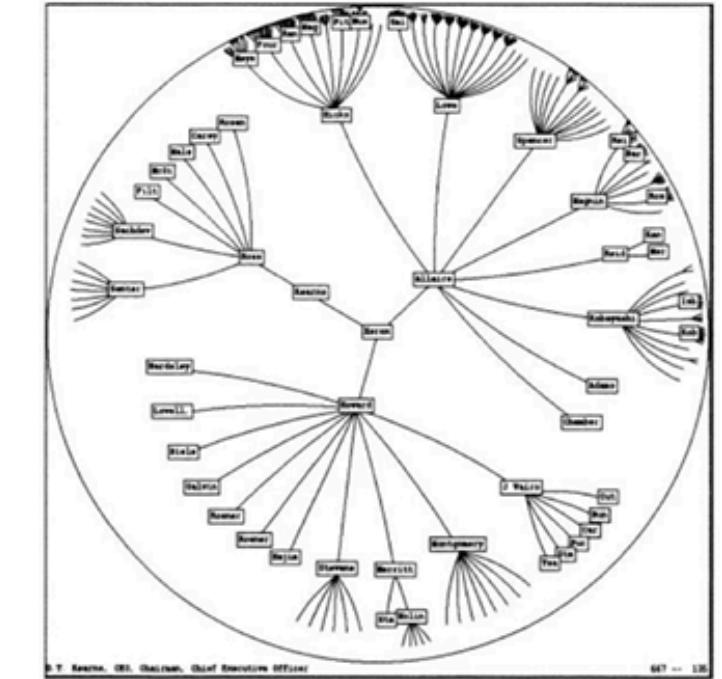
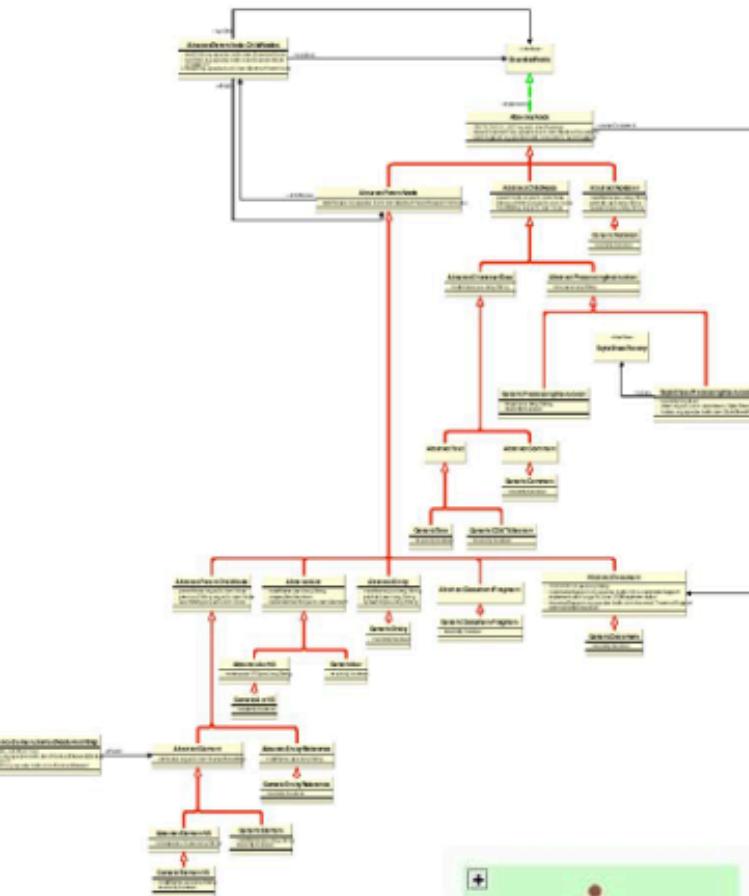
- useful for UML diagrams

Circular:

- emphasizes ring topologies
 - centers specific nodes

Nested:

- recursively apply layout algorithms
 - useful for graphs with hierarchical structure



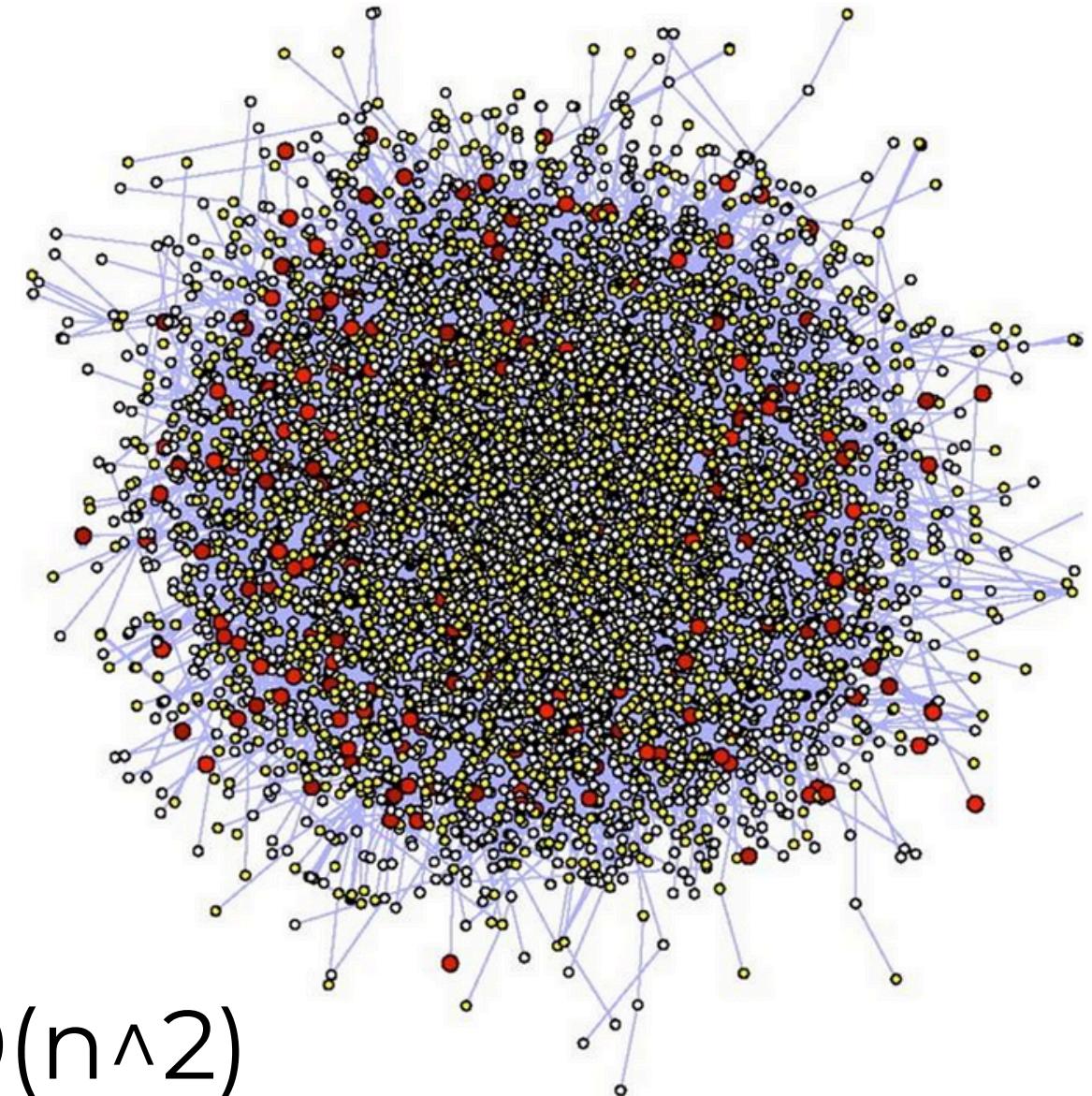
Node-Link Layouts: A Review

Pros:

- understandable visual mapping
- overall structure, clusters, paths visible
- flexible, many variations

Cons:

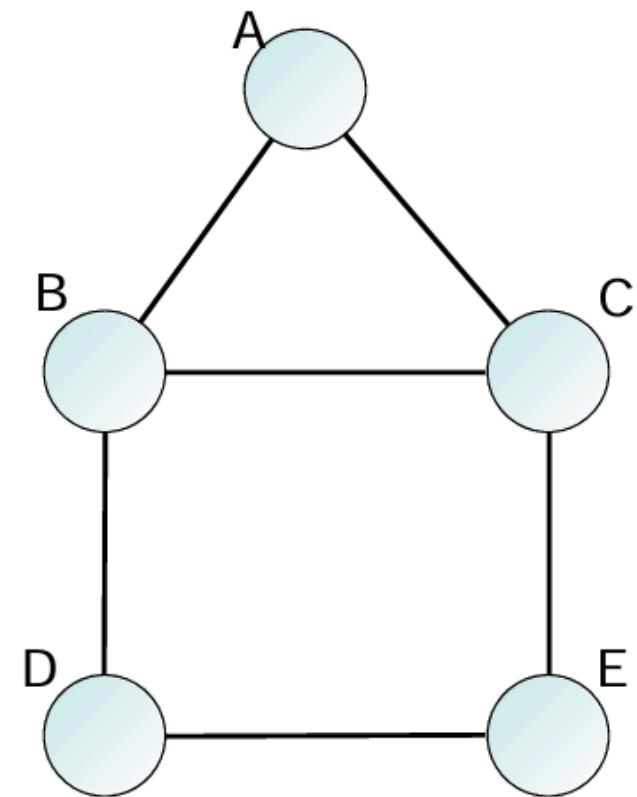
- all but the most trivial algorithms are $> O(n^2)$
- not great for dense graphs
 - hairballs!



Visualizing Graphs: Adjacency Diagrams

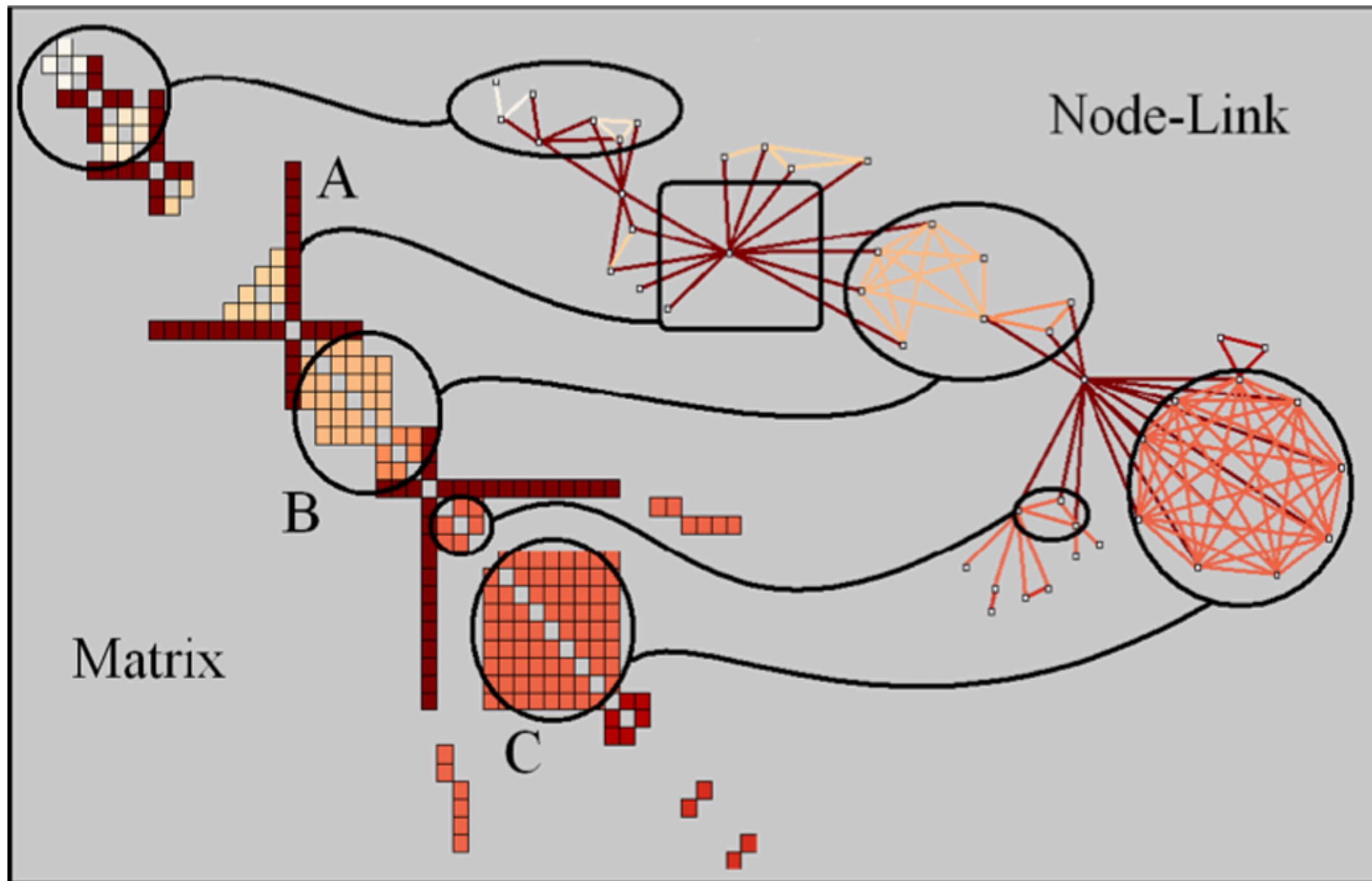
Adjacency Diagrams

Instead of node-link diagrams,
visualize the adjacency matrix

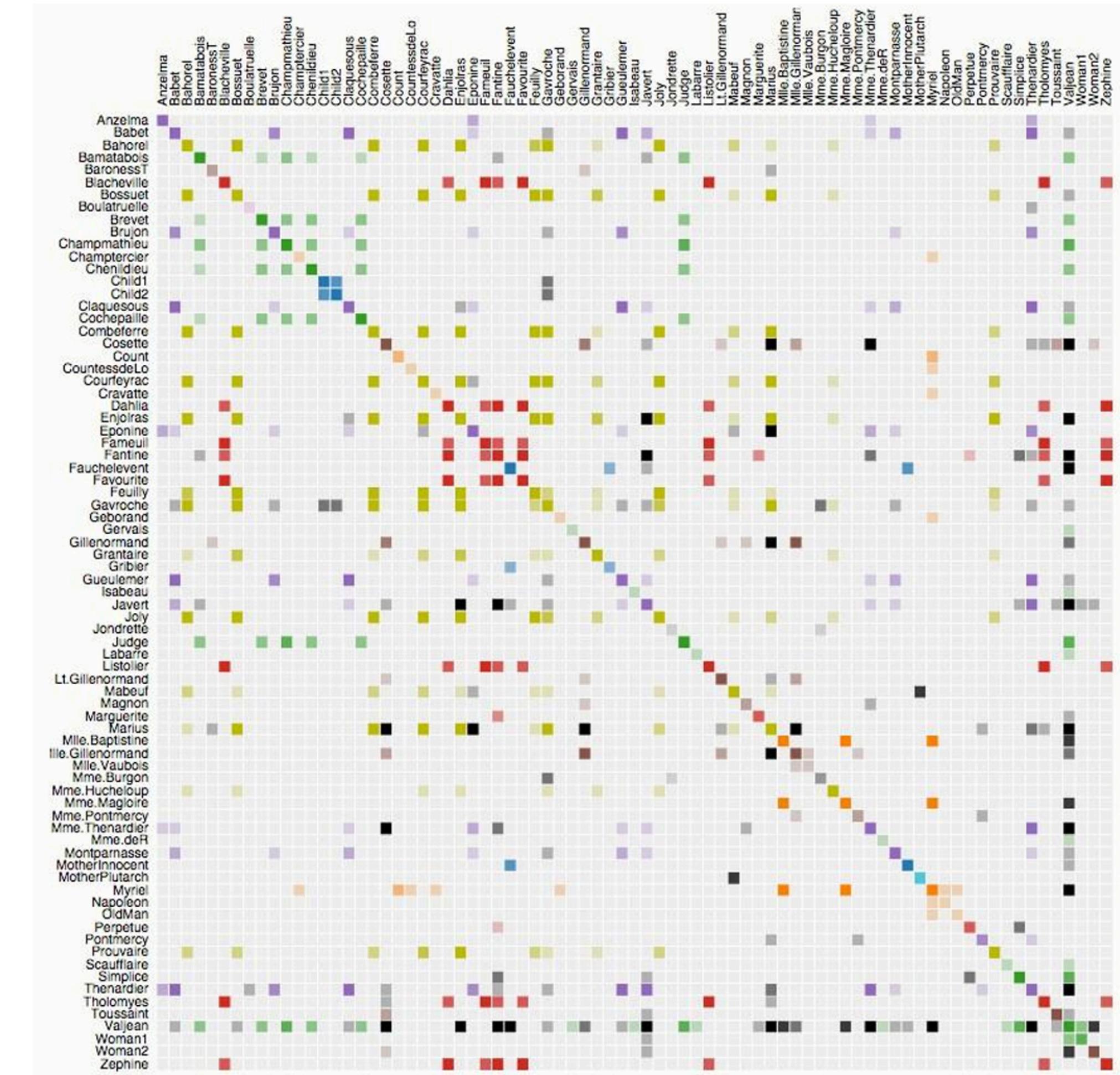


	A	B	C	D	E
A					
B					
C					
D					
E					

Adjacency Diagrams: Spotting Patterns



Adjacency Diagrams



Adjacency Diagrams

Pros:

- great for dense graphs
- visually scalable
- clusters are noticeable

Cons:

- order of rows affects what is visually prominent
- a bit abstract
- hard to follow paths

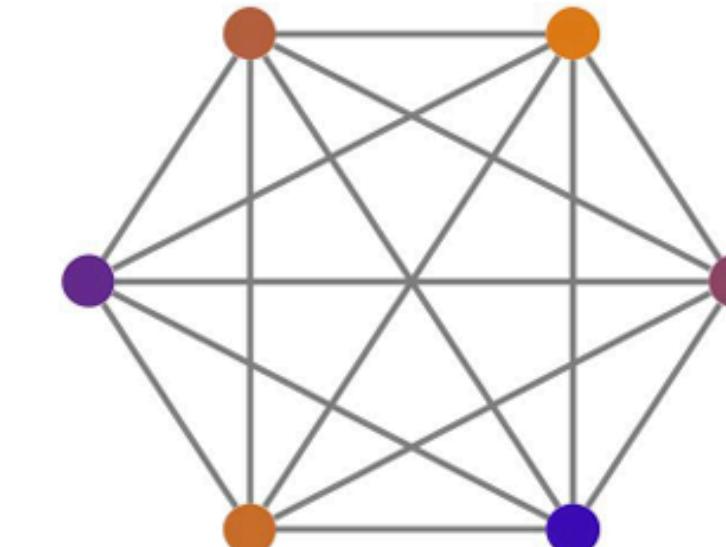
Visualizing Graphs: Motif Glyphs

Motif Glyphs (Aggregate Views)

Connector



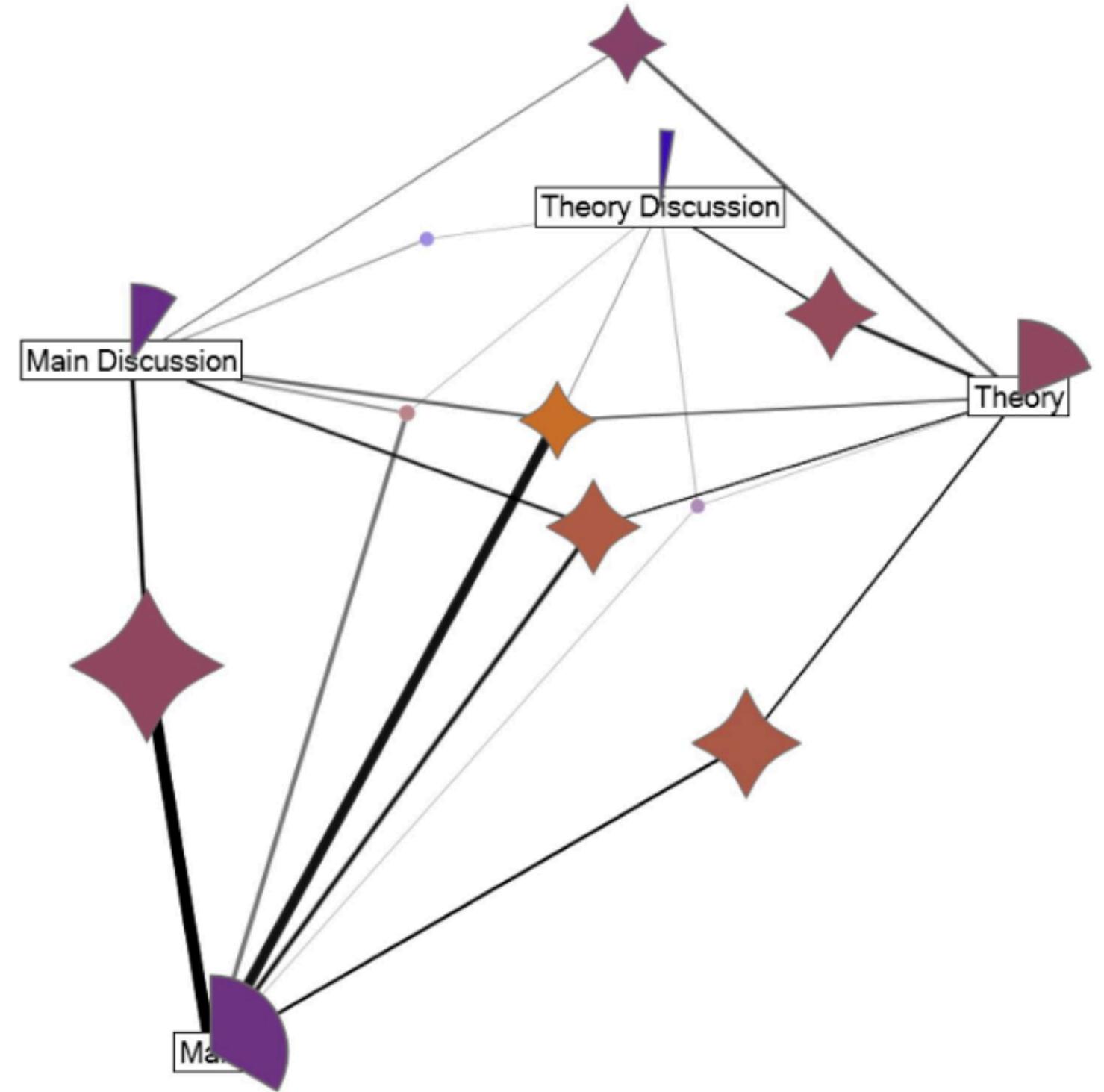
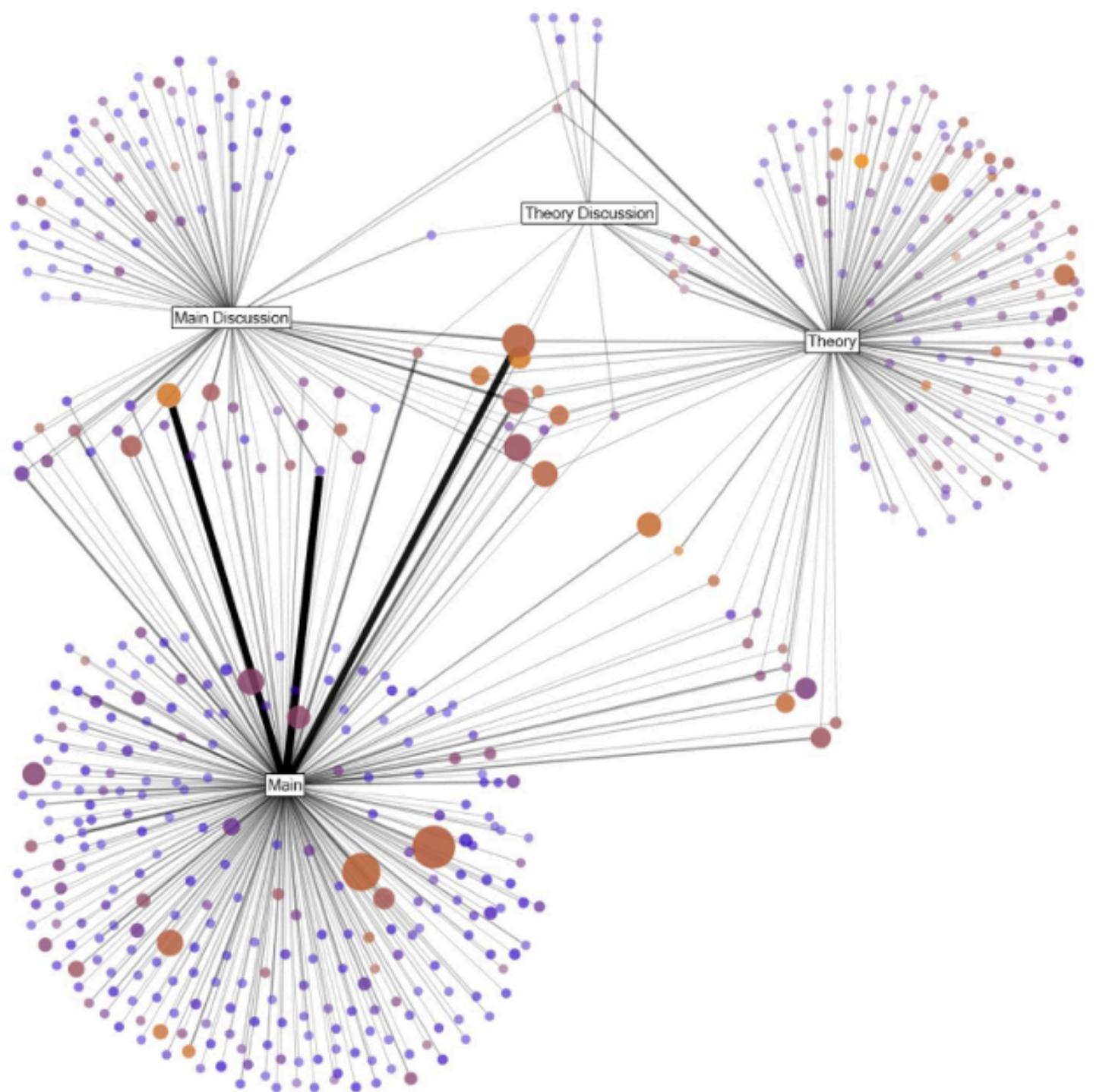
Clique



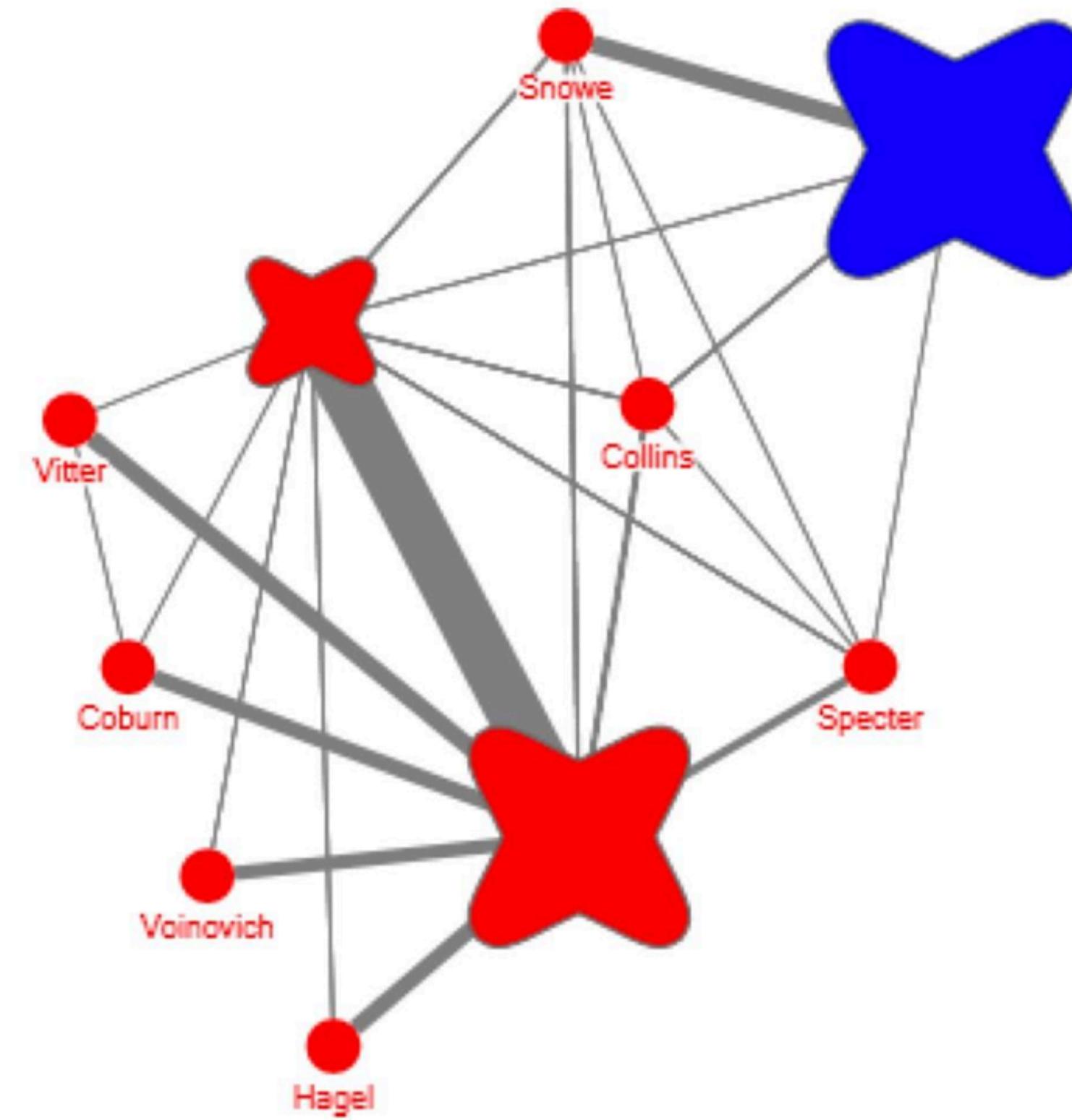
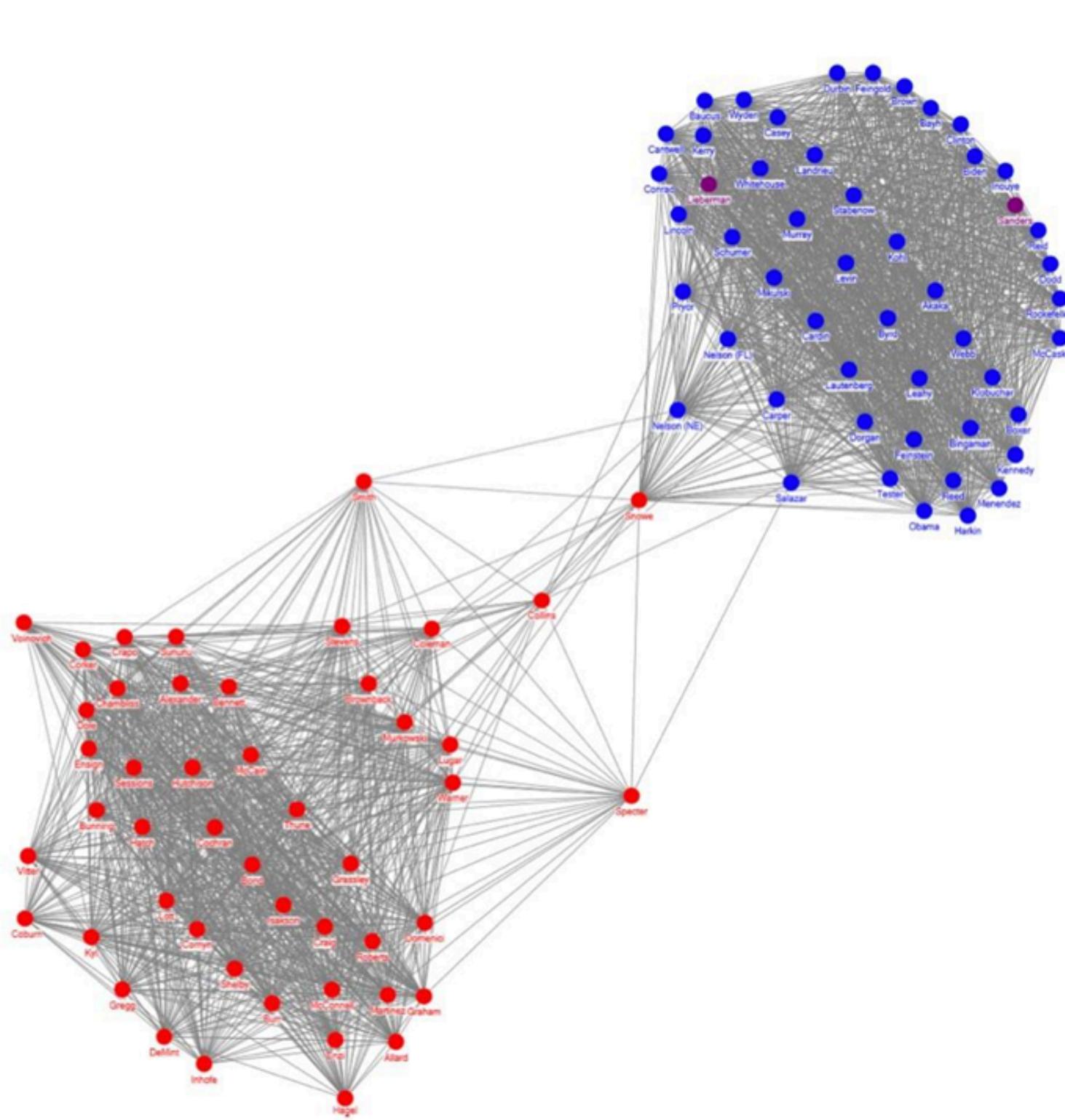
Fan



Motif Glyphs (Aggregate Views)



Motif Glyphs (Aggregate Views)



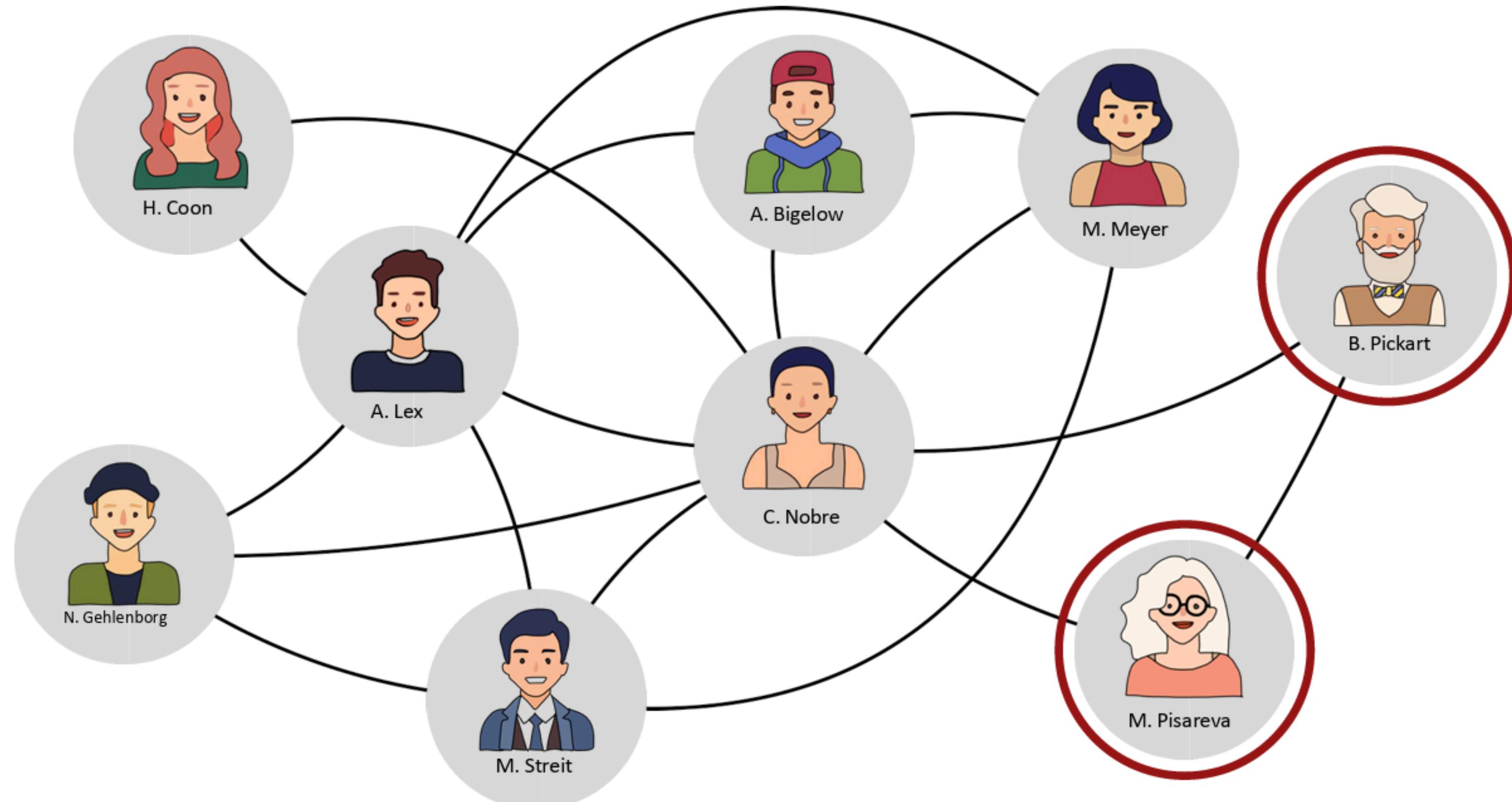
Multivariate Networks (MVN)

Multivariate Network

Network Topology + Node and Edge Attributes

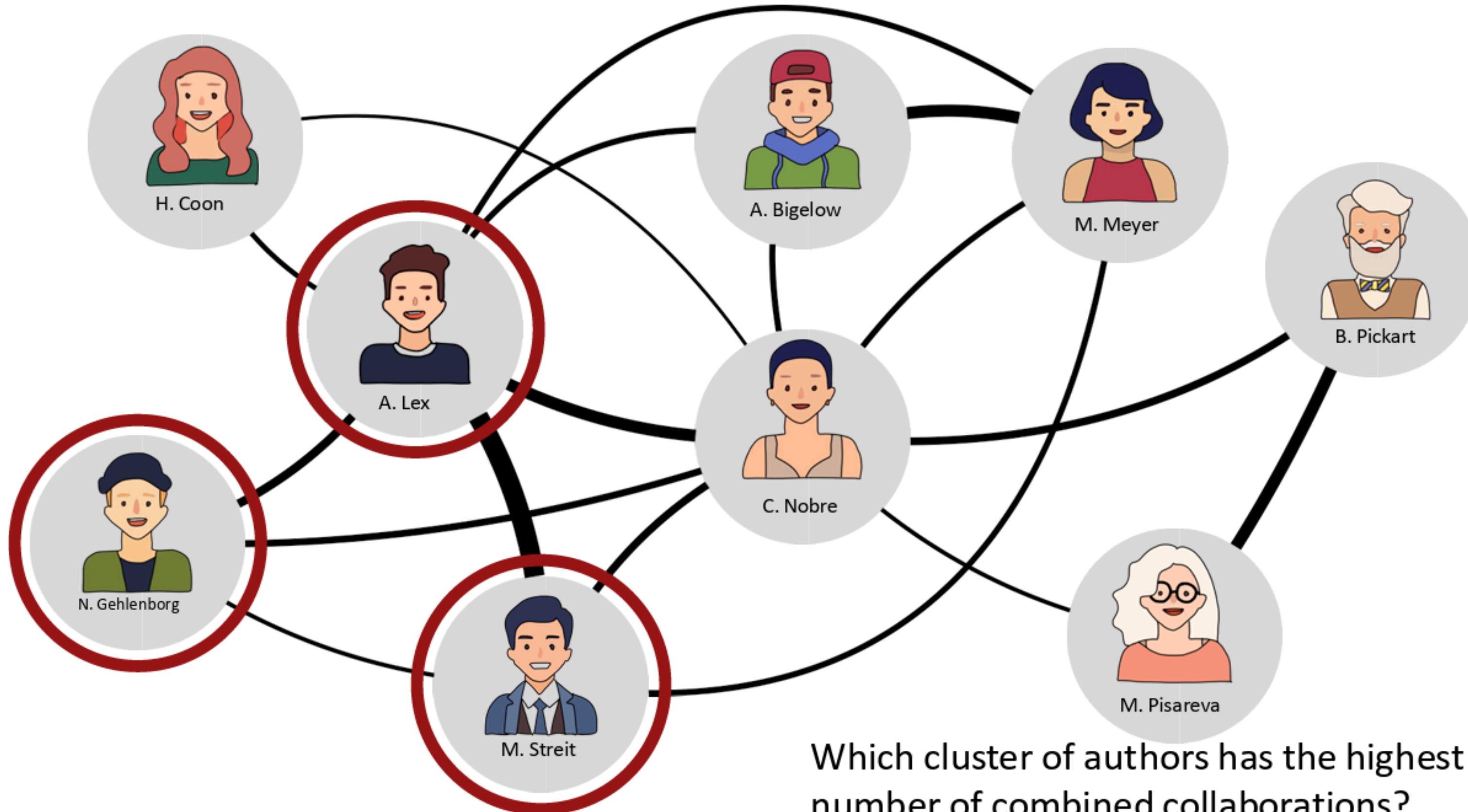
Often a **tradeoff** between visualizing network topology
and attributes

Multivariate Network Tasks

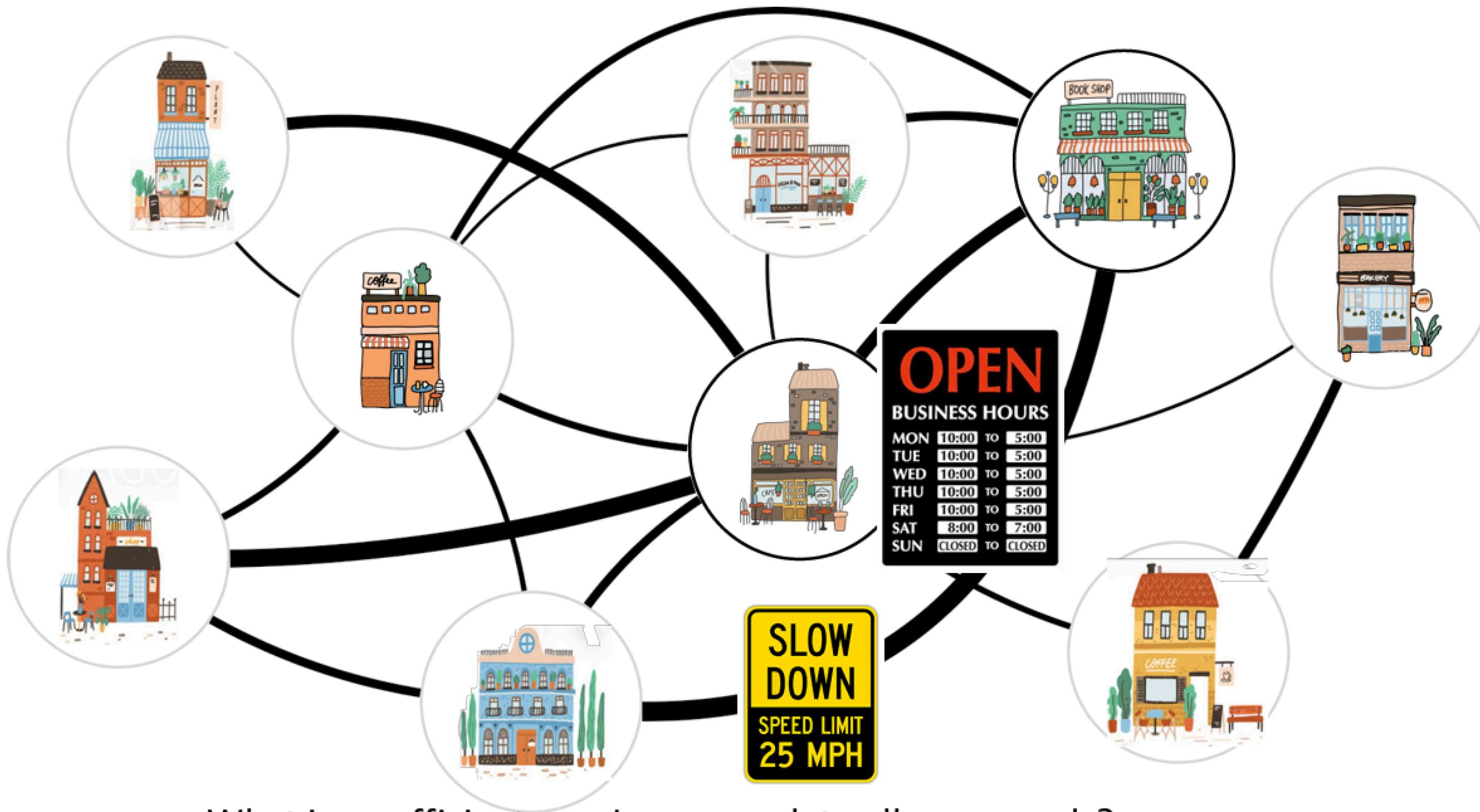


How many of my collaborators are from the oceanography field?

Multivariate Network Tasks



Multivariate Network Tasks



What is an efficient way I can complete all my errands?

Multivariate Network Tasks

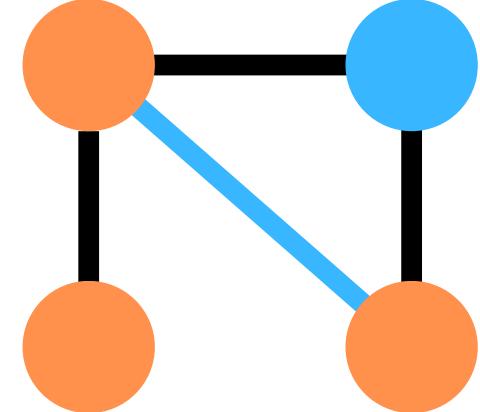
- How many of **my collaborators** are **in the oceanography field**?
- Which **cluster** has **the highest number of collaborations**?
- What is the **fastest route** to get all of my errands done?

Tasks that rely on:

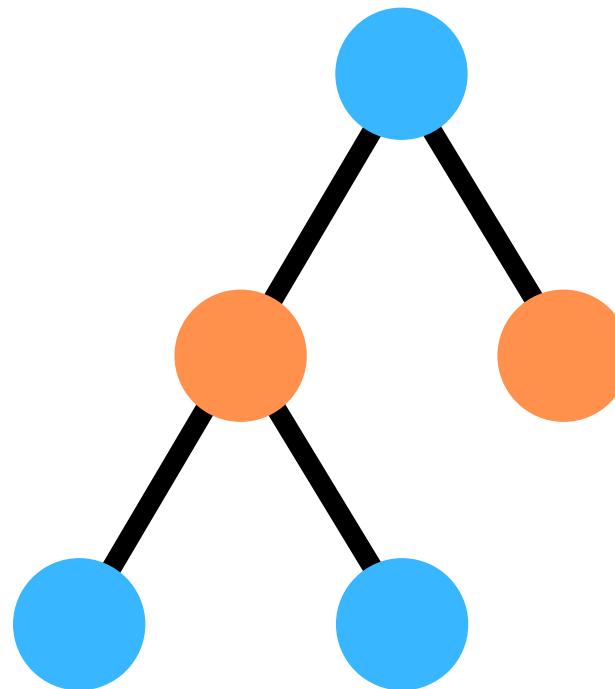
- the **topology** of the network, and
- the **attributes** of the nodes and edges

Multivariate Network Tasks

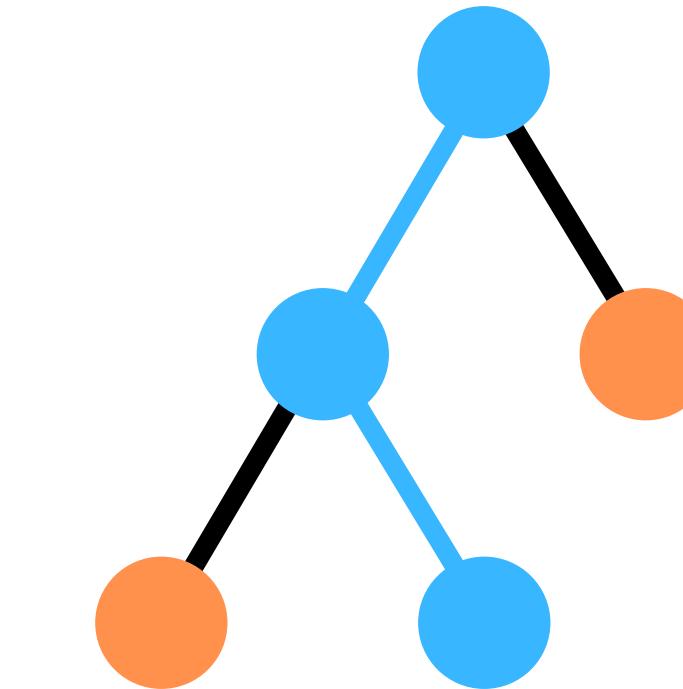
Topological Structures:



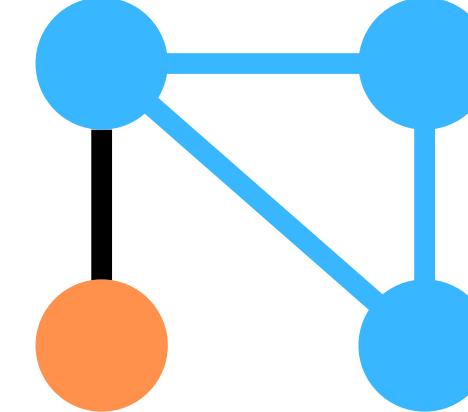
Single Node/Edge



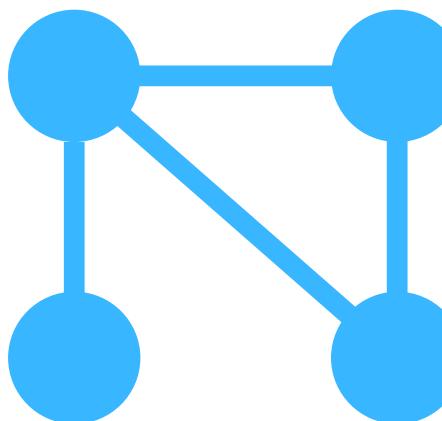
Node Neighbors



Paths



Clusters

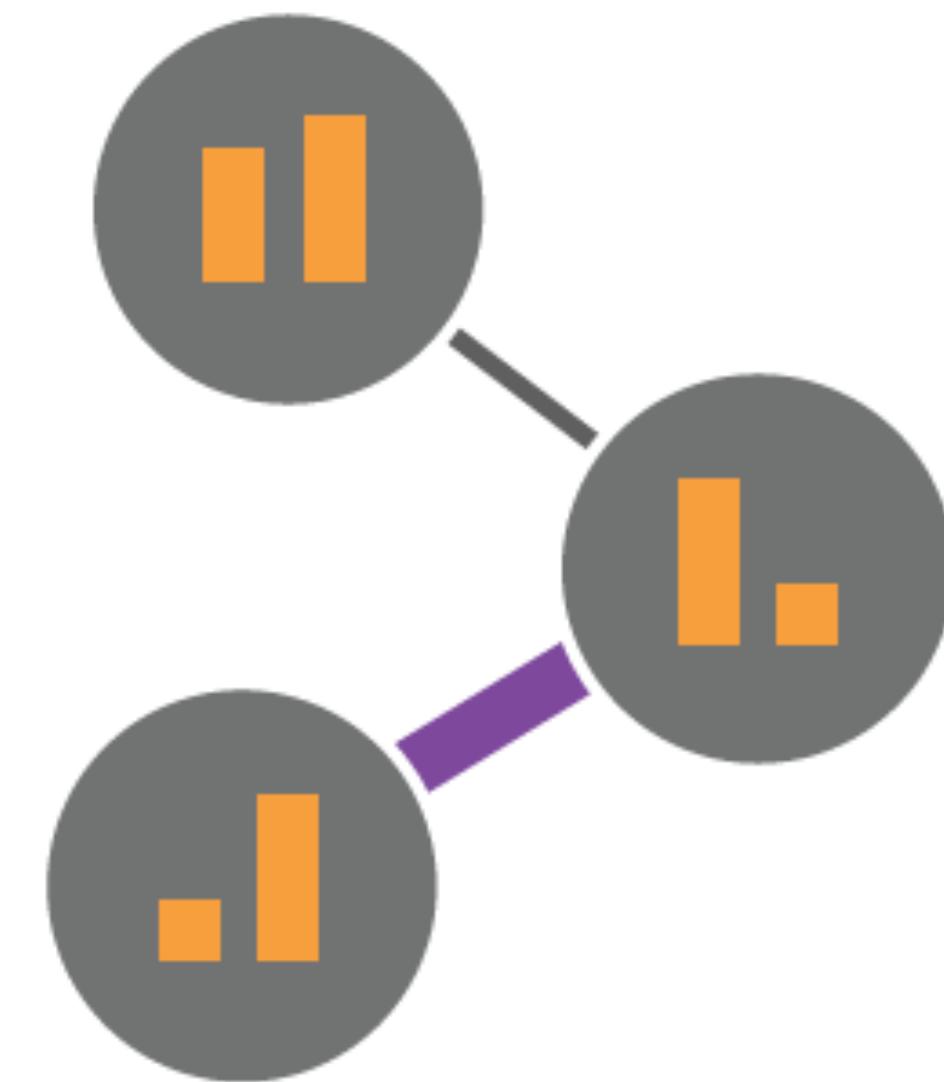


Network / Sub-Network

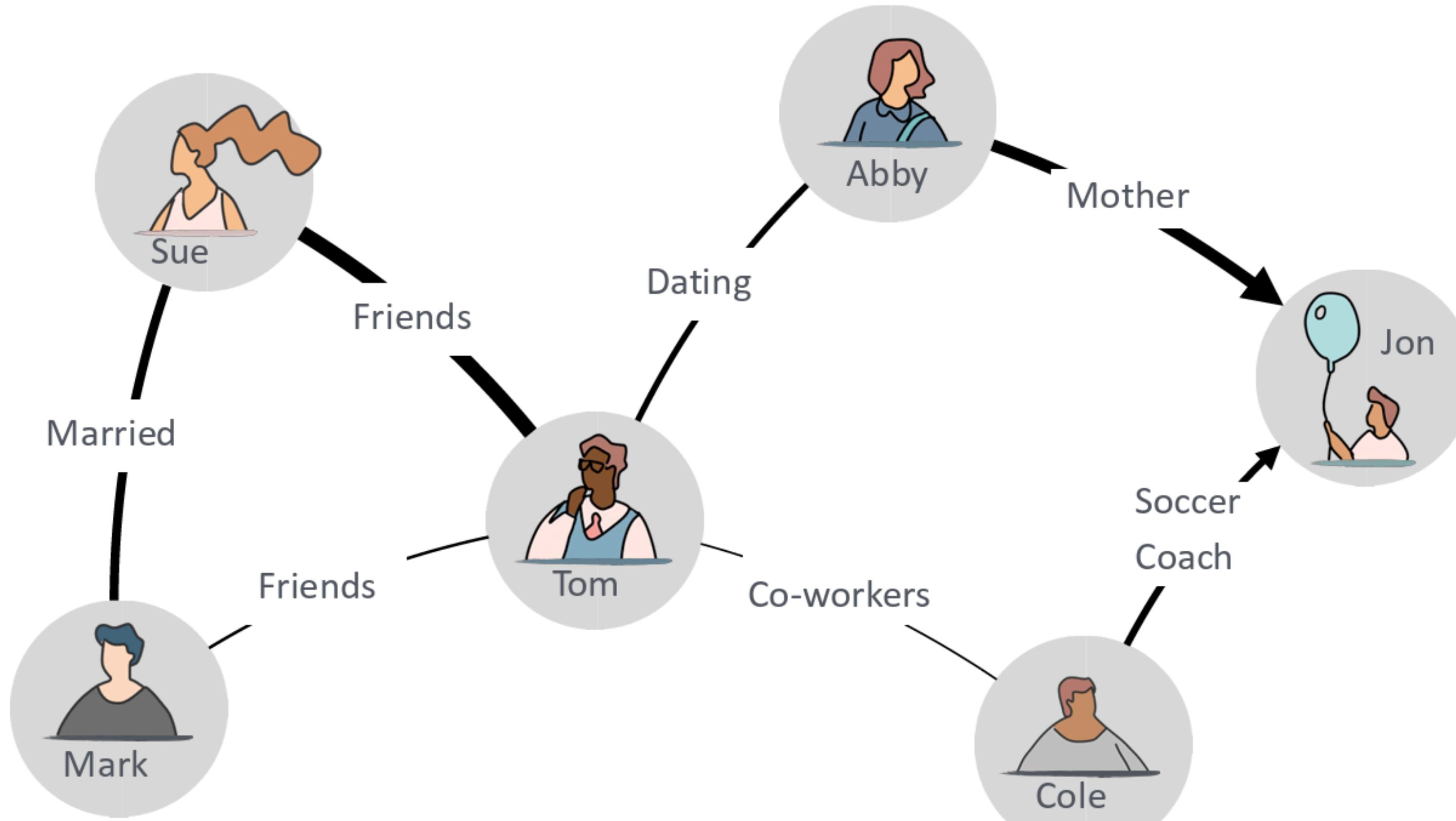
MVN Layouts

MVN Layouts: Node-Link

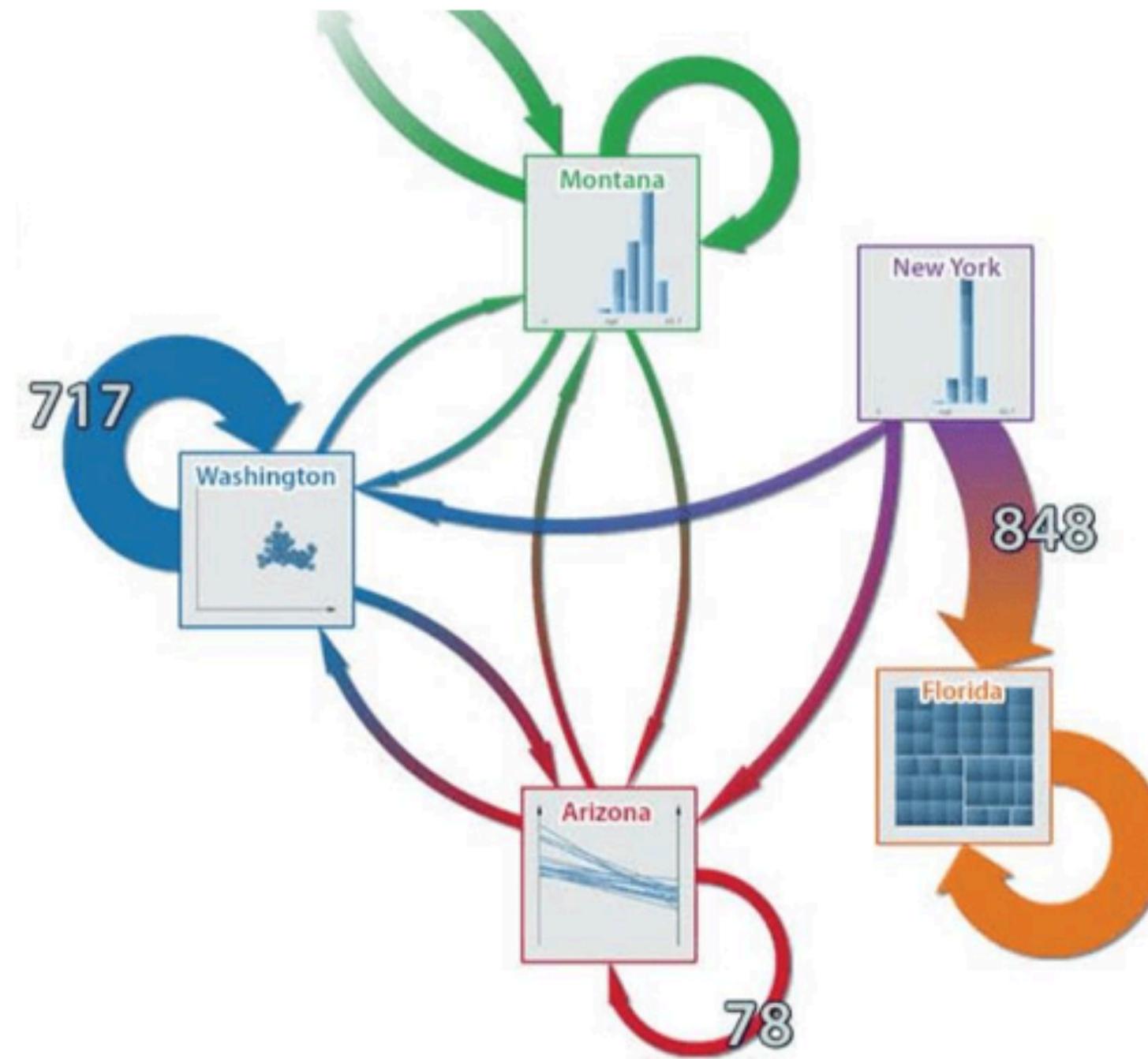
MVN Node-Link Layouts: On-Node / On Edge



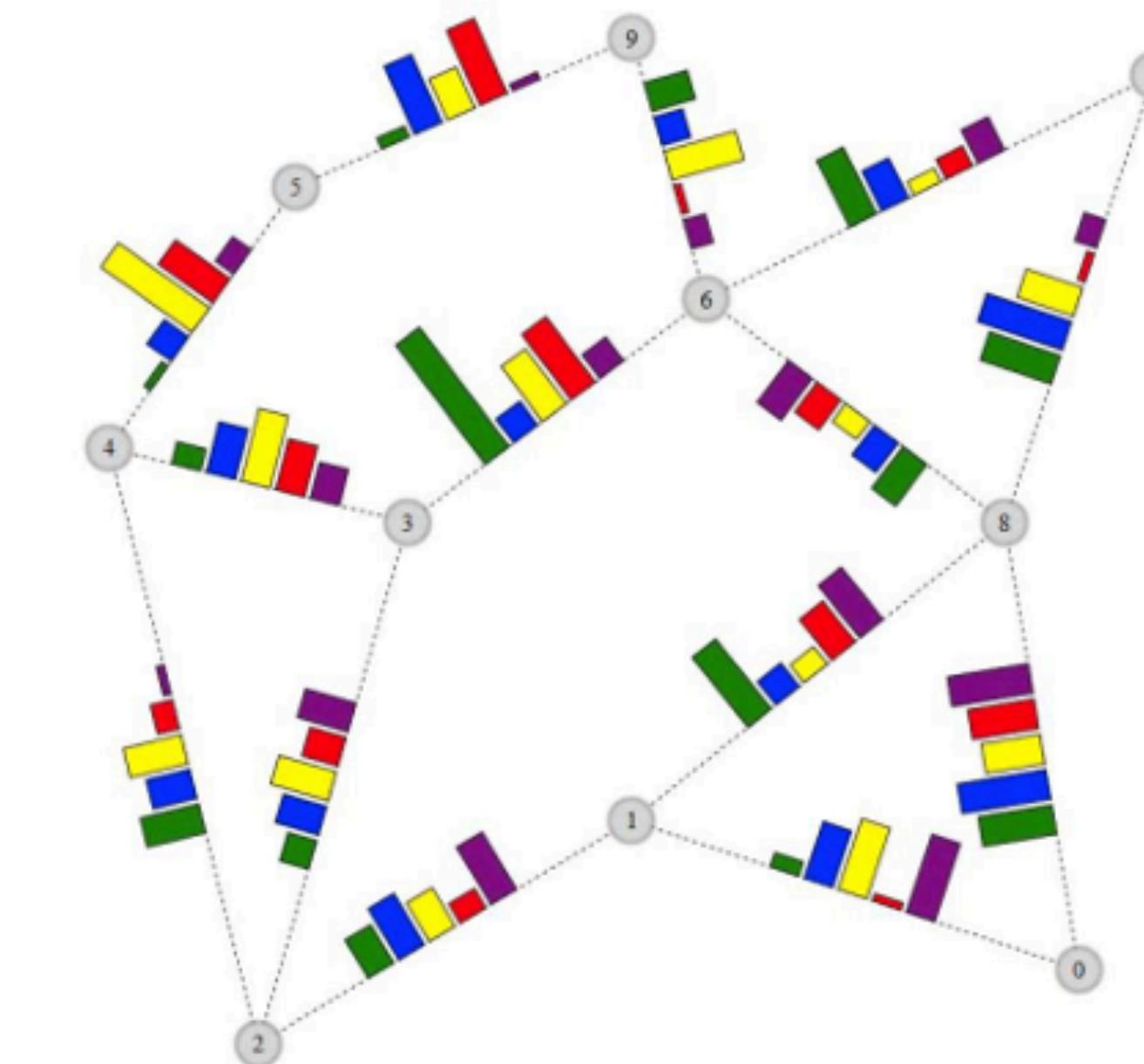
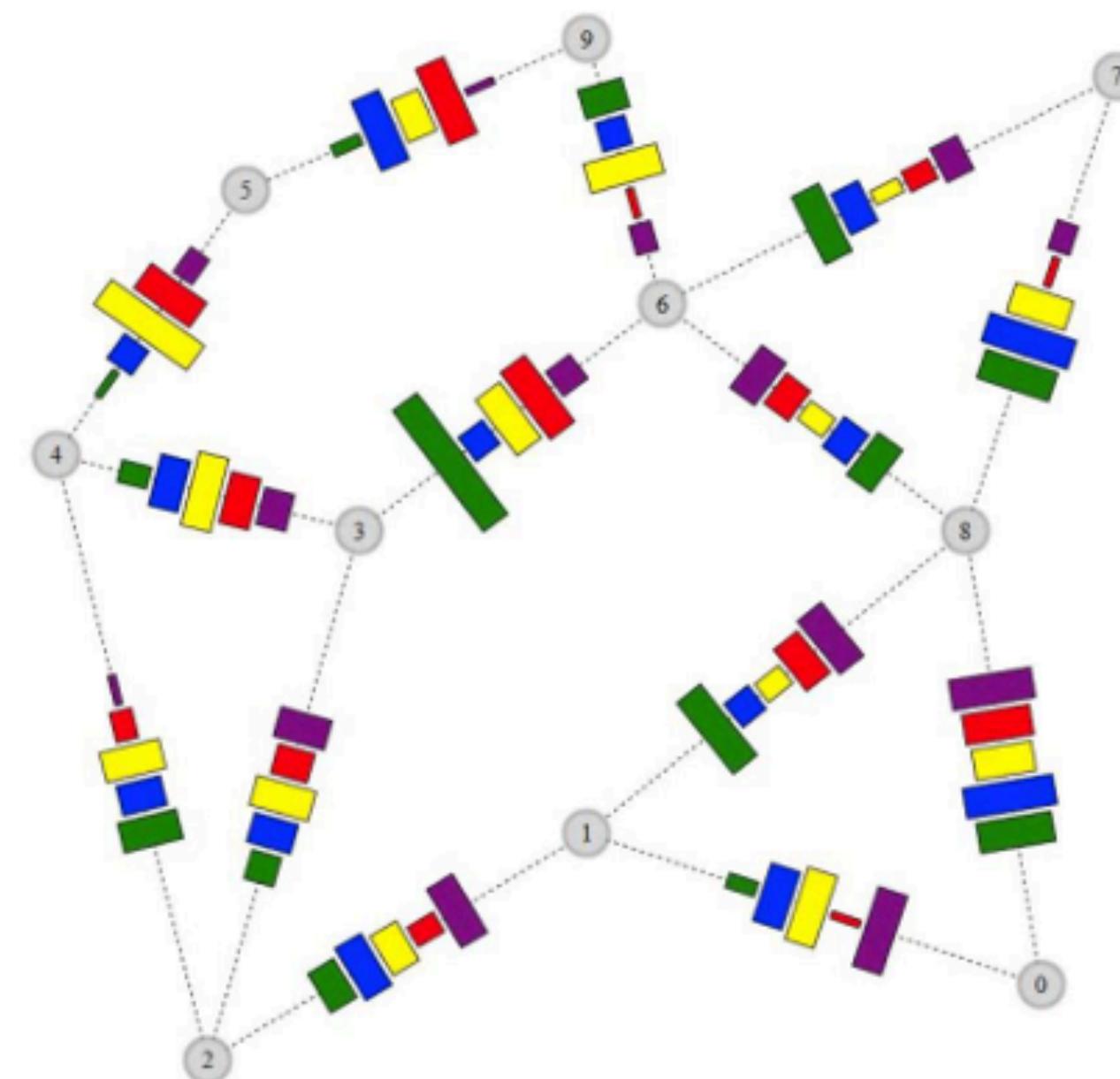
MVN Node-Link Layouts: On-Node / On Edge



MVN Node-Link Layouts: On-Node / On Edge



MVN Node-Link Layouts: On-Node / On Edge



MVN Node-Link Layouts: On-Node / On Edge

Pros:

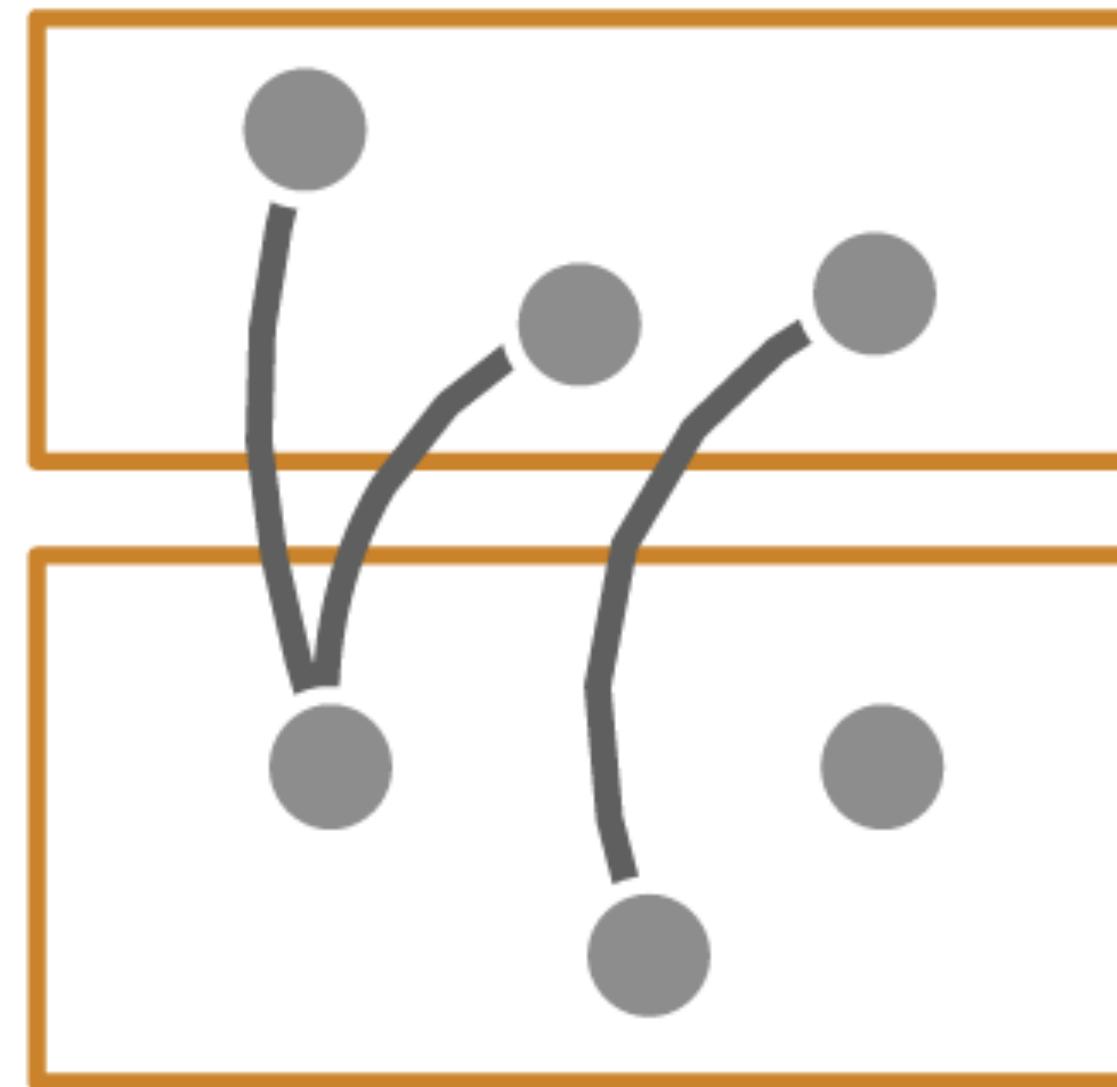
- easily understandable to many users
- works well for all types of networks

Cons:

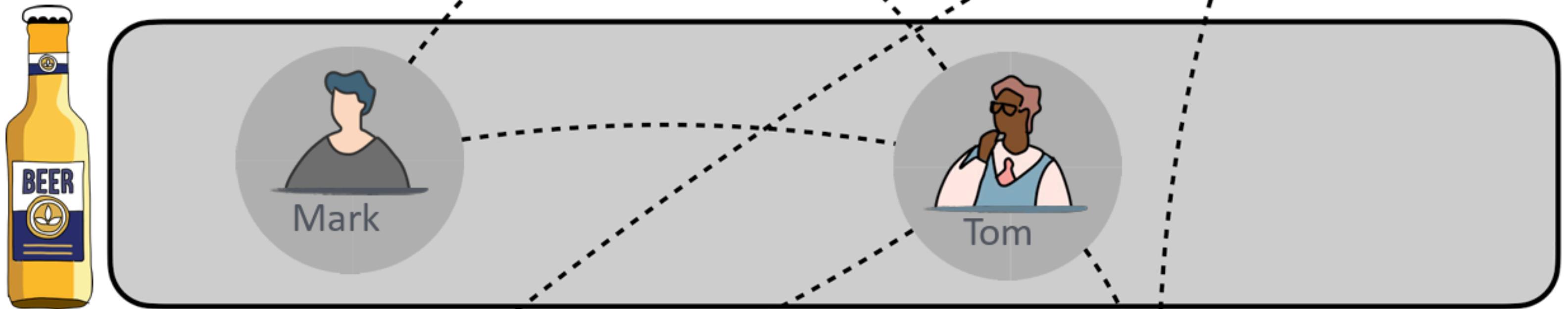
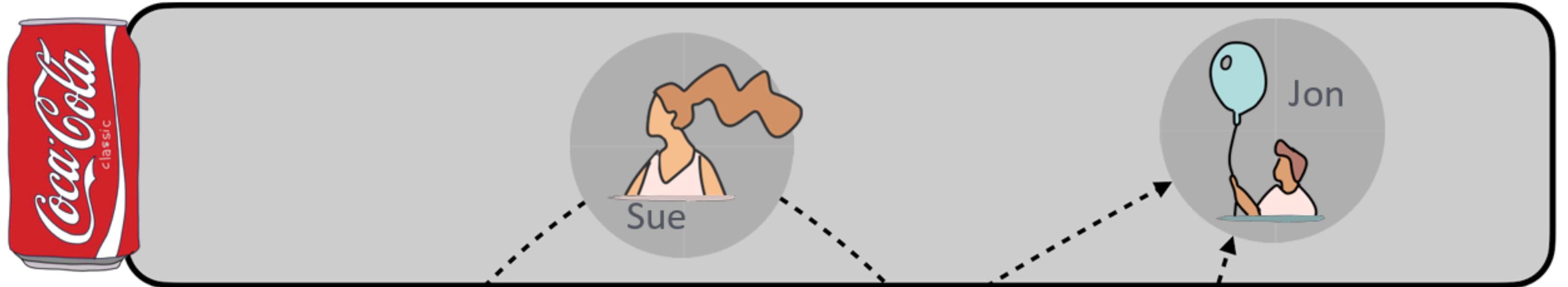
- not very scalable
- node size leaves little space to encode attributes

Recommended for small networks when only a few (<5) attributes are shown (or in combination with zoom+filter)

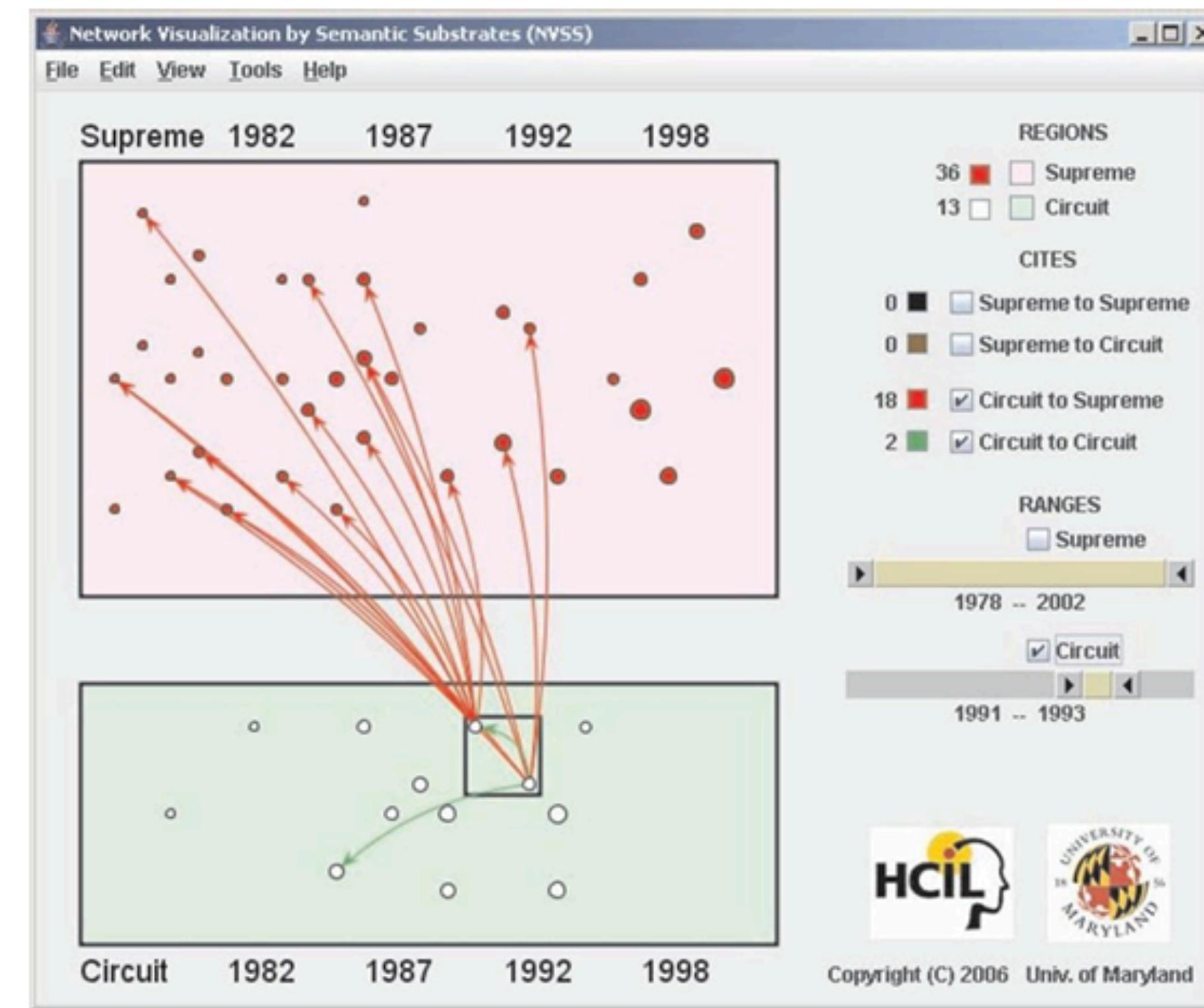
MVN Node-Link Layouts: Attribute Faceting



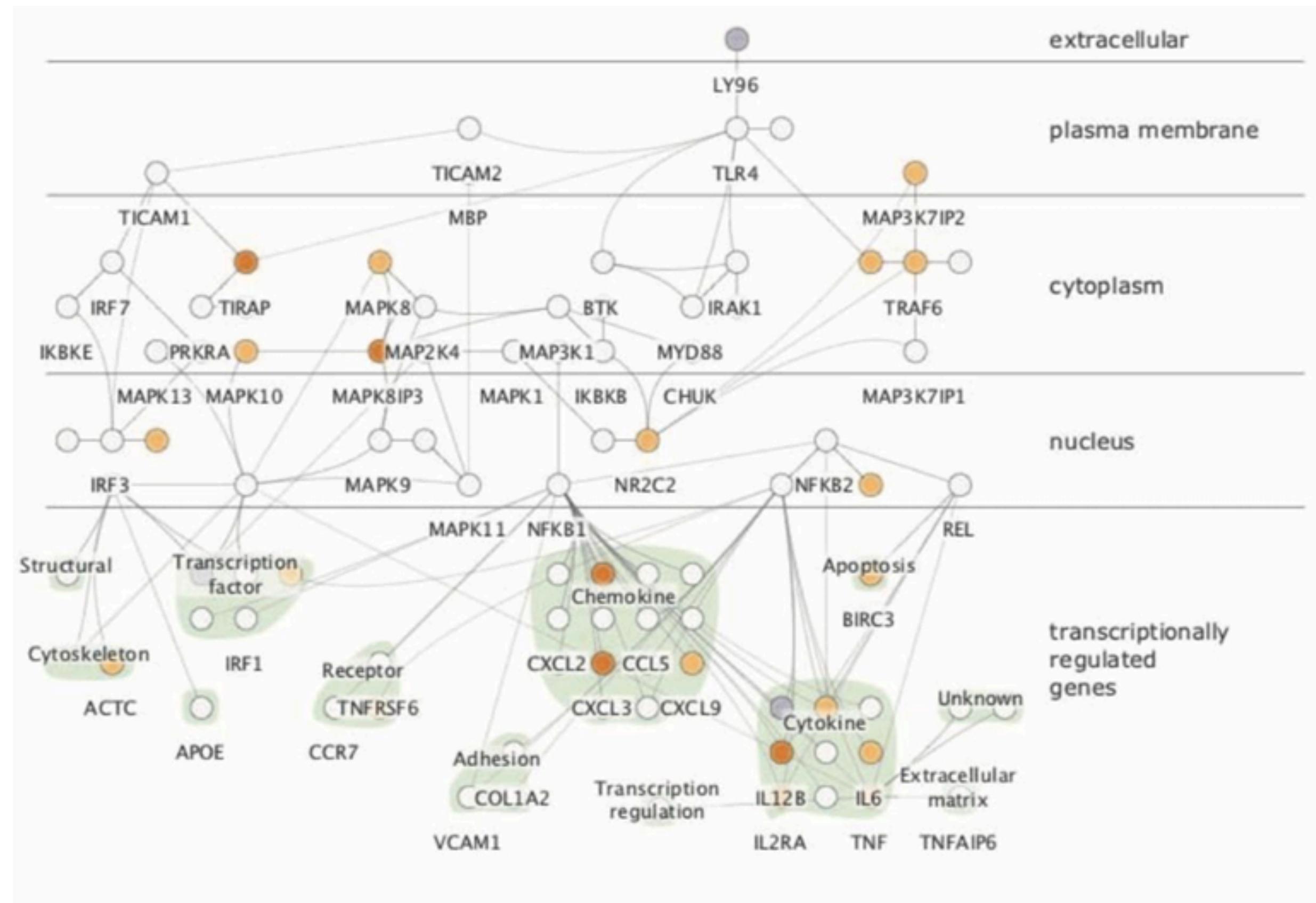
MVN Node-Link Layouts: Attribute Faceting



MVN Node-Link Layouts: Attribute Faceting



MVN Node-Link Layouts: Attribute Faceting



MVN Node-Link Layouts: Attribute Faceting

Pros:

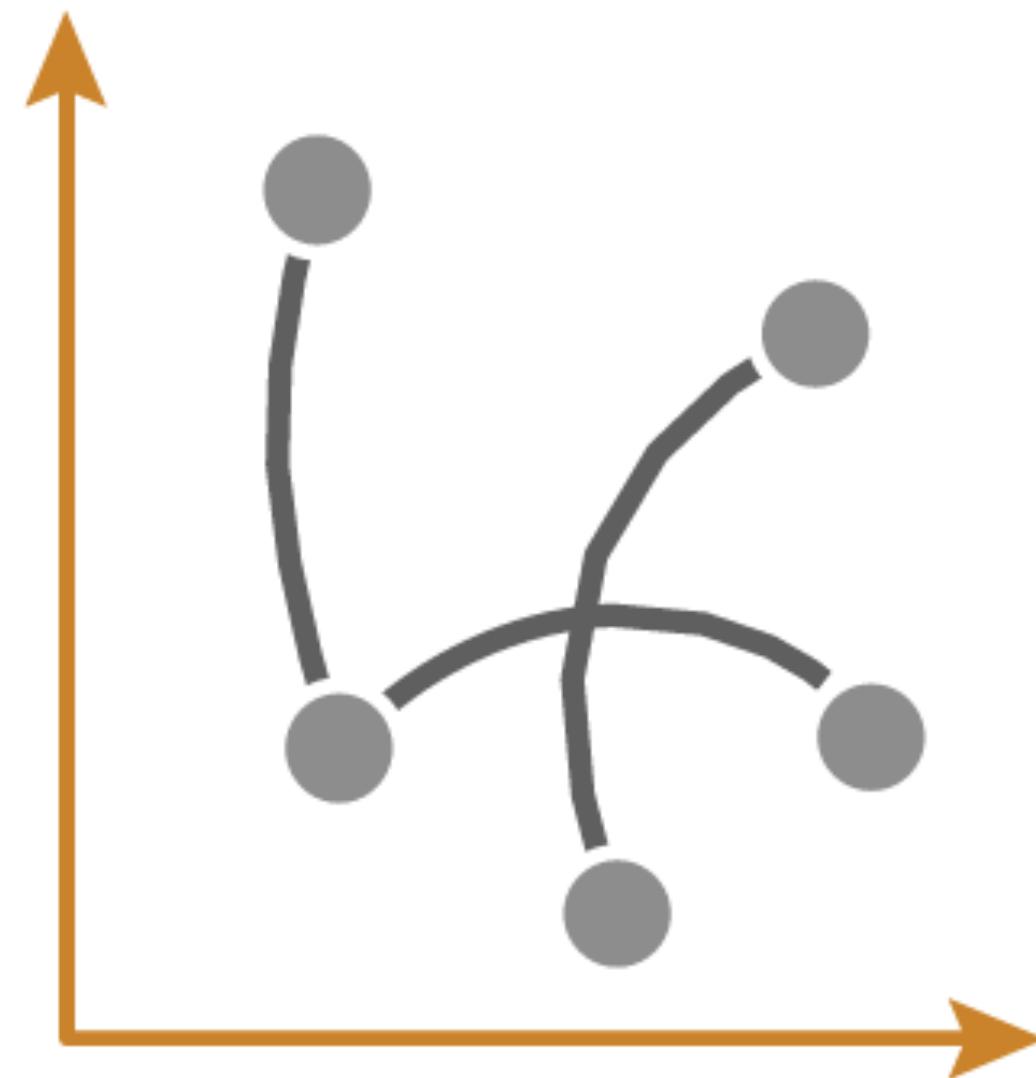
- works well for all types of networks
- good for networks with different node types or with categorical attributes

Cons:

- less scalable
- topological features are not easily visible when they span different facets

Recommended for networks where nodes can be easily separated into groups (and these groups are central to analysis)

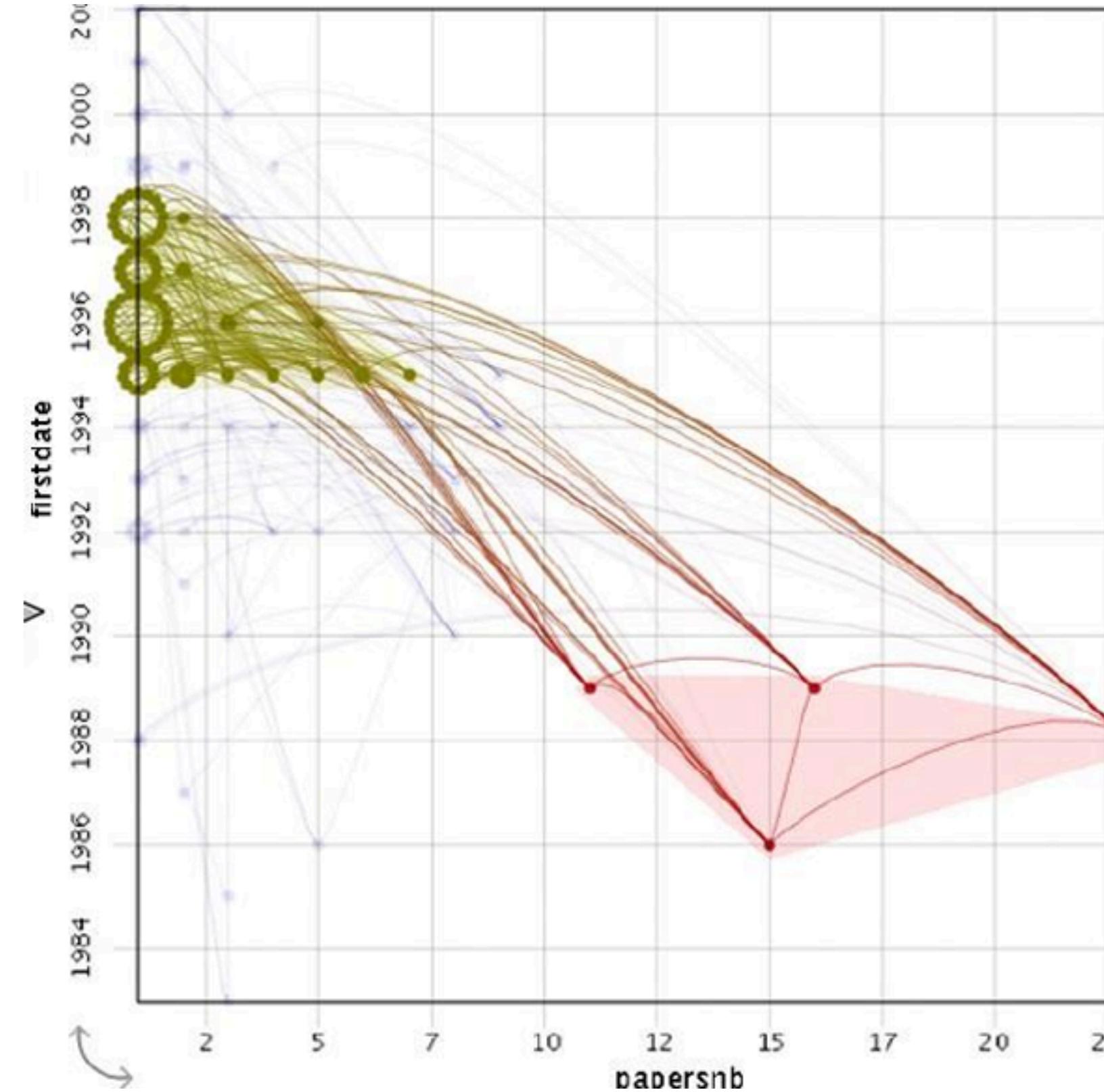
MVN Node-Link Layouts: Attribute Positioning



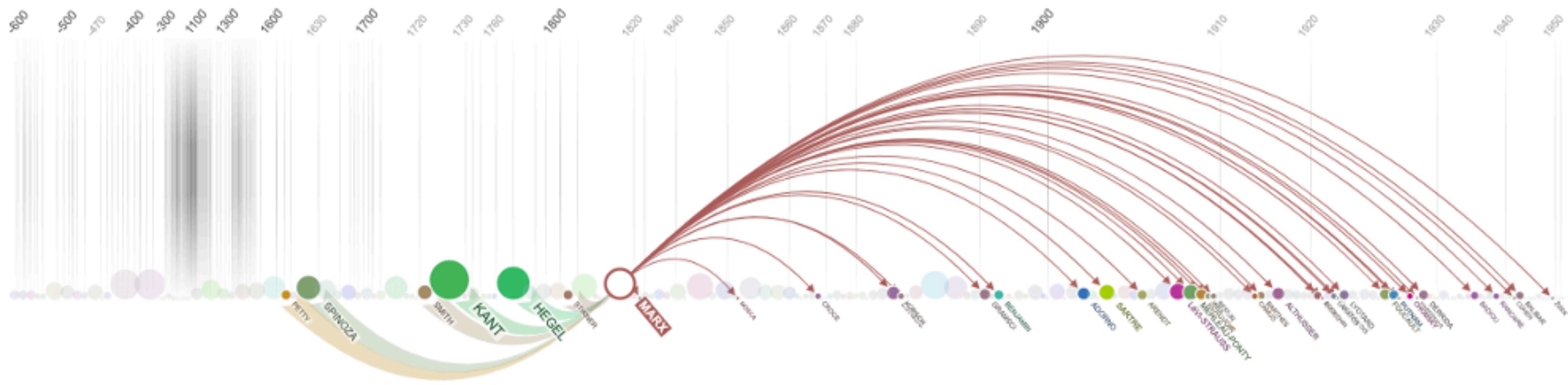
MVN Node-Link Layouts: Attribute Positioning



MVN Node-Link Layouts: Attribute Positioning



MVN Node-Link Layouts: Attribute Positioning



MVN Node-Link Layouts: Attribute Positioning

Pros:

- well suited for quantitative attributes

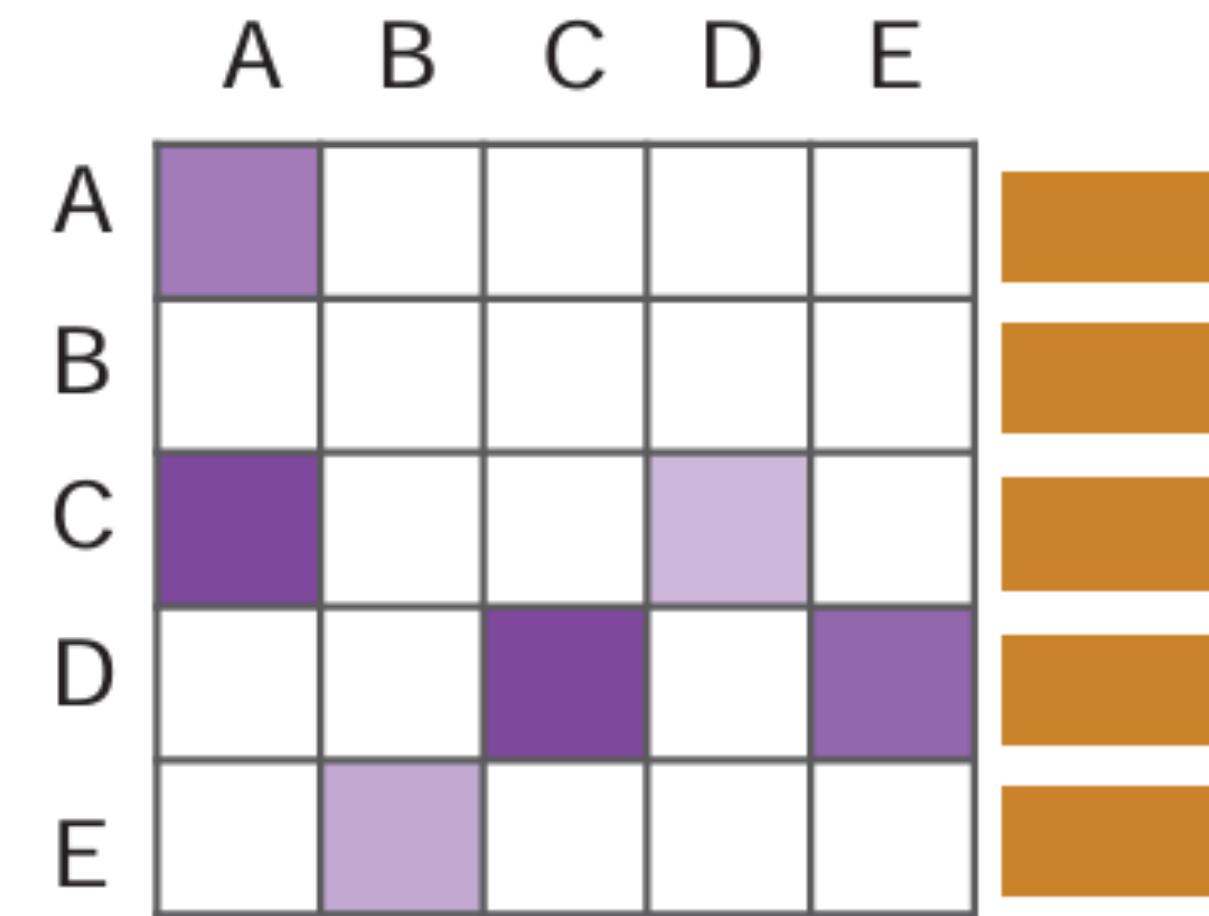
Cons:

- does not lend itself well to visualizing the topology of the network

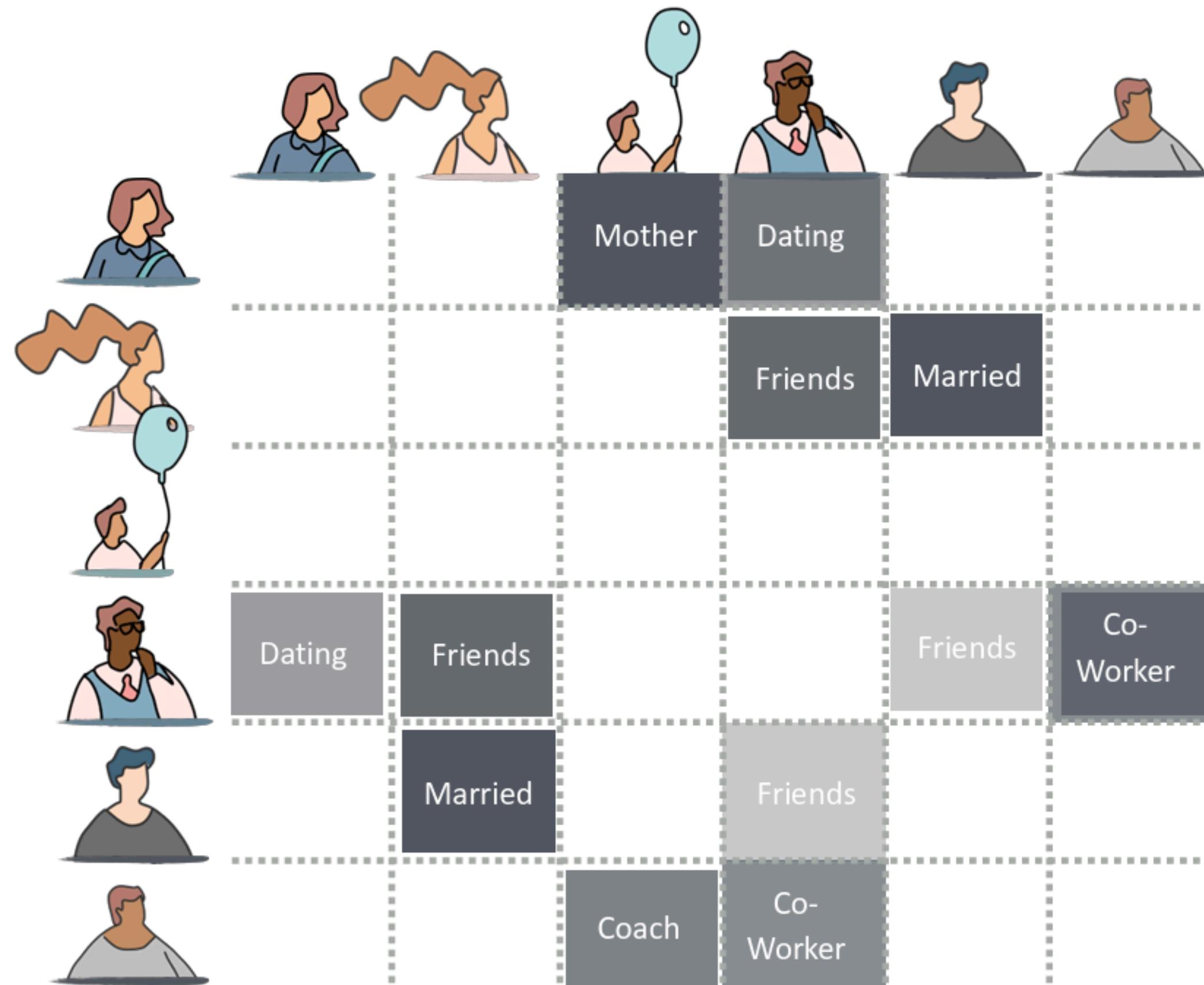
Recommended for smaller, sparse networks where relationships between node attributes are central to the analysis, and topology only provides context

MVN Layouts: Tabular

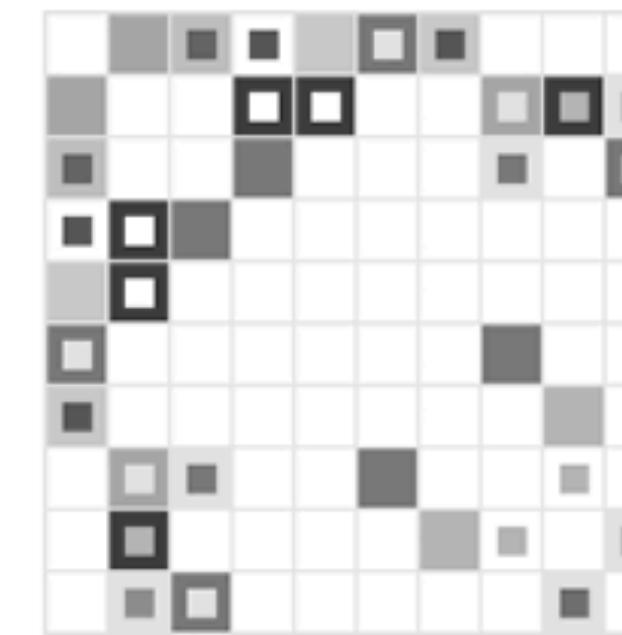
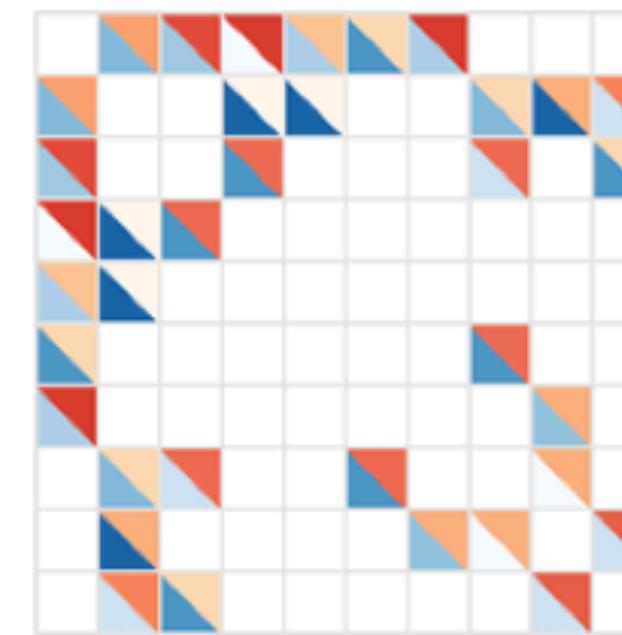
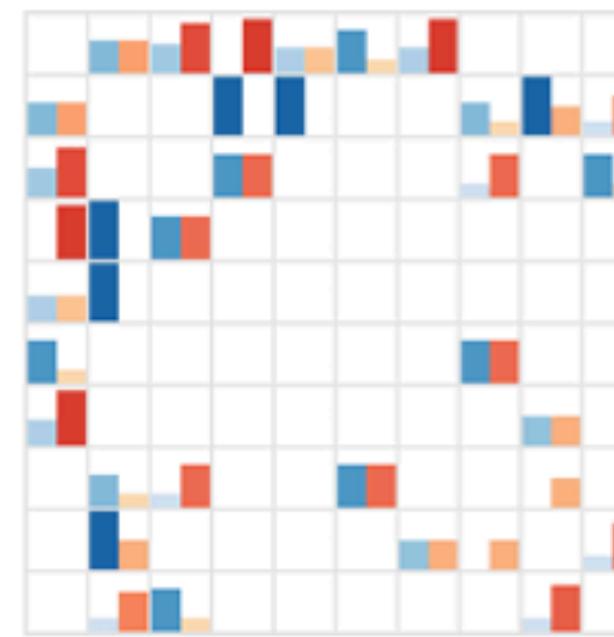
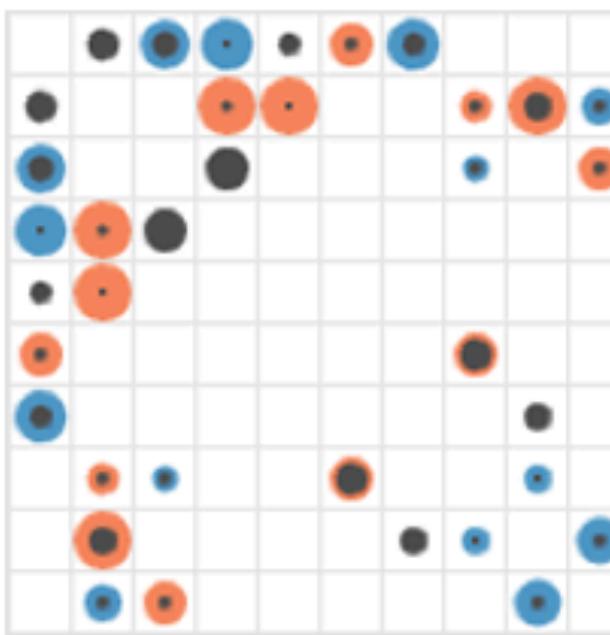
MVN Tabular Layouts: Adjacency Matrix



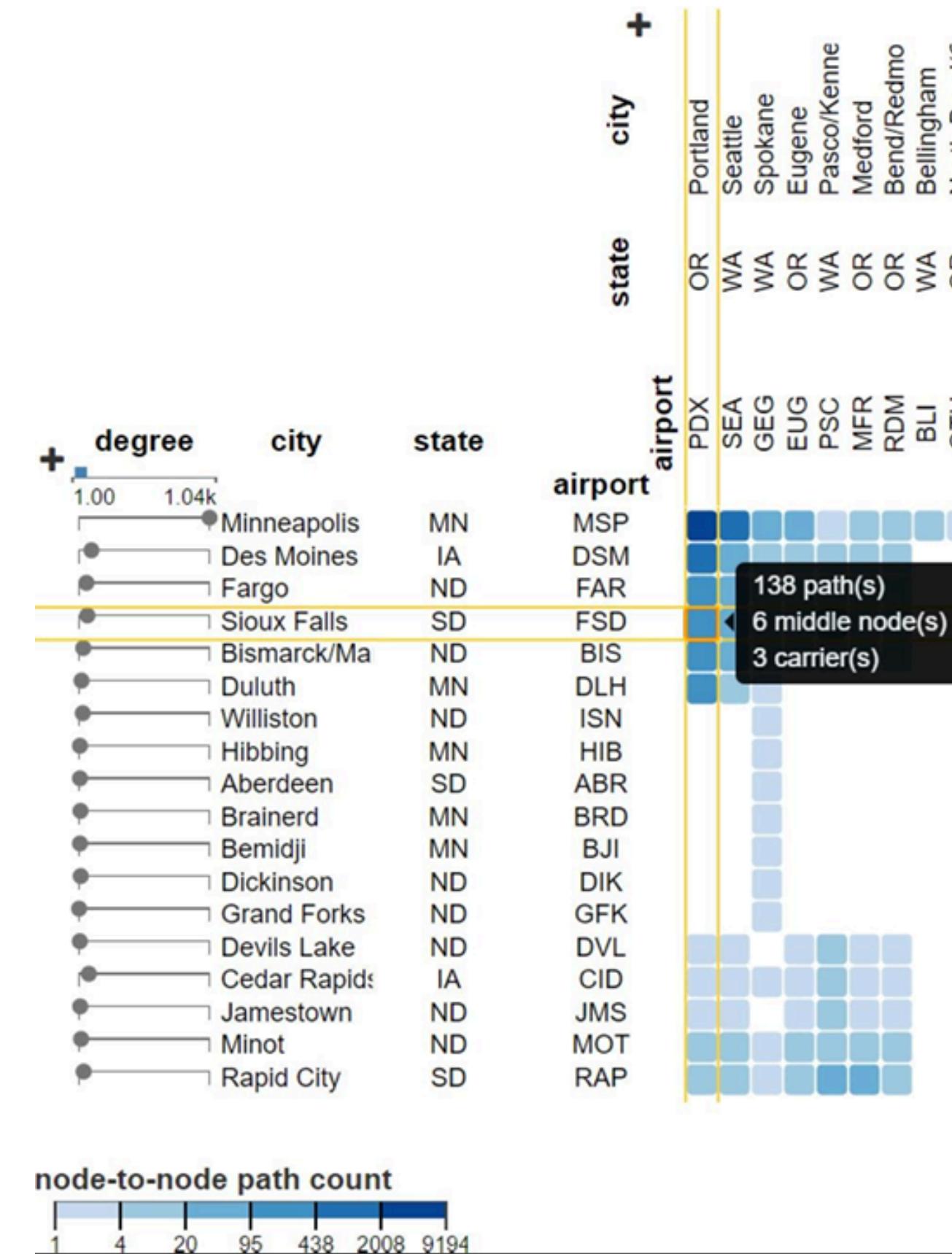
MVN Tabular Layouts: Adjacency Matrix



MVN Tabular Layouts: Adjacency Matrix



MVN Tabular Layouts: Adjacency Matrix



MVN Tabular Layouts: Adjacency Matrix

Pros:

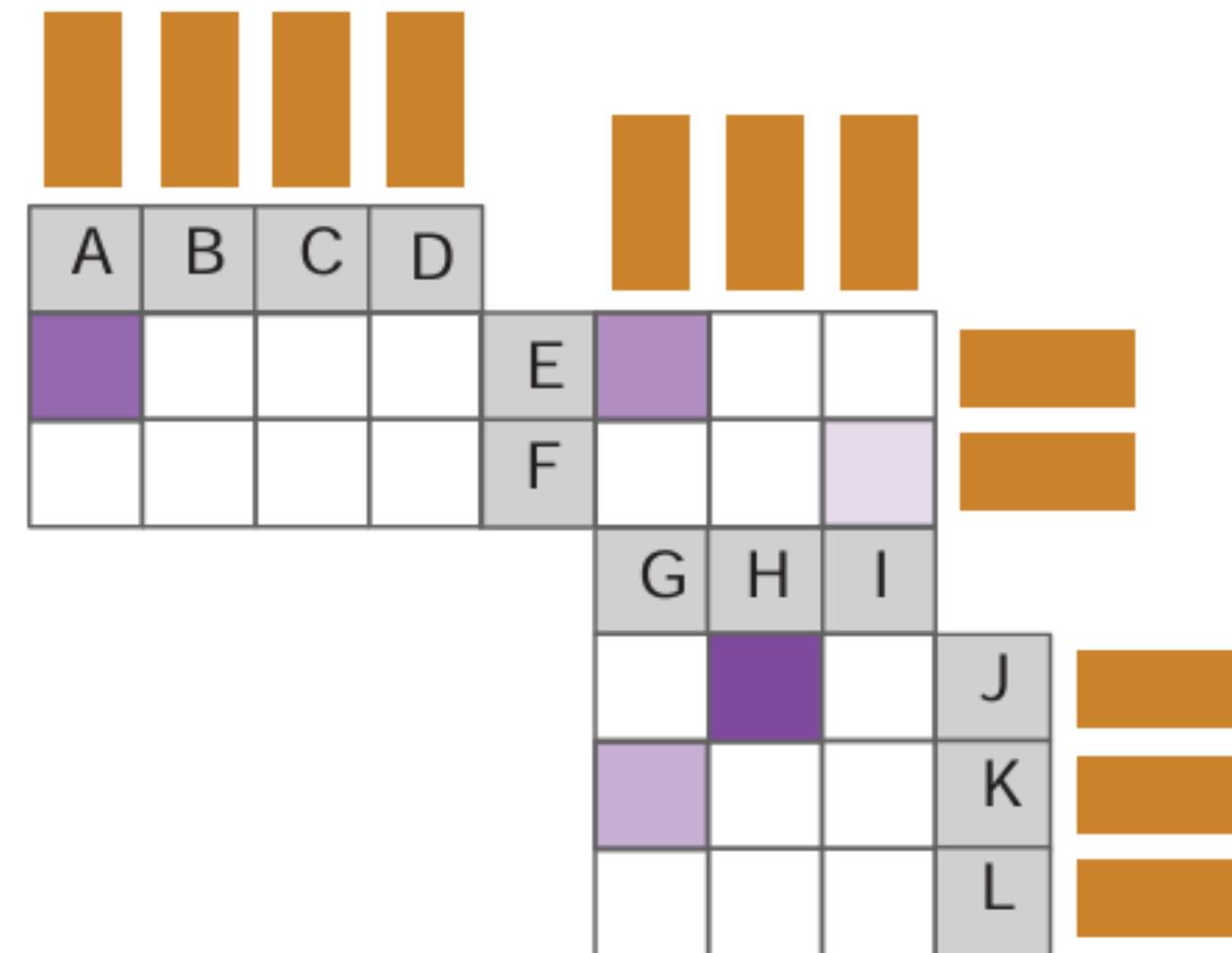
- ideal for dense and completely connected networks

Cons:

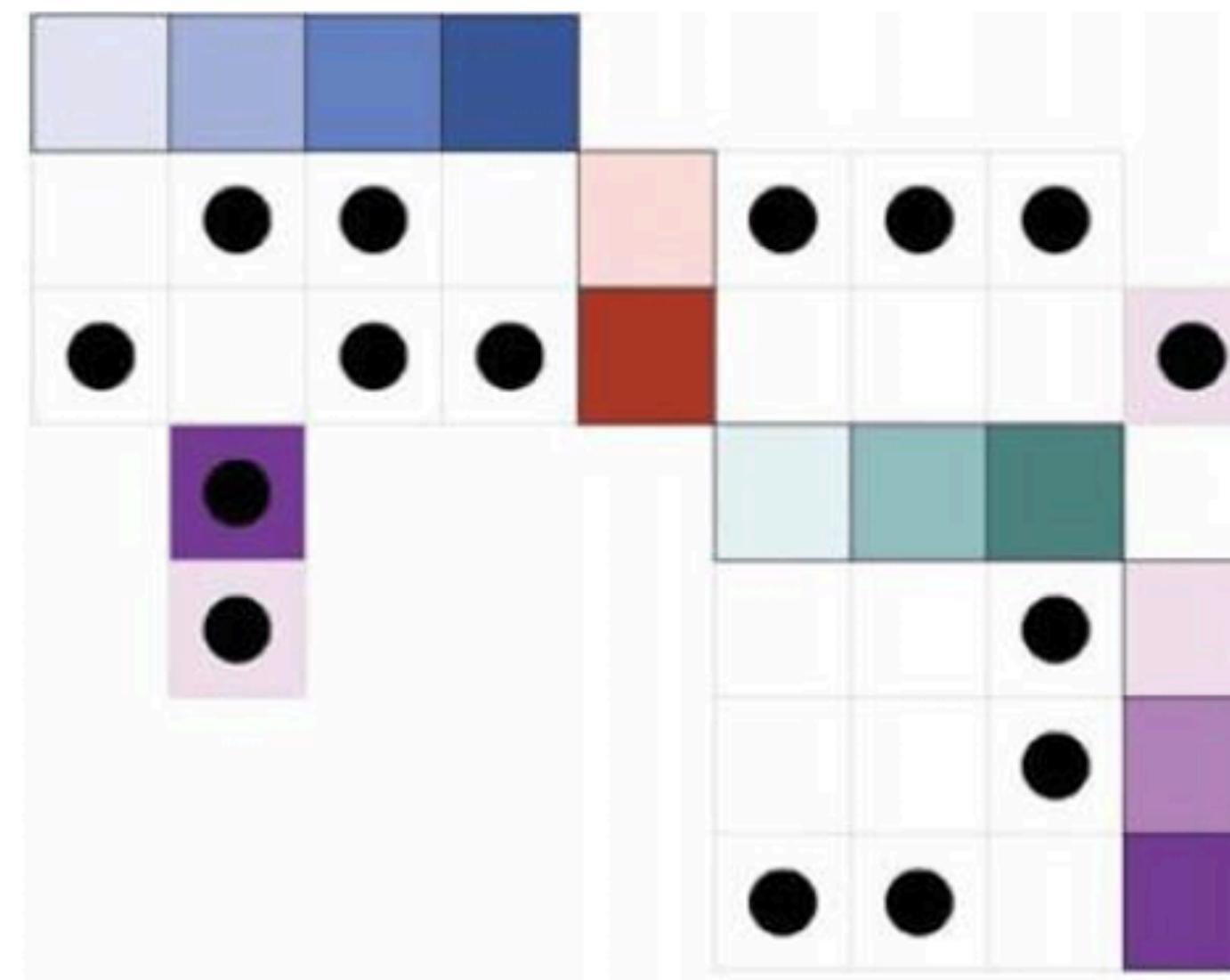
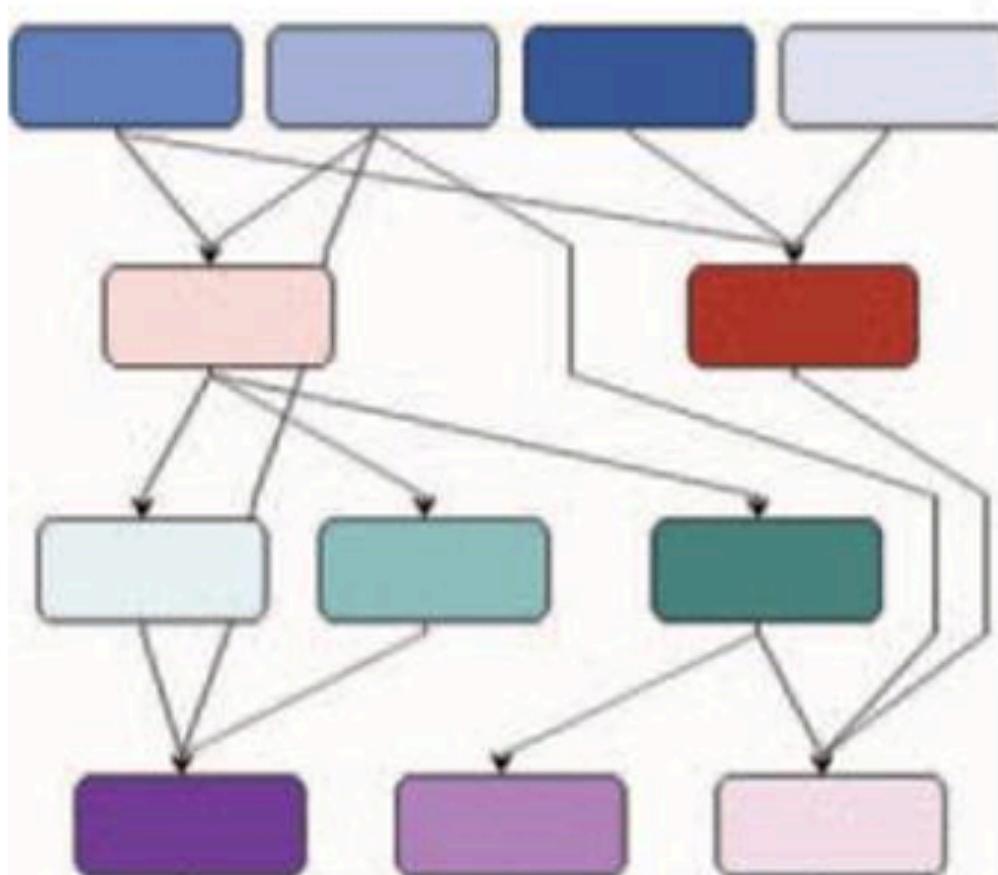
- requires quadratic space with respect to the number of nodes
- ordering-dependent

Recommended for smaller, complex and dense networks with rich node and/or edge attributes (except when the task depends on paths)

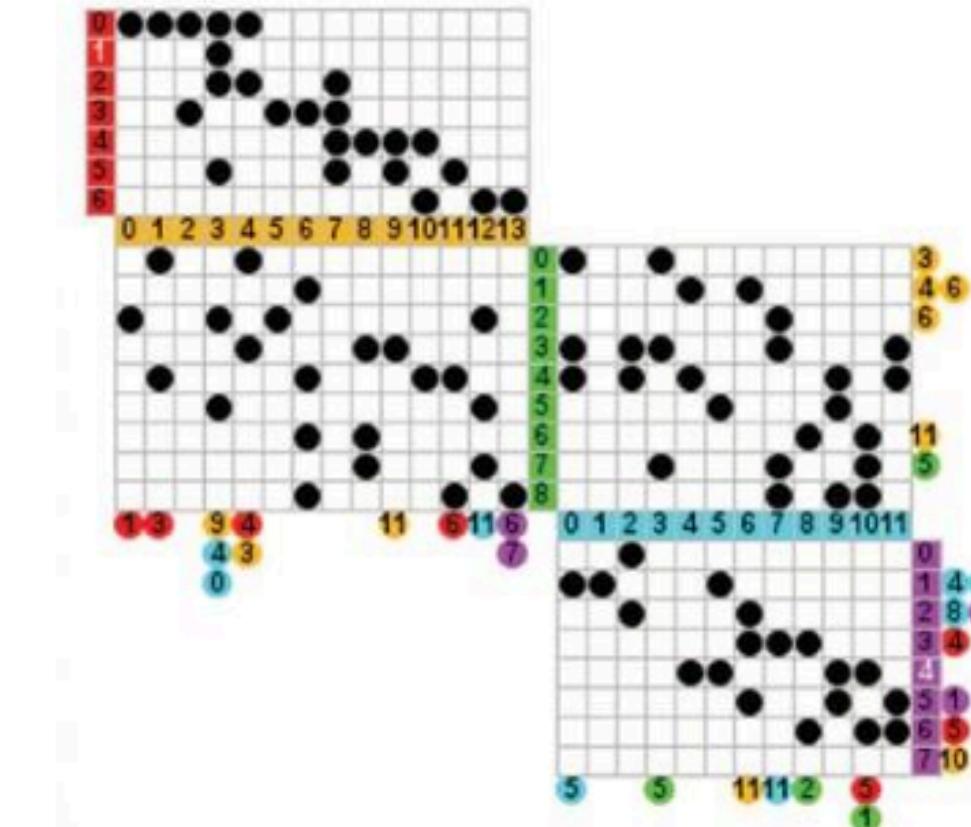
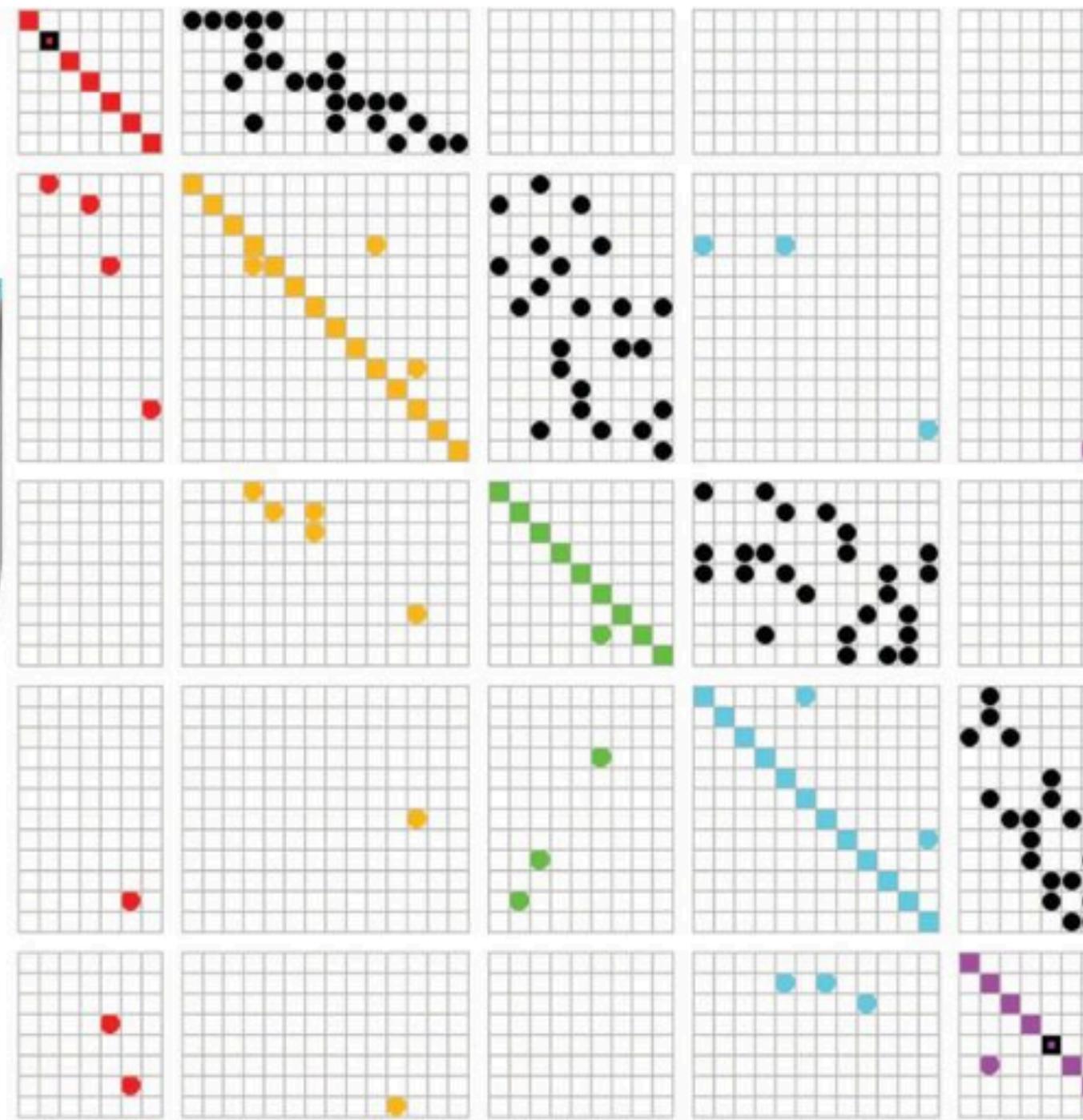
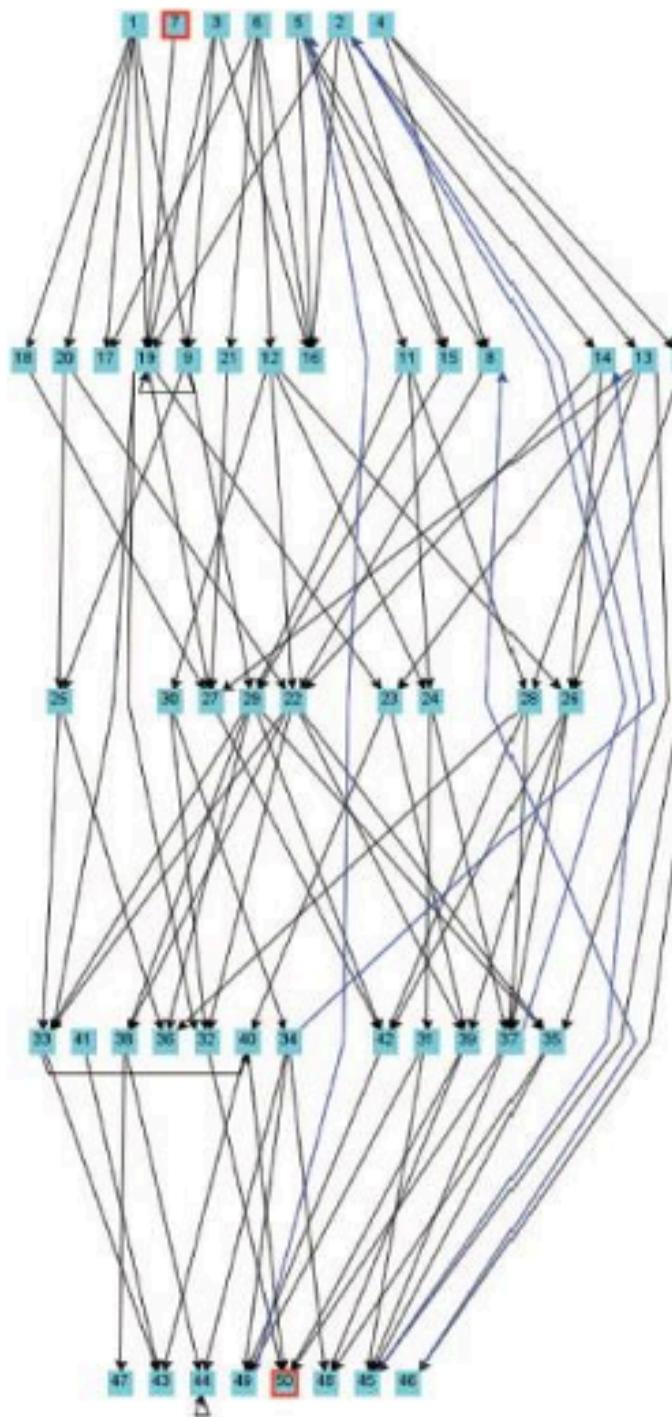
MVN Tabular Layouts: Quilts



MVN Tabular Layouts: Quilts



MVN Tabular Layouts: Quilts



MVN Tabular Layouts: Quilts

Pros:

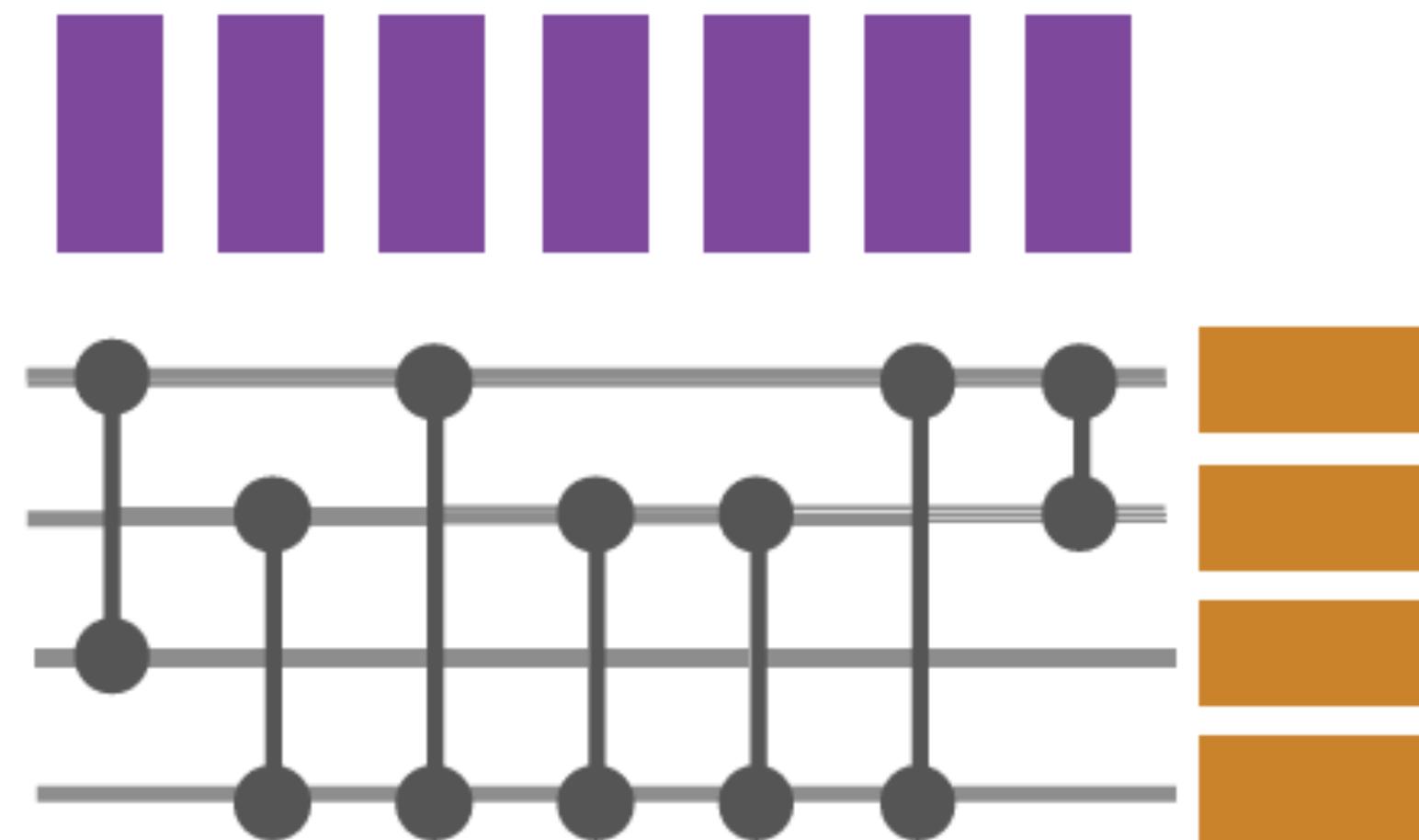
- well suited for layered networks

Cons:

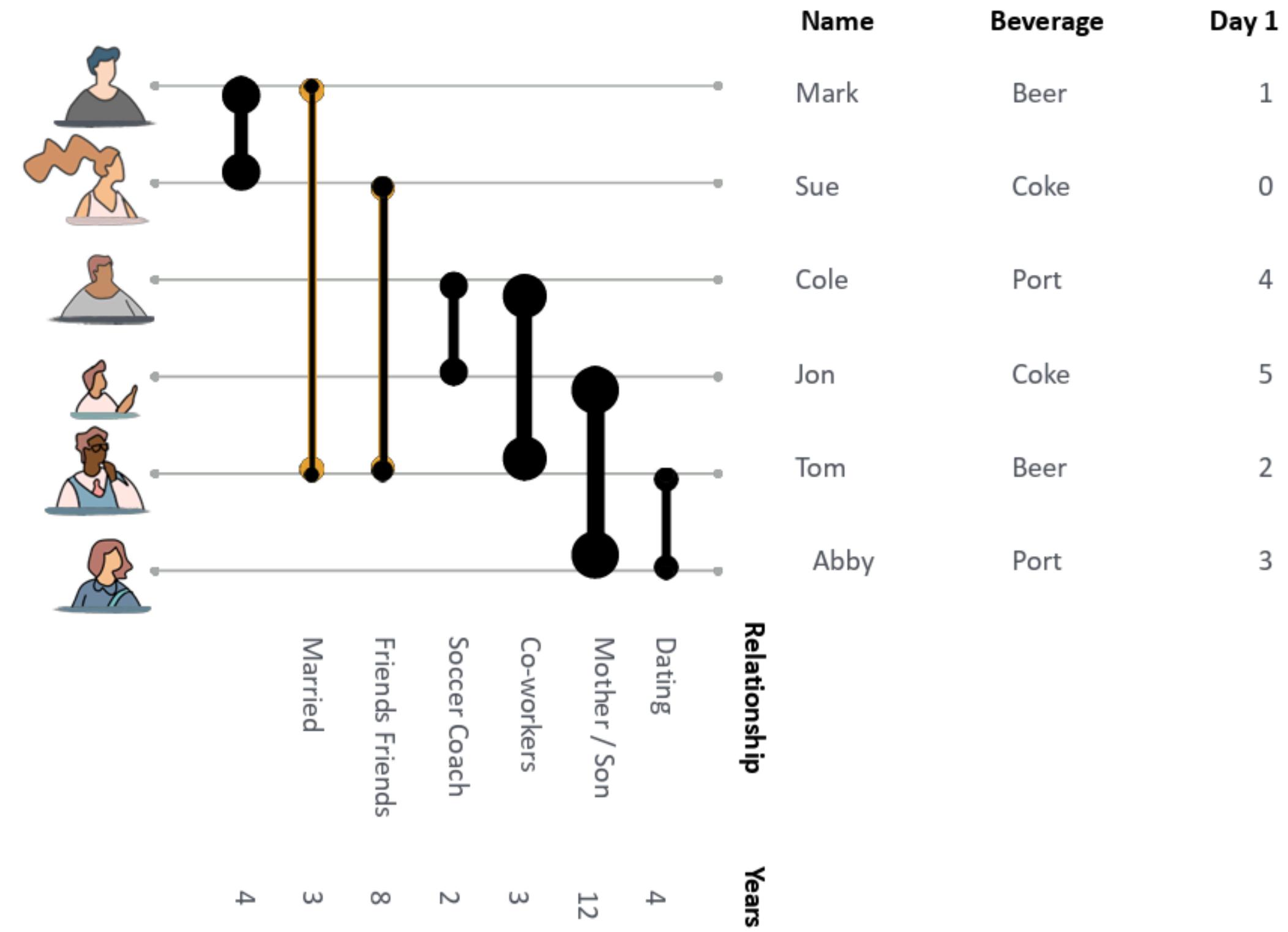
- links between non-consecutive layers can be problematic to visualize and non-intuitive

Recommended for k-partite networks with limited skip links

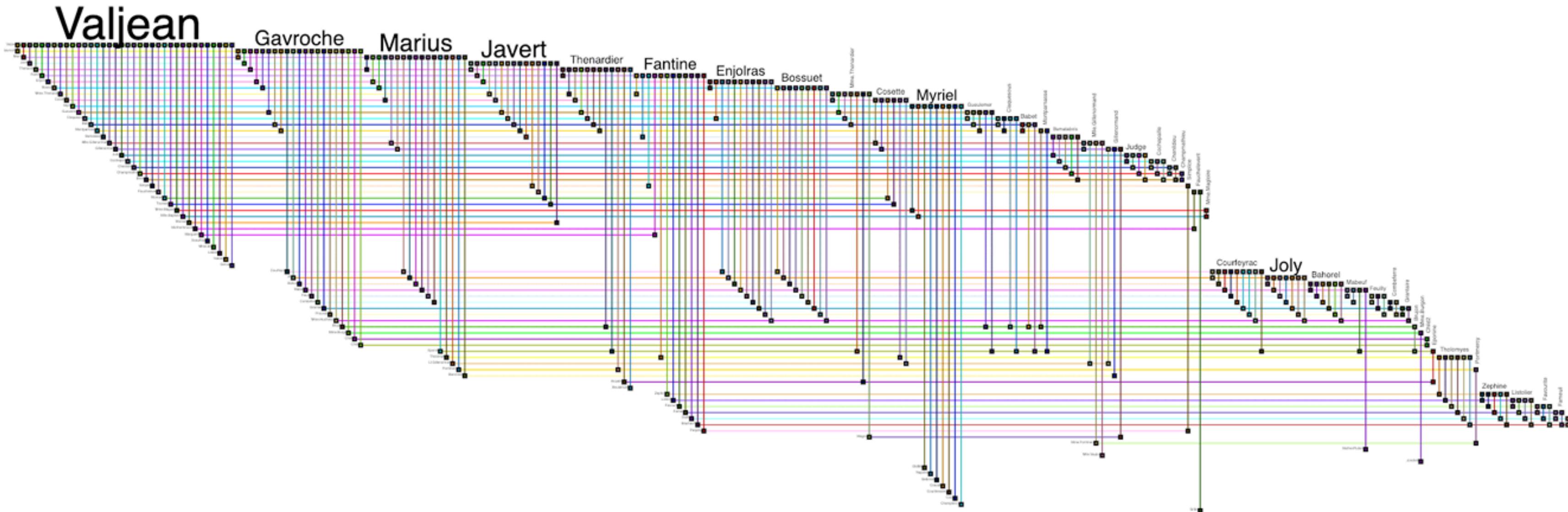
MVN Tabular Layouts: BioFabric



MVN Tabular Layouts: BioFabric



MVN Tabular Layouts: BioFabric



MVN Tabular Layouts: BioFabric

Pros:

- useful for visualizing rich node attributes and edge attributes simultaneously

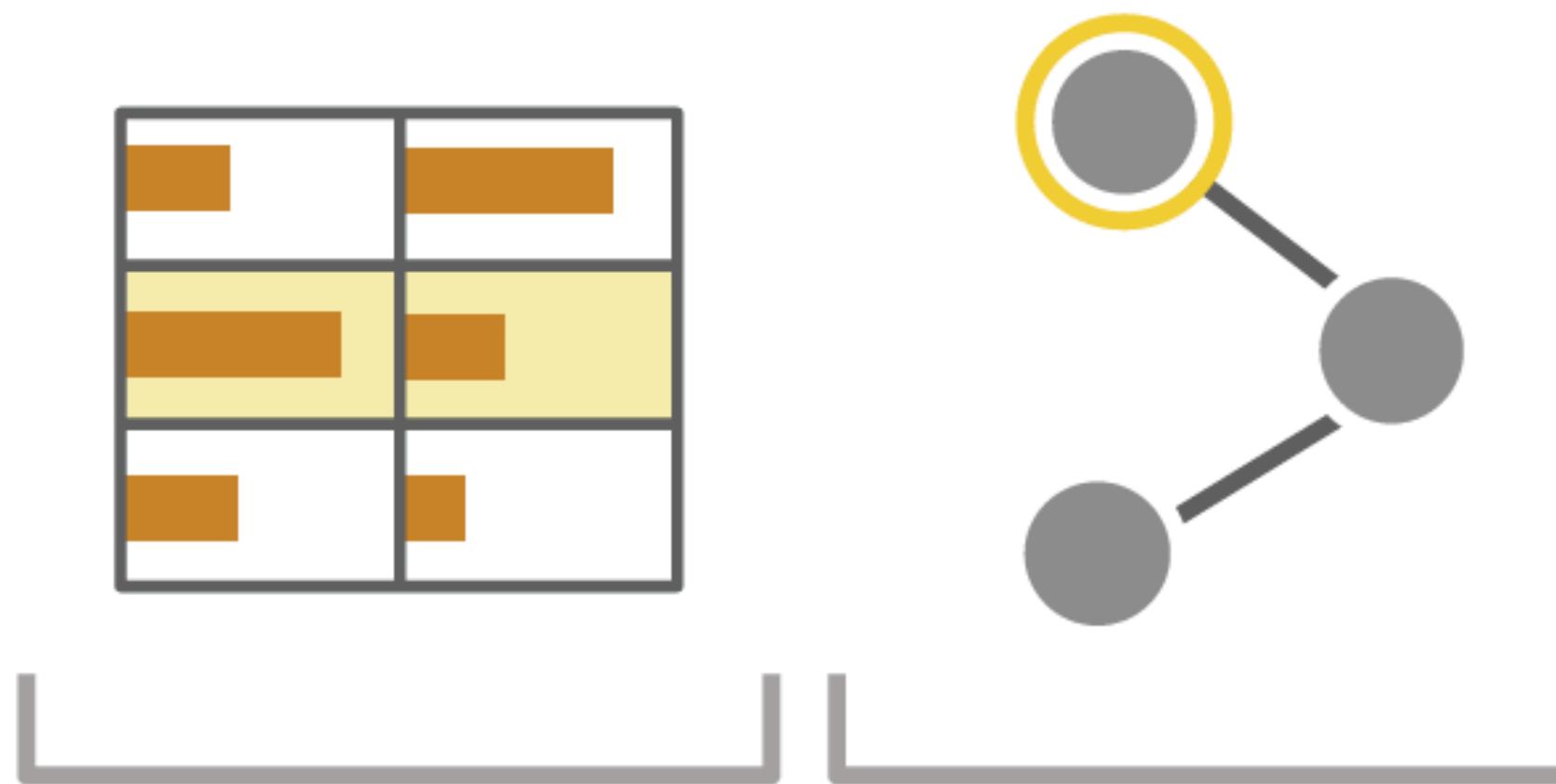
Cons:

- more difficult to discover neighbors

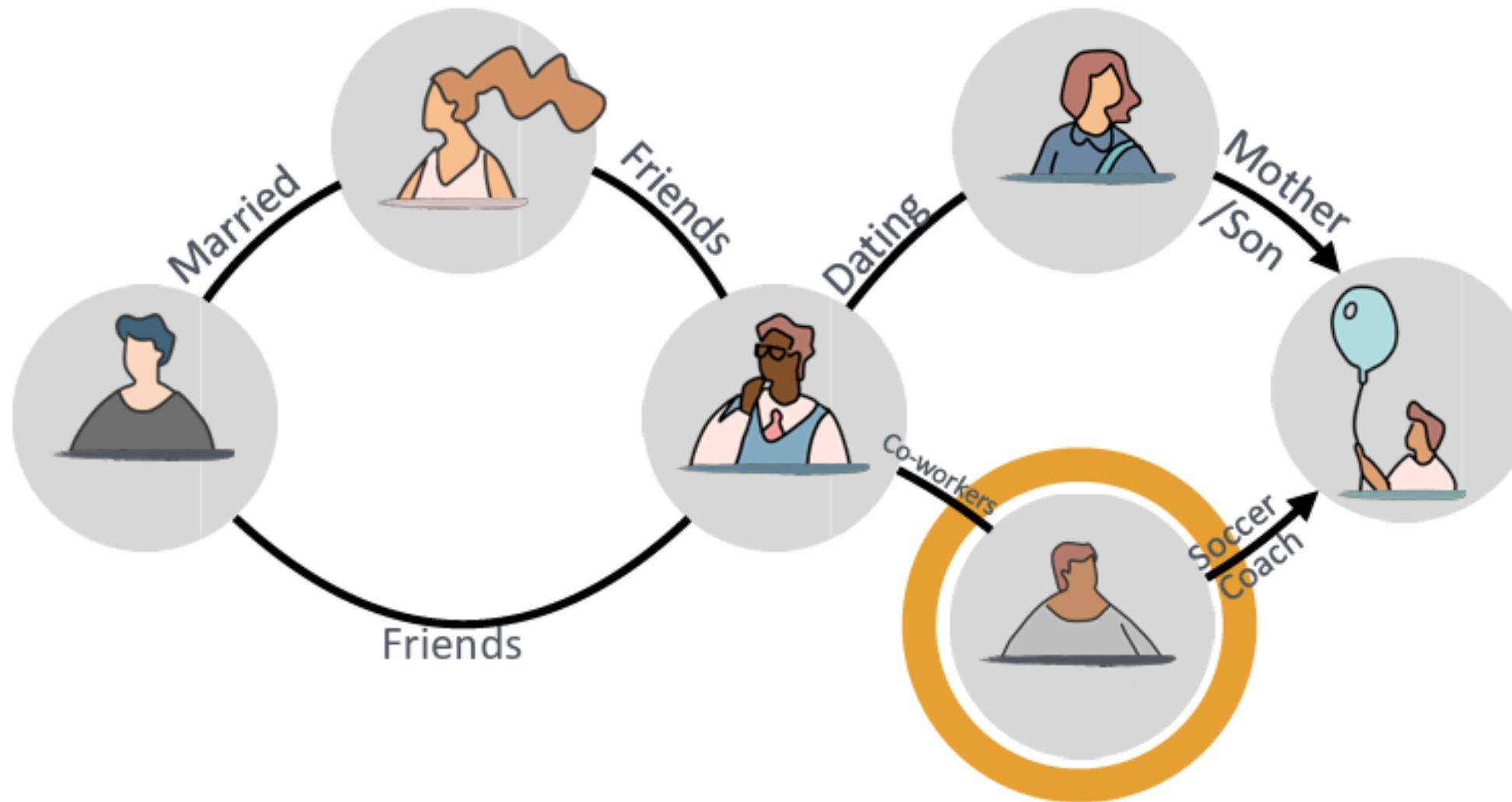
Recommended for smaller, sparse networks with rich node and edge attributes

MVN Operations

MVN Operations: Juxtaposed Views



MVN Operations: Juxtaposed Views



Name	Beverage	Day 1
Mark	Beer	1
Sue	Coke	0
Cole	Port	4
Jon	Coke	5
Tom	Beer	2
Abby	Port	3

MVN Operations: Juxtaposed Views

Pros:

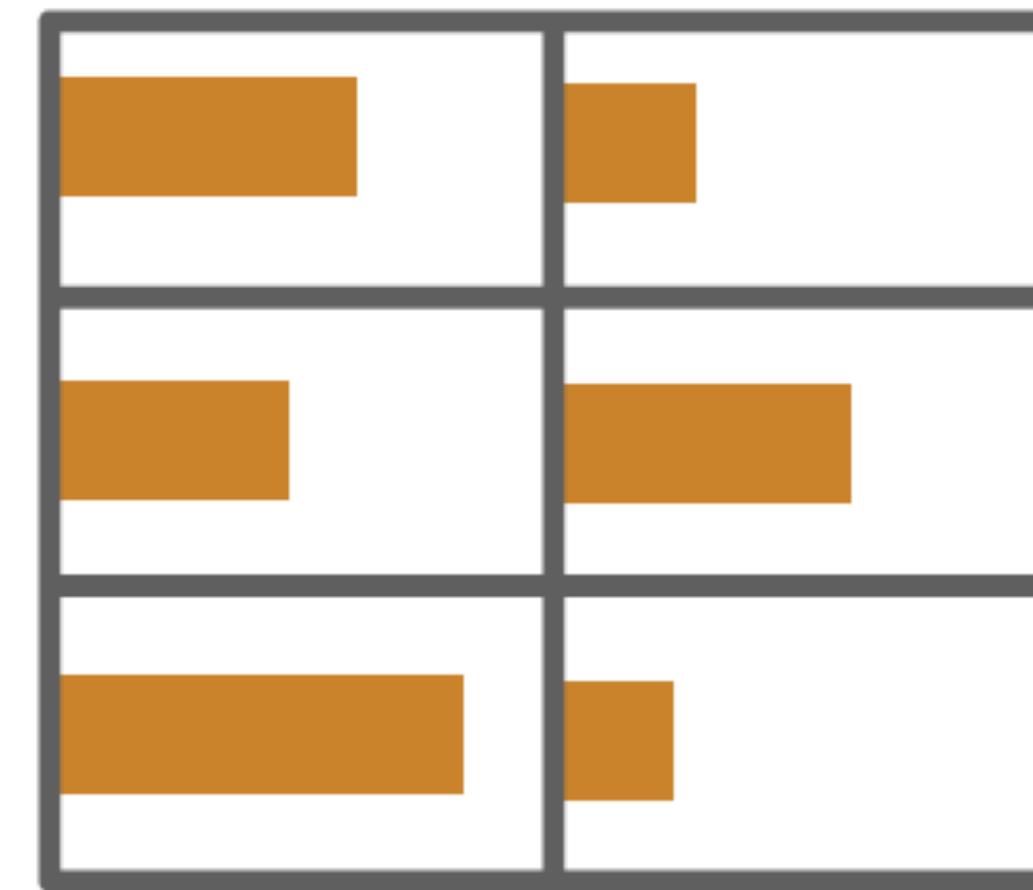
- separate views can optimize for topology and attributes independently

Cons:

- not great for tasks on topological structures beyond a single node or edge

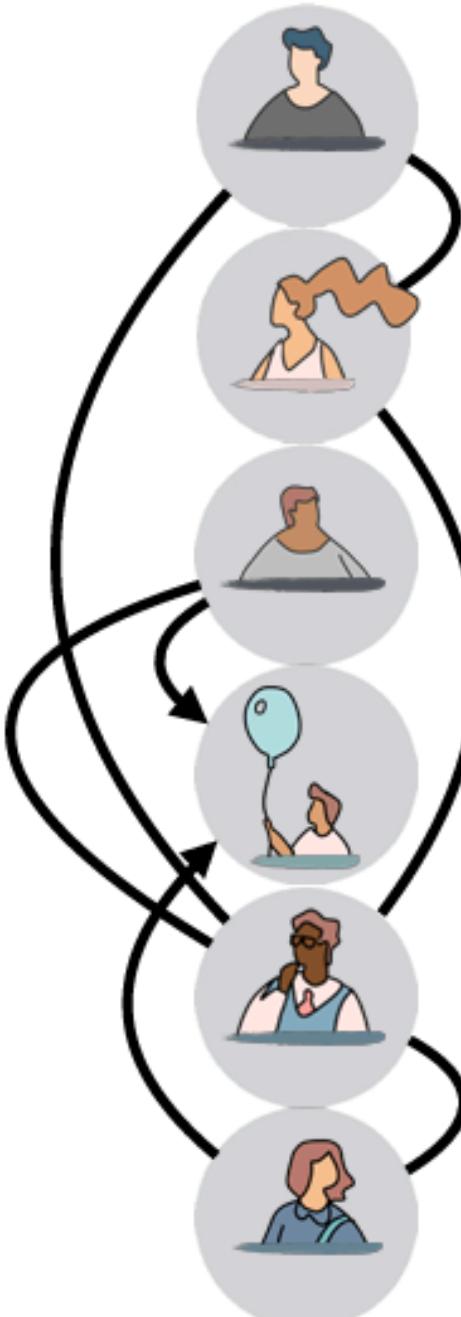
Recommended for large networks and/or heterogenous types of node and link attributes

MVN Operations: Juxtaposed Views



MVN Operations: Juxtaposed Views

Name	Beverage	Day 1
Mark	Beer	1
Sue	Coke	0
Cole	Port	4
Jon	Coke	5
Tom	Beer	2
Abby	Port	3



The diagram illustrates a circular dependency between six user profiles, each represented by a circle containing a user icon. The profiles are arranged vertically from top to bottom: Mark (male), Sue (female), Cole (male), Jon (male), Tom (male), and Abby (female). A thick black curved arrow starts at the top of Mark's profile and loops around all five other profiles, returning to the bottom of Mark's profile, indicating a cyclical relationship between all users.

MVN Operations: Juxtaposed Views

Pros:

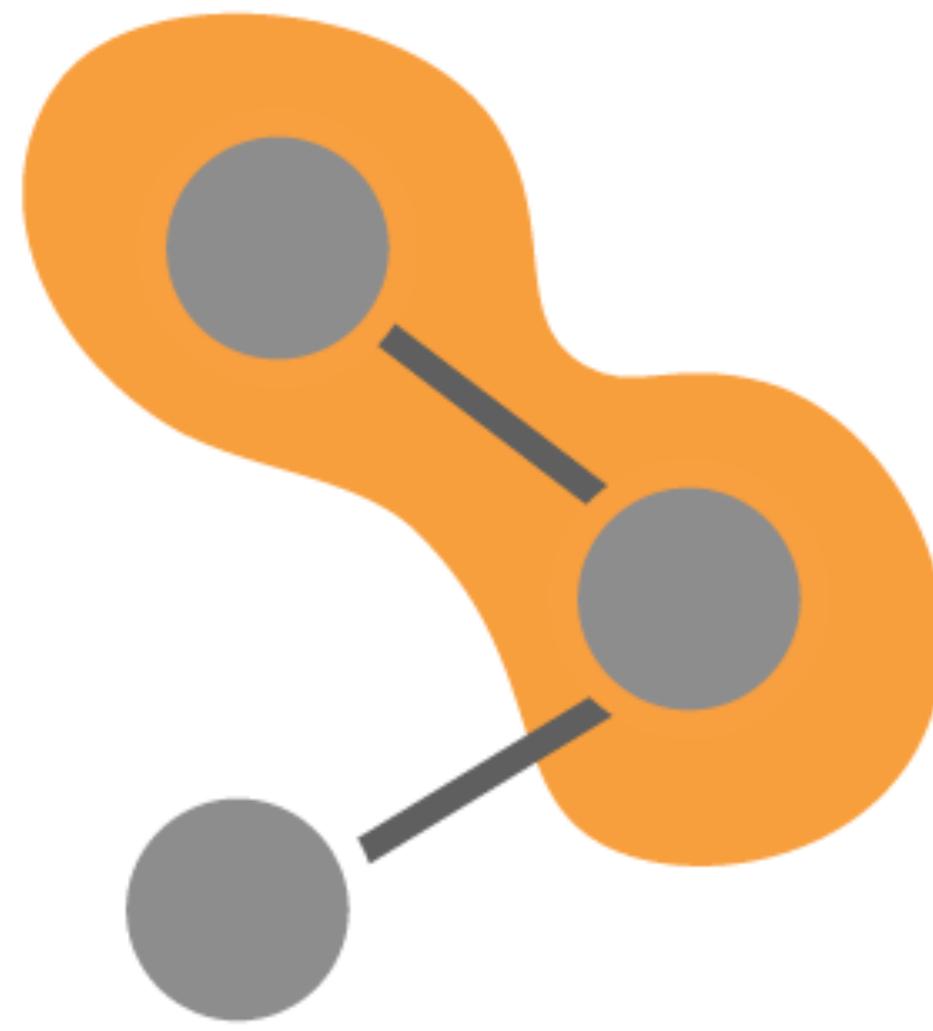
- good at integrating attributes with topology if the topology can be visualized in a linear layout

Cons:

- not suitable for networks that can't be sensibly linearized

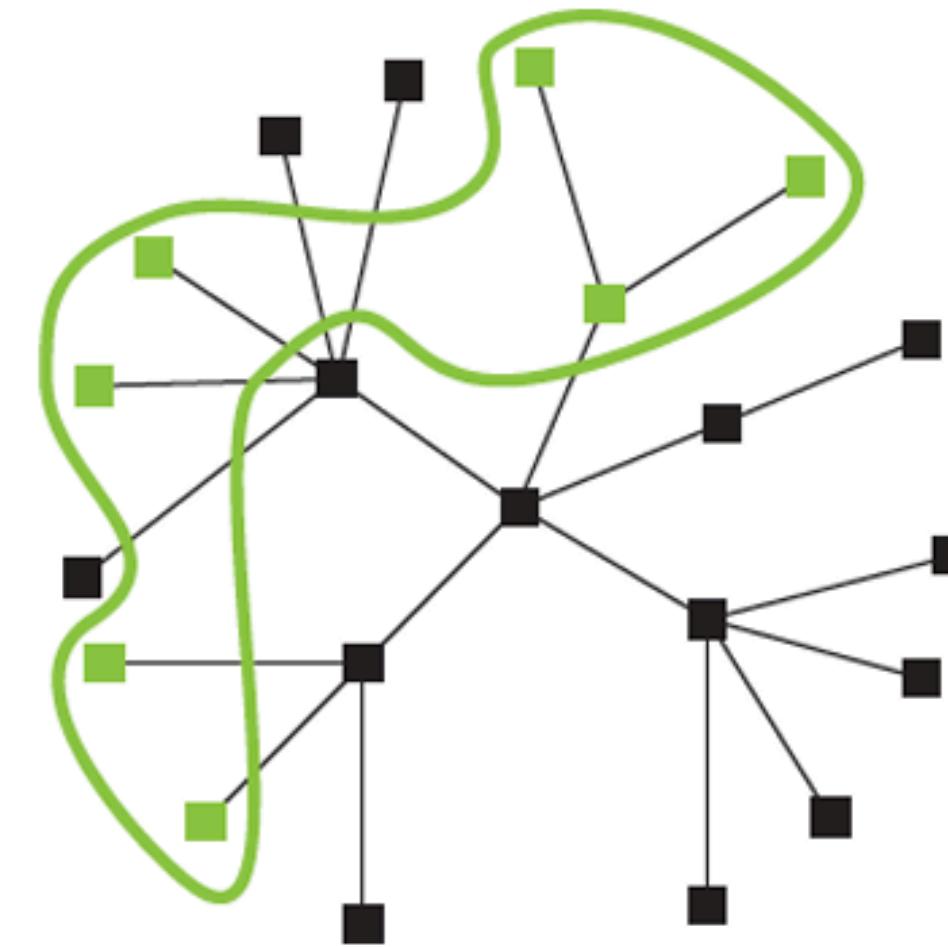
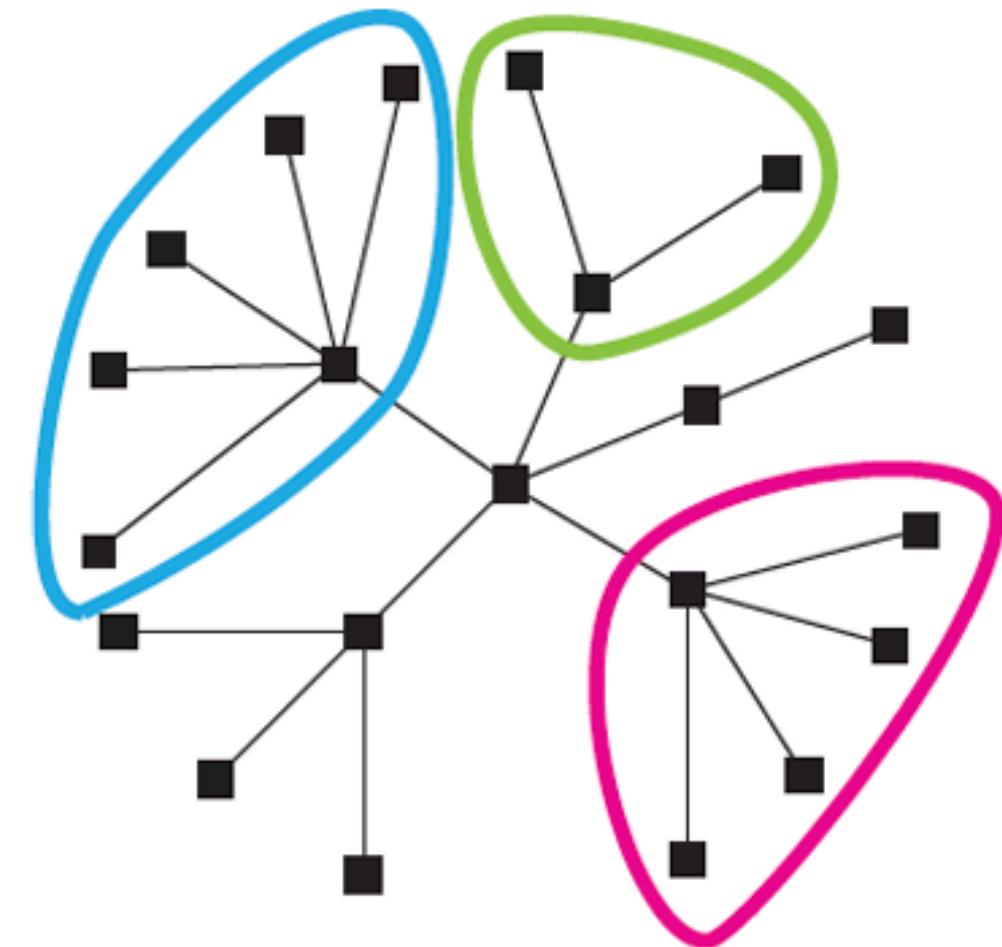
Recommended for networks with several, heterogenous, node attributes

MVN Operations: Overloaded Views



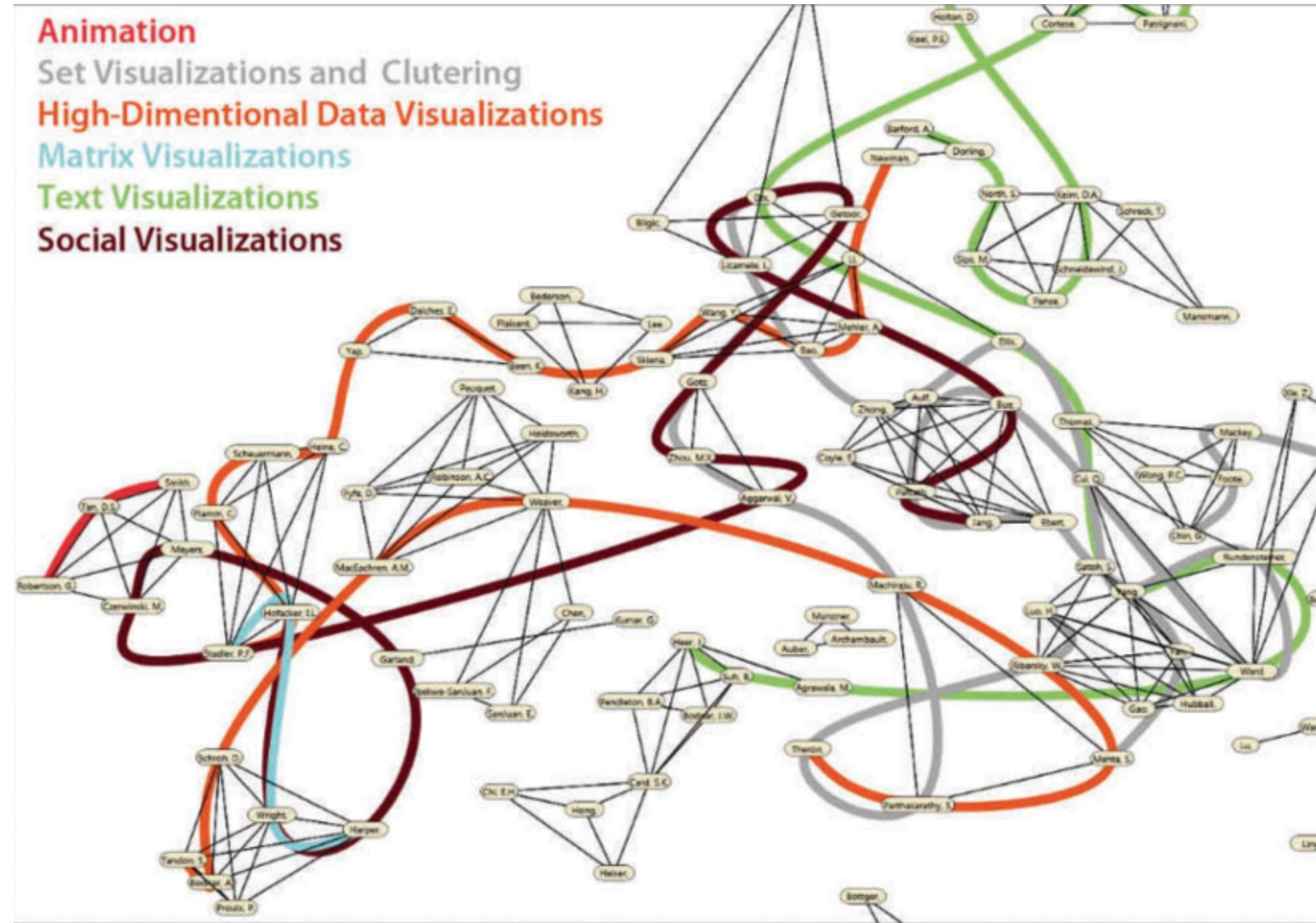
MVN Operations: Overloaded Views

Bubble Sets



MVN Operations: Overloaded Views

Line Sets



MVN Operations: Overloaded Views

Pros:

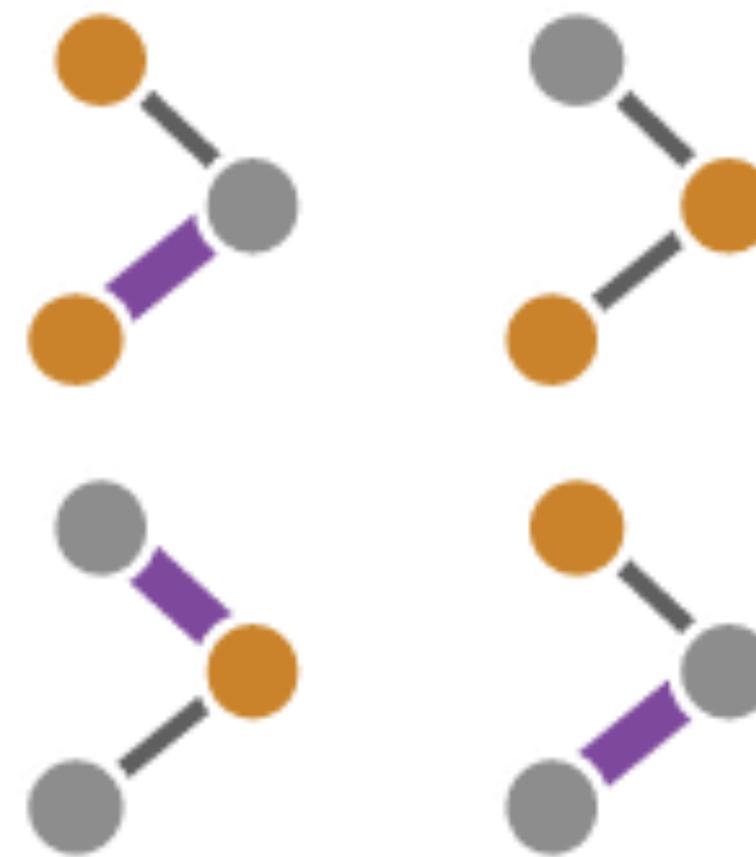
- good at displaying sets and clusters

Cons:

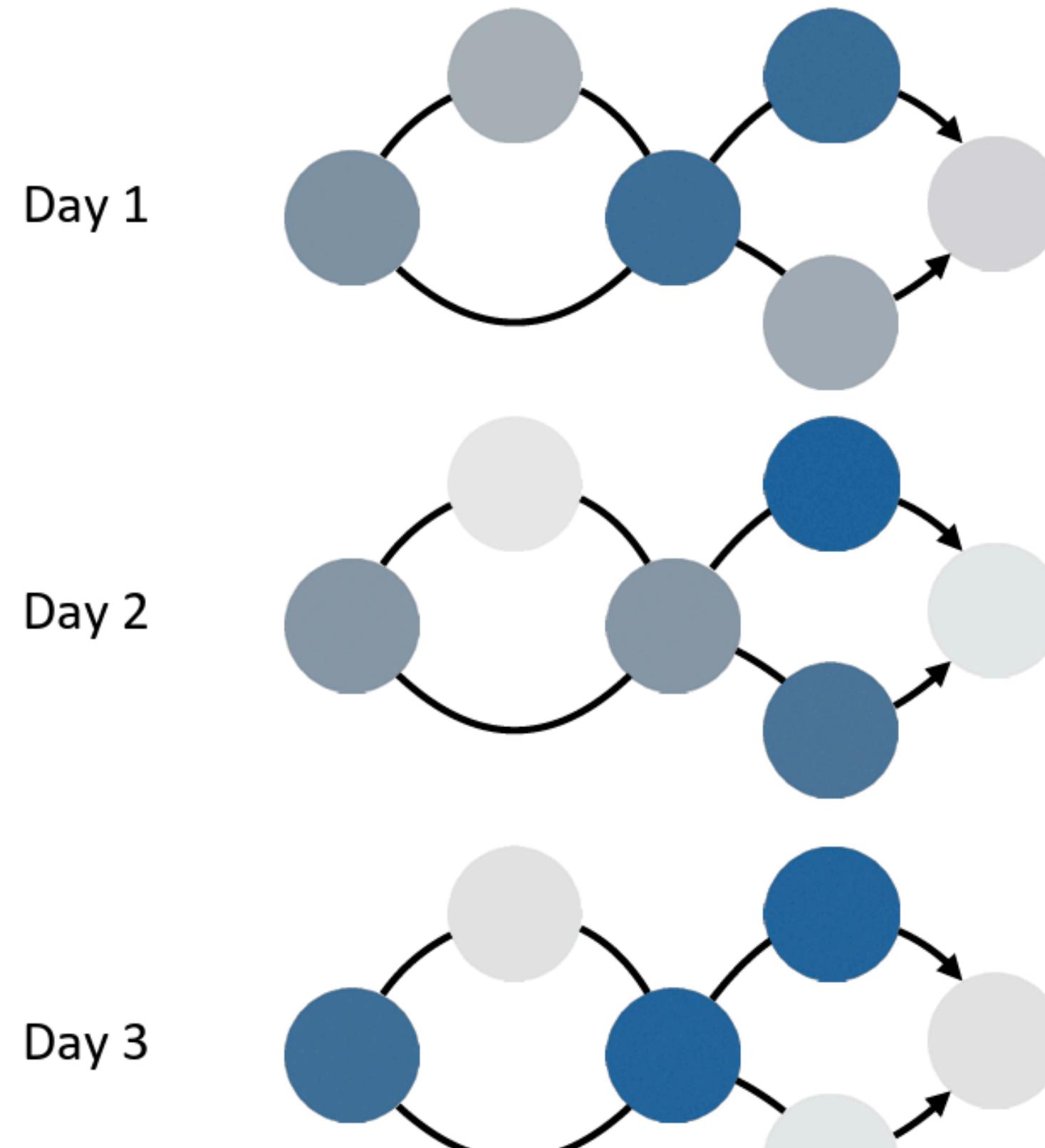
- not suitable for displaying more than one or two attributes at a time

Recommended for the particular use-case of visualizing set-membership or clusters on top of node-link diagrams

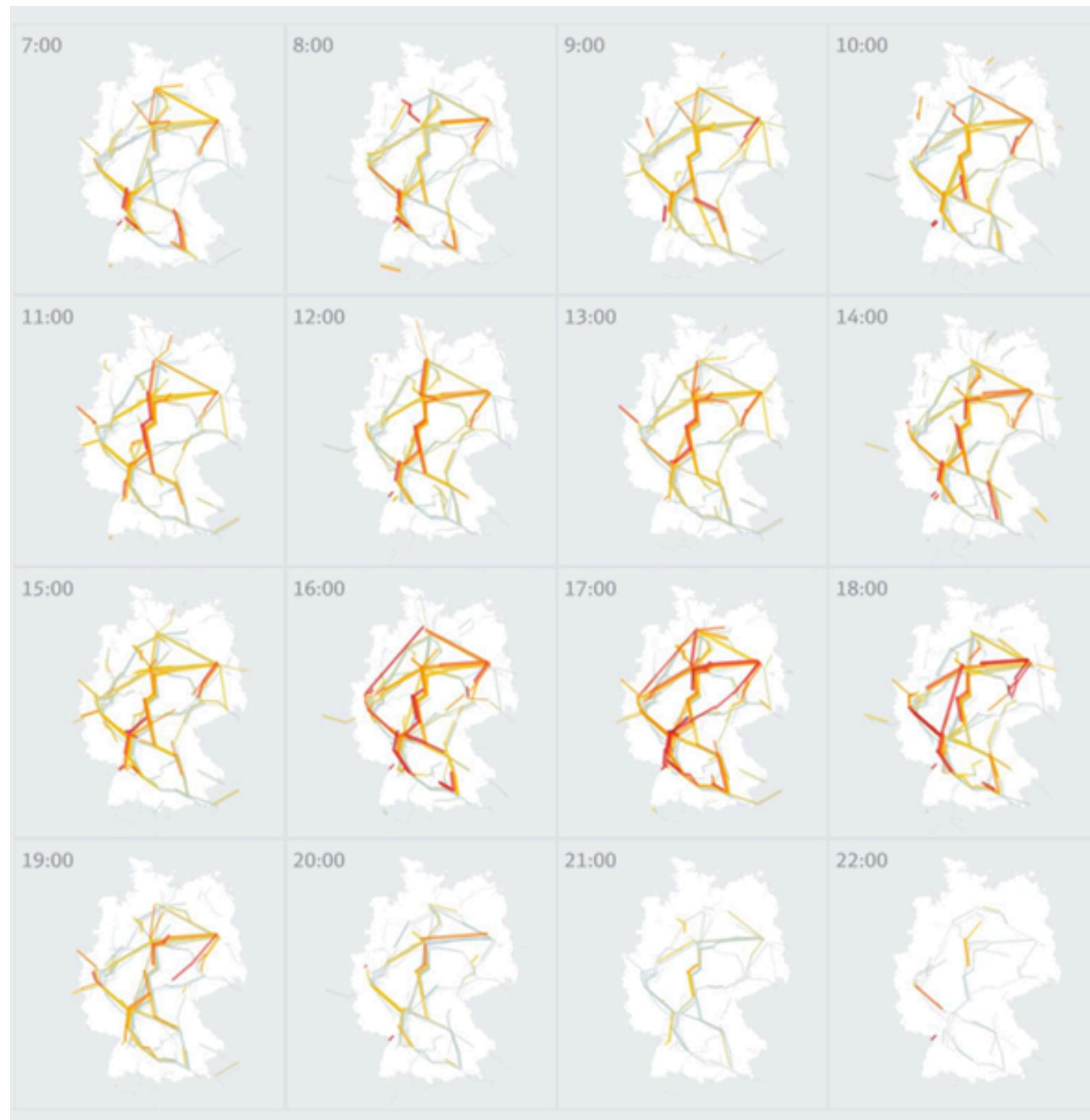
MVN Operations: Small Multiples



MVN Operations: Small Multiples



MVN Operations: Small Multiples



MVN Operations: Small Multiples

Pros:

- facilitates attribute comparisons within specific topological features

Cons:

- not ideal for large networks, or tasks on clusters

Recommended for smaller networks where the tasks are focused on attribute comparison

More at <http://vdl.sci.utah.edu/mvnv/>

FIN

Upcoming Dates

May 2:

- Homework 5 Due*
- Project Screencast Submission Due*
- All Quizzes Due

May 12: Final Project Submission Due

*Request an Extension to May 9 for Project
Screencast and Homework 5