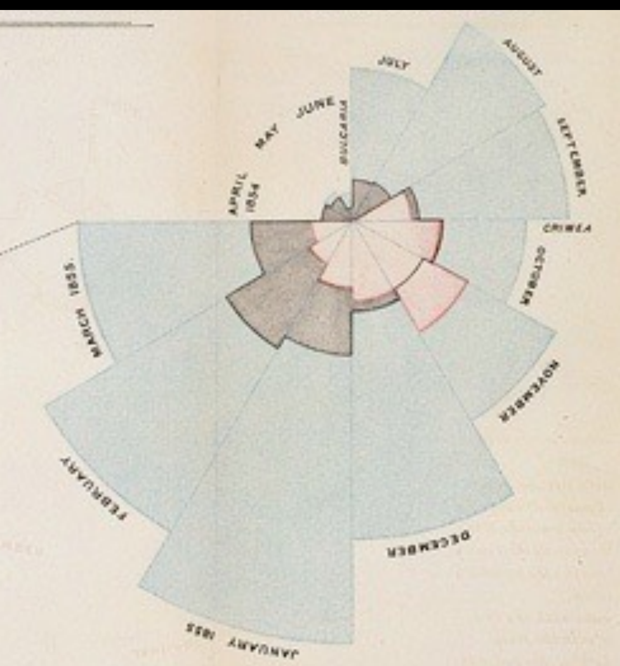
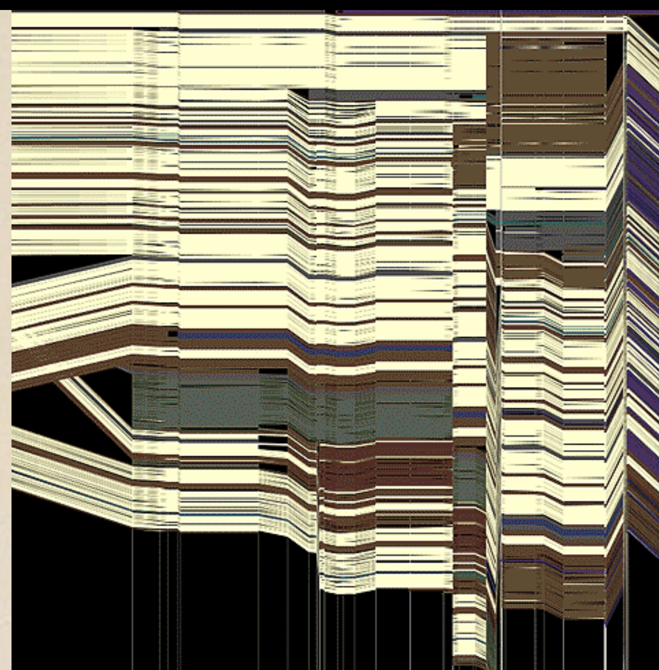
CSE 512 - Data Visualization

Networks





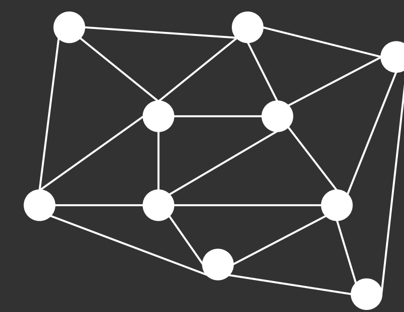
Jeffrey Heer University of Washington

Graphs and Trees

Graphs

Model relations among data

Nodes and *edges*

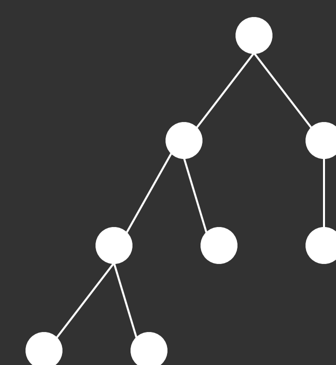


Trees

Graphs with hierarchical structure

Connected graph with N-1 edges

Nodes as *parents* and *children*



Spatial Layout

A primary concern of tree/graph drawing is the spatial arrangement of nodes and edges.

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

- Connectivity, path-following
- Topological distance
- Clustering / grouping
- Ordering (e.g., hierarchy level)

Applications

Tournaments
Organization Charts
Genealogy
Diagramming (e.g., Visio)
Biological Interactions (Genes, Proteins)
Computer Networks
Social Networks
Simulation and Modeling
Integrated Circuit Design

Topics

Tree Visualization

Graph Layout: Node-Link Diagrams

- Sugiyama-Style Layout
- Force-Directed Layout

Alternatives to Node-Link Diagrams

- Matrix Diagrams
- Attribute-Driven Layout & Hive Plots

Tree Visualization

Indentation

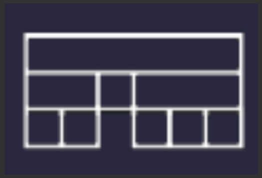
Linear list, indentation encodes depth

Node-Link diagrams

Nodes connected by lines/curves

Enclosure diagrams

Represent hierarchy by enclosure



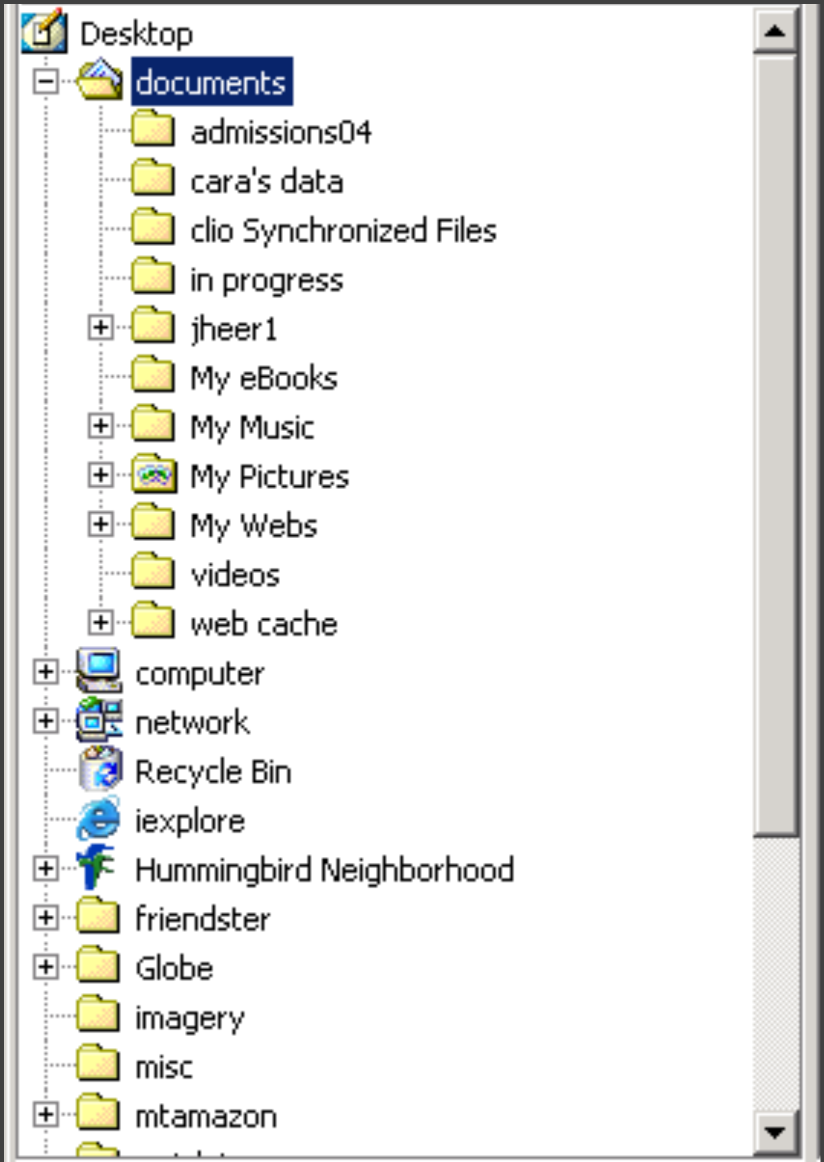
Layering

Relative position and alignment

Typically fast: O(n) or O(n log n), interactive layout

Tree Layout

Indentation



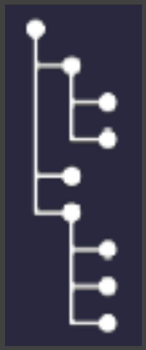
Places all items along vertically spaced rows

Indentation used to show parent/child relationships

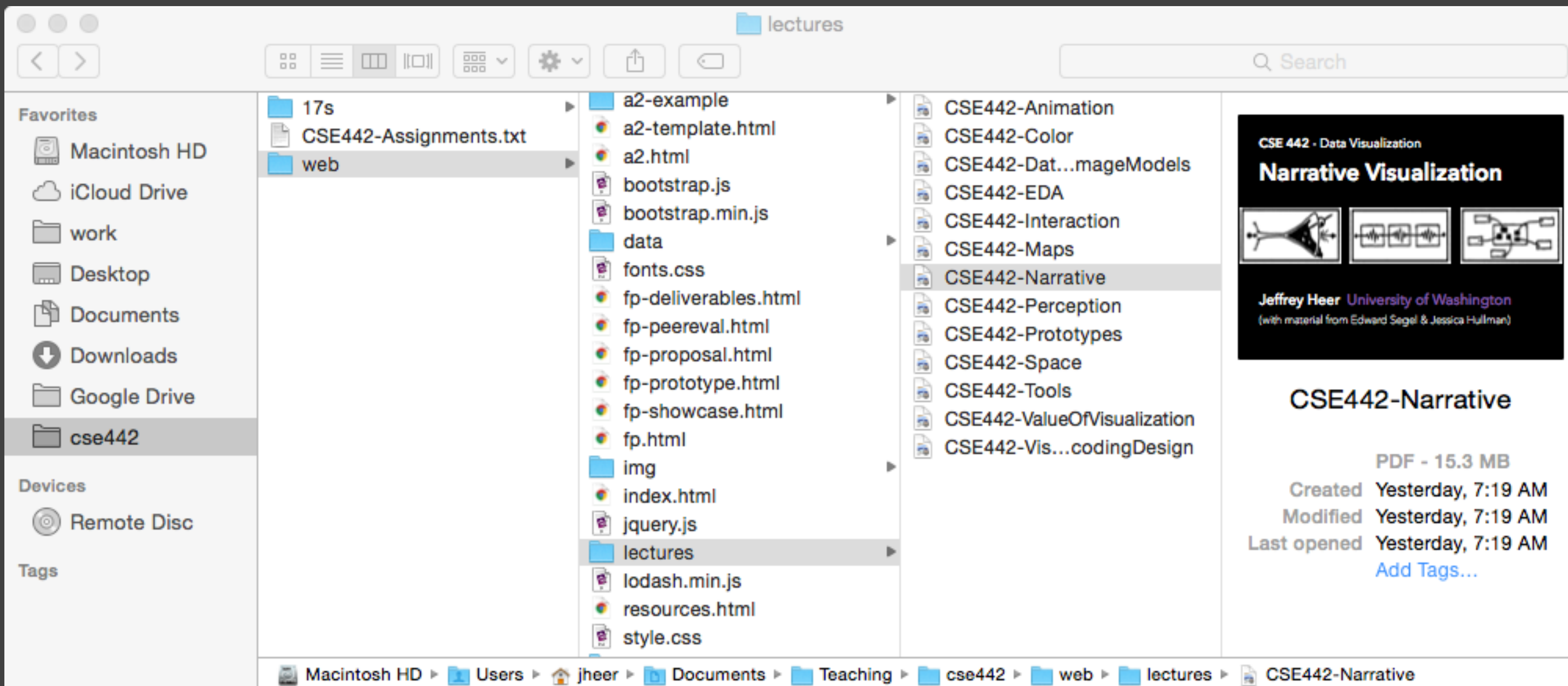
Commonly used as a component in an interface

Breadth and depth contend for space

Often requires a great deal of scrolling



Single-Focus (Accordion) List



Separate breadth & depth along 2D.
Focus on a single path at a time.

Node-Link Diagrams

Nodes are distributed in space, connected by straight or curved lines

Typical approach is to use 2D space to break apart breadth and depth

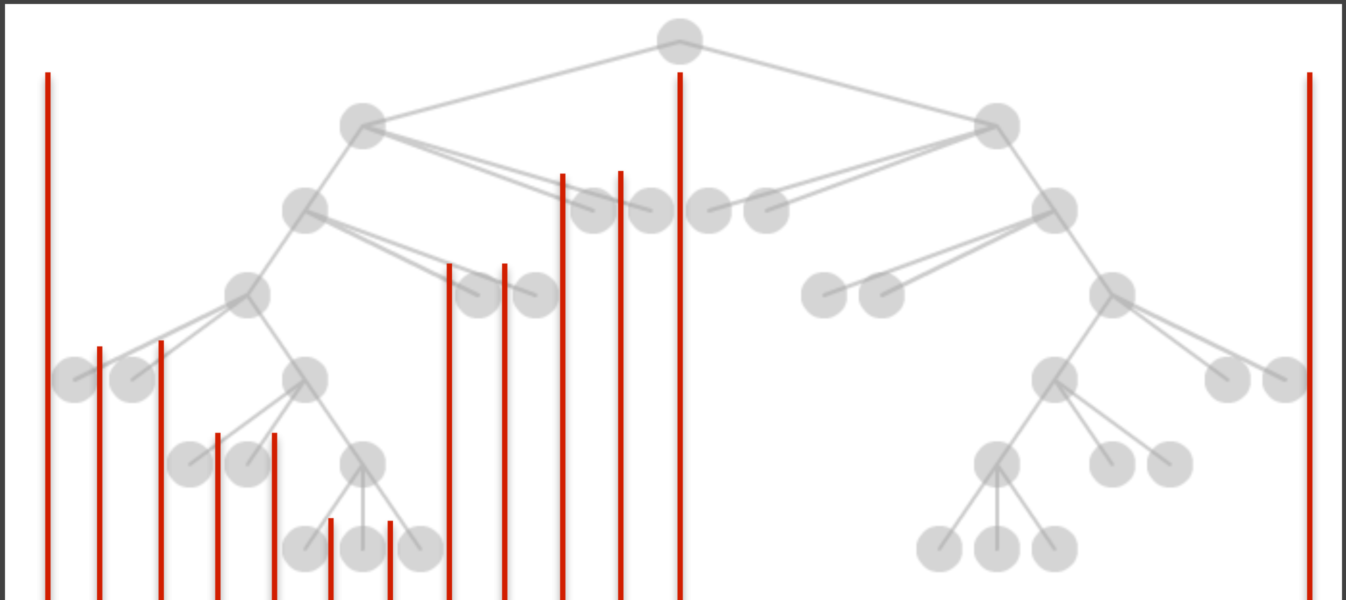
Often space is used to communicate hierarchical orientation (e.g., towards authority or generality)

Naïve Recursive Layout

Repeatedly divide space for subtrees by leaf count

Breadth of tree along one dimension

Depth along the other dimension



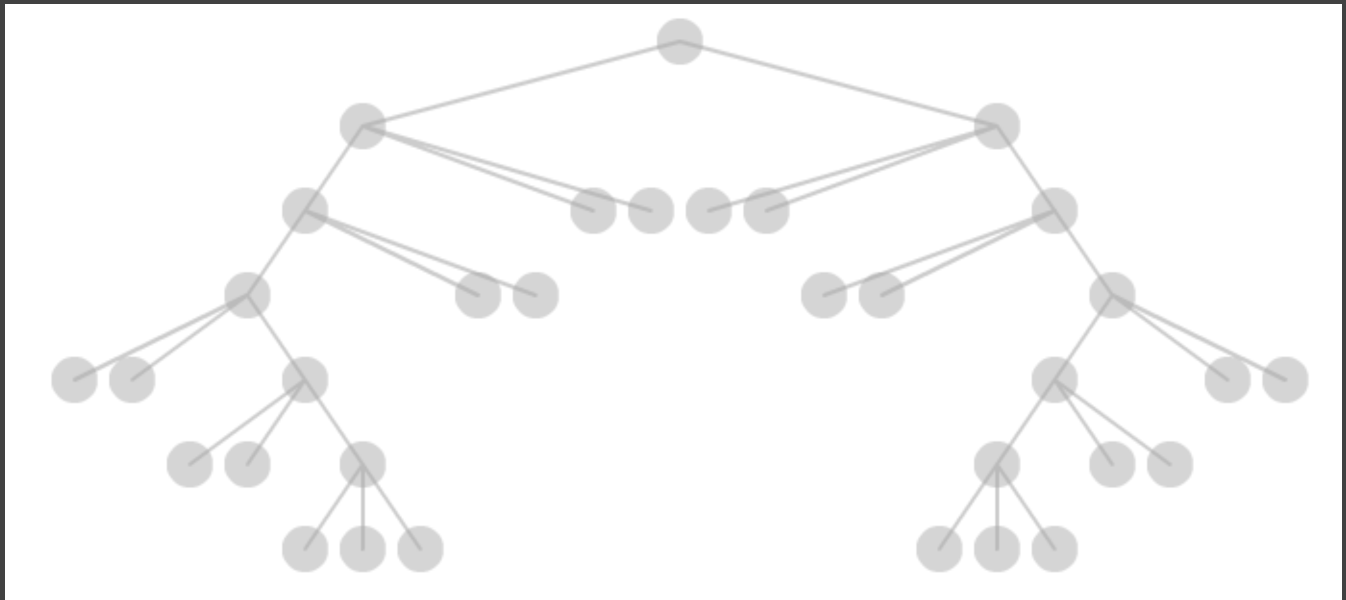
Naïve Recursive Layout

Repeatedly divide space for subtrees by leaf count

Breadth of tree along one dimension

Depth along the other dimension

Problems?



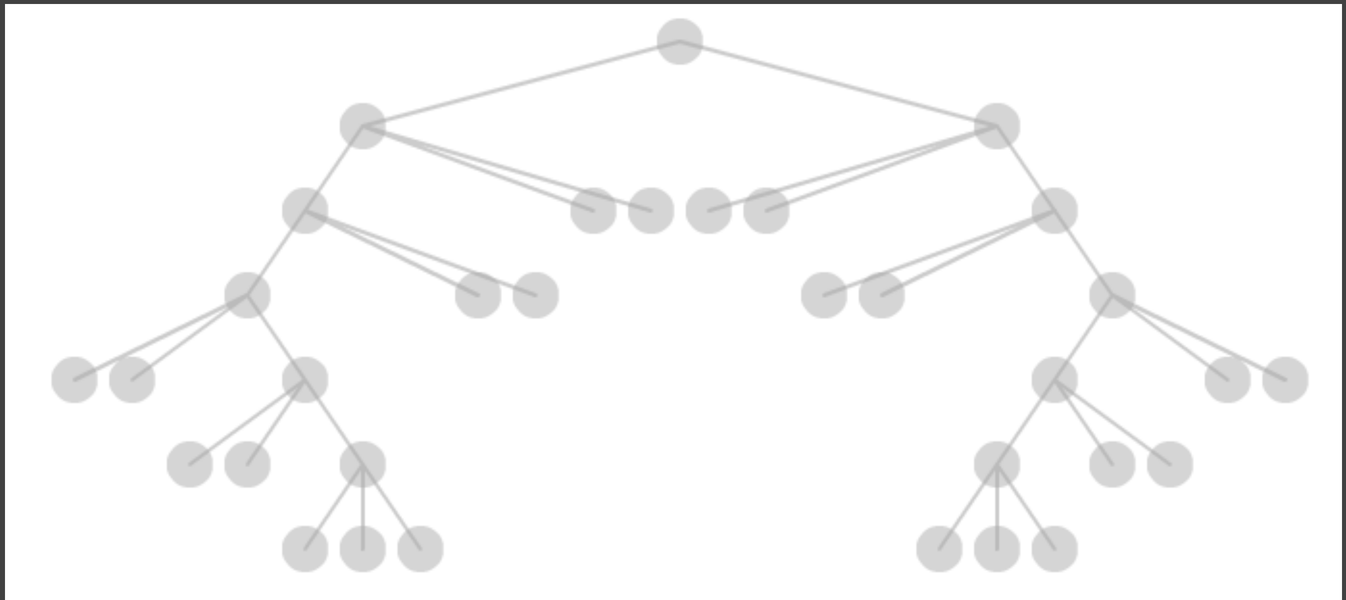
Naïve Recursive Layout

Repeatedly divide space for subtrees by leaf count

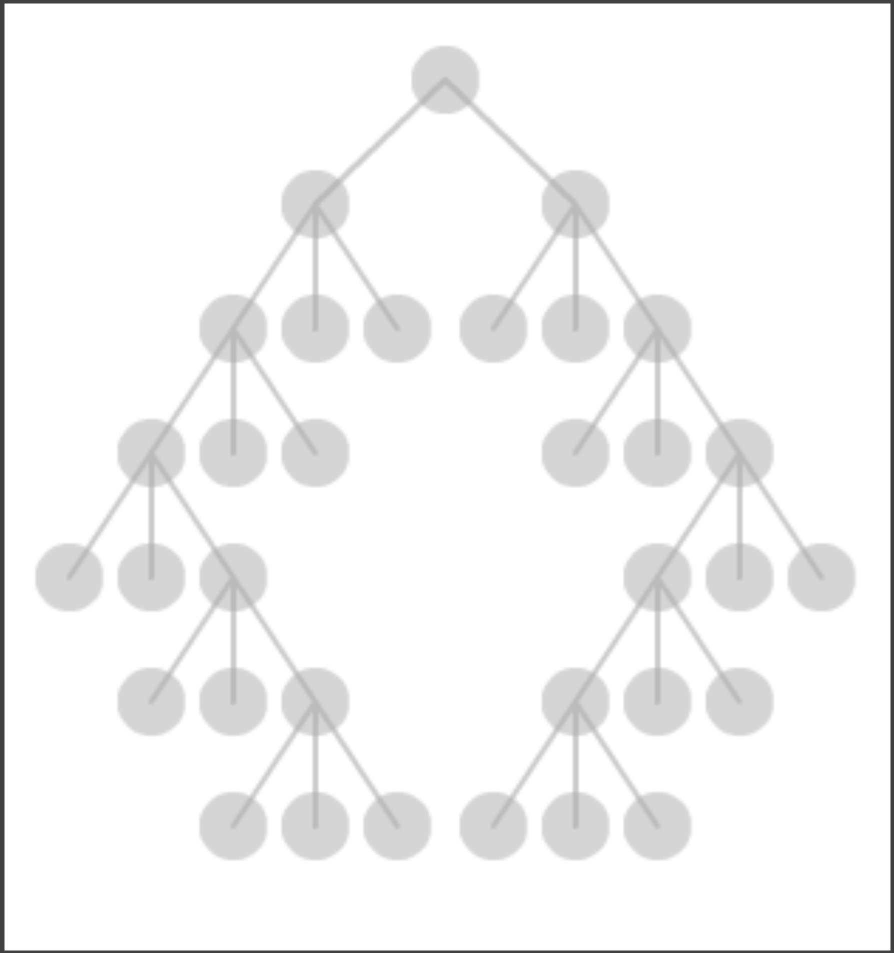
Breadth of tree along one dimension

Depth along the other dimension

Problem: exponential growth of breadth



Reingold & Tilford's "Tidy" Layout



Goal: make smarter use of space, maximize density and symmetry.

Originally binary trees, extended by Walker to cover general case.

Corrected by Buchheim et al. to achieve a linear time algorithm.

Reingold-Tilford Layout

Design Considerations

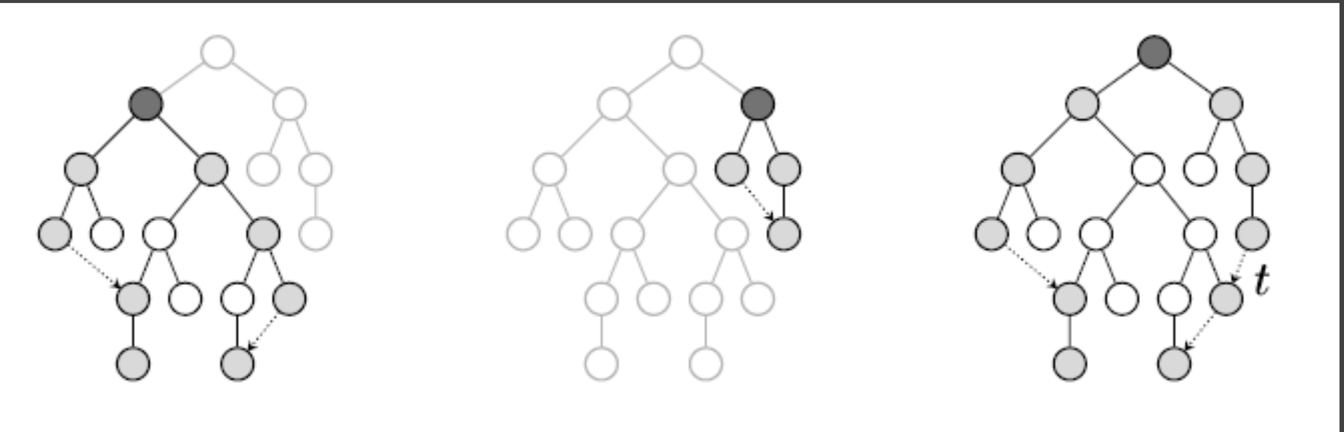
Clearly encode depth level

No edge crossings

Isomorphic subtrees drawn identically

Ordering and symmetry preserved

Compact layout (don't waste space)



Reingold-Tilford Layout

Initial bottom-up (post-order) traversal of the tree

- Y-coordinates based on tree depth
- X-coordinates initialized to zero

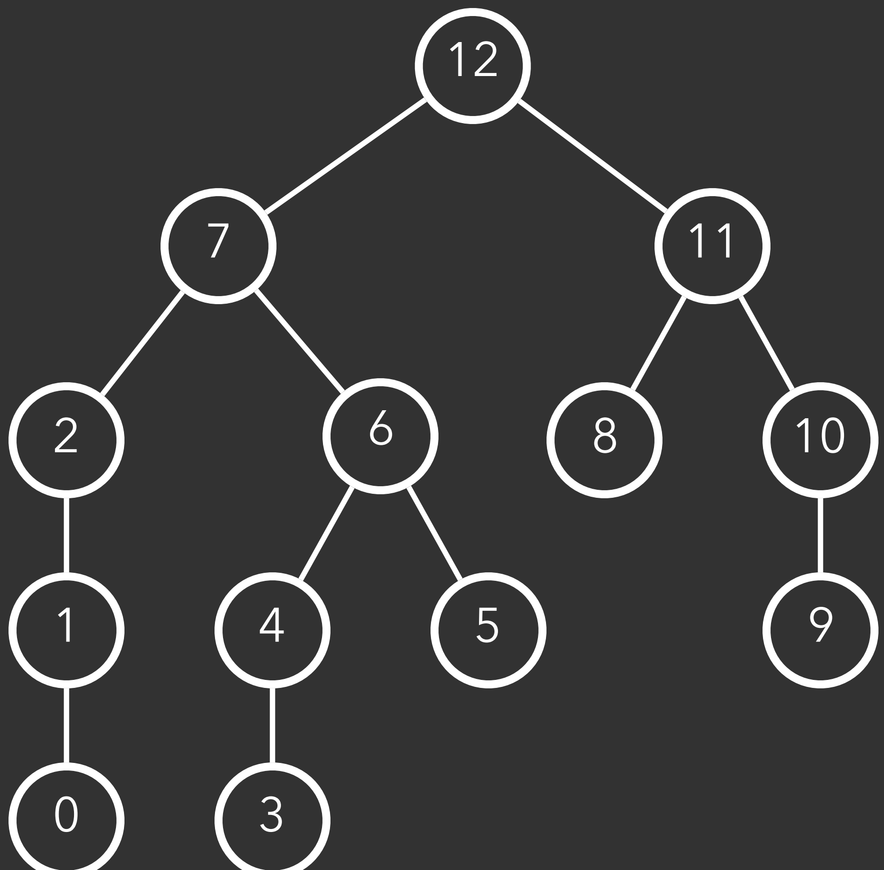
At each parent node: merge left and right subtrees

- Shift right subtree as close as possible to the left
- Compute efficiently by maintaining subtree boundaries
- Center the parent node above its children
- Record "shift" position offset for right subtree

Final top-down (pre-order) traversal to set X-coordinates

- Sum the aggregated shifts

Reingold-Tilford Layout

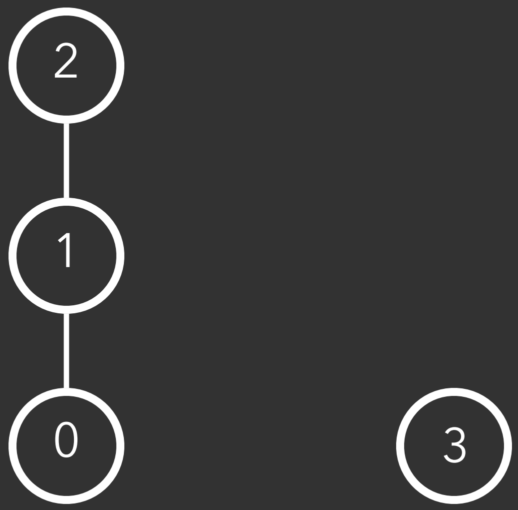


Reingold-Tilford Layout

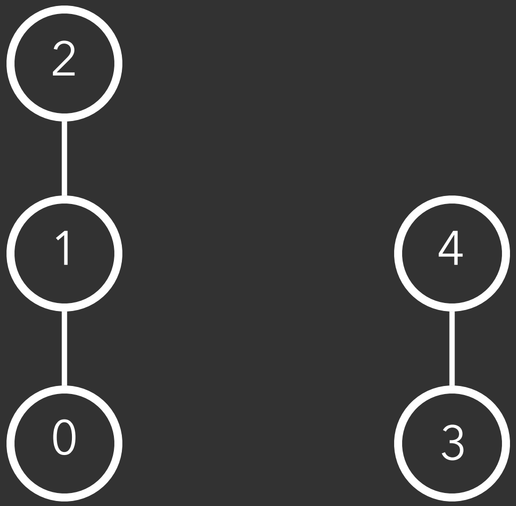
Reingold-Tilford Layout

Reingold-Tilford Layout

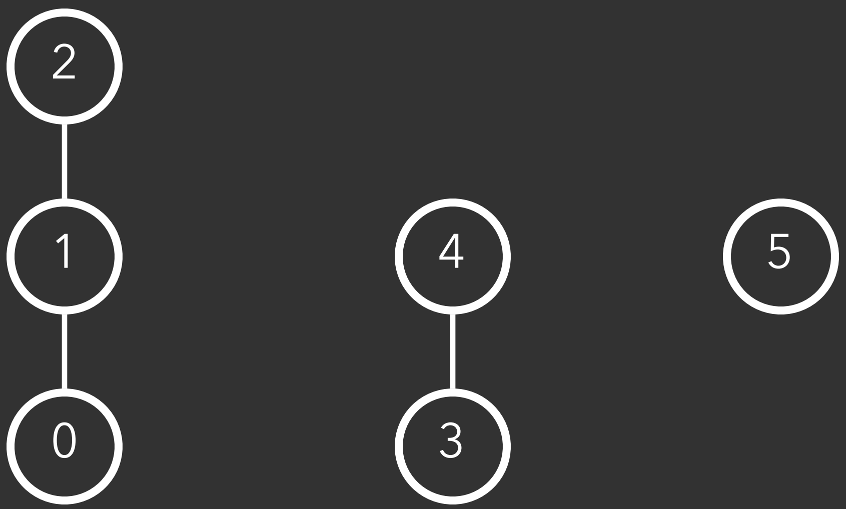
Reingold-Tilford Layout



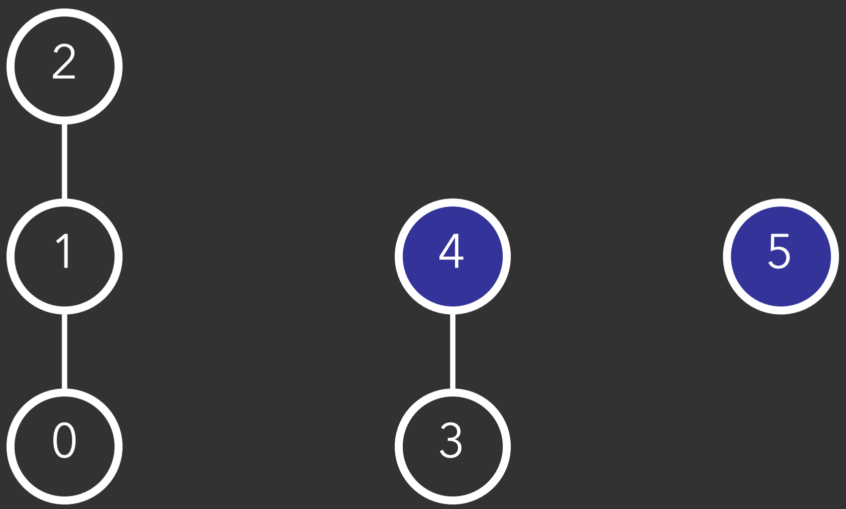
Reingold-Tilford Layout



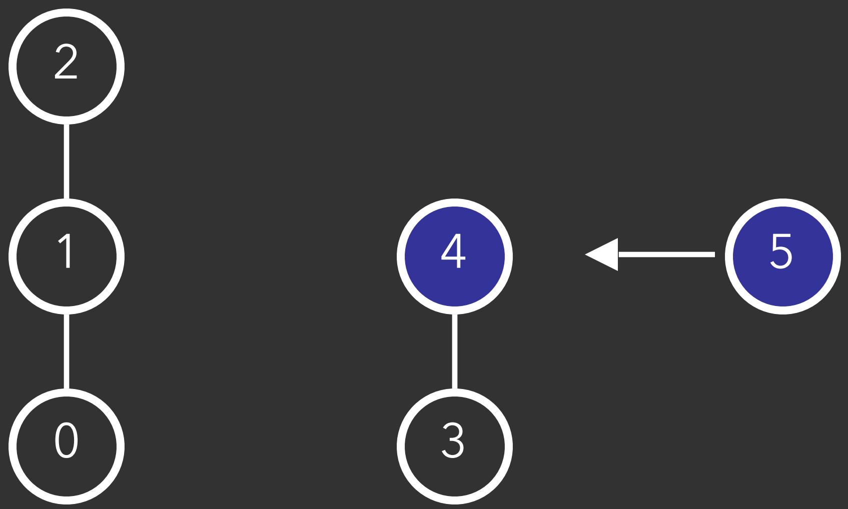
Reingold-Tilford Layout



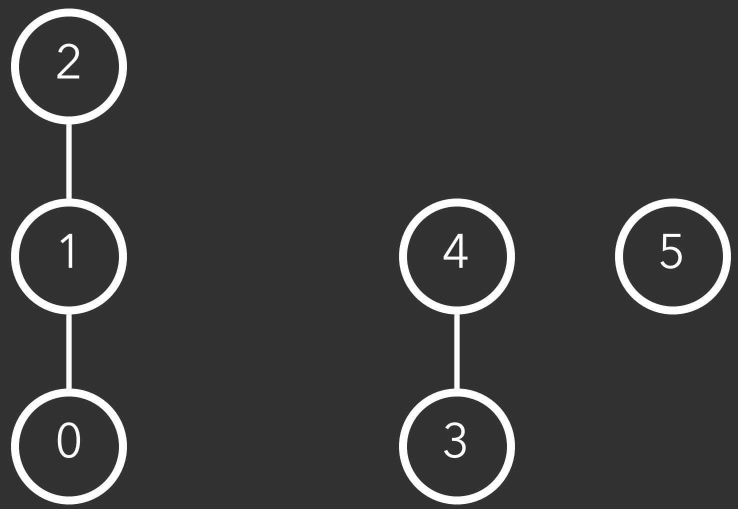
Reingold-Tilford Layout



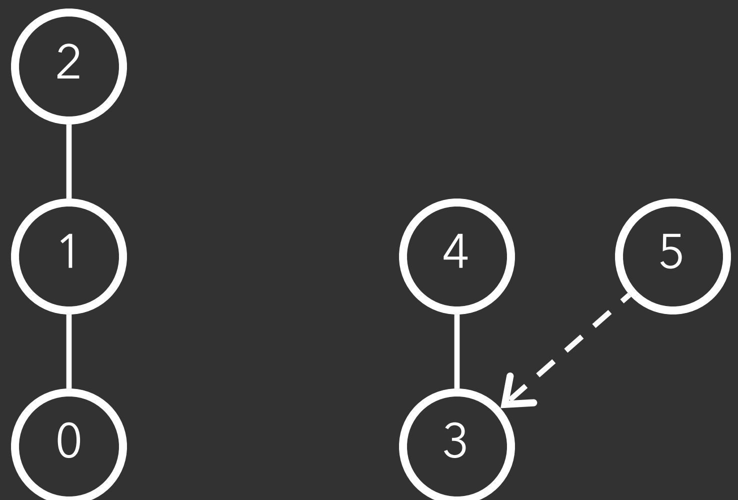
Reingold-Tilford Layout



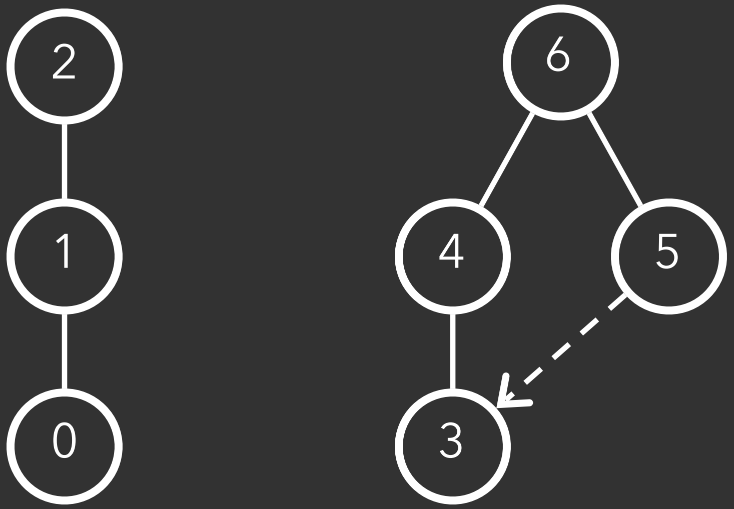
Reingold-Tilford Layout



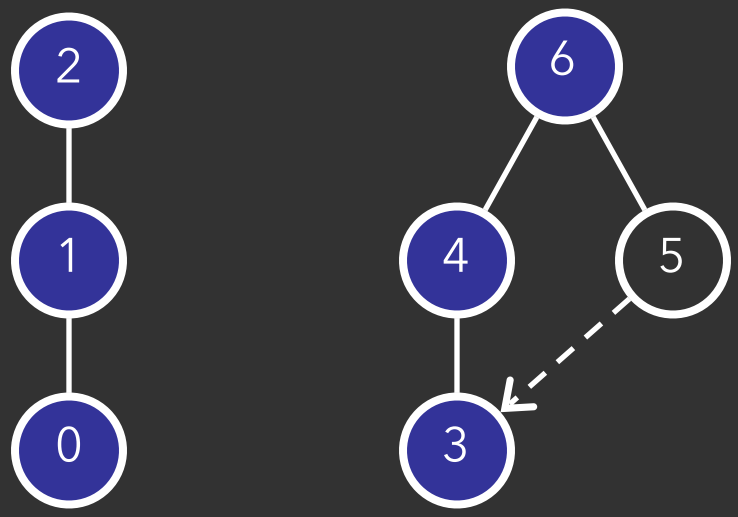
Reingold-Tilford Layout



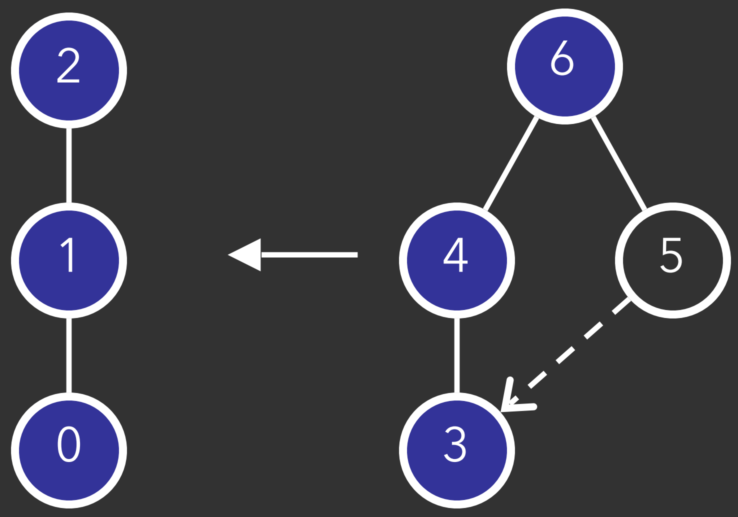
Reingold-Tilford Layout



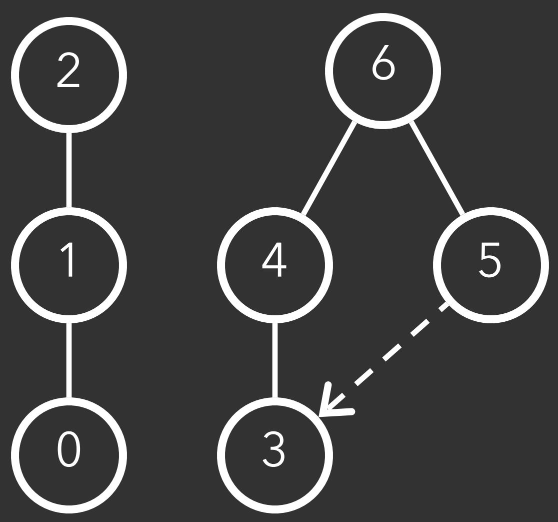
Reingold-Tilford Layout



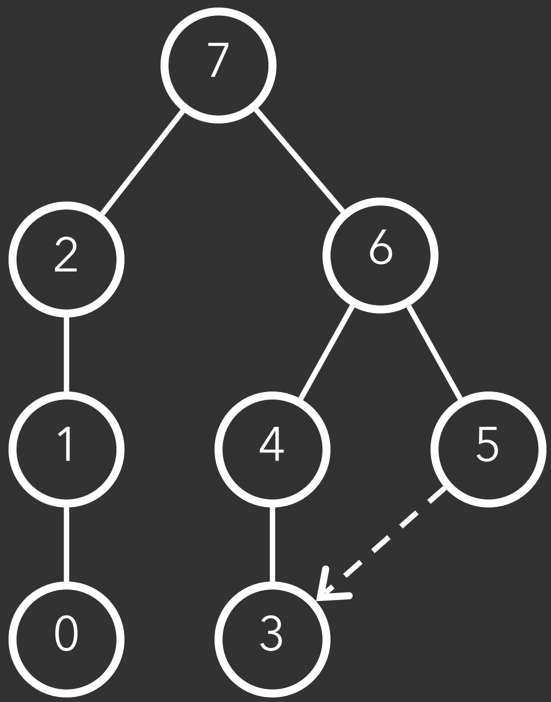
Reingold-Tilford Layout



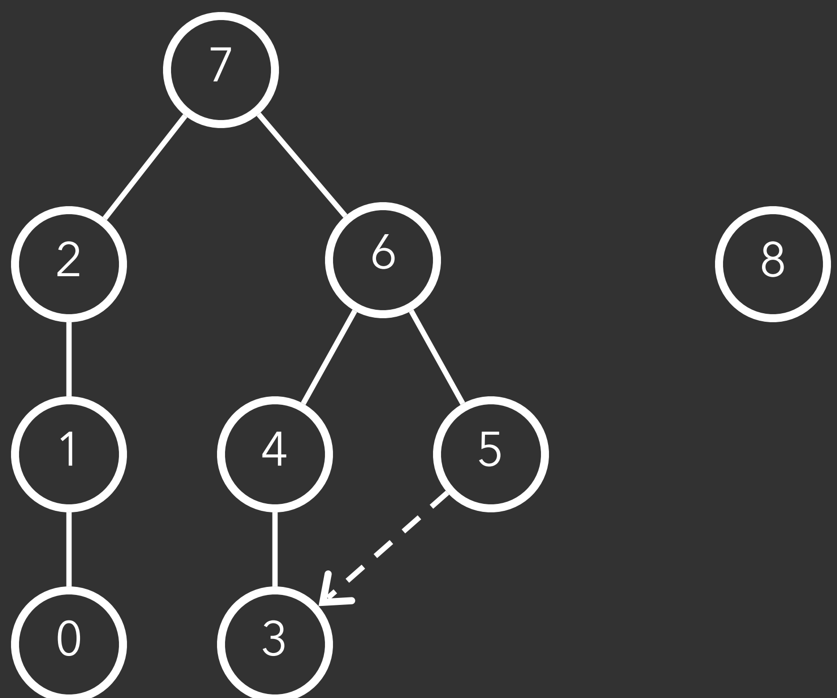
Reingold-Tilford Layout



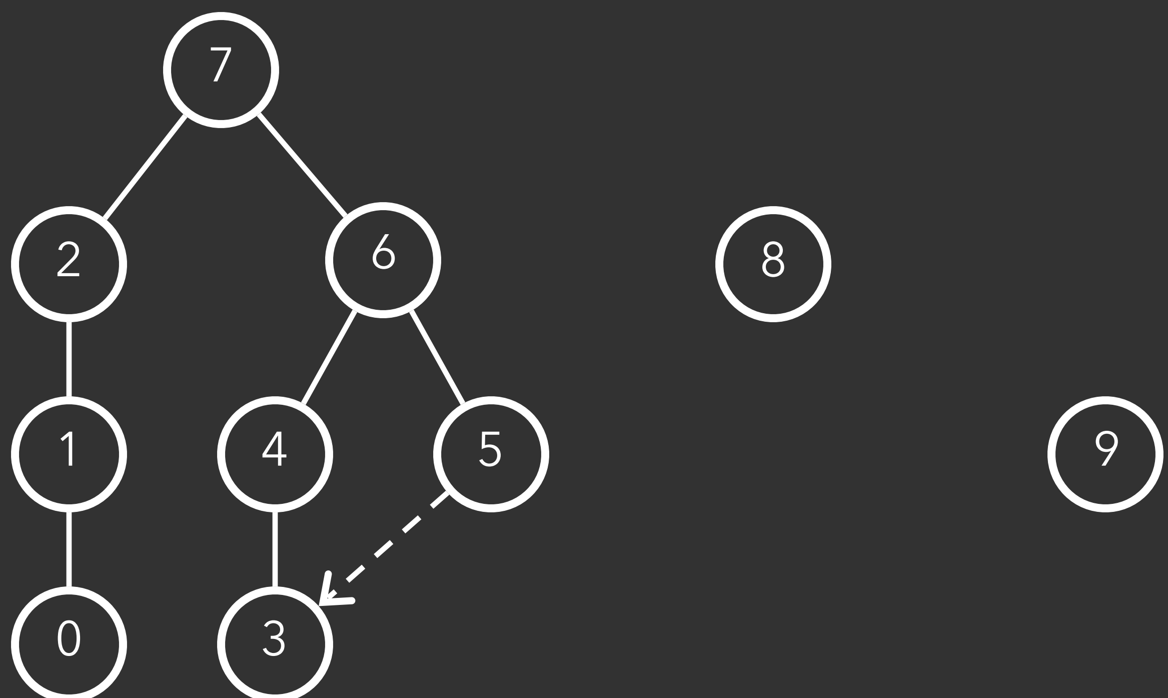
Reingold-Tilford Layout



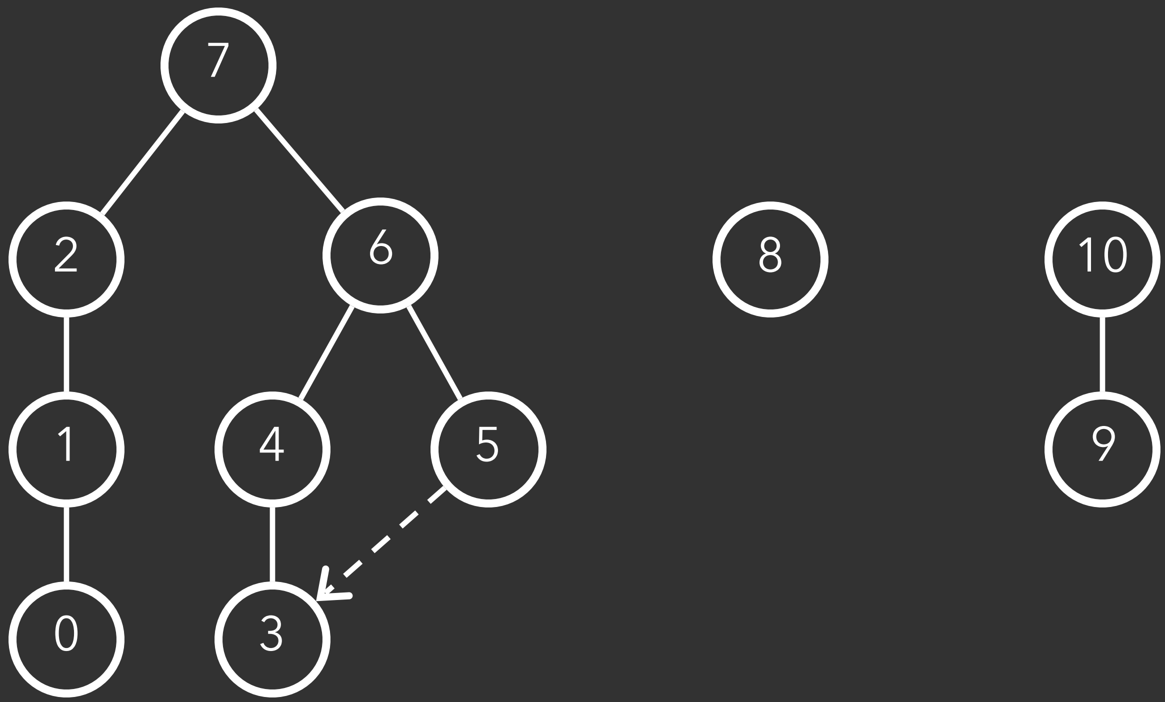
Reingold-Tilford Layout



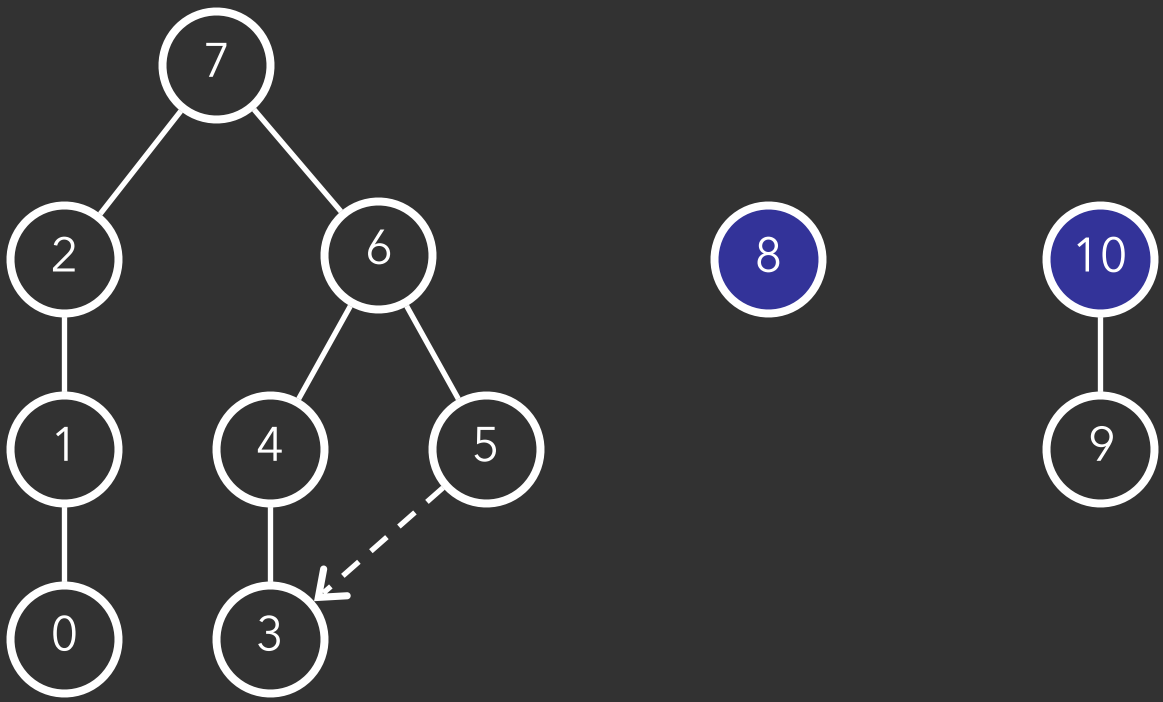
Reingold-Tilford Layout



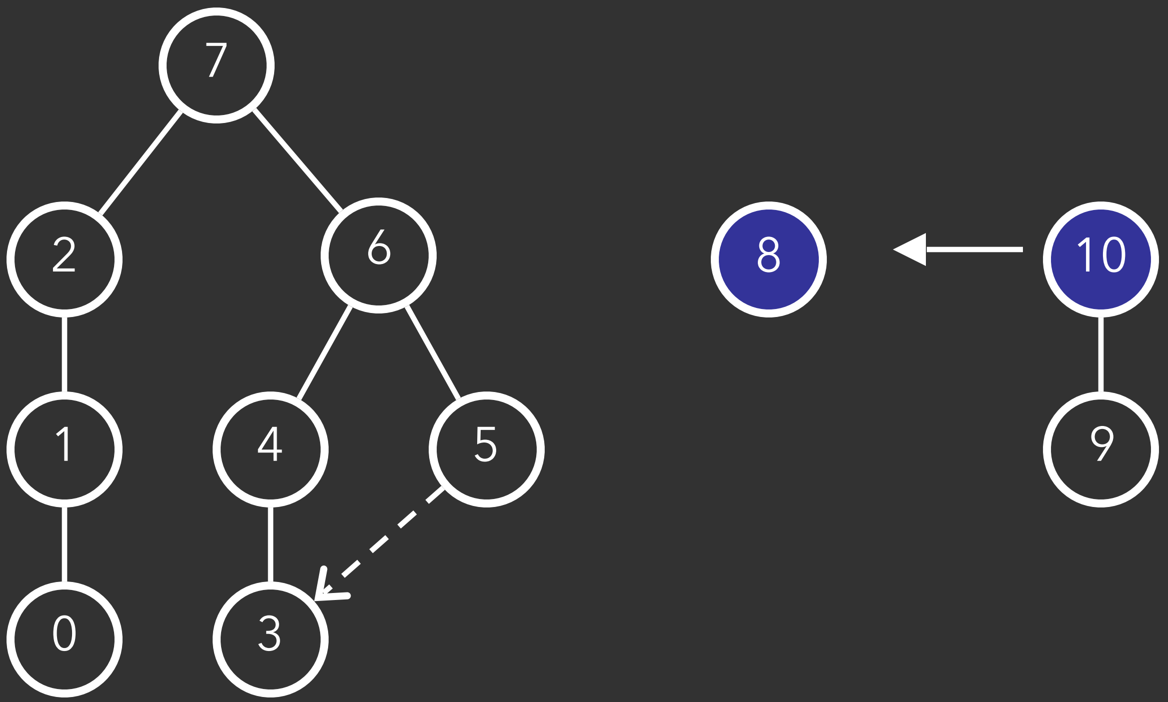
Reingold-Tilford Layout



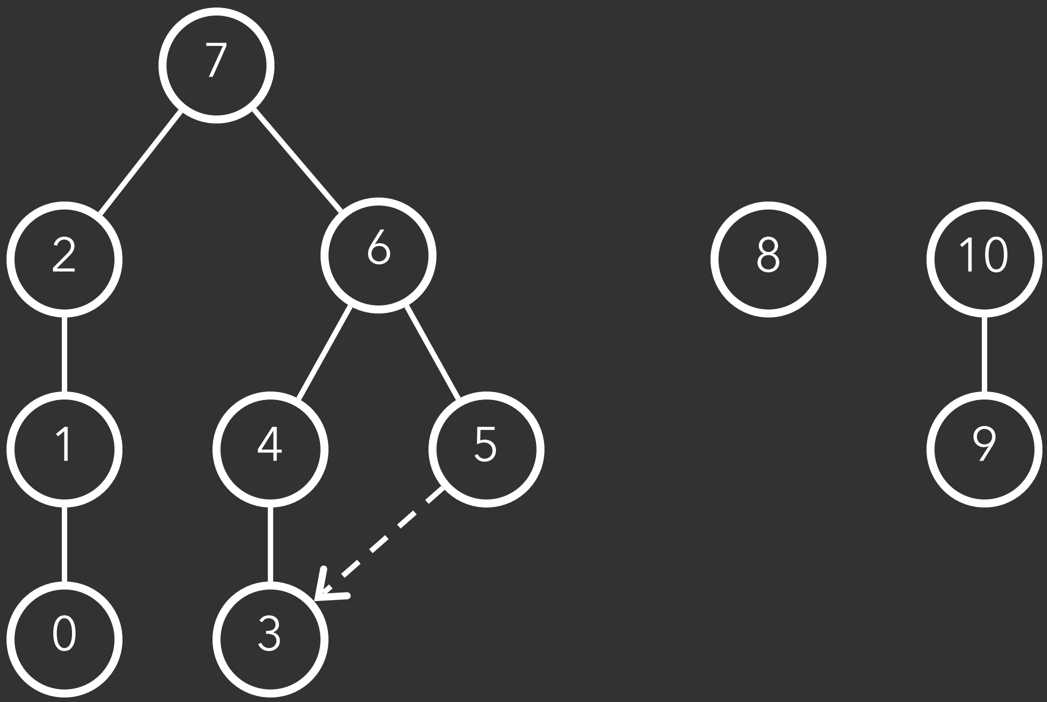
Reingold-Tilford Layout



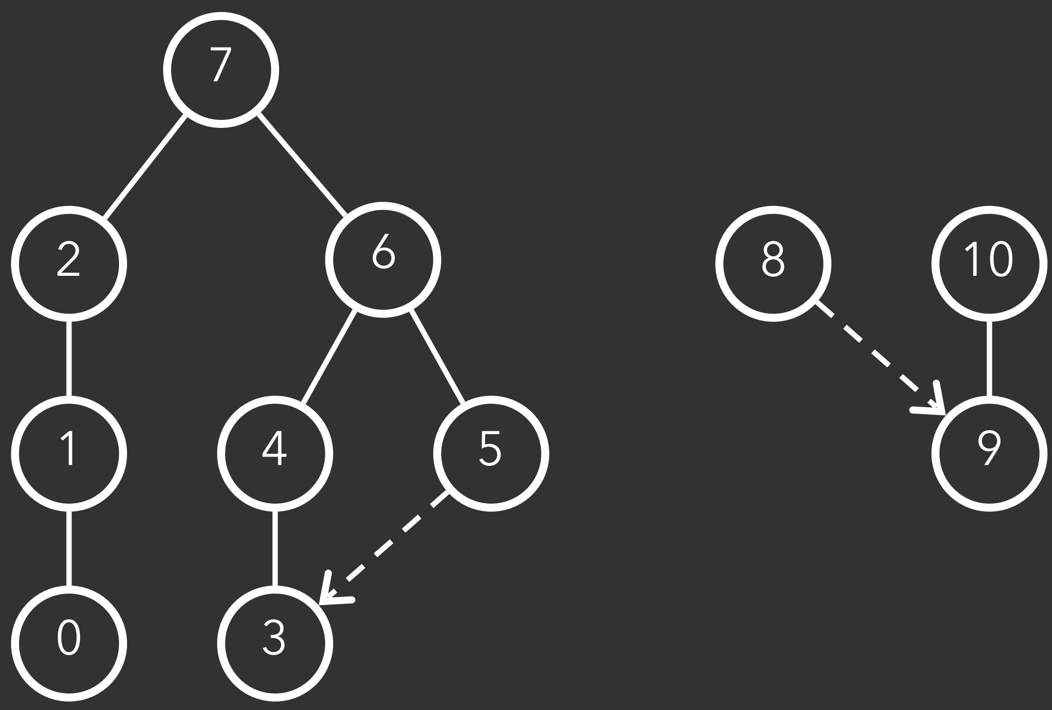
Reingold-Tilford Layout



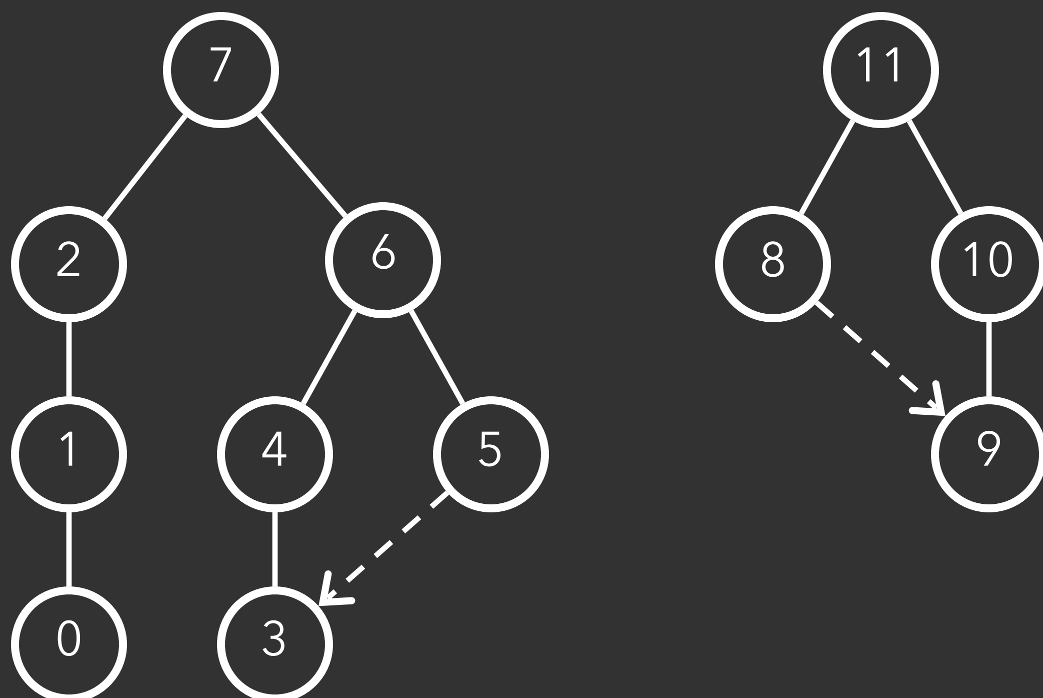
Reingold-Tilford Layout



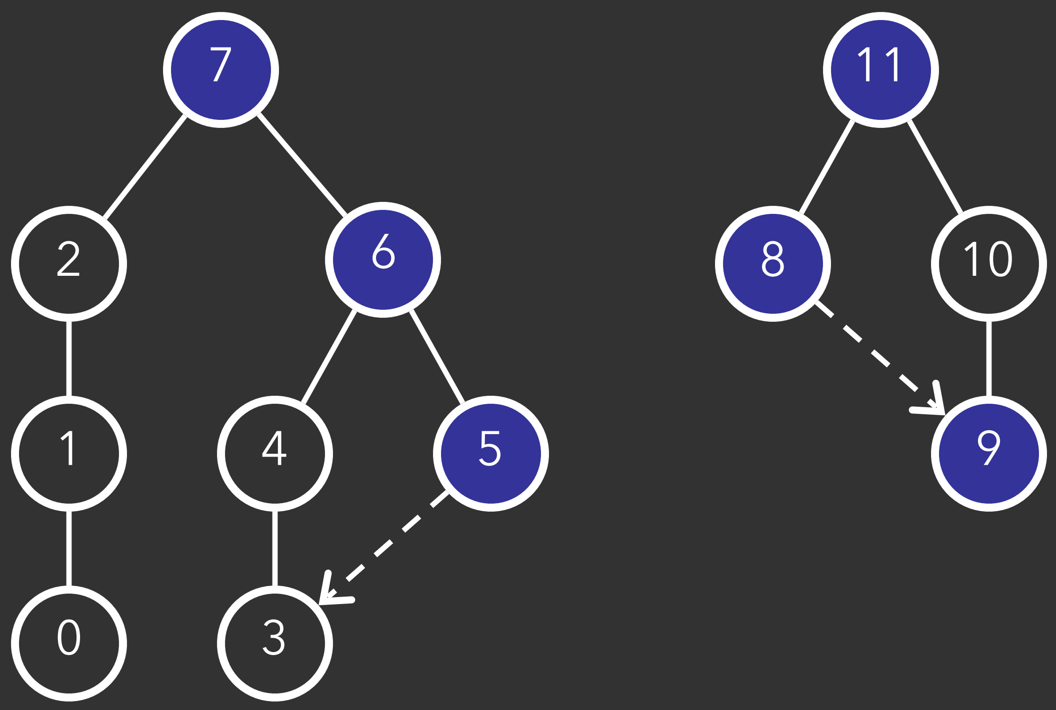
Reingold-Tilford Layout



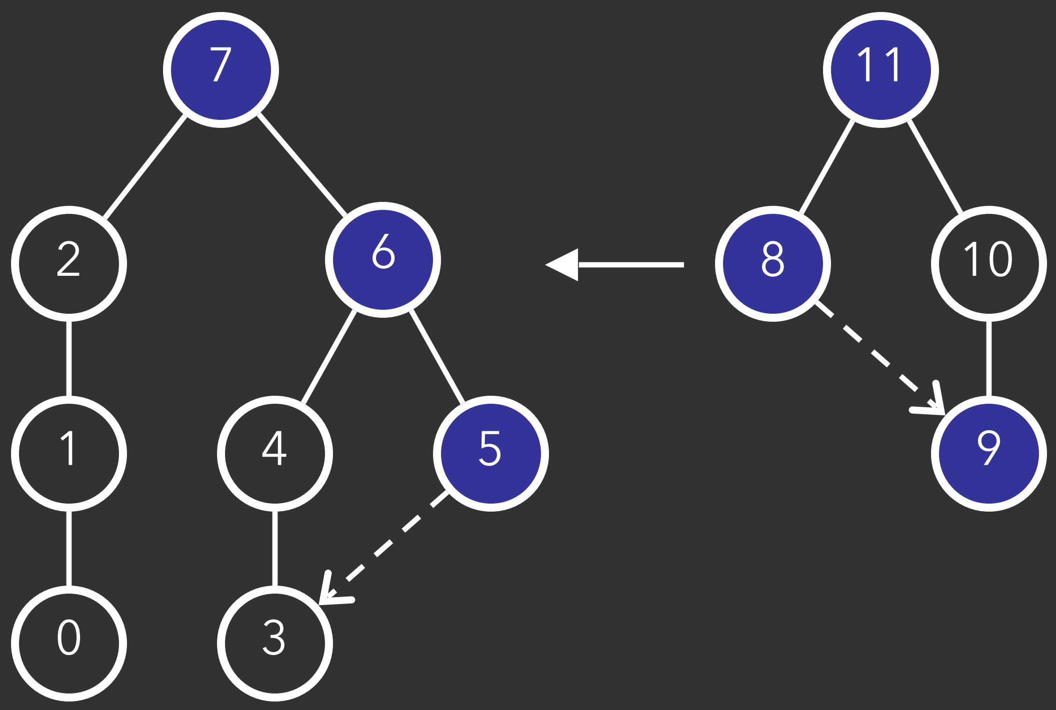
Reingold-Tilford Layout



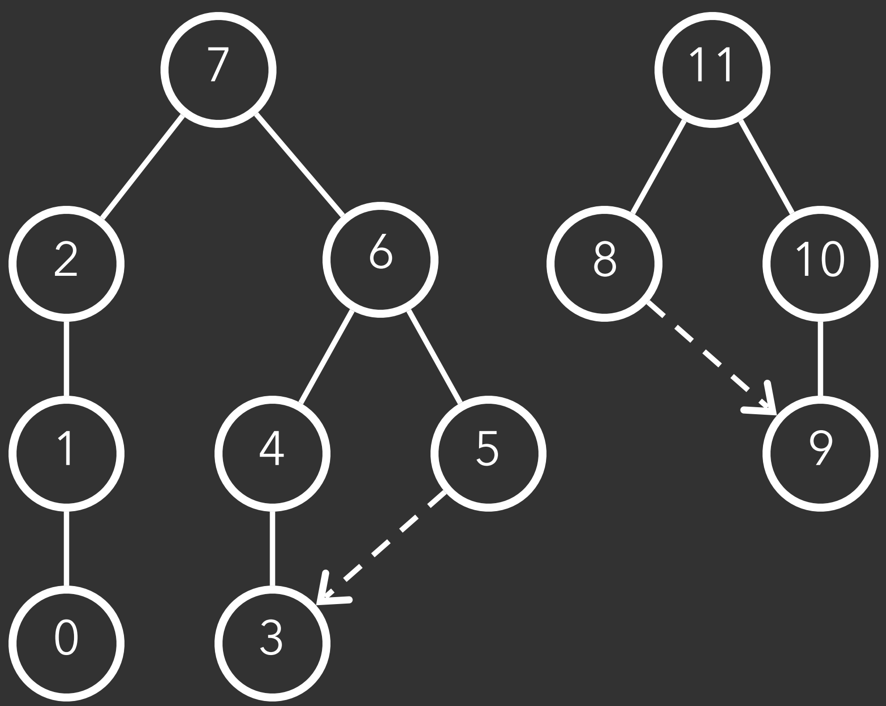
Reingold-Tilford Layout



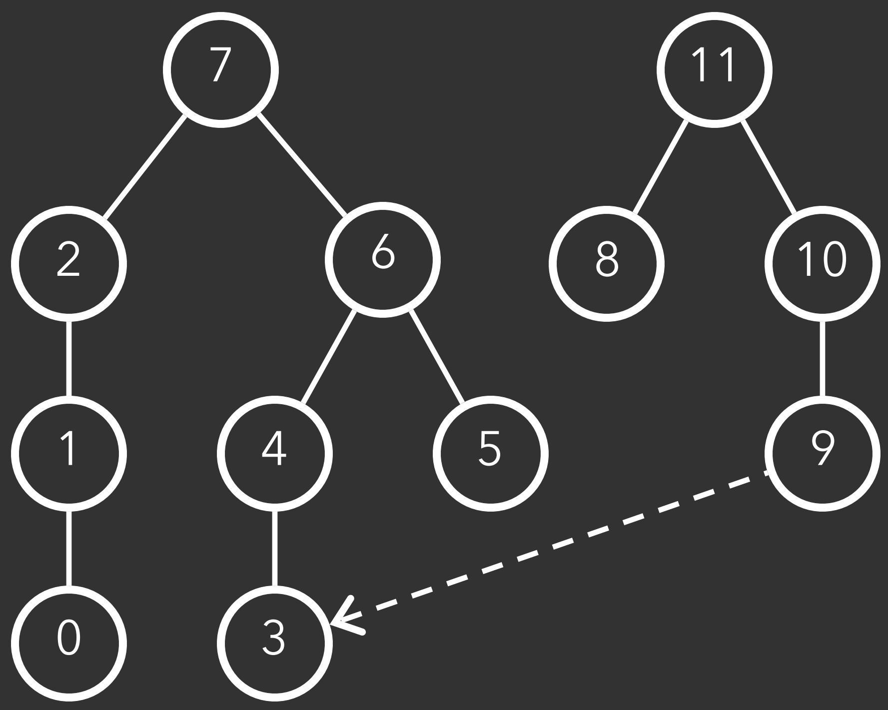
Reingold-Tilford Layout



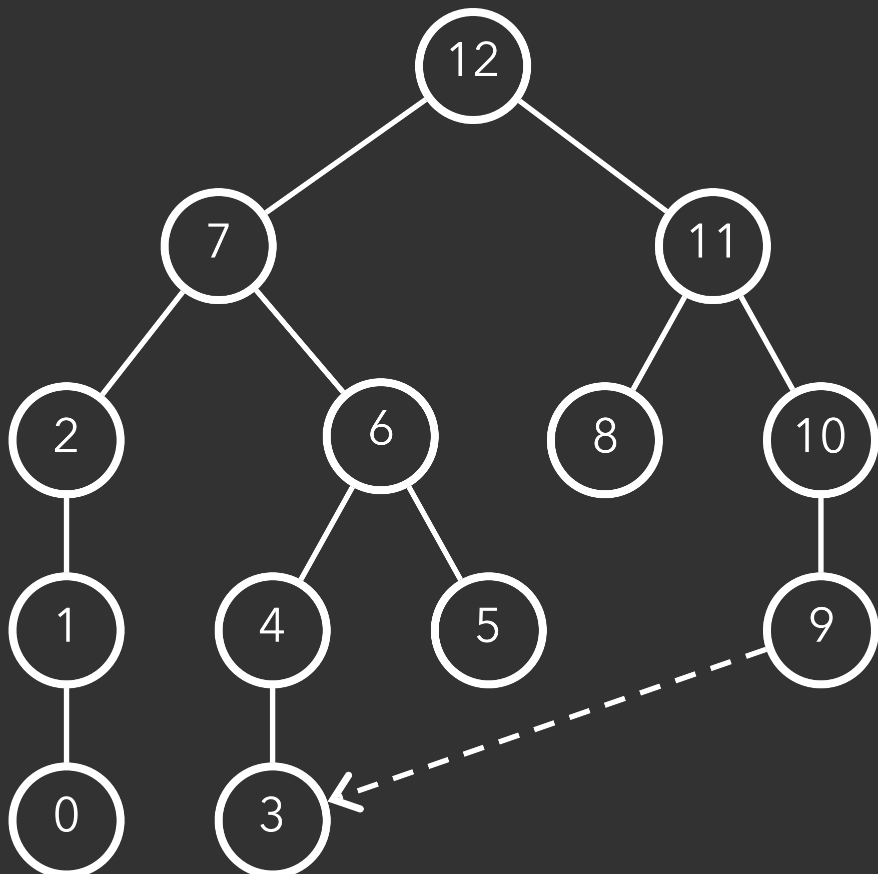
Reingold-Tilford Layout



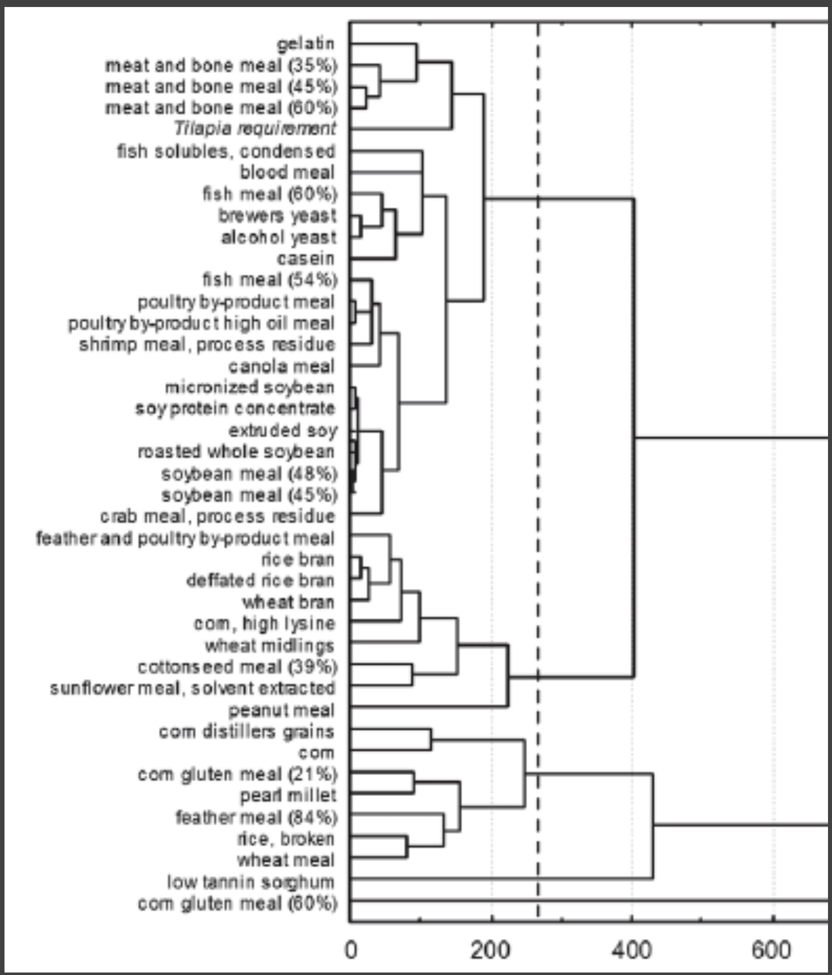
Reingold-Tilford Layout



Reingold-Tilford Layout



Cluster Dendrograms



Depicts cluster trees produced by hierarchical clustering algorithms.

Leaf nodes arranged in a line, internal node depth indicates order/value at which clusters merge.

Naïve recursive layout with orthogonal two-segment edges.

Radial Tree Layout

Node-link diagram in polar co-ordinates.

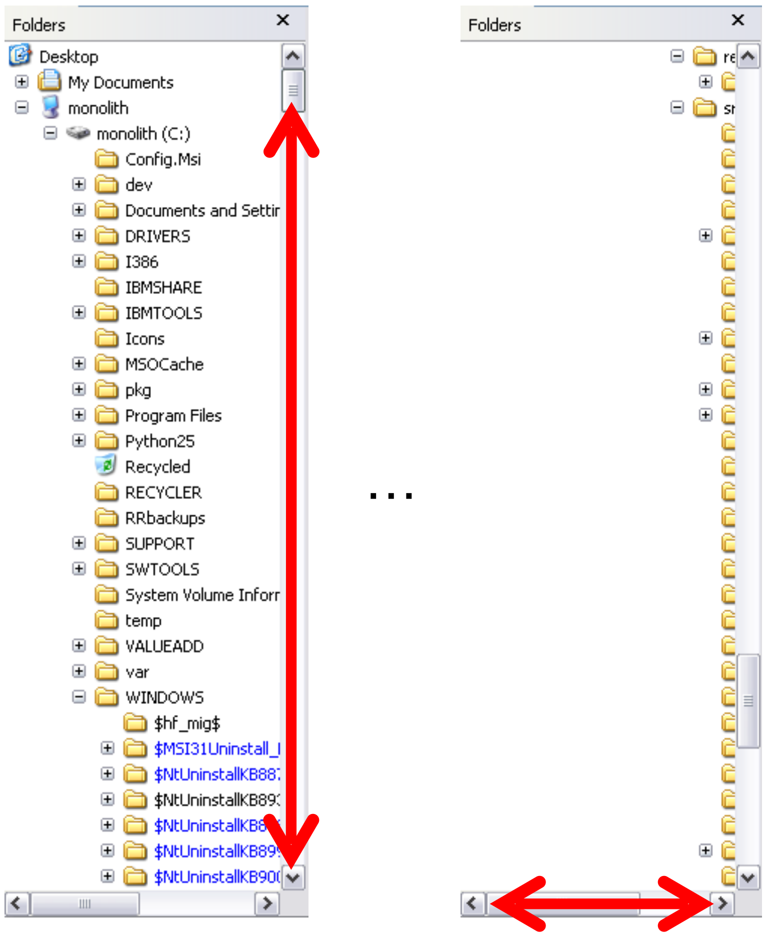
Radius encodes depth, with root in the center.

Angular sectors assigned to subtrees (often with naïve recursive layout).

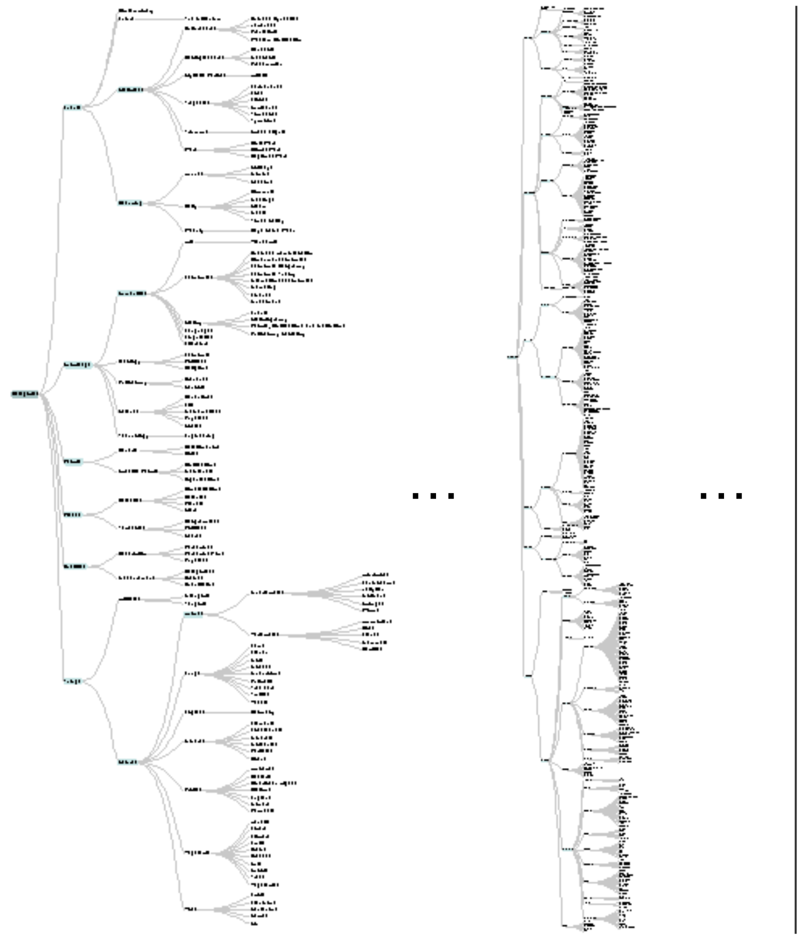
Reingold-Tilford method can also be applied here.

Focus + Context

Visualizing Large Hierarchies



Indented Layout



Reingold-Tilford Layout

More Nodes, More Problems…

Scale

Tree breadth often grows exponentially

Even with tidy layout, quickly run out of space

Possible Solutions

Filtering

Focus+Context

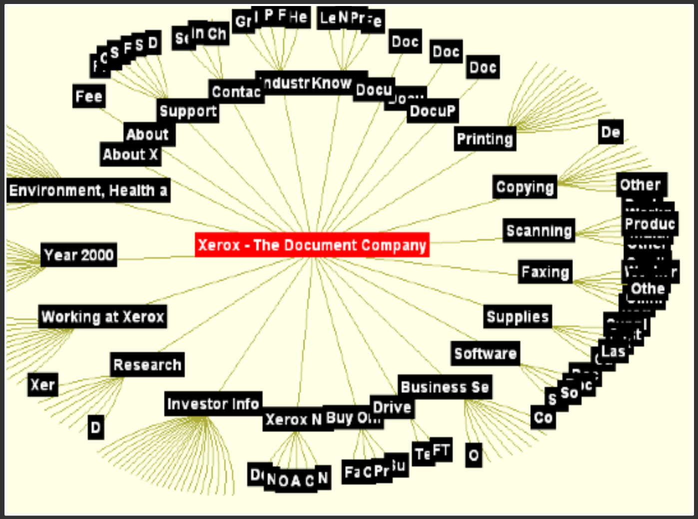
Scrolling or Panning

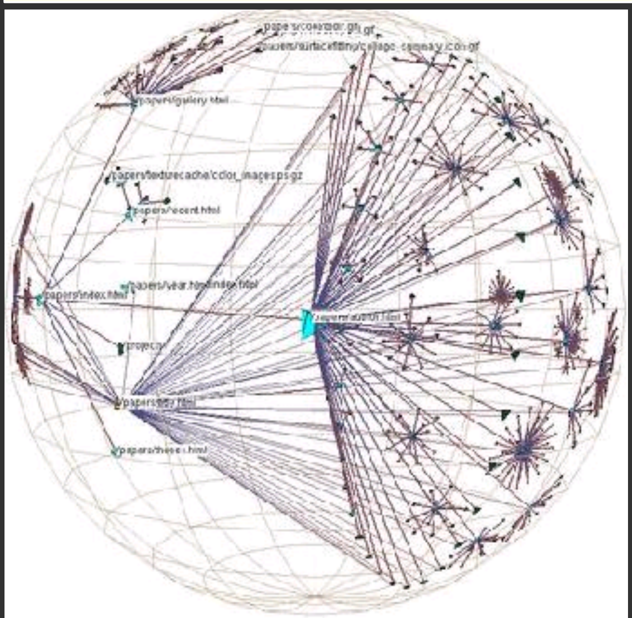
Zooming

Aggregation

MC Escher, *Circle Limit IV*

Hyperbolic Layout



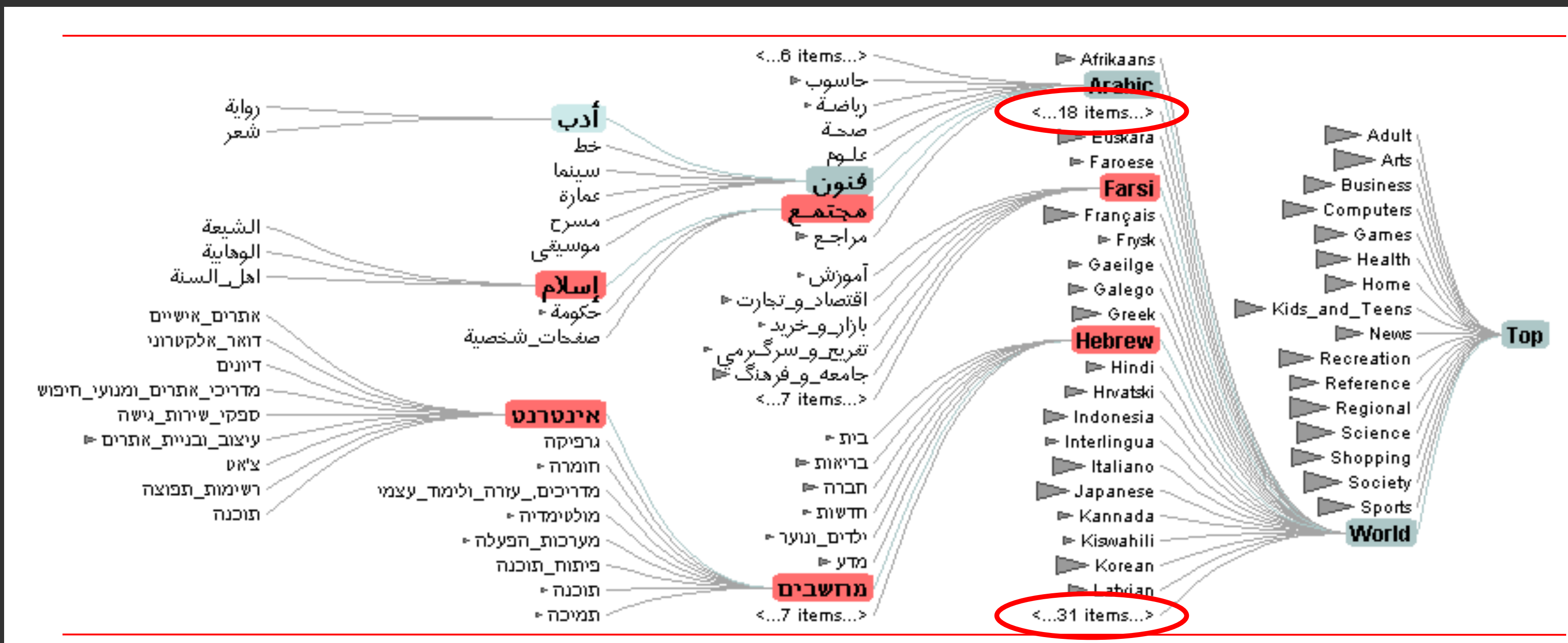


Perform tree layout in hyperbolic geometry, project the result on to the Euclidean plane.

Why? Like tree breadth, the hyperbolic plane expands exponentially!

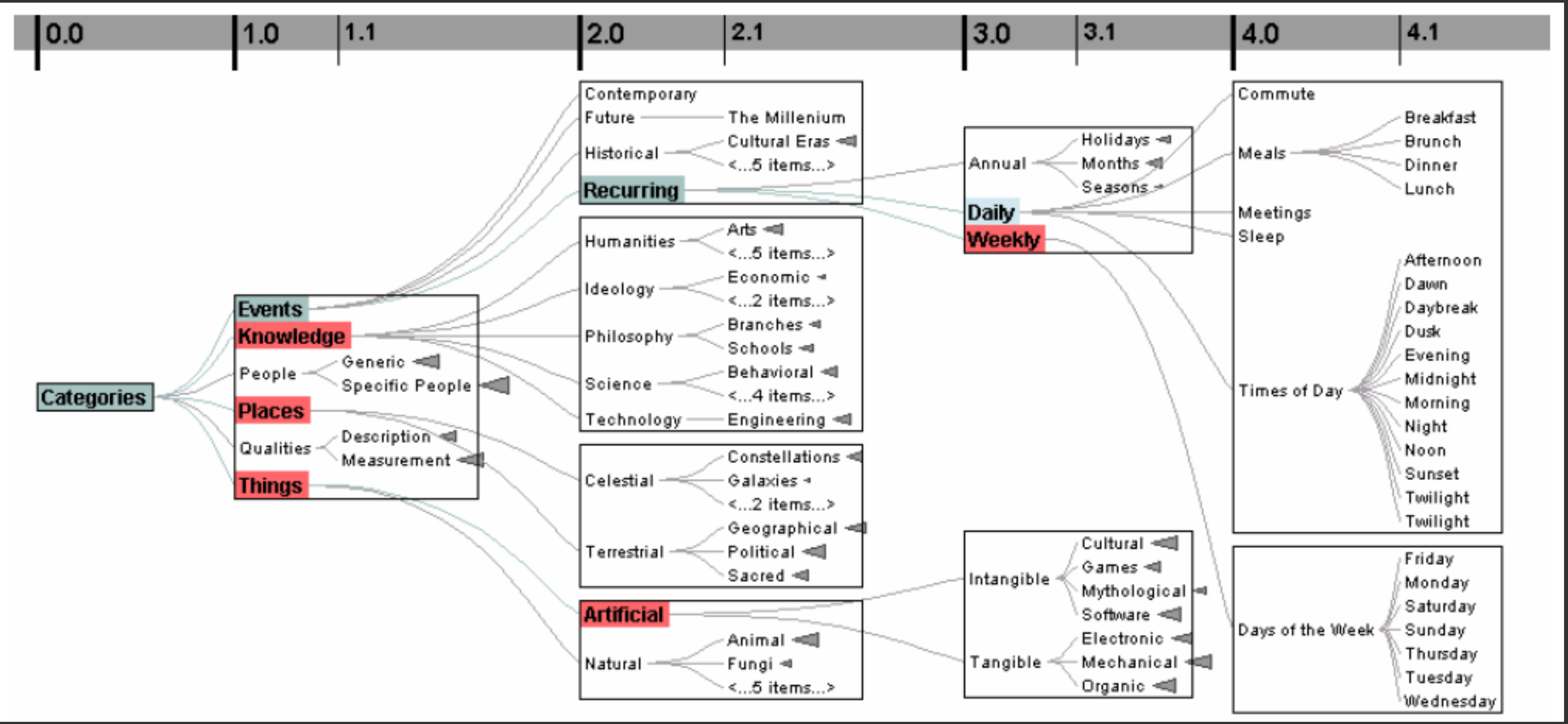
Also computable in 3D, projected into a sphere.

Degree-of-Interest Trees



Space-constrained, multi-focal tree layout

Degree-of-Interest Trees



Remove “low interest” nodes at a given depth level until all blocks on a level fit within bounds.

Attempt to center child blocks beneath parents.

Enclosure

Enclosure Diagrams

Encode structure using **spatial enclosure**
Popularly known as **treemaps**

Benefits
Provides a single view of an entire tree
Easier to spot large/small nodes

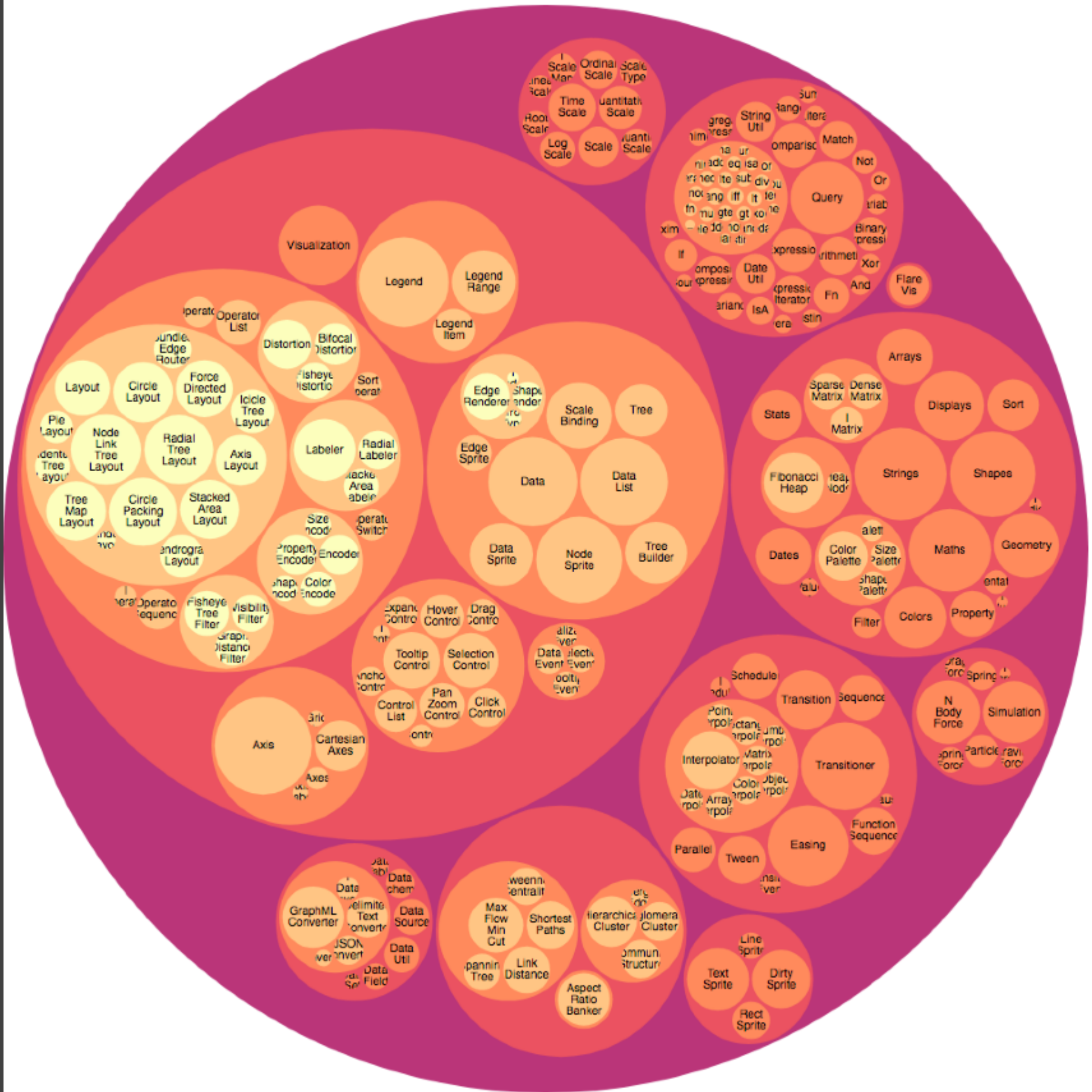
Problems
Difficult to accurately read structure / depth

Circle Packing Layout

Nodes are represented as sized circles.

Nesting shows parent-child relationships.

Issues?
Inefficient use of space.
Parent size misleading?

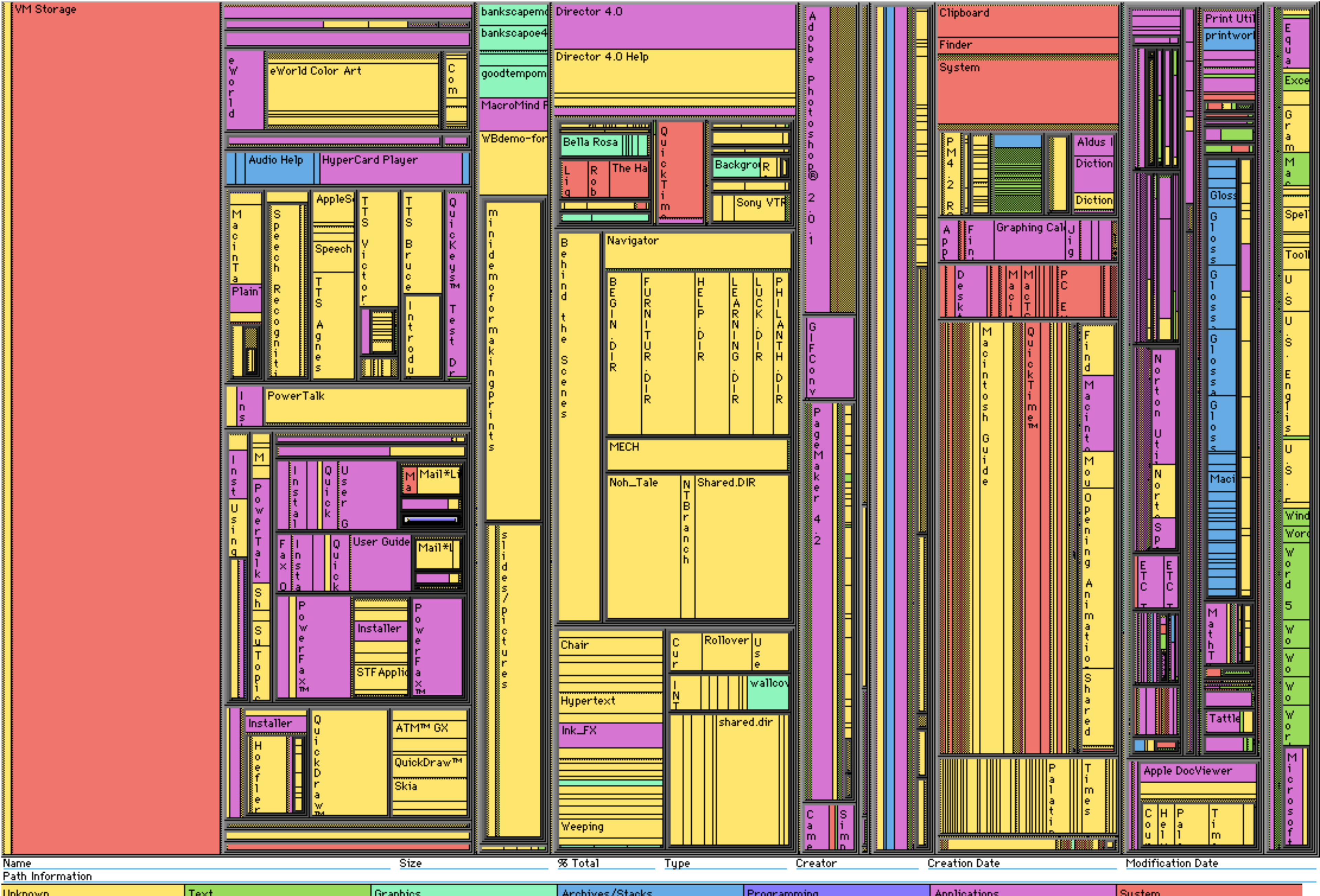


Treemaps

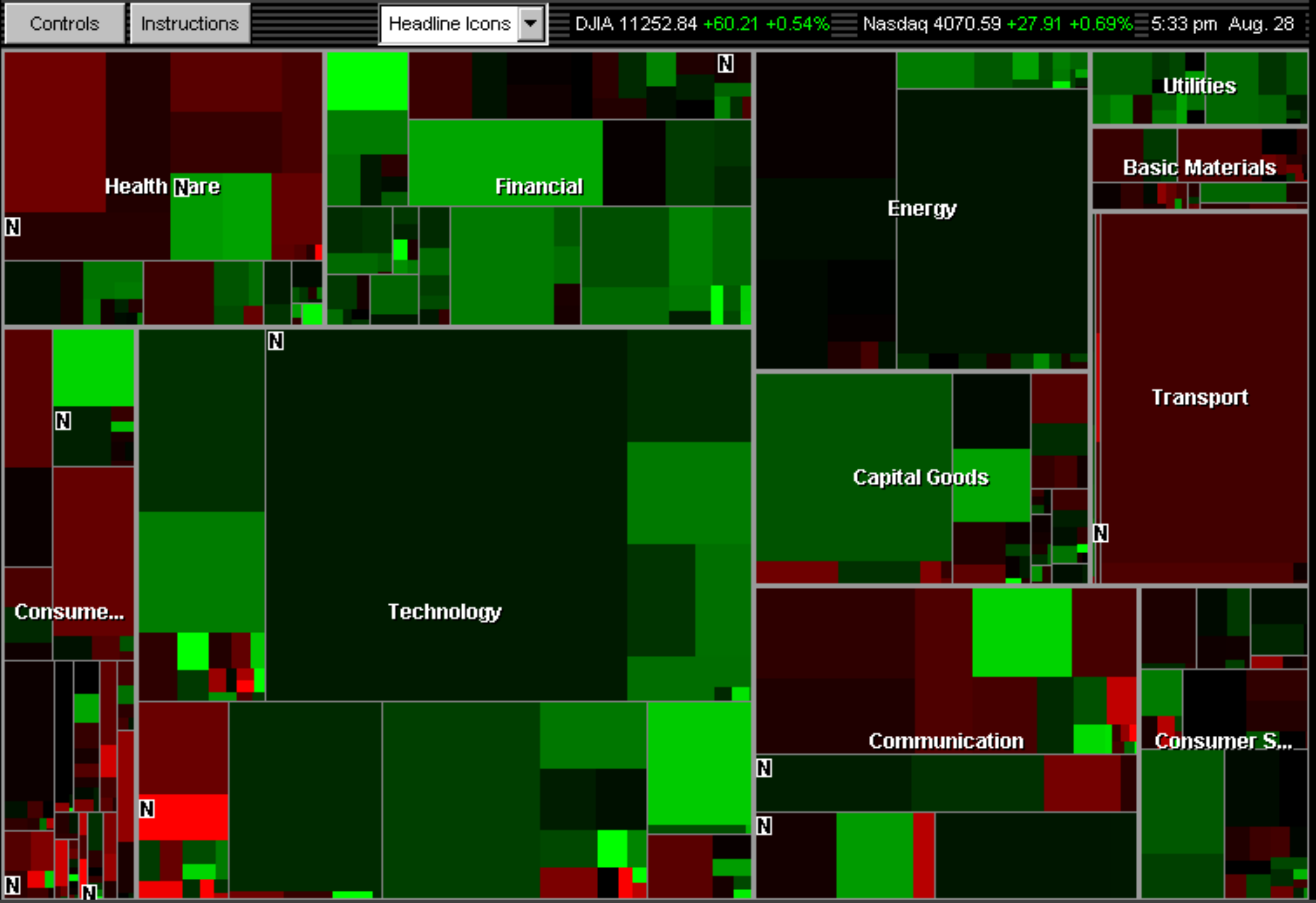
Hierarchy visualization that emphasizes values of nodes via area encoding.

Partition 2D space such that leaf nodes have sizes proportional to data values.

First layout algorithms proposed by Shneiderman et al. in 1990, with focus on showing file sizes on a hard drive.



Slice & Dice layout: Alternate horizontal / vertical partitions.

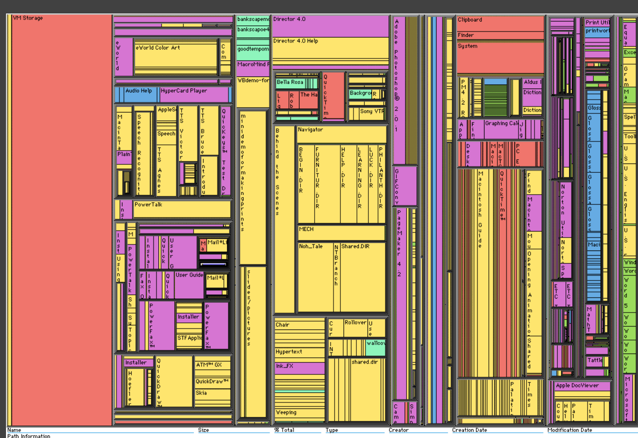


Squarifed layout: Try to produce square (1:1) aspect ratios

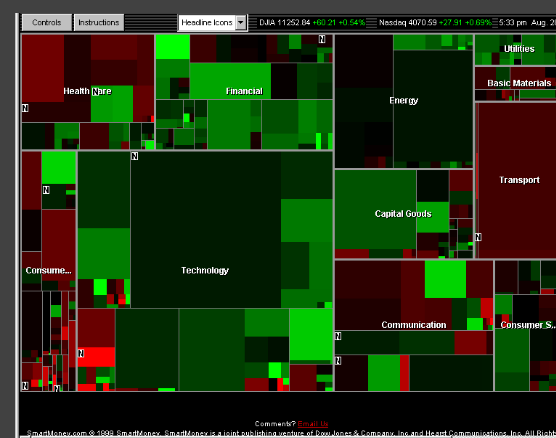
Squarified Treemaps [Bruls et al. ’00]

Slice & Dice layout suffers from extreme aspect ratios. How might we do better?

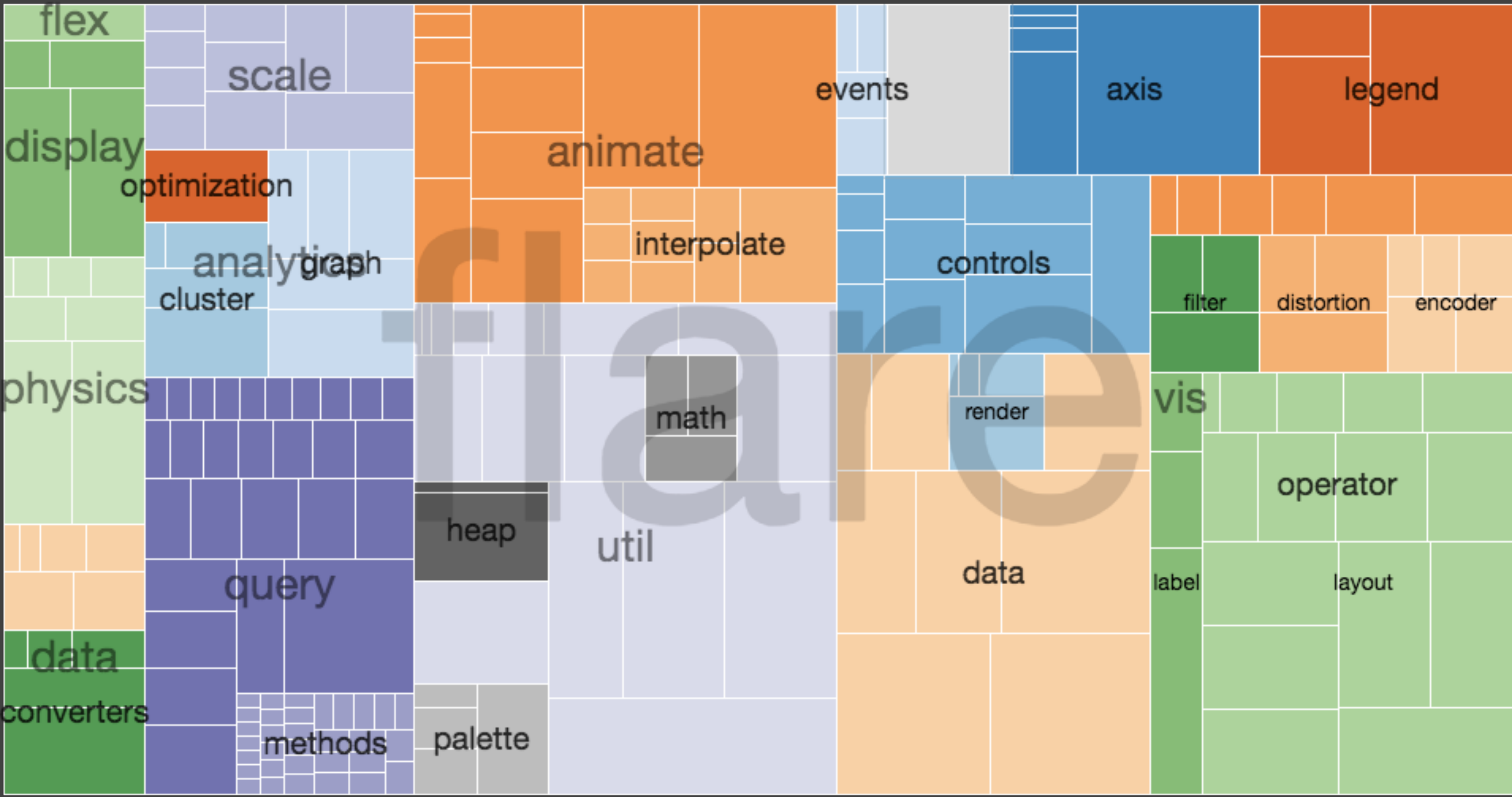
Squarified layout: greedy optimization for objective of square rectangles. Slice/dice within siblings; alternate whenever ratio worsens.



vs.



Interactive Example…



Why Squares? [Bruls et al. ’00]

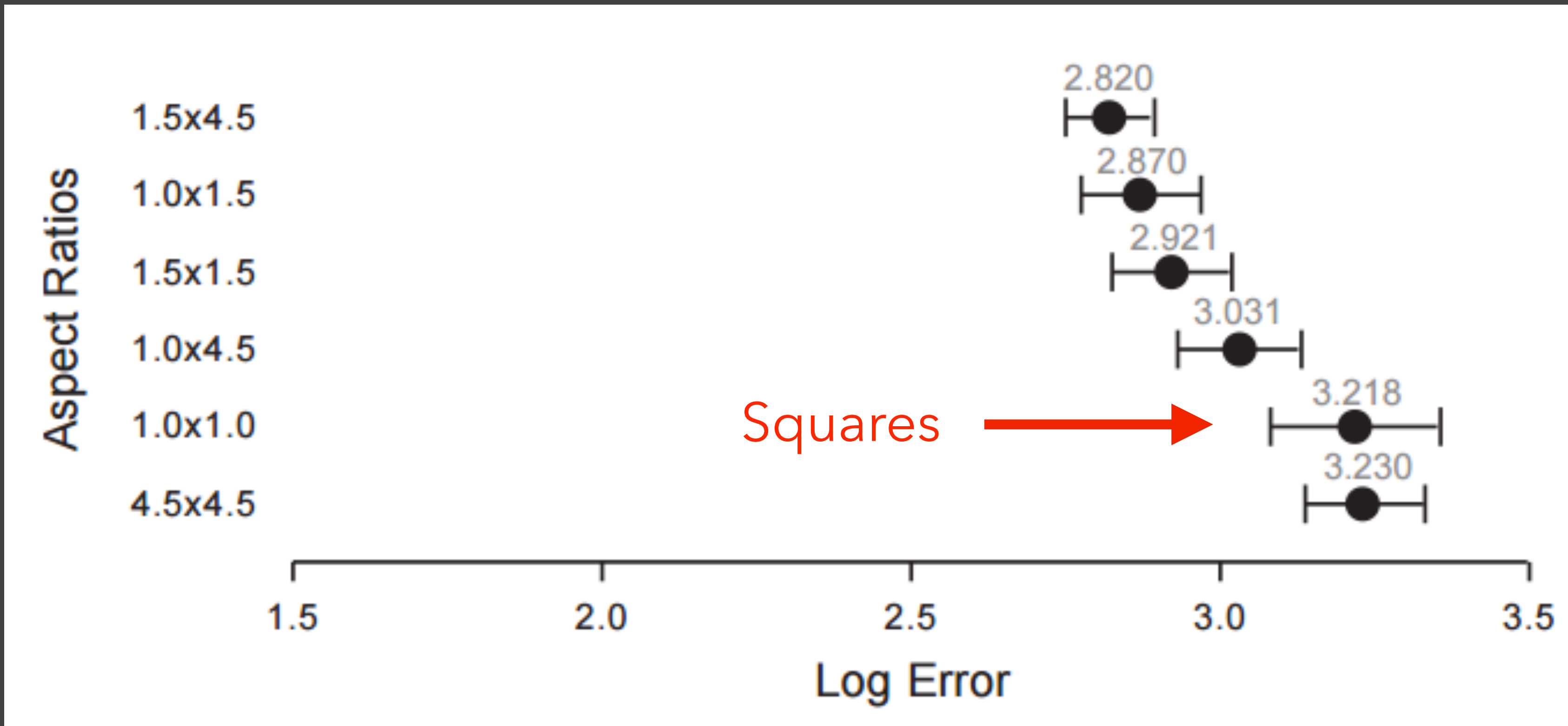
Posited Benefits of 1:1 Aspect Ratios

1. Minimize perimeter, reducing border ink.
   Mathematically true!

2. Easier to select with a mouse cursor.
   Validated by empirical research & Fitt’s Law!

3. Similar aspect ratios are easier to compare.
   Seems intuitive, but is this true?

Comparison Error vs. Aspect Ratio



Study by Kong, Heer & Agrawala, InfoVis '10.
Comparison of squares has higher error!
"Squarify" works because it fails to meet its objective?

Why Squares? [Bruls et al. ’00]

Posited Benefits of 1:1 Aspect Ratios

1. Minimize perimeter, reducing border ink.
   Mathematically true!

2. Easier to select with a mouse cursor.
   Validated by empirical research & Fitt’s Law!

3. Similar aspect ratios are easier to compare.
   Seems intuitive, but is this true?

Why Squares? [Bruls et al. ’00]

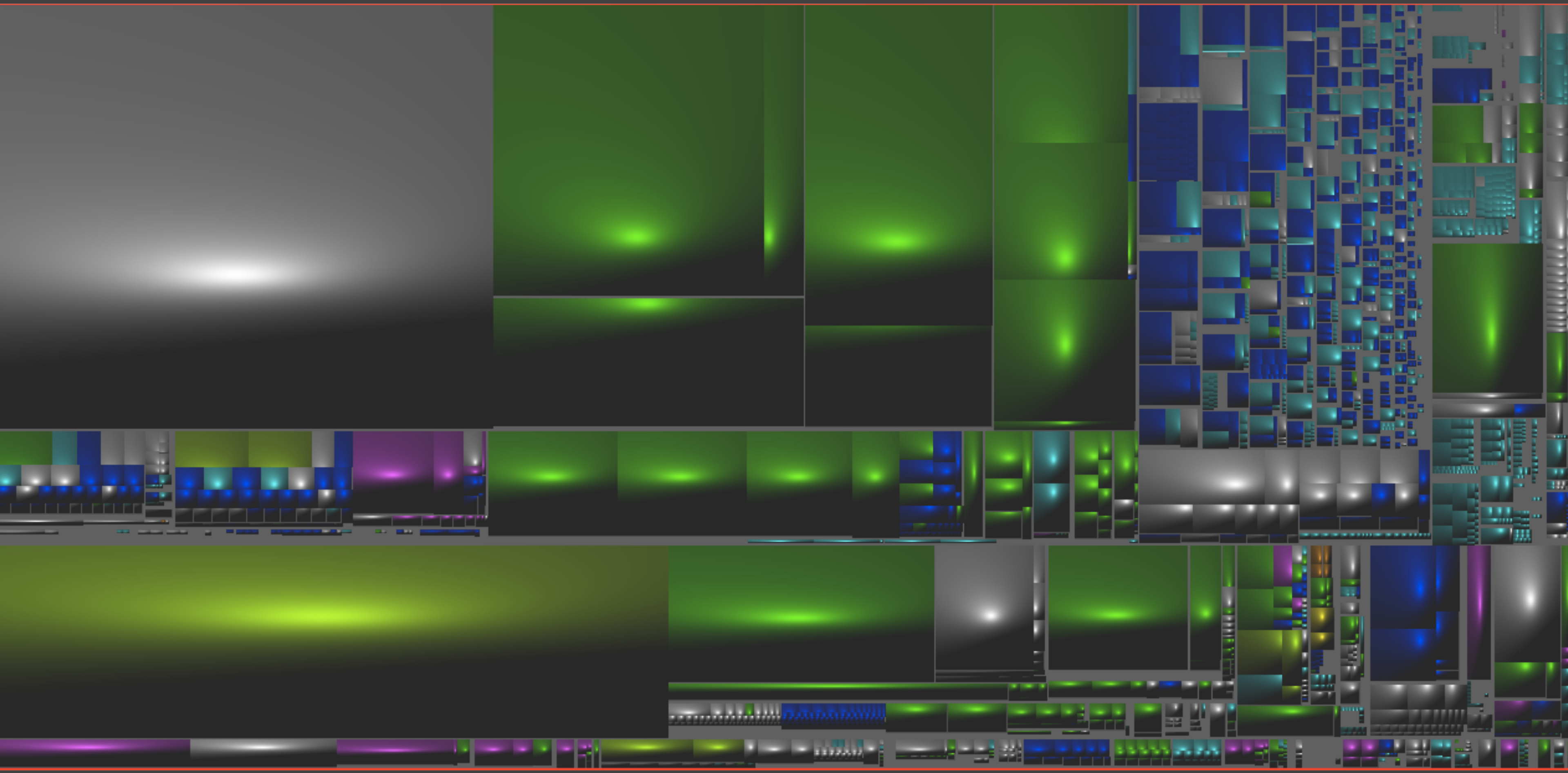
Posited Benefits of 1:1 Aspect Ratios

1. Minimize perimeter, reducing border ink.
   Mathematically true!

2. Easier to select with a mouse cursor.
   Validated by empirical research & Fitt’s Law!

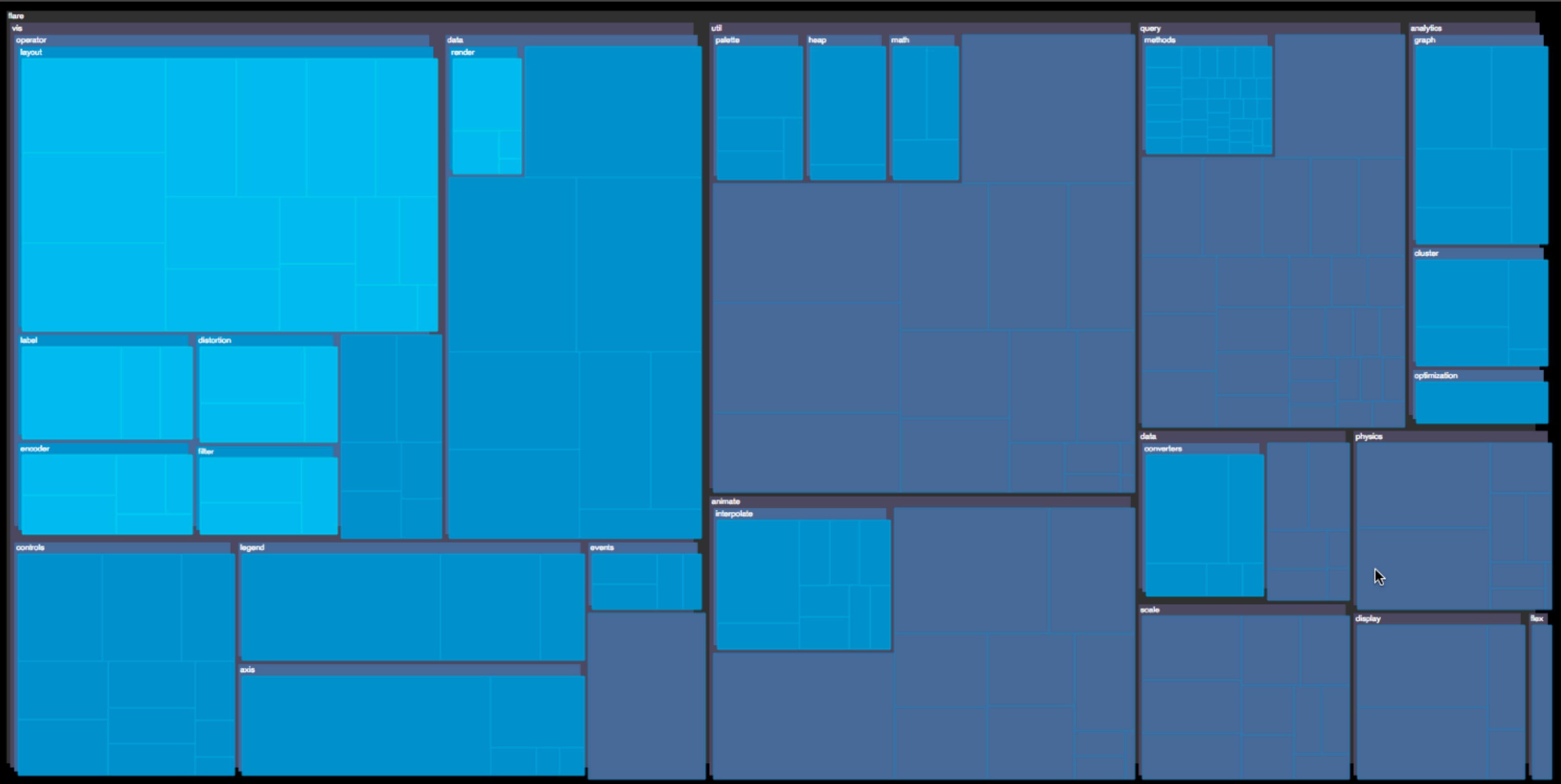
3. ~~Similar aspect ratios are easier to compare.~~
   Extreme ratios & squares-only more inaccurate.
   Balanced ratios better? Target golden ratio?

Cushion Treemaps [van Wijk & Wetering '99]



Uses shading to emphasize hierarchal structure.

Cascaded Treemaps [Lü & Fogarty ’08]

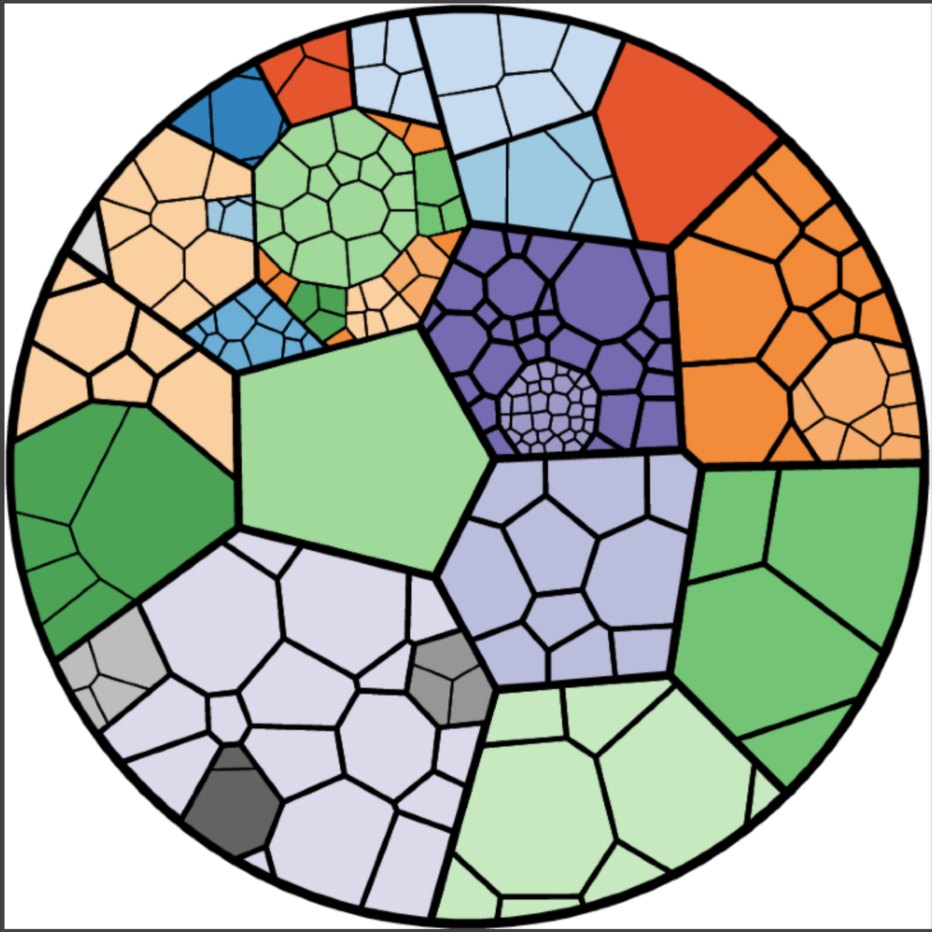


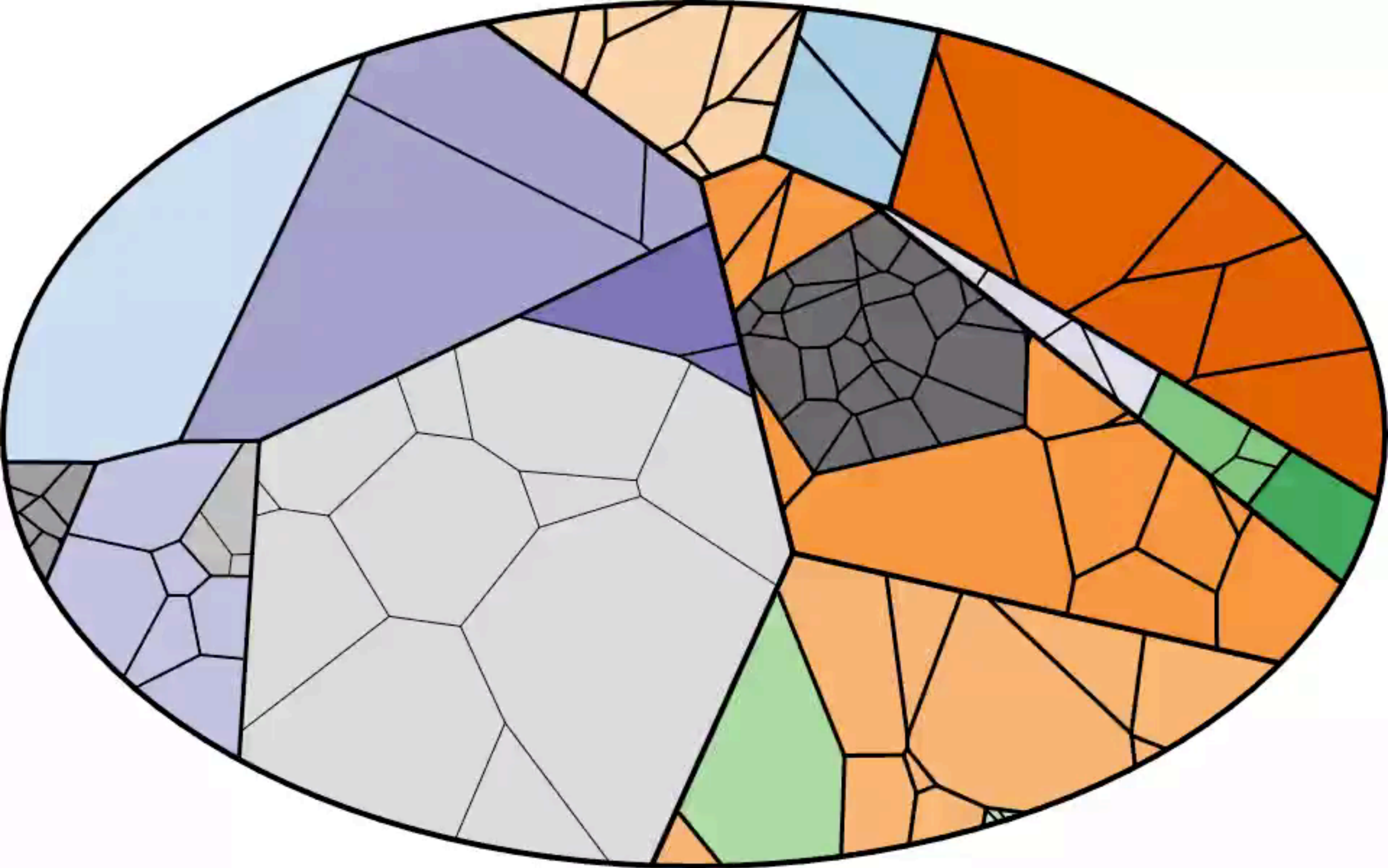
Uses 2.5D effect to emphasize hierarchy relations.

Voronoi Treemaps [Balzer et al. '05]

Instead of rectangles, create treemaps with arbitrary polygonal shapes and boundary.

Use iterative, weighted Voronoi tessellations to achieve cells with value-proportional areas.

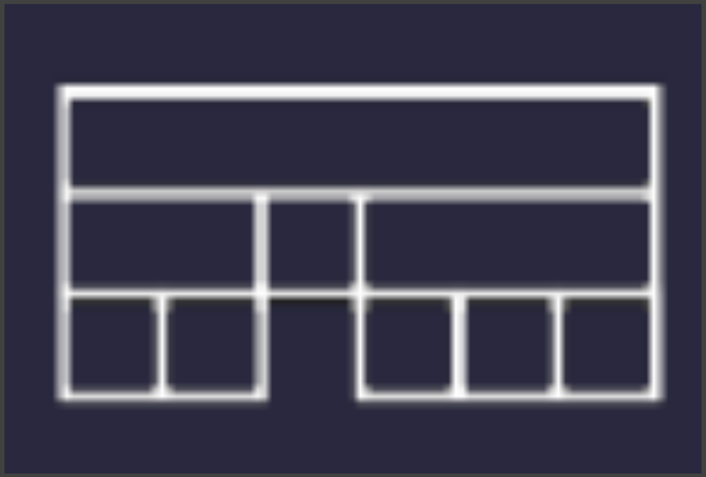




Iterative Voronoi Tesselations [Jason Davies]

Layering

Layered Diagrams

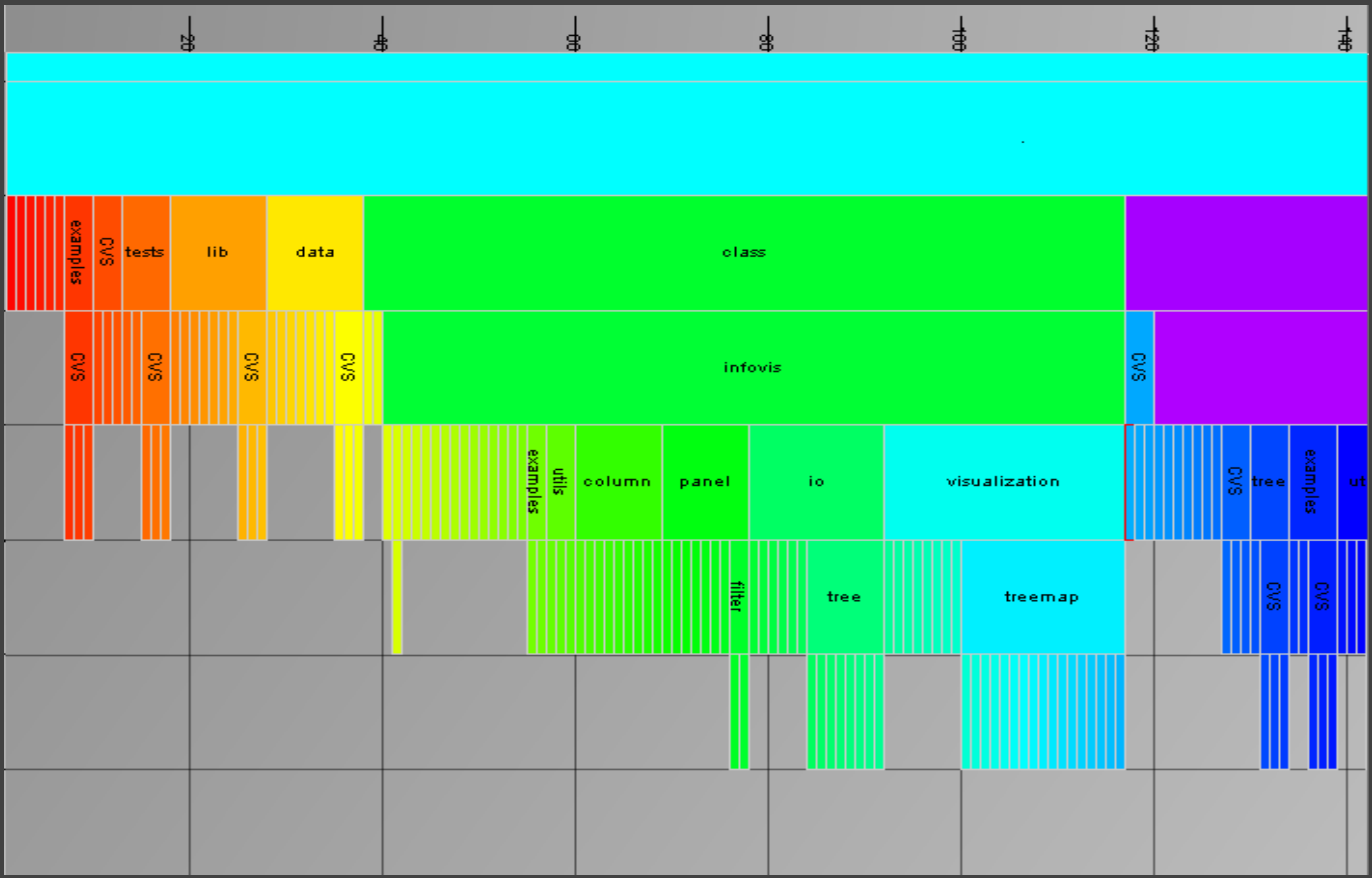


Signify tree structure using:
- Layering
- Adjacency
- Alignment

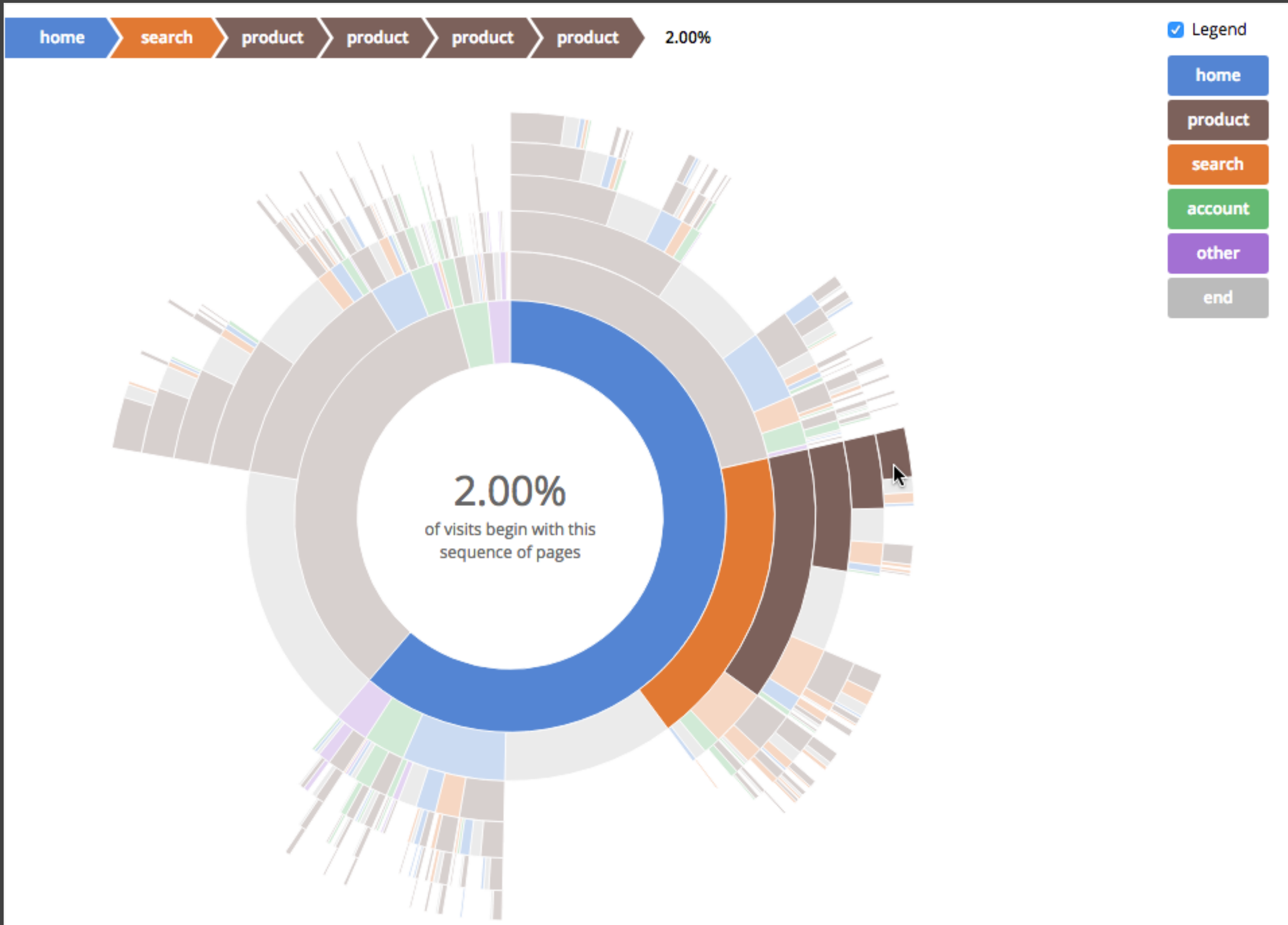
Involves recursive sub-division of space.

Leaf nodes may be sized by value, parent size visualizes sum of descendant leaf values.

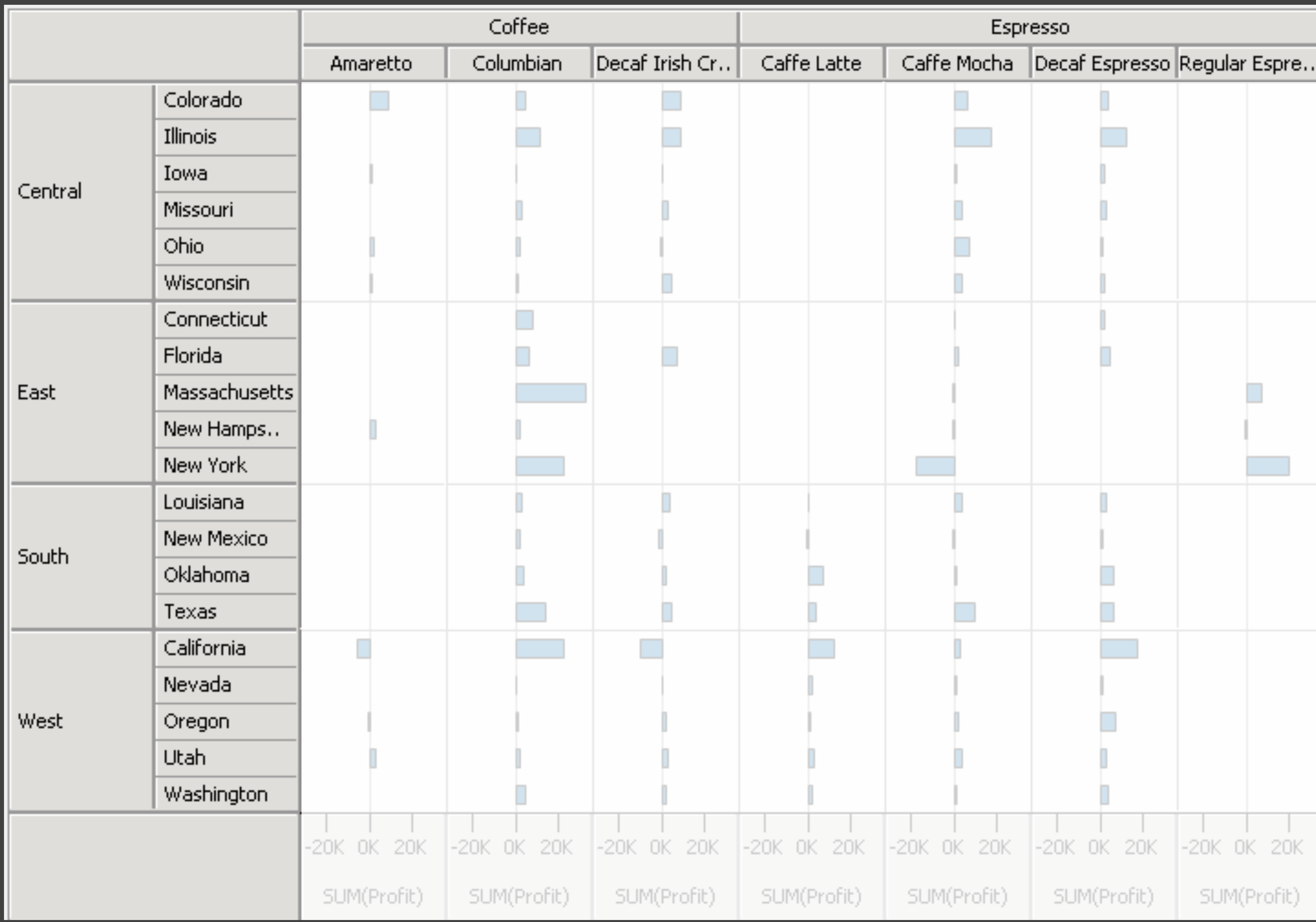
Icicle Trees: Cartesian Partition



"Sunburst" Trees: Polar Partition



Layered Trees Useful Elsewhere…



Node-Link Graph Layout

Spanning Tree Layout

Spanning Tree Layout

Many graphs have useful spanning trees

Websites, Social Networks

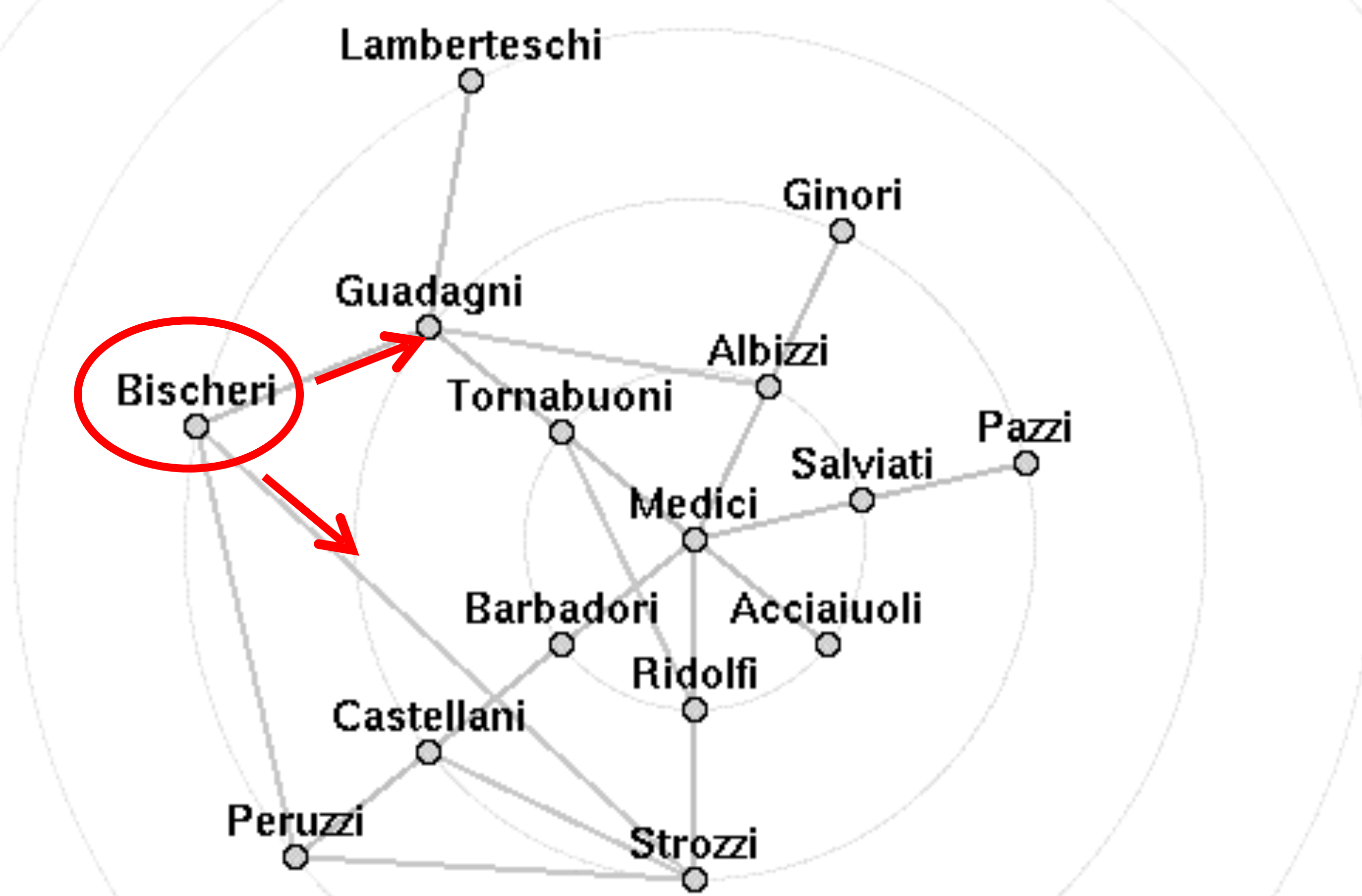
Use tree layout on spanning tree of graph

Trees created by BFS / DFS

Min/max spanning trees

Fast tree layouts allow graph layouts to be recalculated at interactive rates

Heuristics may further improve layout



Spanning tree layout may result in arbitrary parent node!

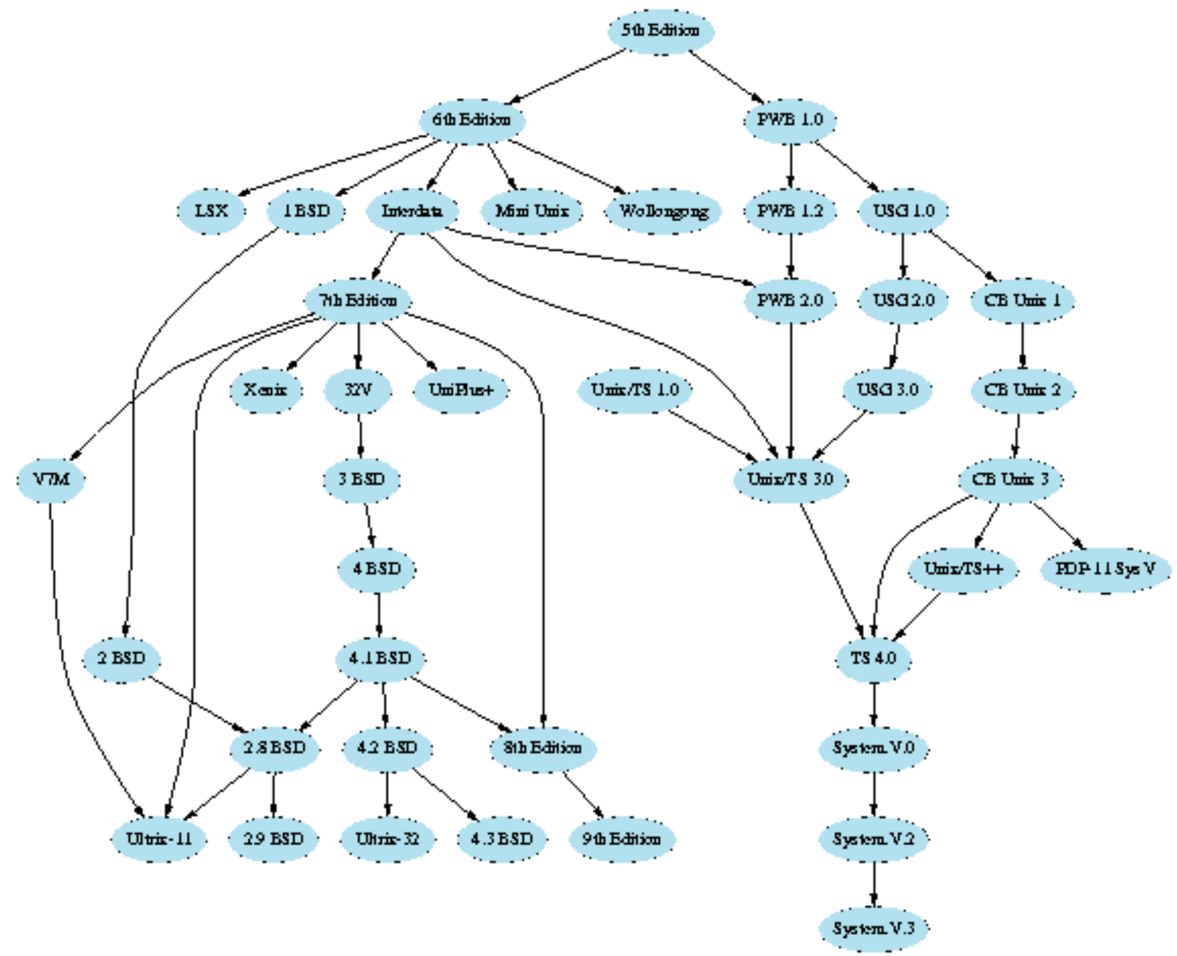
Sugiyama-Style Layout

Sugiyama-Style Layout

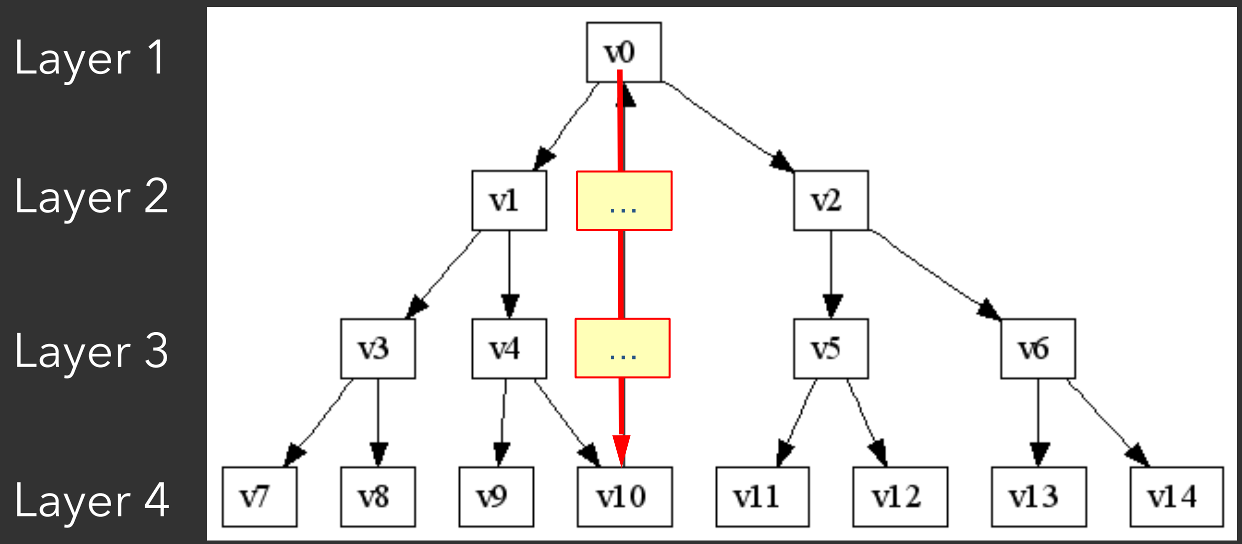
Evolution of the UNIX operating system

Hierarchical layering based on descent

GraphViz package!

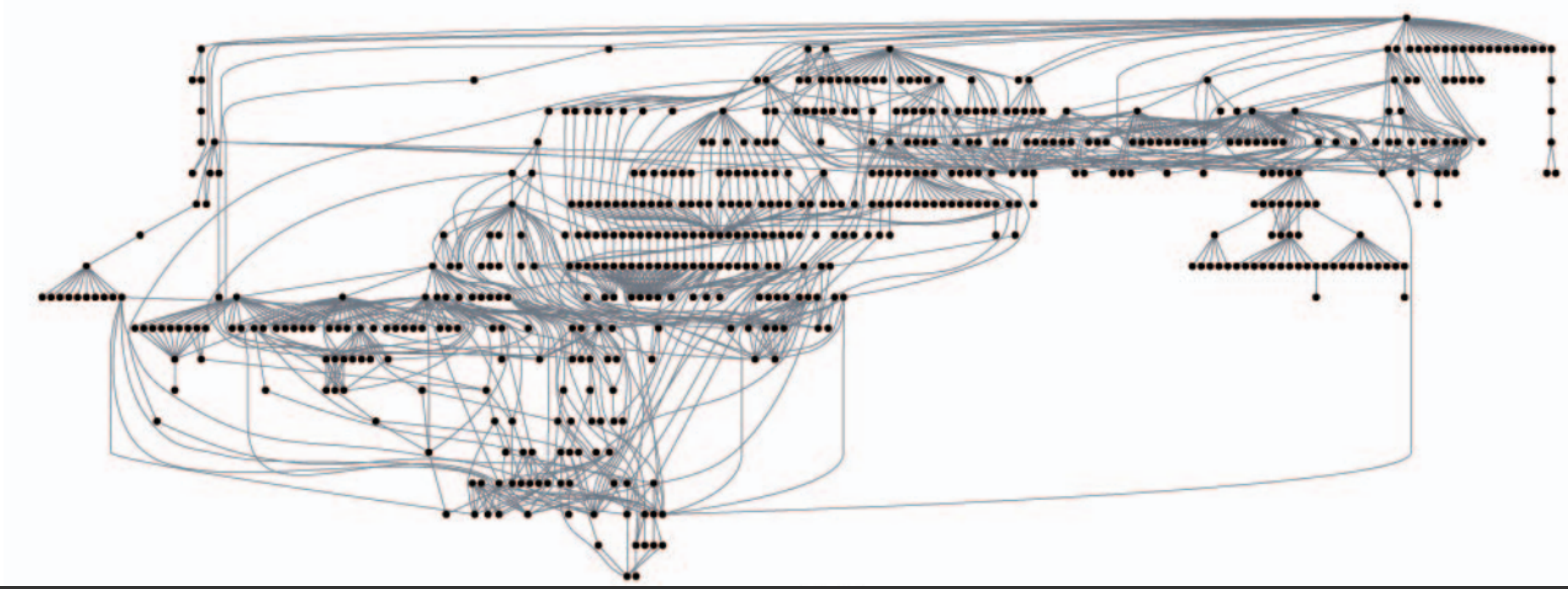


Sugiyama-Style Layout



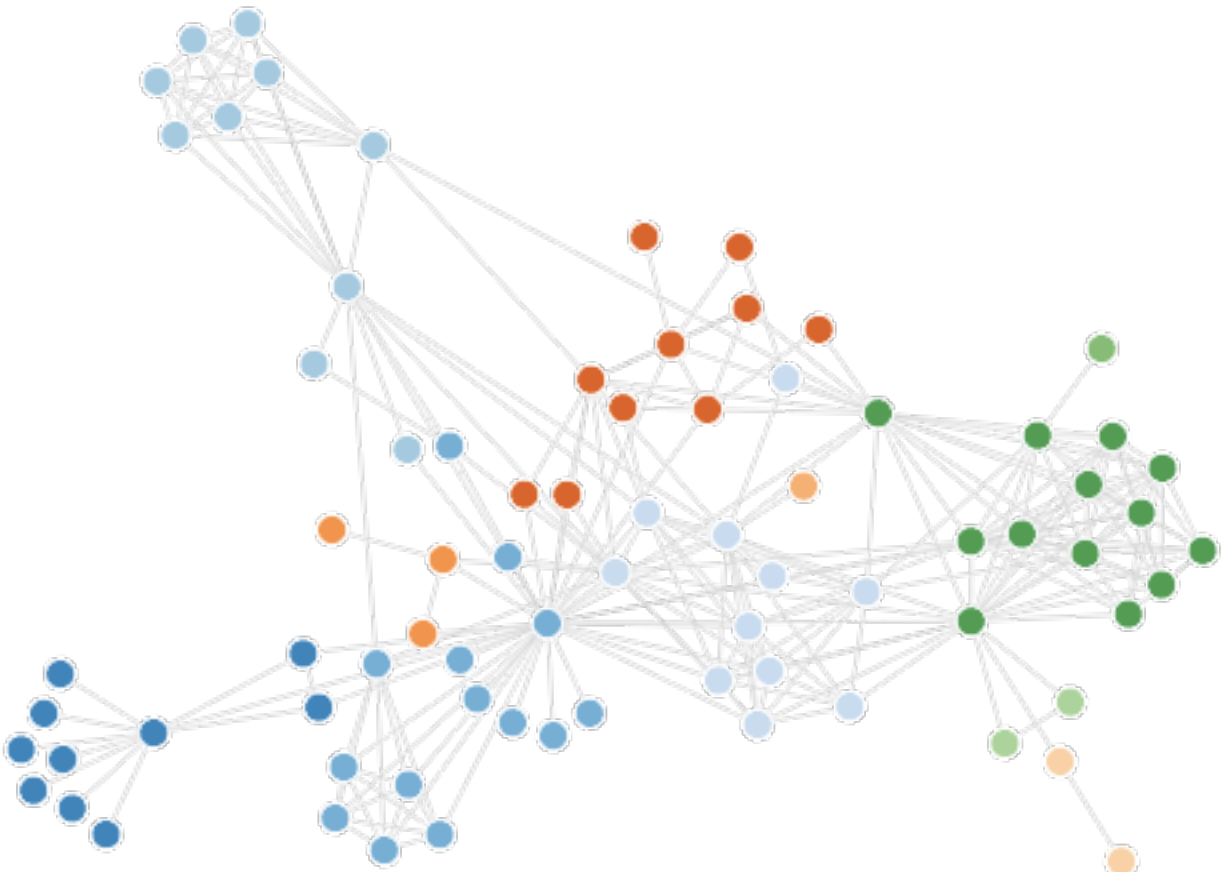
Reverse edges to remove cycles
Assign nodes to hierarchy layers
Create dummy nodes to “fill in” missing layers
Arrange nodes within layer, minimize edge crossings
Route edges – layout splines if needed

Produces Hierarchical Layouts



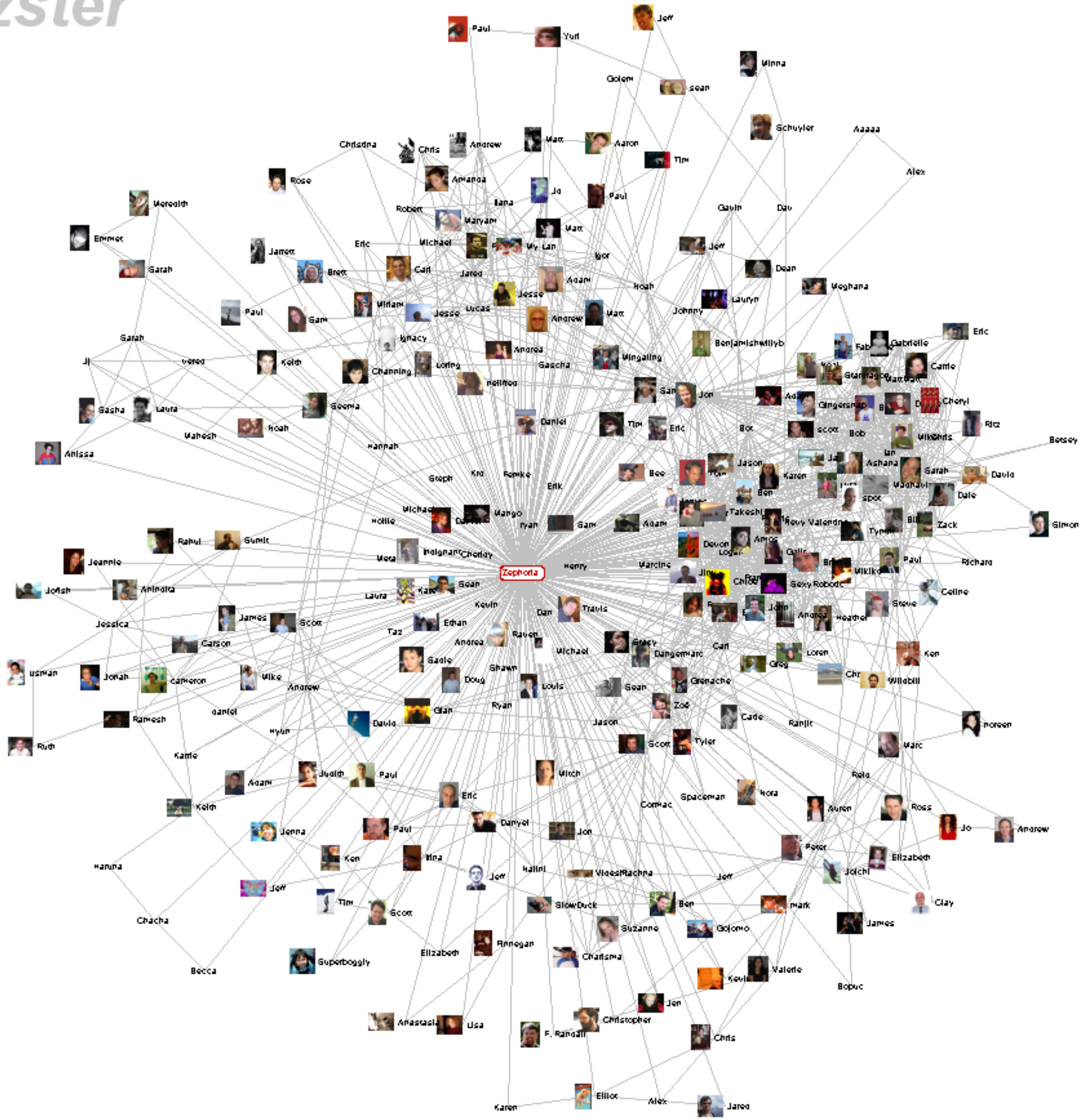
Sugiyama-style layout emphasizes hierarchy.
However, cycles in the graph may mislead.
Long edges can impede perception of proximity.

Force-Directed Layout



Interactive Example: Configurable Force Layout

vizster



community >> Enable

search >>

Zephoria

User ID	21721
Friends	266
Age	??
Gender	Female
Status	Single
Location	San Francisco, CA
Hometown	Lancaster, PA
Occupation	researcher: social networks, identity, context
Interests	apophenia, observing people, culture, questioning power, reading, buddhism, ipseity, computer-mediated communication, social networks, technology, anthropology, stomping
Music	psytrance/goa/trance [Infected Mushroom, Son Kite... Iboga/Digital Structures], Ani Difranco, downtempo, Thievery Corporation, Beth Orton, Morcheeba, Ween, White Stripes
Books	Authors: Erving Goffman, Stanley Milgram, Jeanette Winterson, Eric Schlosser, Leslie Feinberg, Dorothy Allison, Italo Calvino, Hermann Hesse
TV Shows	??
Movies	Koyaanisqatsi, Amelie, Waking Life, Tank Girl, The Matrix, Clockwork Orange, American Beauty, Fight Club, Boys Don't Cry
Member Since	??
Last Login	2003-10-21
Last Updated	2003-10-21
About	[Some know me as danah...] I'm a geek, an activist and an academic, fascinated by people and society. I see life as a very large playground and enjoy exploring its intricacies. I revel in life's chaos, while simultaneously providing my own insane element. My musings: http://www.zephoria.org/thoughts/
Want to Meet	Someone who makes life's complexities seem simply elegant

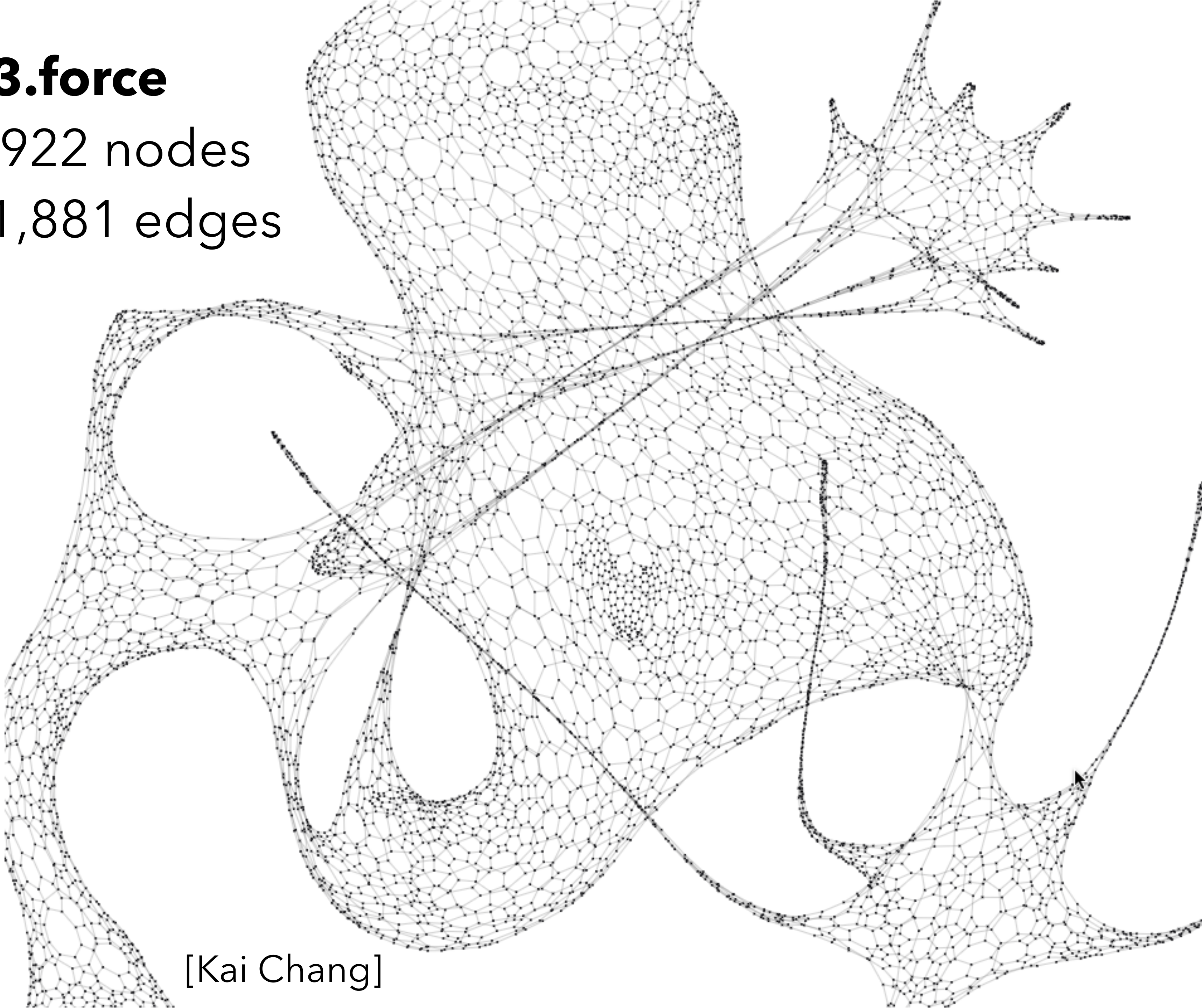
Use the Force!

http://mbostock.github.io/d3/talk/20110921/

d3.force

7,922 nodes

11,881 edges



[Kai Chang]

Force-Directed Layout

Nodes = charged particles $F = q_i * q_j / d_{ij}^2$

with air resistance $F = -b * v_i$

Edges = springs $F = k * (L - d_{ij})$

At each timestep, calculate forces acting on nodes.
Integrate for updated velocities and positions.

D3’s force layout uses **velocity Verlet** integration.
Assume uniform mass ***m*** and timestep ***Δt***:
$F = ma \rightarrow F = a \rightarrow F = \Delta v / \Delta t \rightarrow F = \Delta v$
Forces simplify to velocity offsets!

N-Body Force

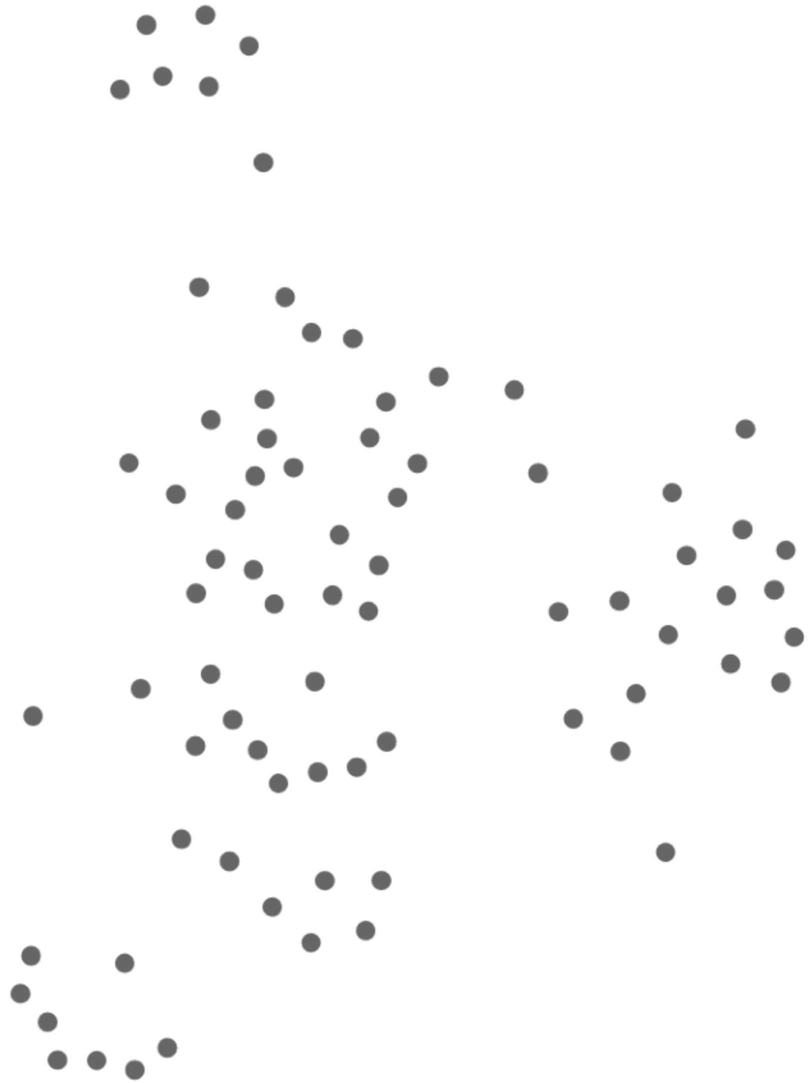
Naïve calculation of repulsive force doesn't scale!

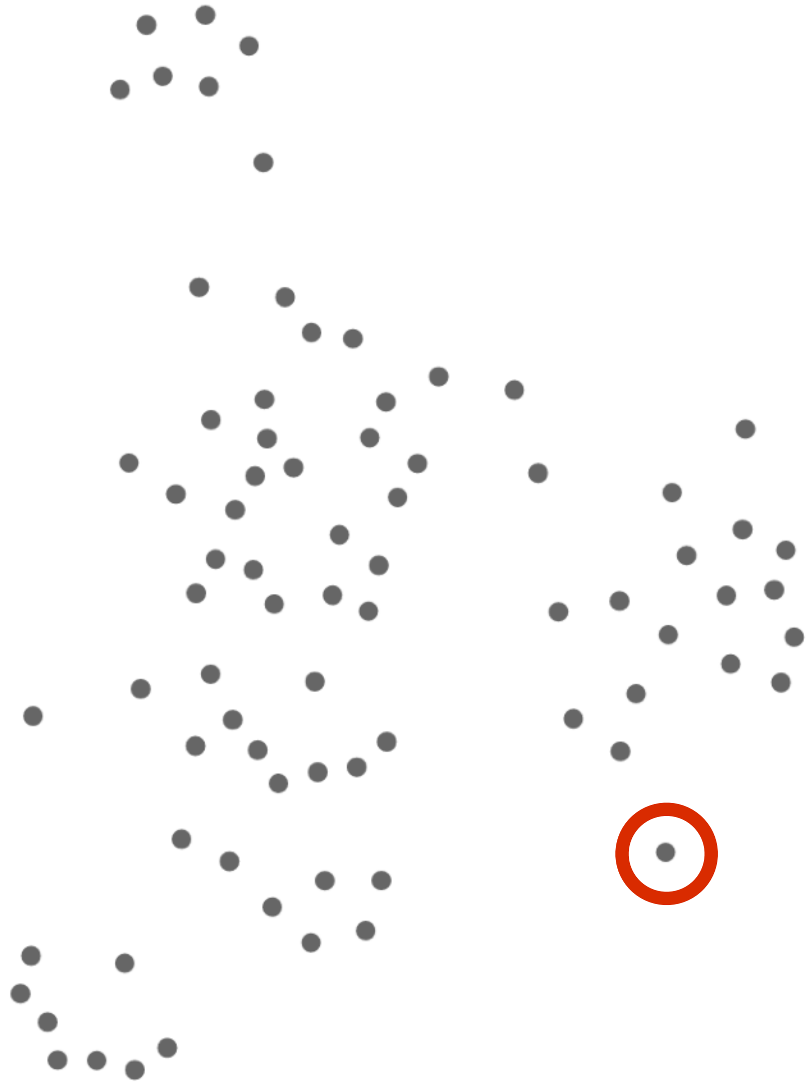
Comparing all pairs of nodes is $O(V^2)$

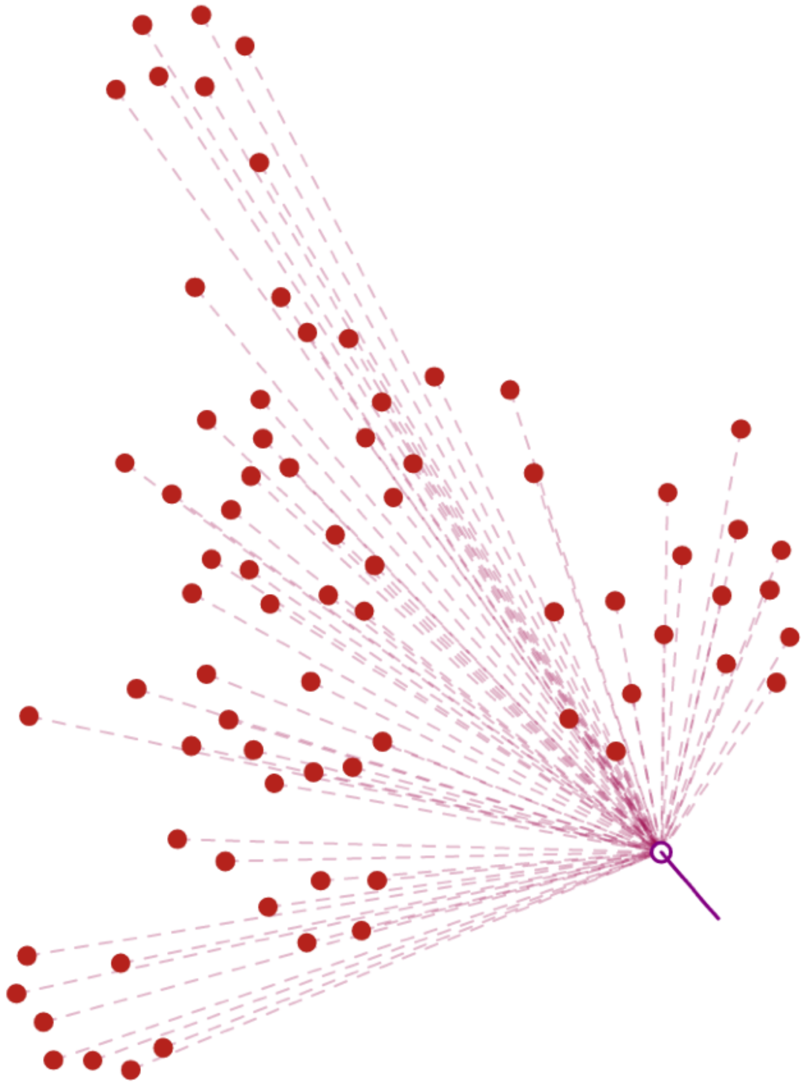
We can approximate force calculations using a spatial index (e.g., quadtree) to achieve O(V log V)

One such approach is the **Barnes-Hut algorithm**, originally created for astronomical simulations.

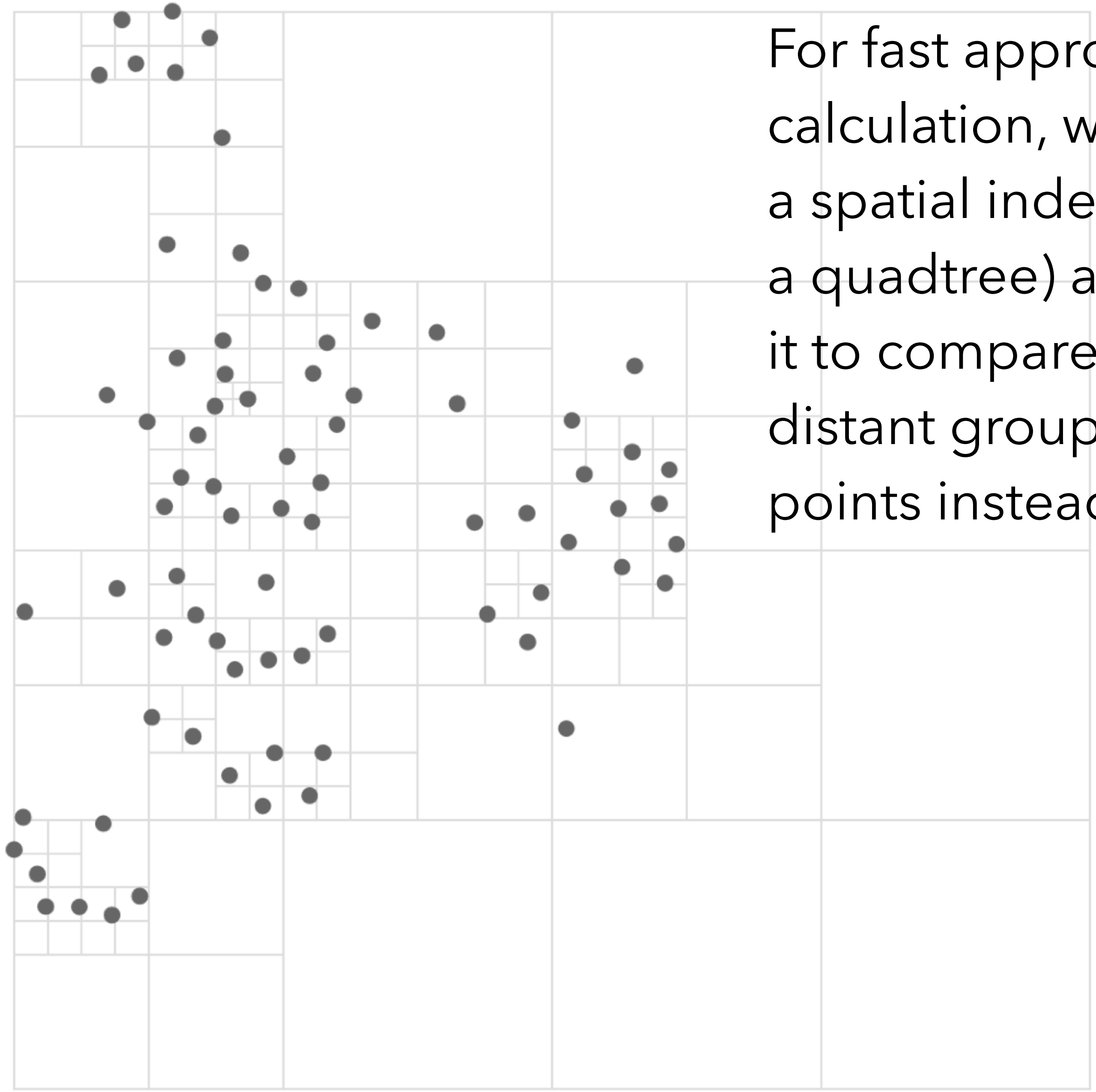
The key idea is to approximate forces from distant nodes by comparing to aggregate centers of charge rather than individual nodes.



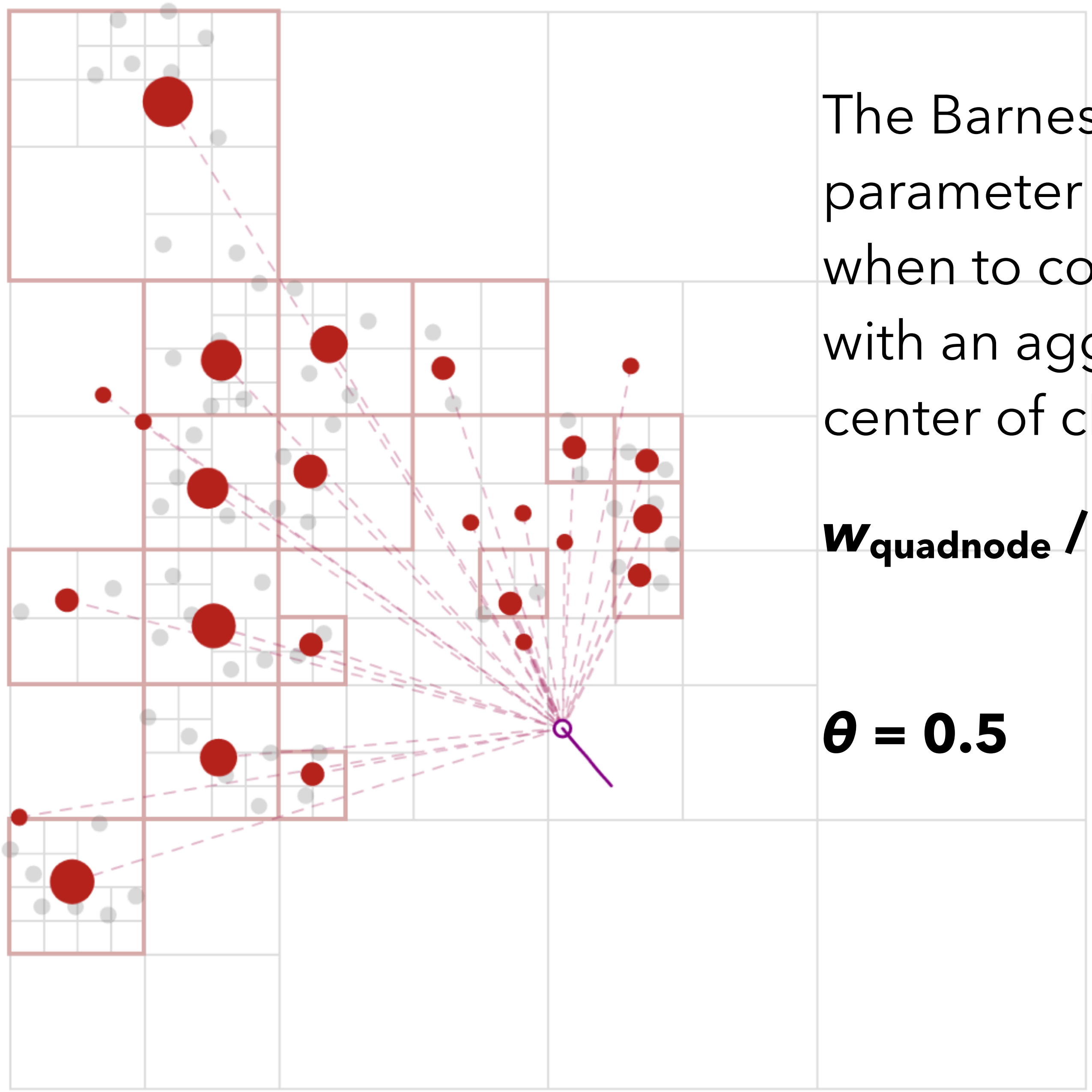




Naive calculation of forces at a point uses sum of forces from all other n-1 points.



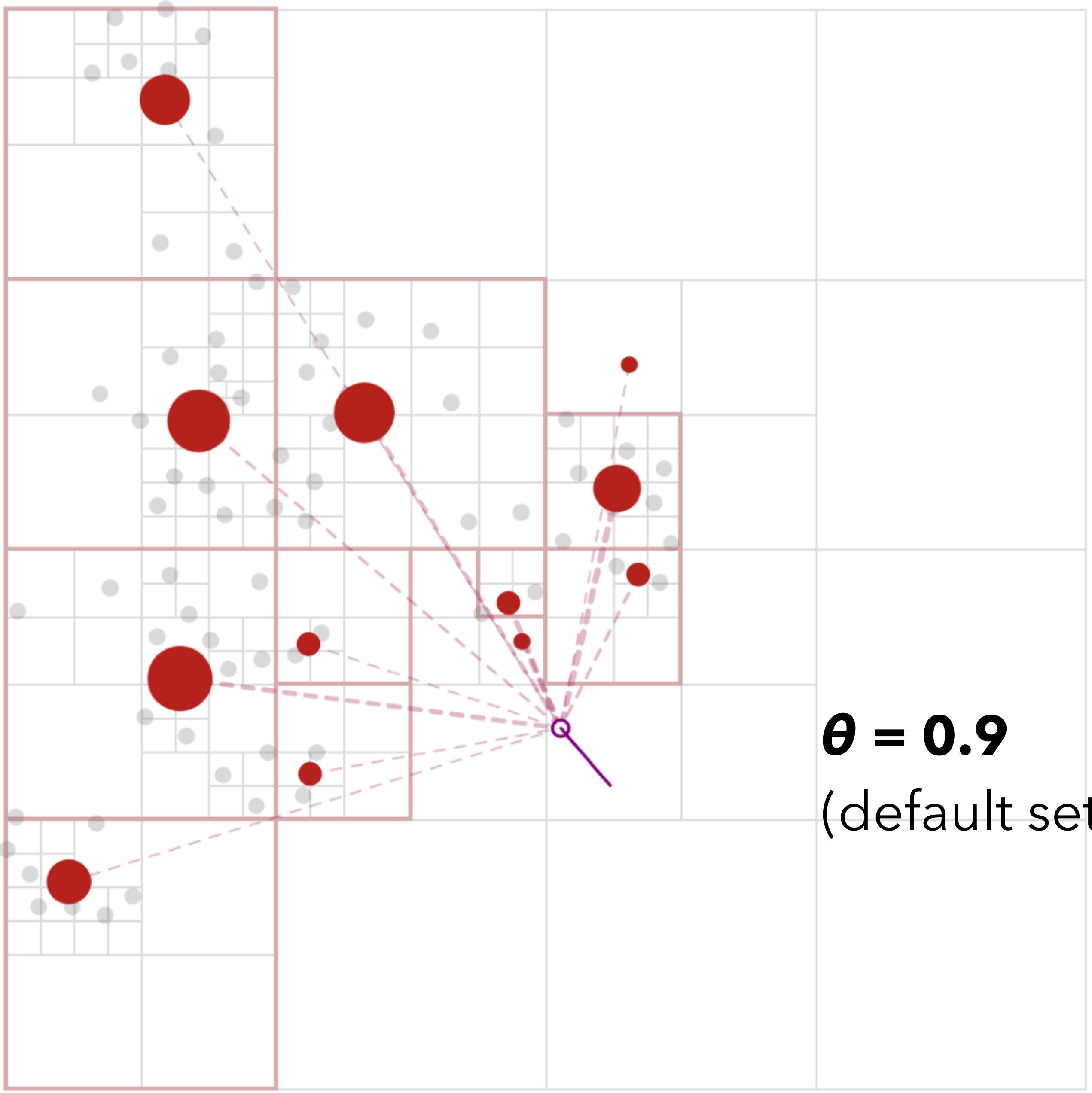
For fast approximate calculation, we build a spatial index (here, a quadtree) and use it to compare with distant groups of points instead.



The Barnes-Hut θ parameter controls when to compare with an aggregate center of charge.

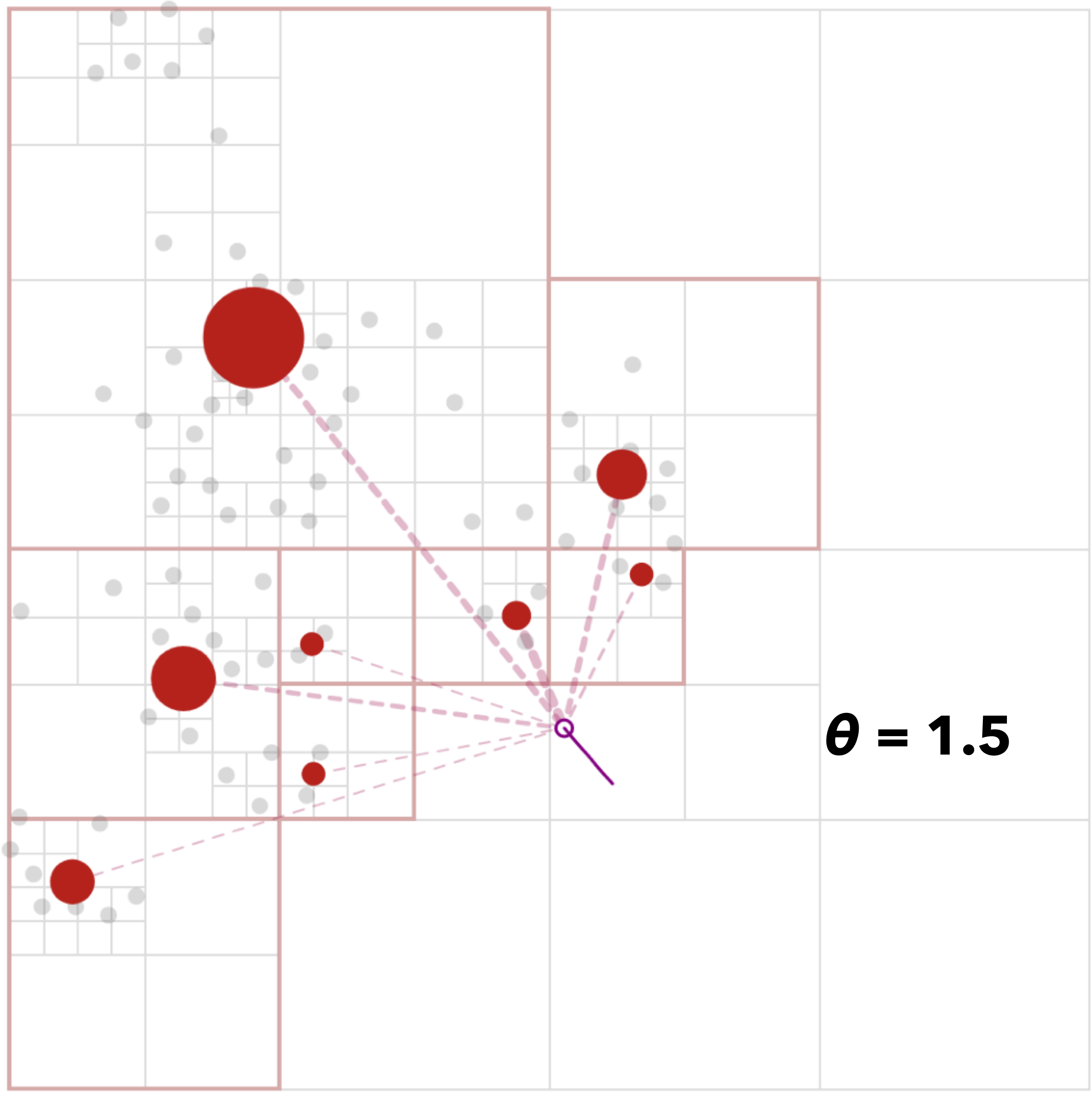
$\boldsymbol{w_{quadnode} / d_{ij} < \theta}$ **?**

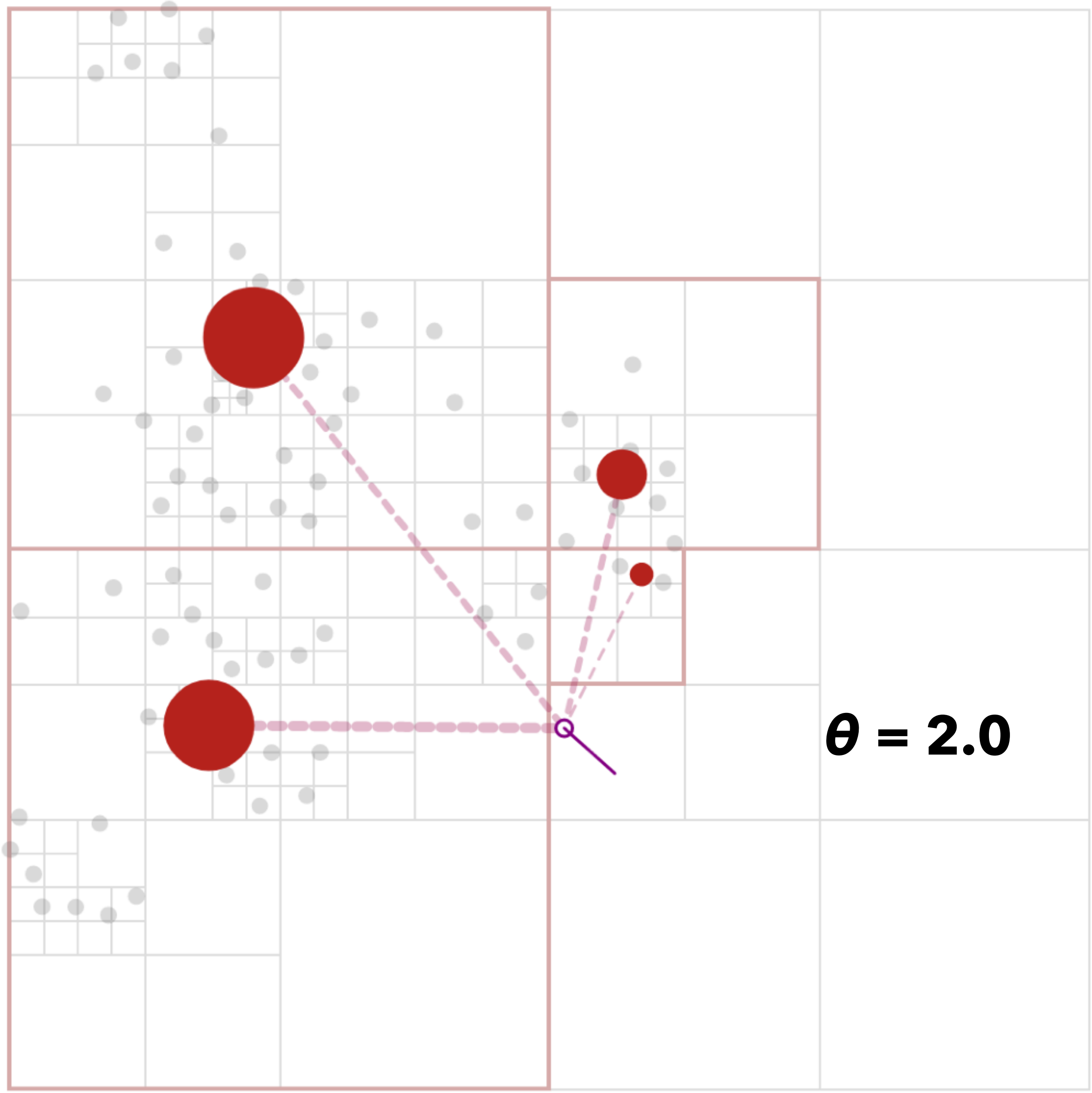
$\boldsymbol{\theta = 0.5}$



θ **= 0.9**

(default setting)


θ = 1.5


θ = 2.0

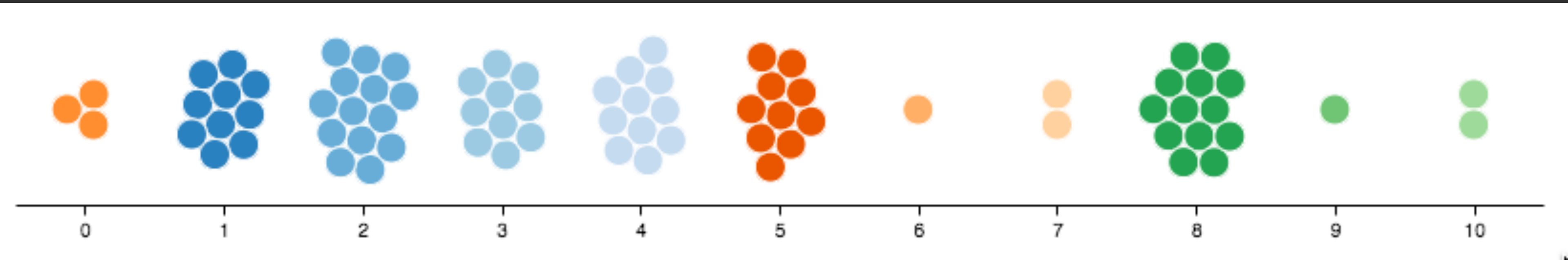
Customized Force Layouts

Different forces can be composed to create an expressive space of custom layouts.

A **beeswarm plot** can be made by combining:

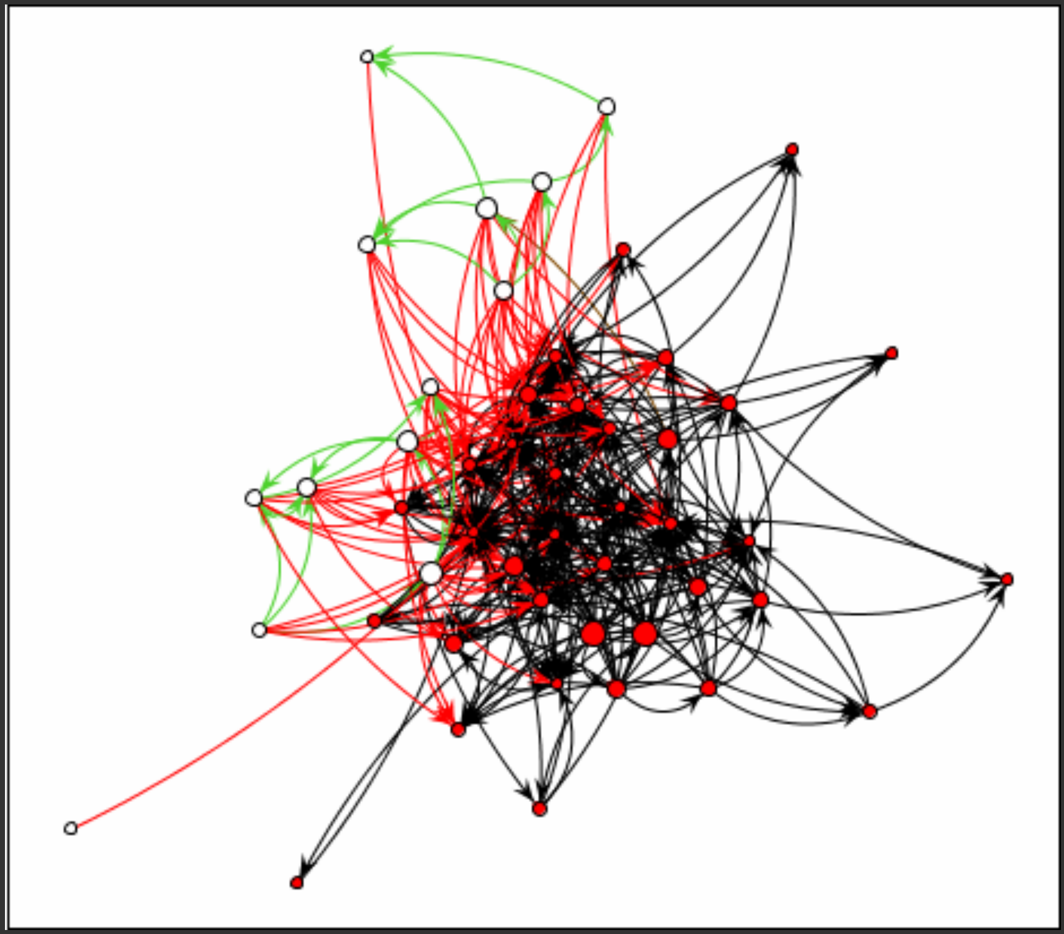
Attractive **X** and **Y** forces to draw nodes of a certain category to a desired point

Collide force to detect collision & remove overlap

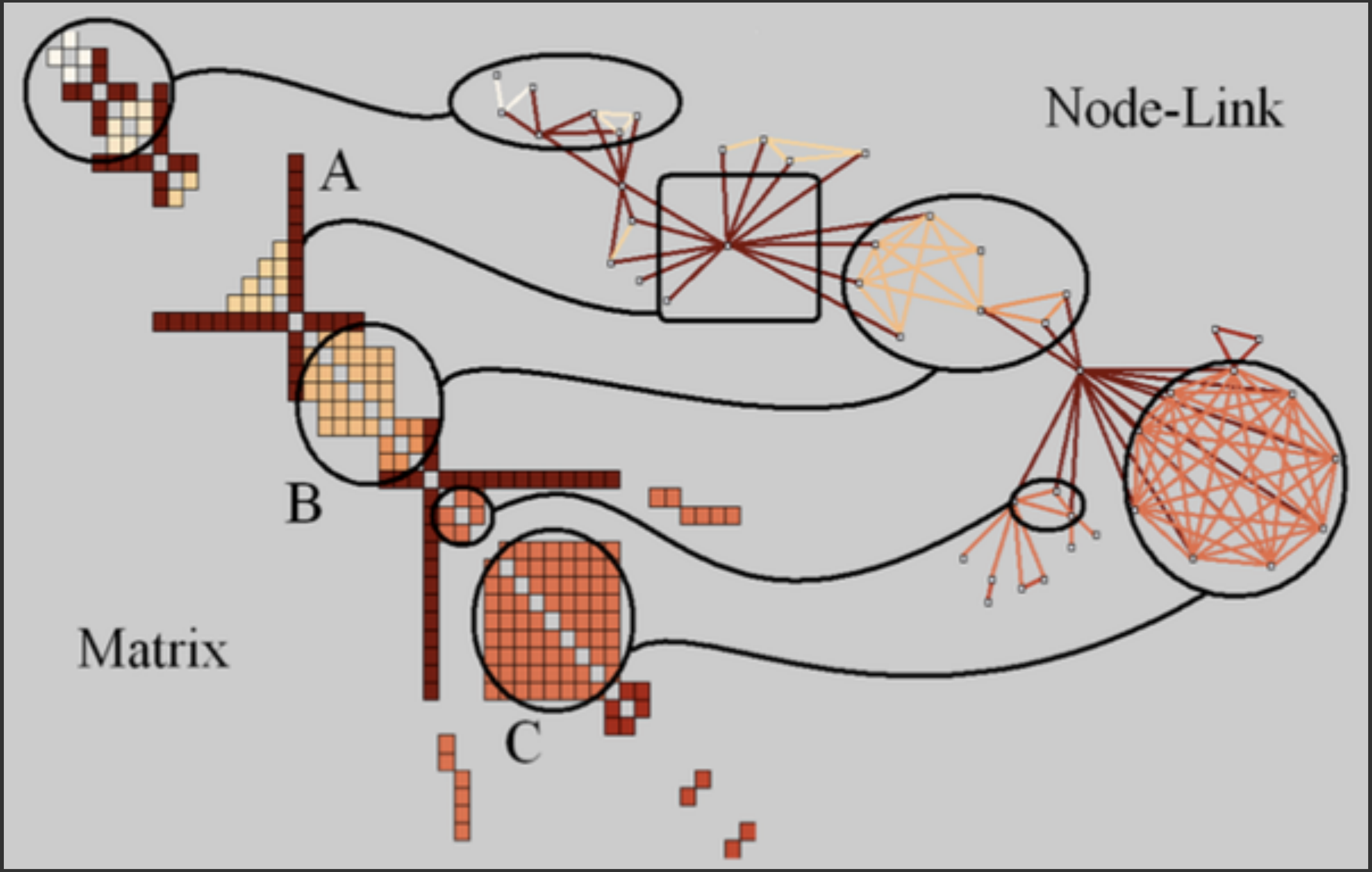


Matrix Diagrams

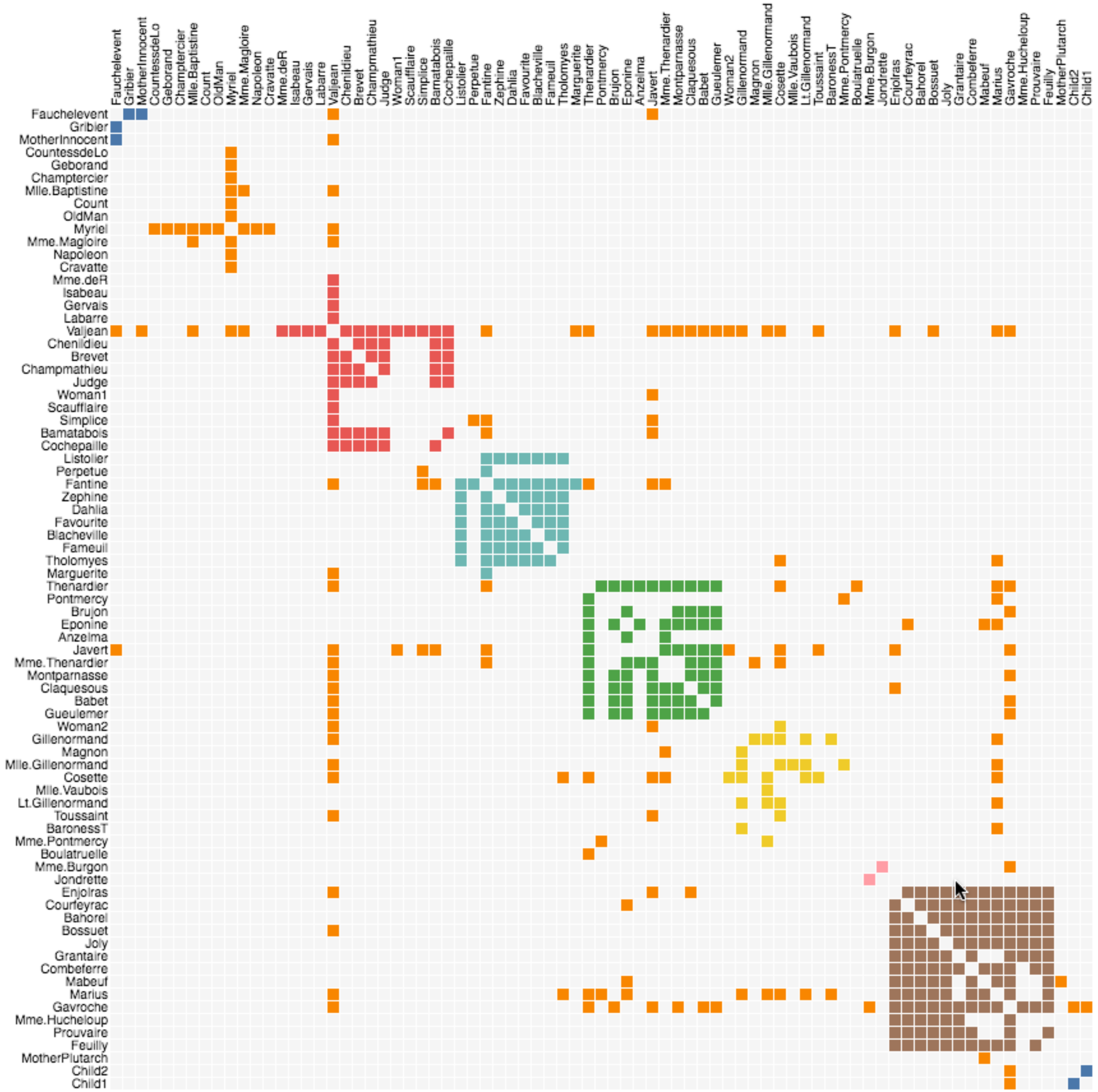
Limitations of Node-Link Layouts

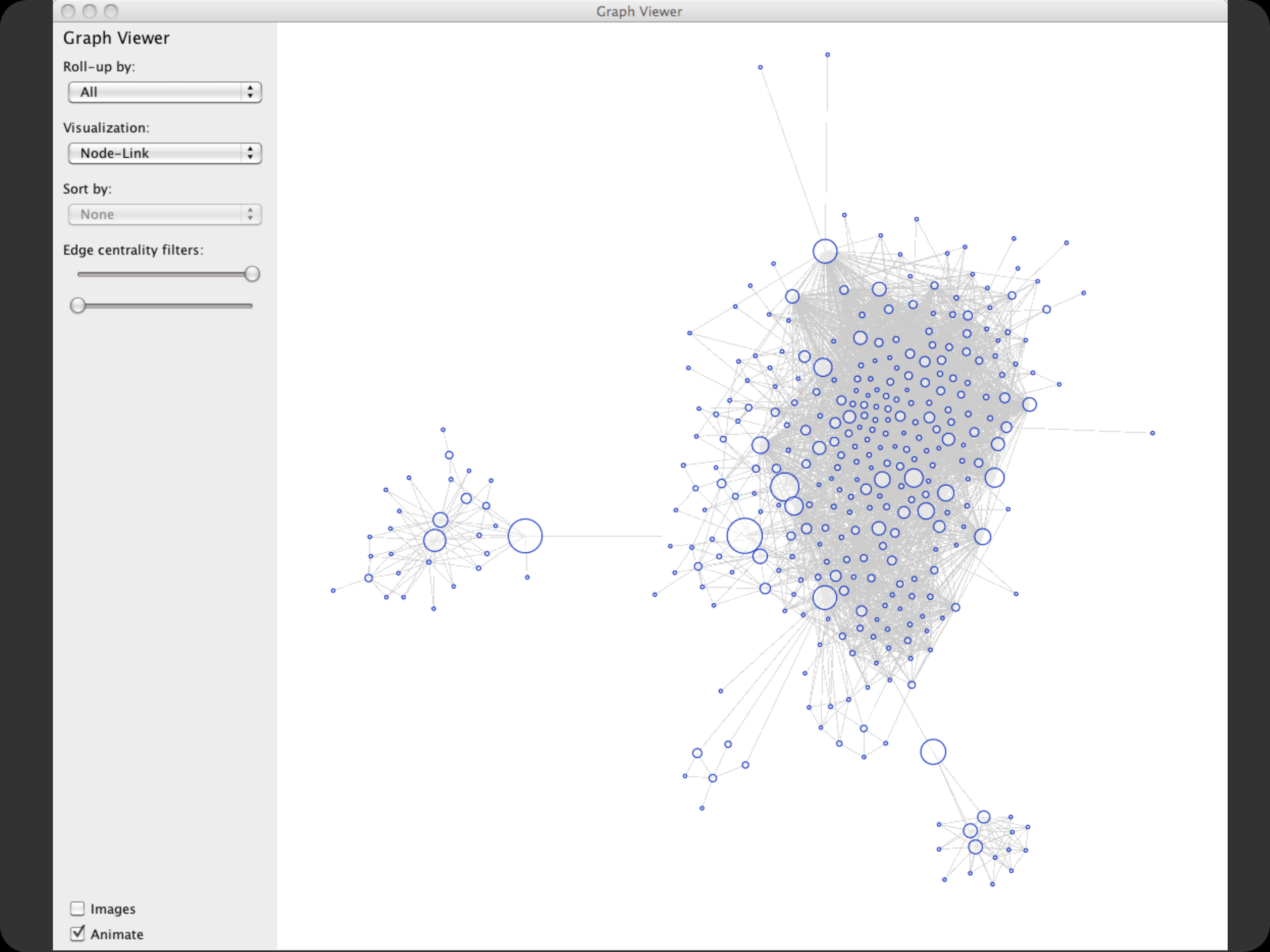


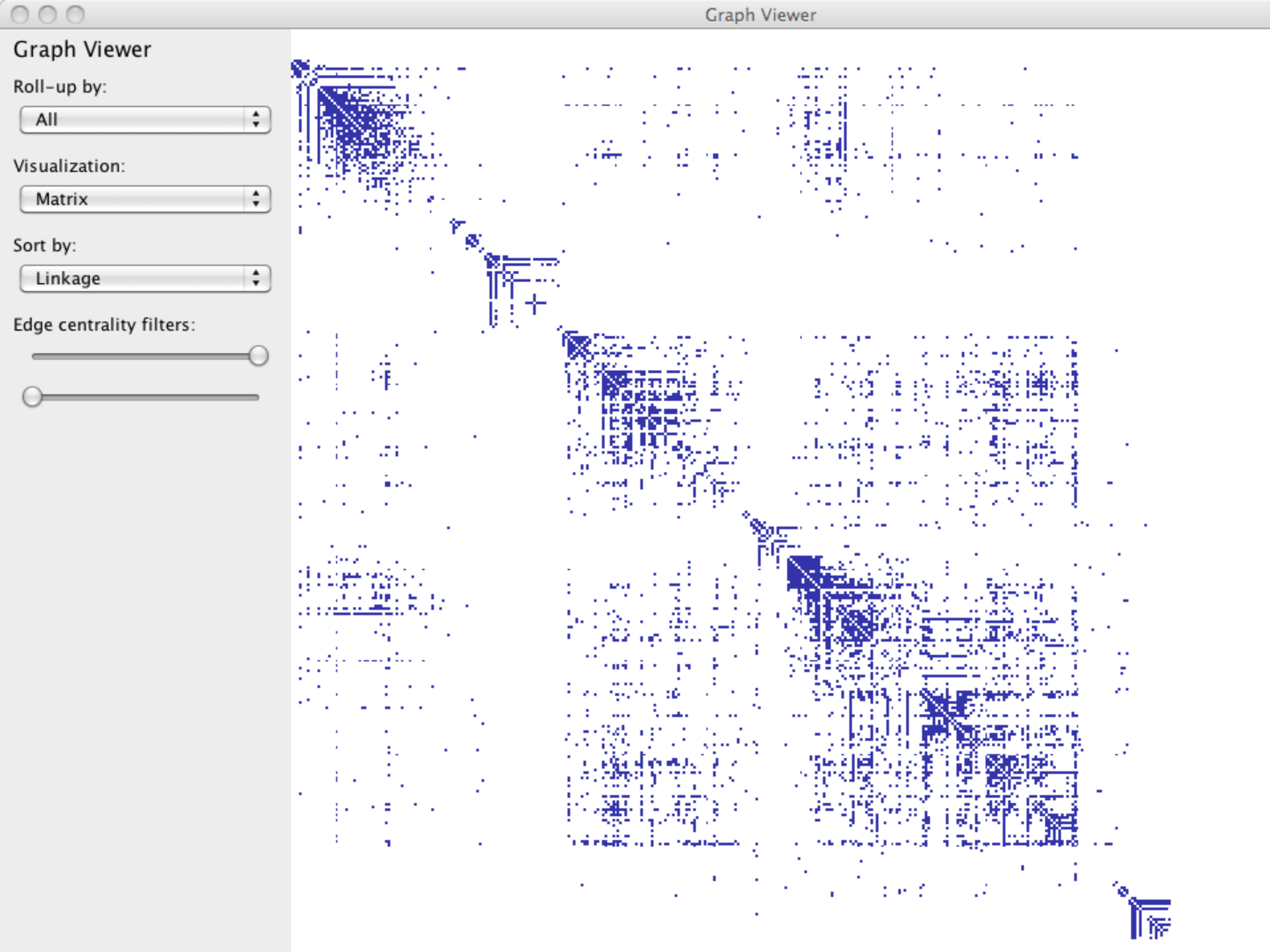
Edge-crossings and occlusion! Poor scalability….

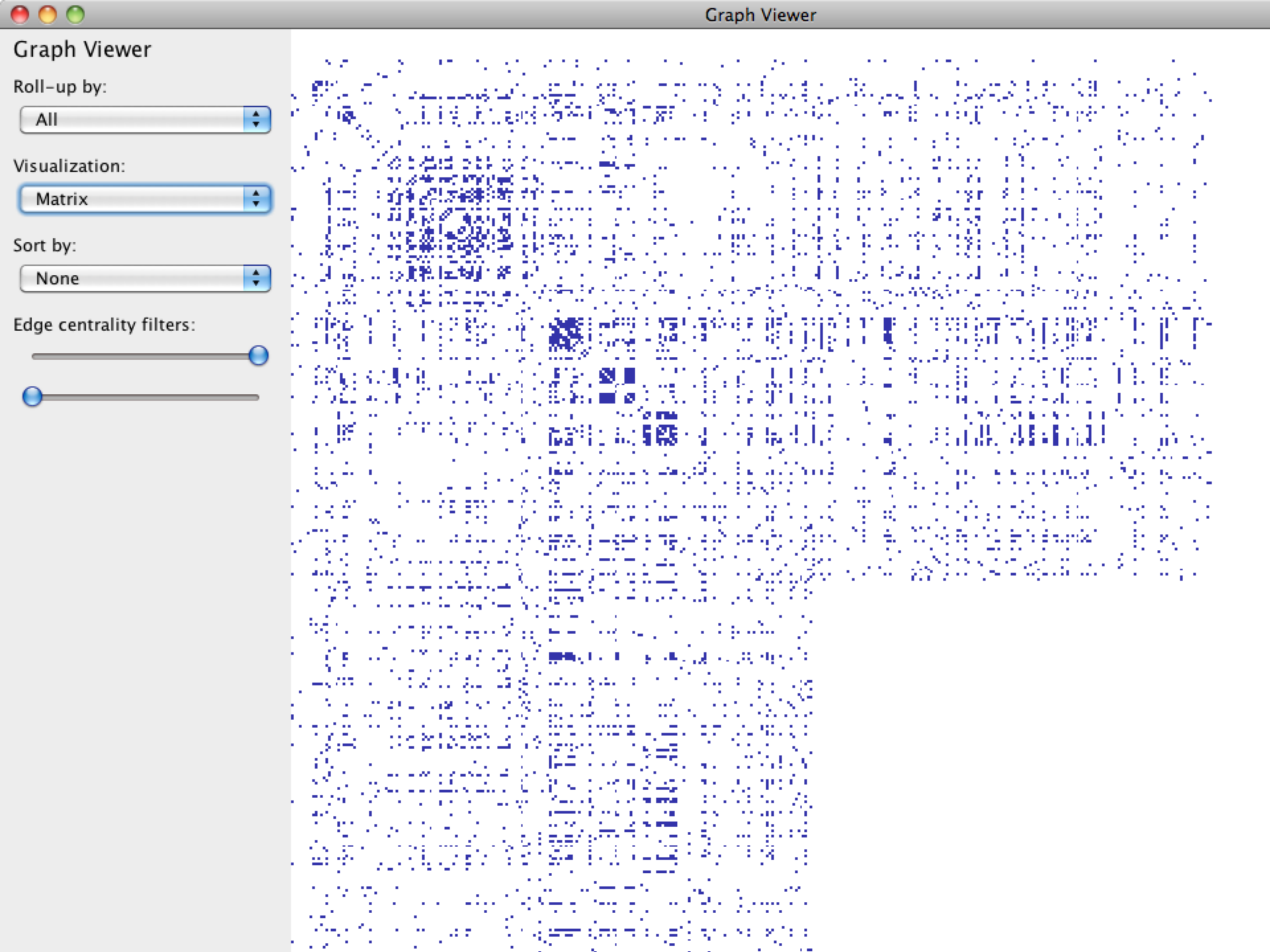


Adjacency Matrices


Fauchelevent
Gribier
MotherInnocent
CountessdeLo
Geborand
Champtercier
Mlle.Baptistine
Count
OldMan
Myriel
Mme.Magloire
Napoleon
Cravatte
Mme.deR
Isabeau
Gervais
Labarre
Valjean
Chenildieu
Brevet
Champmathieu
Judge
Woman1
Scaufflaire
Simplice
Bamatabois
Cochepaille
Listolier
Perpetue
Fantine
Zephine
Dahlia
Favourite
Blacheville
Fameuil
Tholomyes
Marguerite
Thenardier
Pontmercy
Brujon
Eponine
Anzelma
Javert
Mme.Thenardier
Montparnasse
Claquesous
Babet
Gueulemer
Woman2
Gillenormand
Magnon
Mlle.Gillenormand
Cosette
Mlle.Vaubois
Lt.Gillenormand
Toussaint
BaronessT
Mme.Pontmercy
Boulatruelle
Mme.Burgon
Jondrette
Enjolras
Courfeyrac
Bahorel
Bossuet
Joly
Grantaire
Combeferre
Mabeuf
Marius
Gavroche
Mme.Hucheloup
Prouvaire
Feuilly
MotherPlutarch
Child2
Child1
Fauchelevent
Gribier
MotherInnocent
CountessdeLo
Geborand
Champtercier
Mlle.Baptistine
Count
OldMan
Myriel
Mme.Magloire
Napoleon
Cravatte
Mme.deR
Isabeau
Gervais
Labarre
Valjean
Chenildieu
Brevet
Champmathieu
Judge
Woman1
Scaufflaire
Simplice
Bamatabois
Cochepaille
Listolier
Perpetue
Fantine
Zephine
Dahlia
Favourite
Blacheville
Fameuil
Tholomyes
Marguerite
Thenardier
Pontmercy
Brujon
Eponine
Anzelma
Javert
Mme.Thenardier
Montparnasse
Claquesous
Babet
Gueulemer
Woman2
Gillenormand
Magnon
Mlle.Gillenormand
Cosette
Mlle.Vaubois
Lt.Gillenormand
Toussaint
BaronessT
Mme.Pontmercy
Boulatruelle
Mme.Burgon
Jondrette
Enjolras
Courfeyrac
Bahorel
Bossuet
Joly
Grantaire
Combeferre
Mabeuf
Marius
Gavroche
Mme.Hucheloup
Prouvaire
Feuilly
MotherPlutarch
Child2
Child1


Graph Viewer
Graph Viewer
Roll-up by:
All
Visualization:
Node-Link
Sort by:
None
Edge centrality filters:
Images
Animate


Graph Viewer
Graph Viewer
Roll-up by:
All
Visualization:
Matrix
Sort by:
Linkage
Edge centrality filters:


Graph Viewer
Graph Viewer
Roll-up by:
All
Visualization:
Matrix
Sort by:
None
Edge centrality filters:

Seriation / Ordination / Permutation

Goal: Ensure similar items placed near each other.
E.g., minimize sum of distances of adjacent items.

Requires combinatorial optimization: **NP-Hard!**

Instead, approximate / heuristic approaches used:
- Perform hierarchical clustering, sort cluster tree.
- Apply approximate traveling salesperson solver.

Seriation initially used in **archaeology** for relative dating of artifacts based on observed properties.

Attribute-Driven Layout

Attribute-Driven Layout

Large node-link diagrams **get messy**!
Is there additional structure we can exploit?

Idea: Use **data attributes** to perform layout
For example, scatter plot based on node values

Attributes may be associated with nodes or edges
or may be statistical properties of the graph.

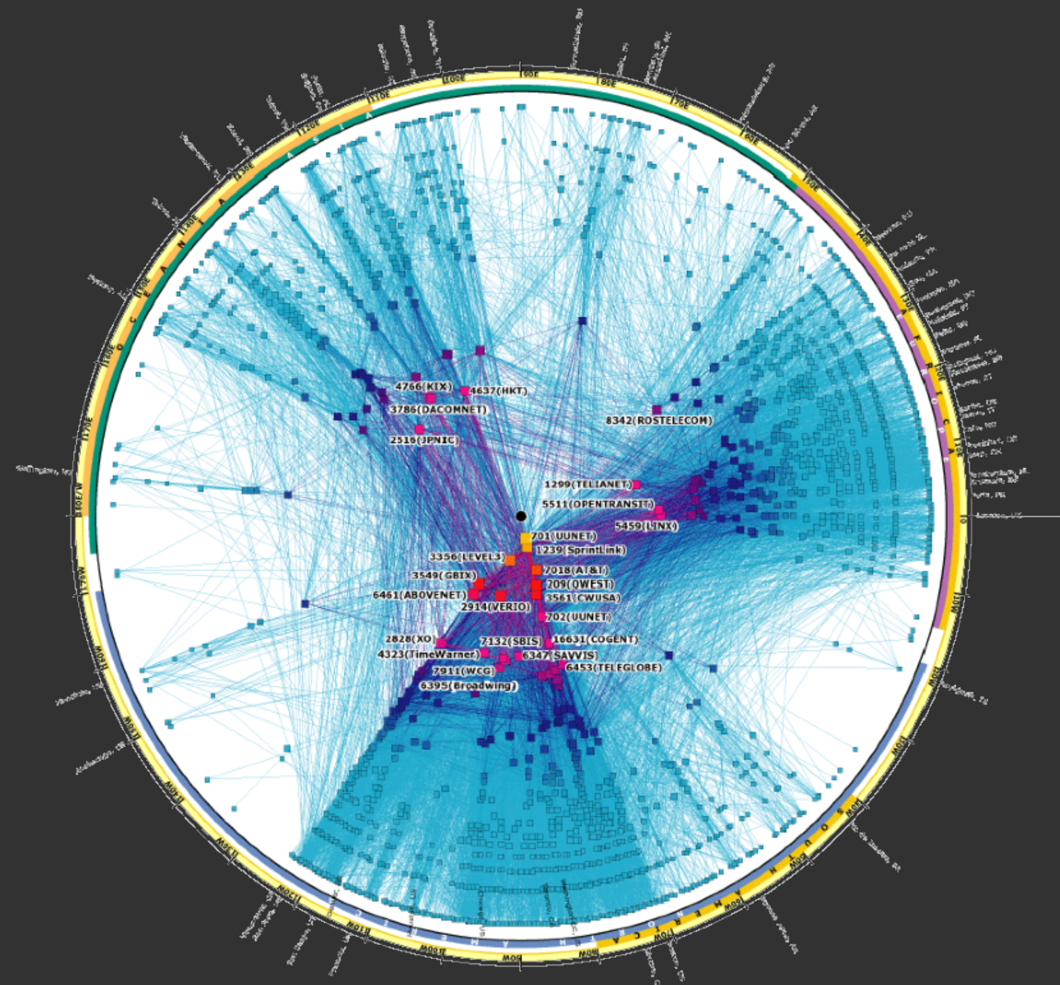
Use dynamic queries / brushing to explore…

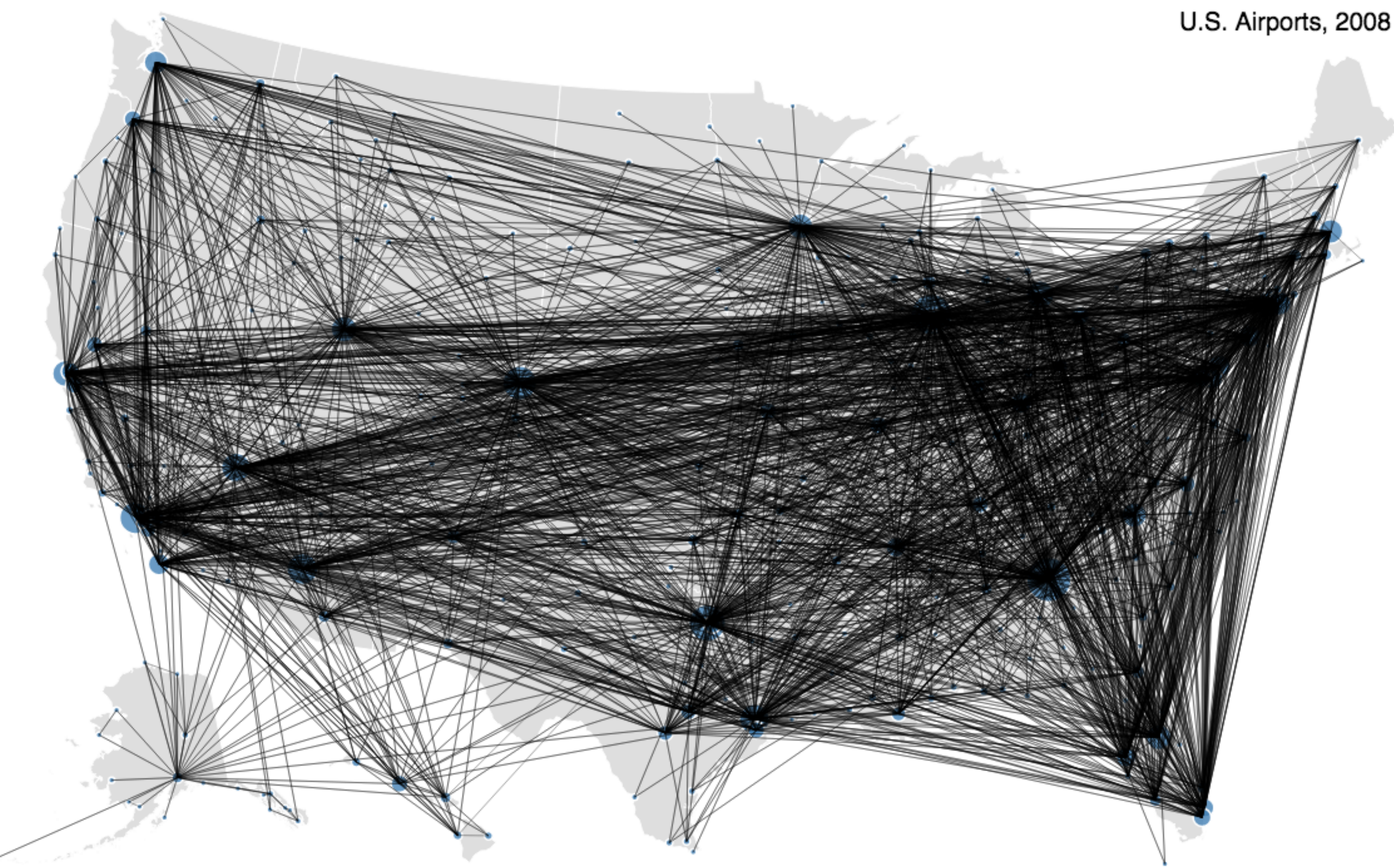
Attribute-Driven Layout

The "Skitter" Layout
Internet Connectivity
Radial Scatterplot

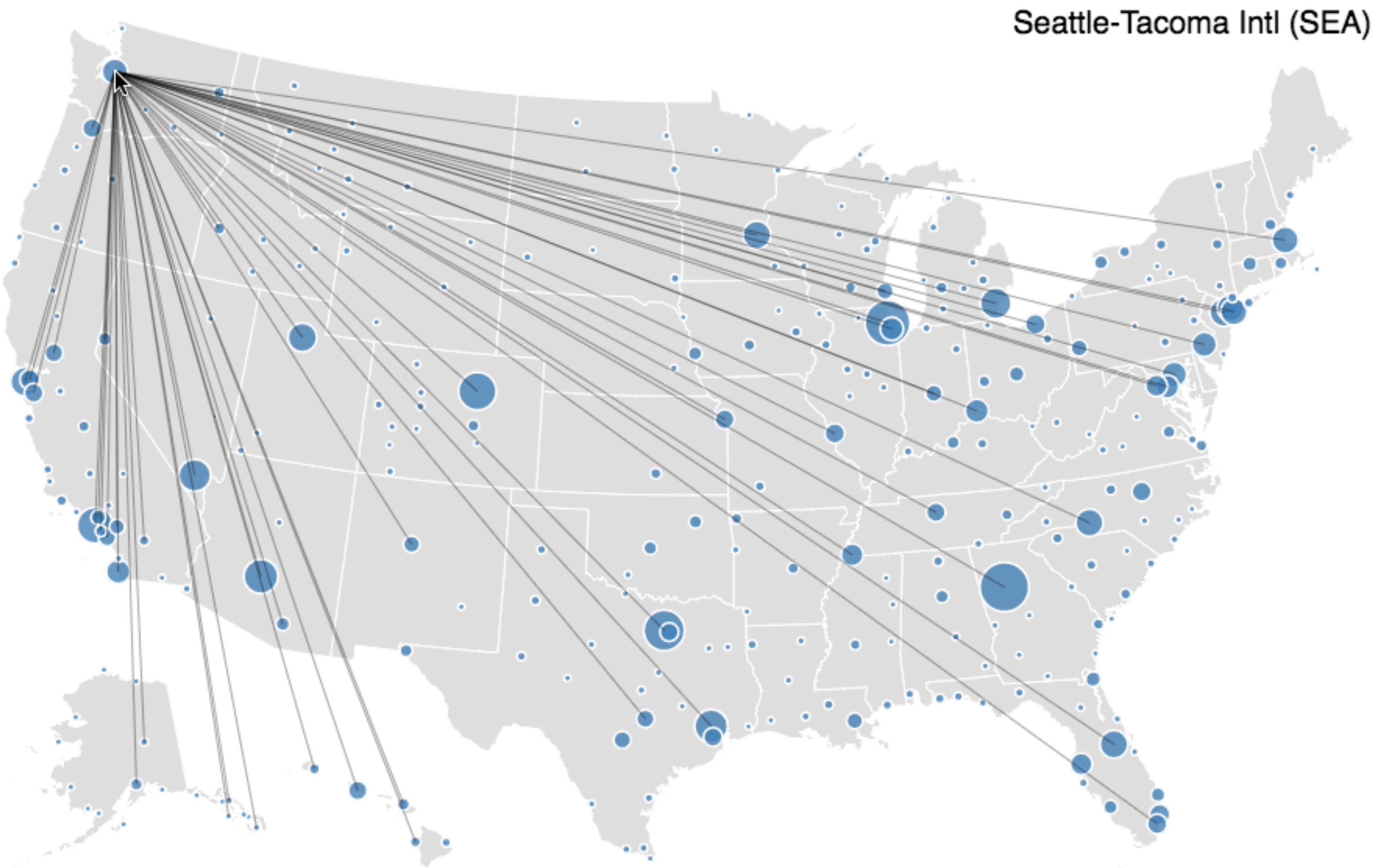
Angle = Longitude
Geography

Radius = Degree
# of connections
(a statistic of the nodes)



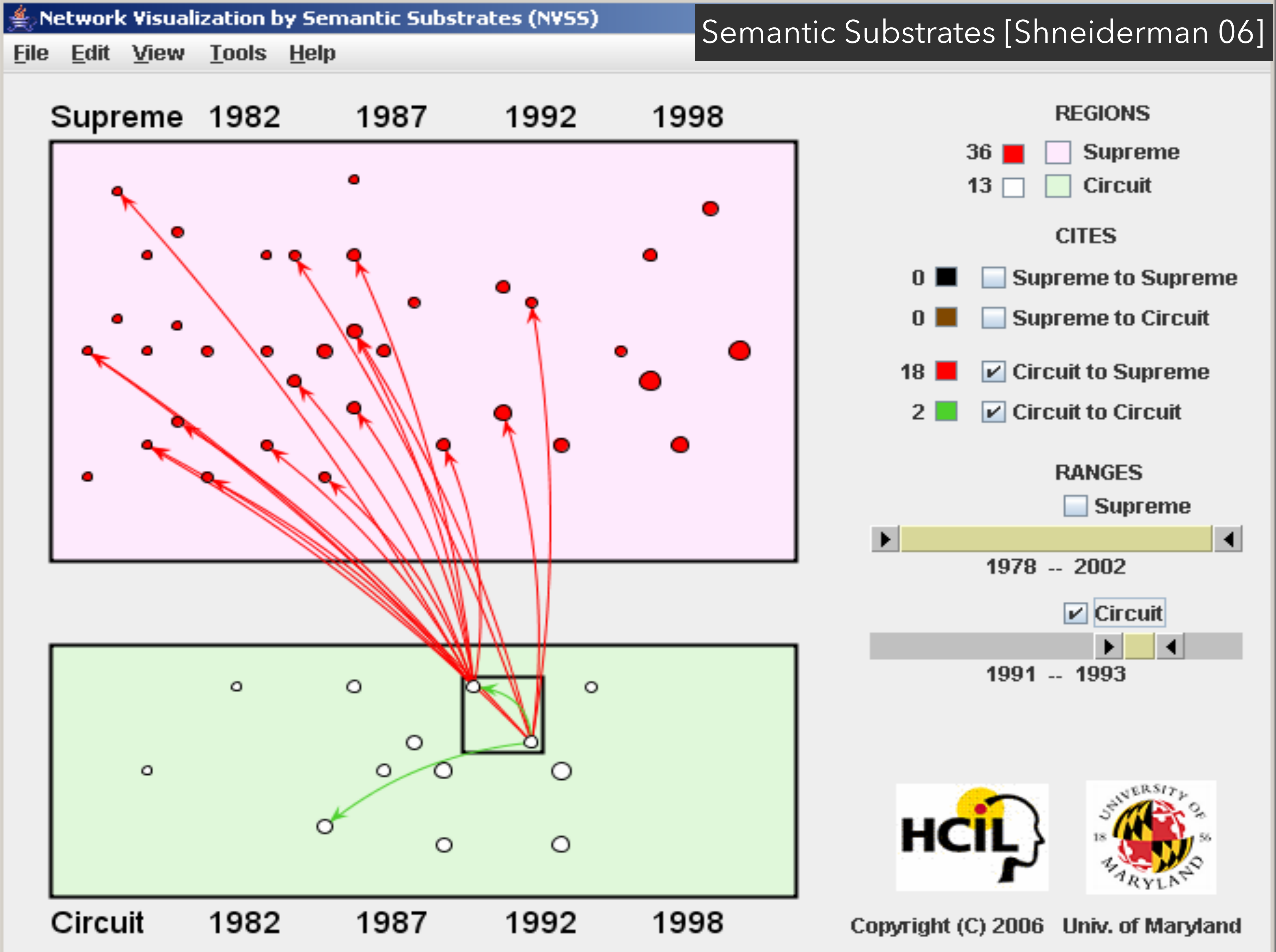


Drawing all edges is not particularly useful here…

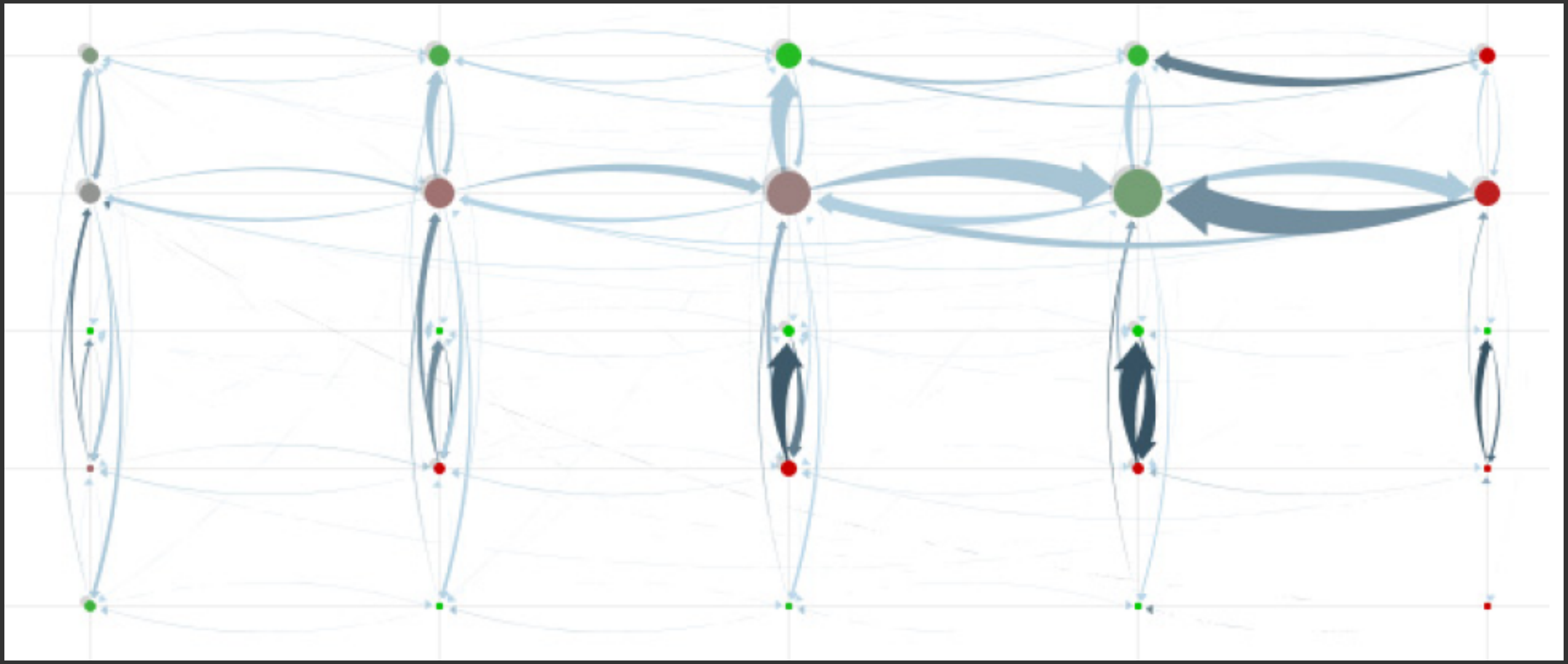


Node layout determined by geographic location.
Adjacent edges shown on node selection.

Semantic Substrates [Shneiderman 06]

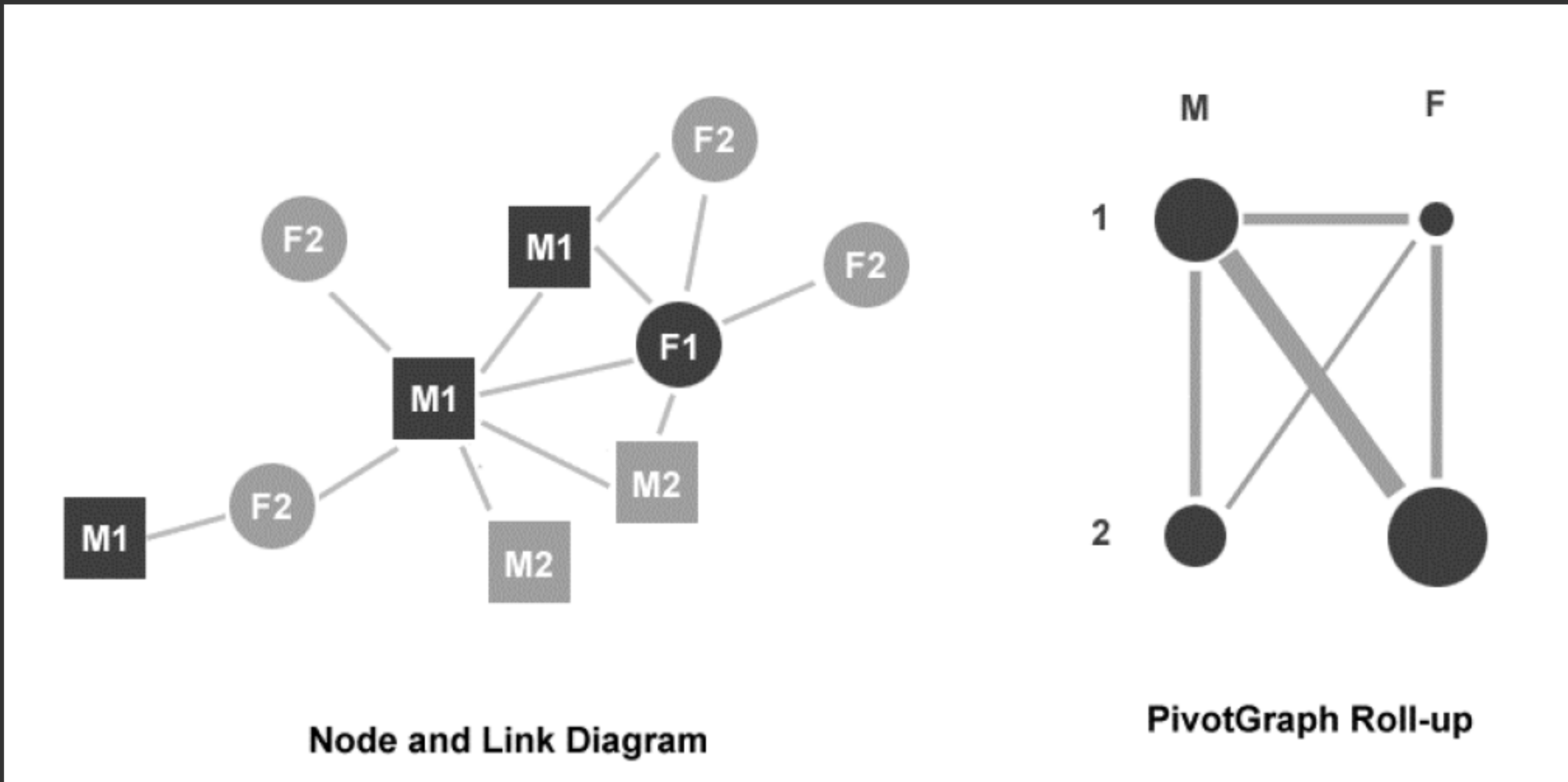


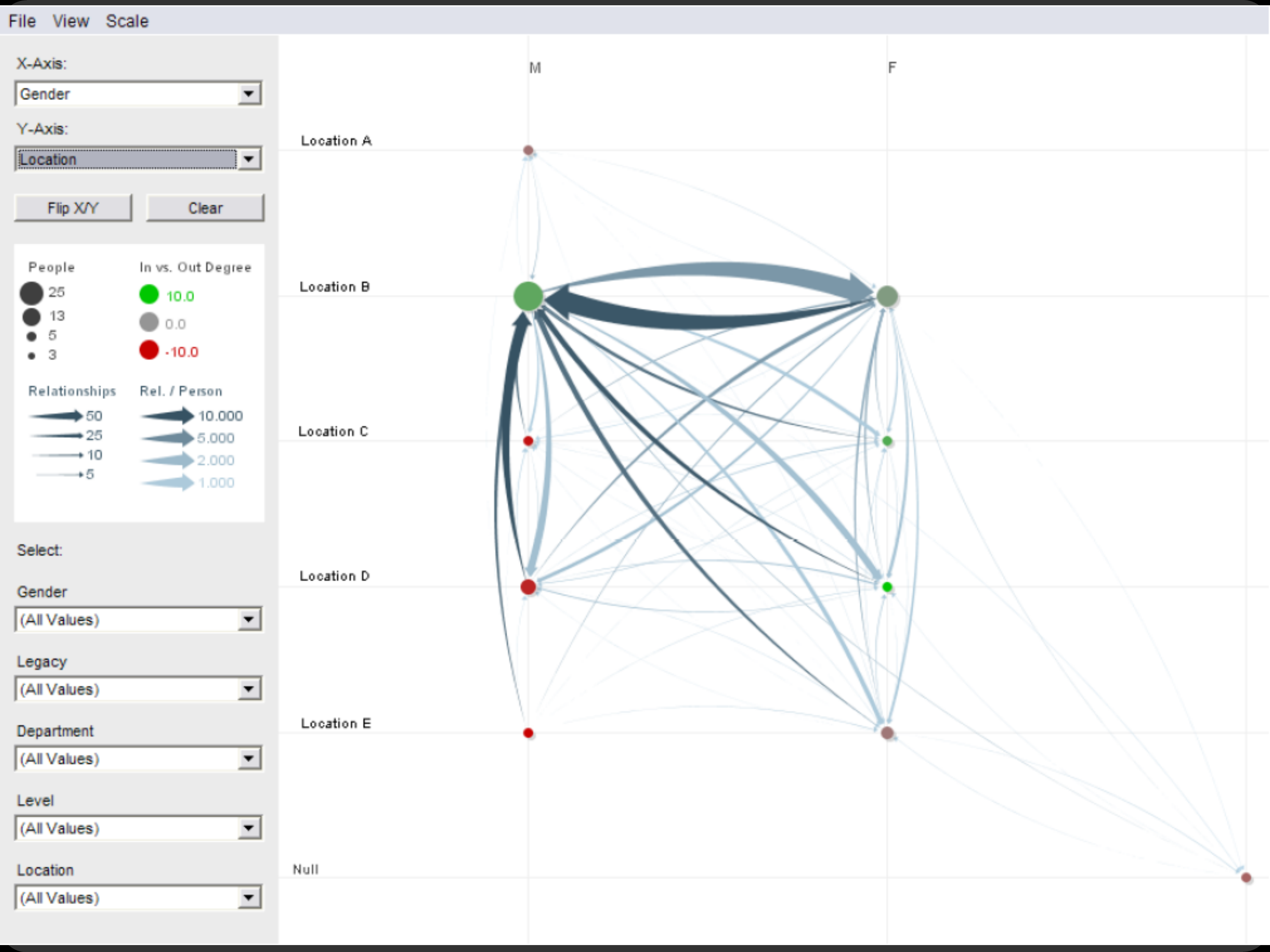
PivotGraph [Wattenberg '06]



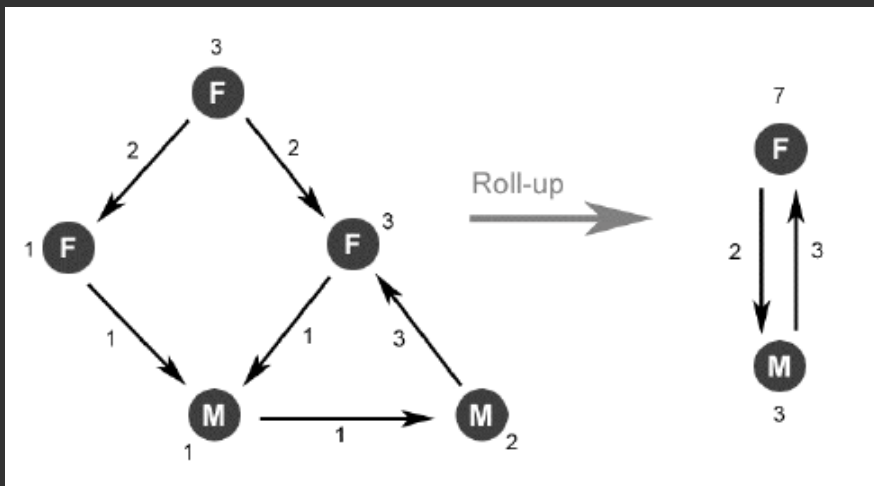
Layout aggregate graphs using node attributes.
Analogous to pivot tables and trellis display.

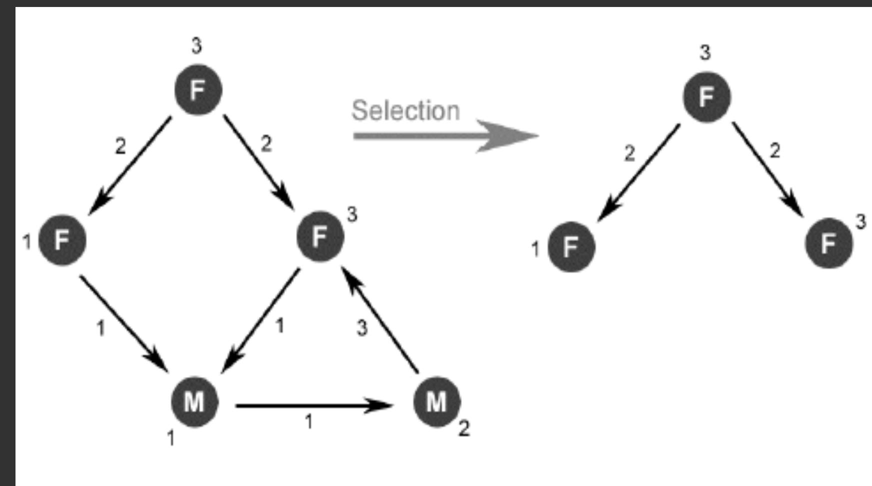
PivotGraph


F2
F2
M1
F2
F1
M1
M2
M1
F2
M2
Node and Link Diagram
M
F
1
2
PivotGraph Roll-up


File
View
Scale
X-Axis:
Gender
Y-Axis:
Location
Flip X/Y
Clear
People
25
13
5
3
In vs. Out Degree
10.0
0.0
-10.0
Relationships
50
25
10
5
Rel. / Person
10.000
5.000
2.000
1.000
Select:
Gender
(All Values)
Legacy
(All Values)
Department
(All Values)
Level
(All Values)
Location
(All Values)
M
F
Location A
Location B
Location C
Location D
Location E
Null

Operators



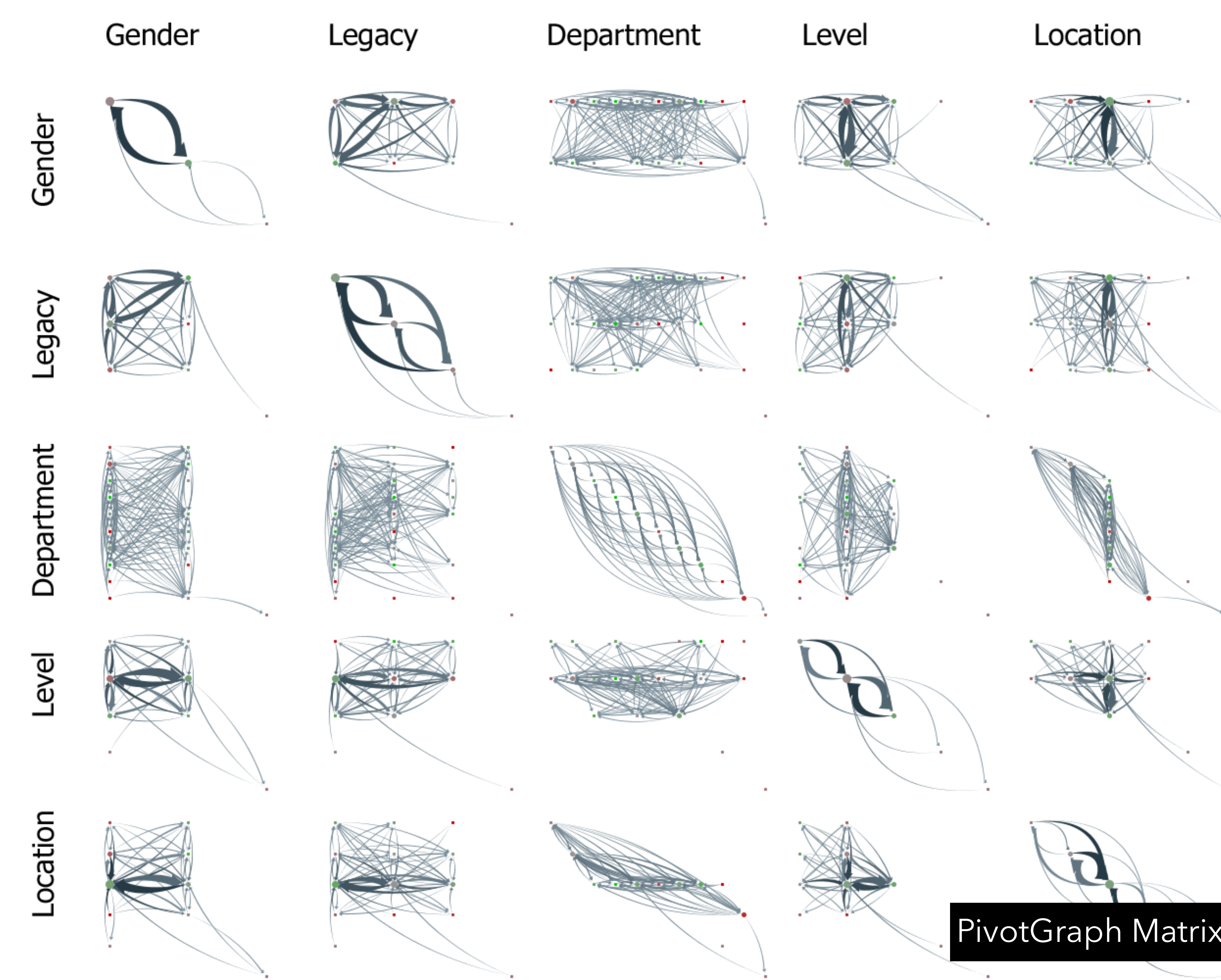


Roll-Up

Aggregate items with matching data values

Selection

Filter on data values



Gender
Legacy
Department
Level
Location
Gender
Legacy
Department
Level
Location
PivotGraph Matrix

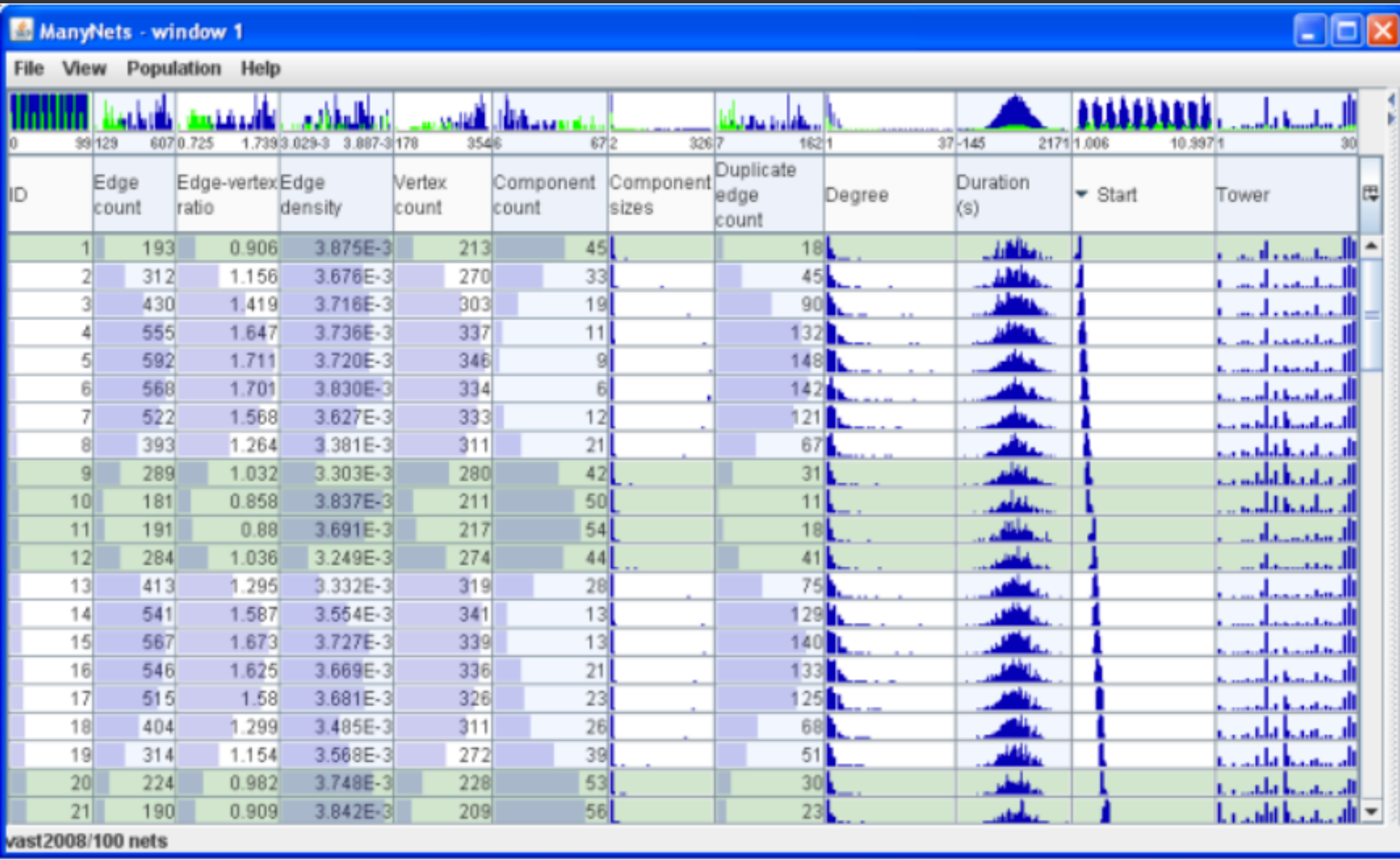
Limitations of PivotGraph

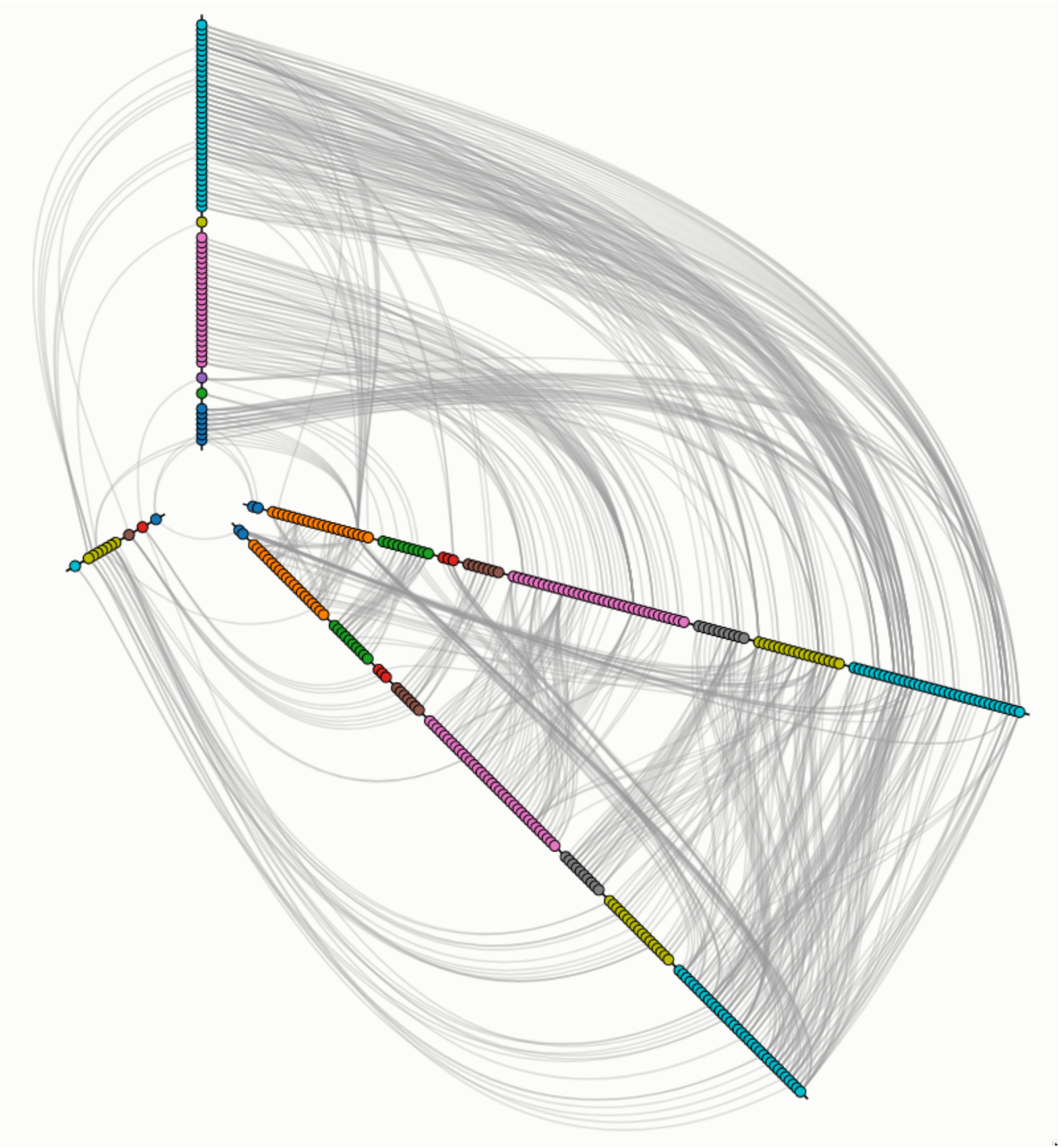
Only 2 variables (no nesting as in Tableau)

Doesn’t support continuous variables

Multivariate edges?

ManyNets [Freire et al. '10]





HivePlots

[Krzywinski '11]

Nodes (dots) may be replicated.

Nodes sorted on radial axes by network statistics (e.g., by degree).

Different axes may contain different subsets of nodes.

egweb.bcgsc.ca

Summary: Hierachies & Networks

Tree Layout

Indented / Node-Link / Enclosure / Layers

Focus+Context techniques for scale

Graph Layout

Spanning Tree Layout, “Sugiyama” Layout

Arc Diagrams

Force-Directed Layout, Optimization Methods

Matrix Diagrams

Attribute-Driven Layout