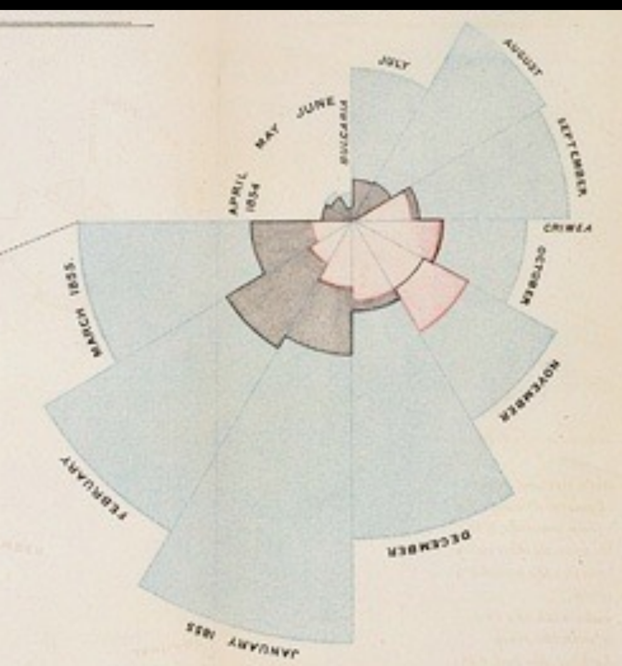
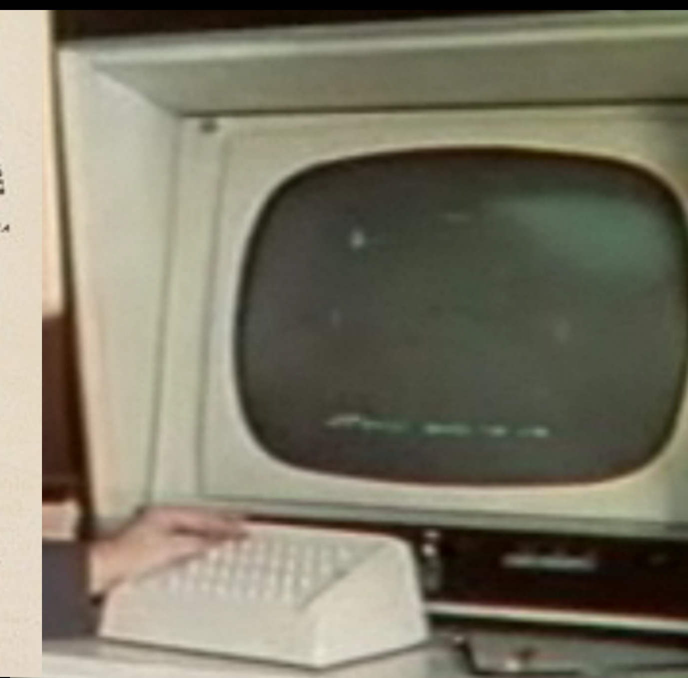
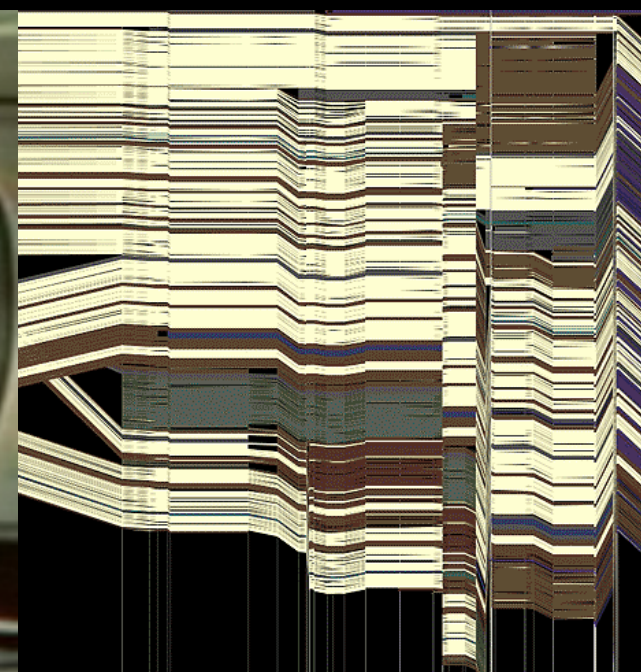
CSE 512 - Data Visualization

# Networks





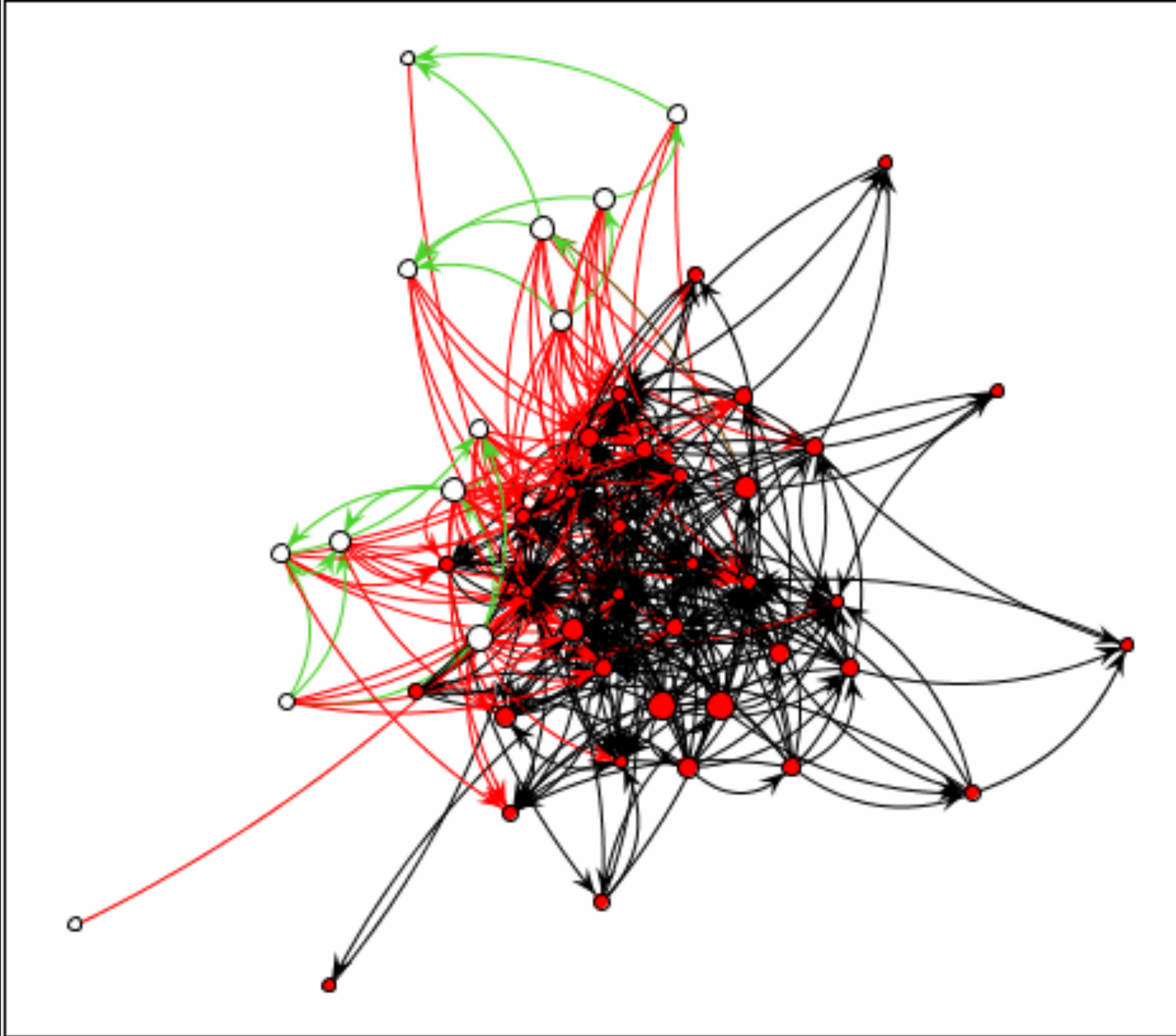
Jeffrey Heer University of Washington

# Topics

**Visualizing Trees**
**Visualizing Graphs**

## Goals

Overview of layout approaches
Assess strengths and weaknesses
Insight into implementation techniques

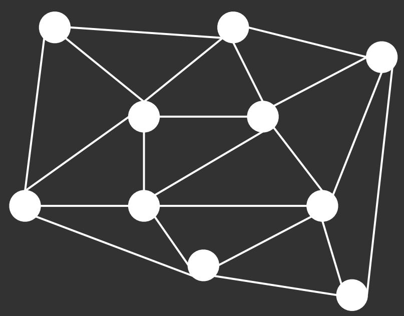


# Graphs and Trees

## Graphs

Model relations among data

*Nodes* and *edges*

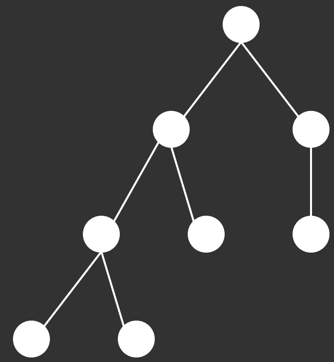


## Trees

Graphs with hierarchical structure

- Connected graph with N-1 edges

Nodes as *parents* and *children*



# Spatial Layout

A primary concern of 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
- Network distance
- Clustering
- 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

# Tree Layout

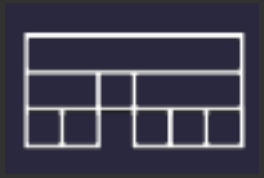
# Tree Visualization

**Indentation**
Linear list, indentation encodes depth

**Node-Link diagrams**
Nodes connected by lines/curves

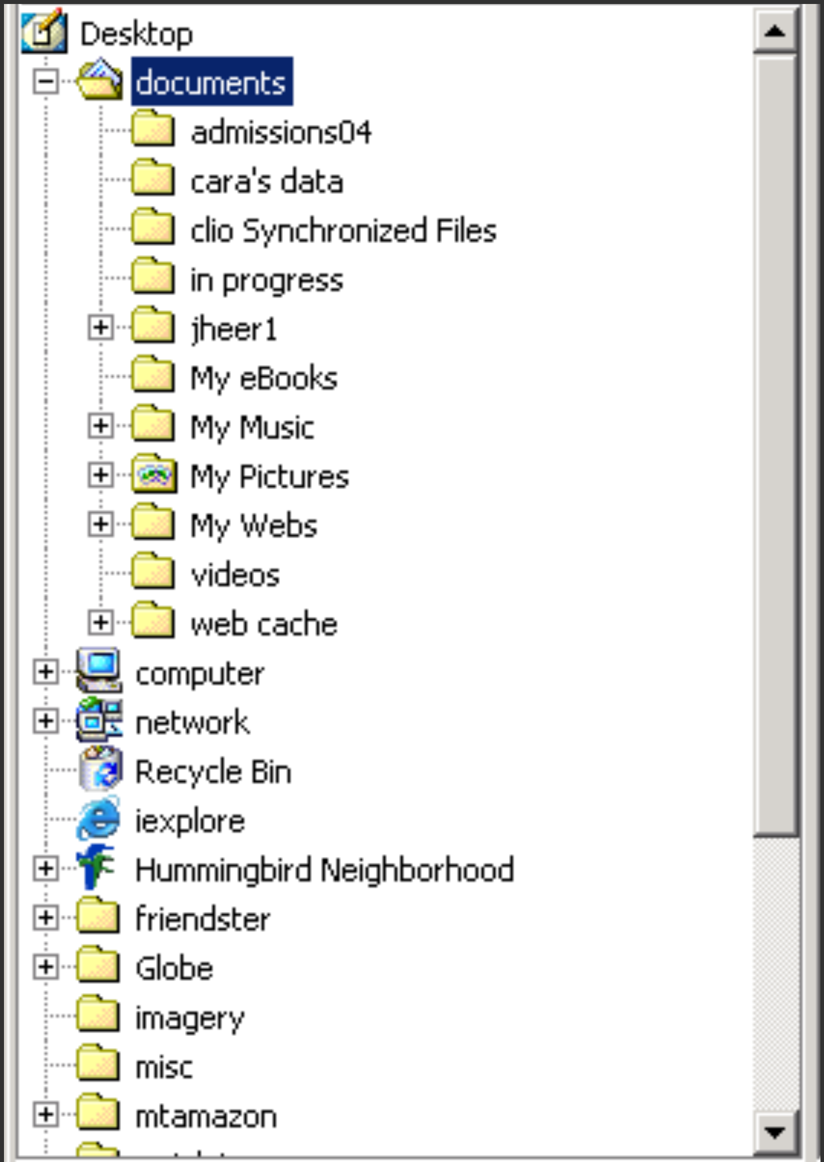
**Enclosure diagrams**
Represent hierarchy by enclosure

**Layering**
Relative position and alignment



*Fast*: O(n) or O(n log n), interactive 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

# Node-Link Diagram

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

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

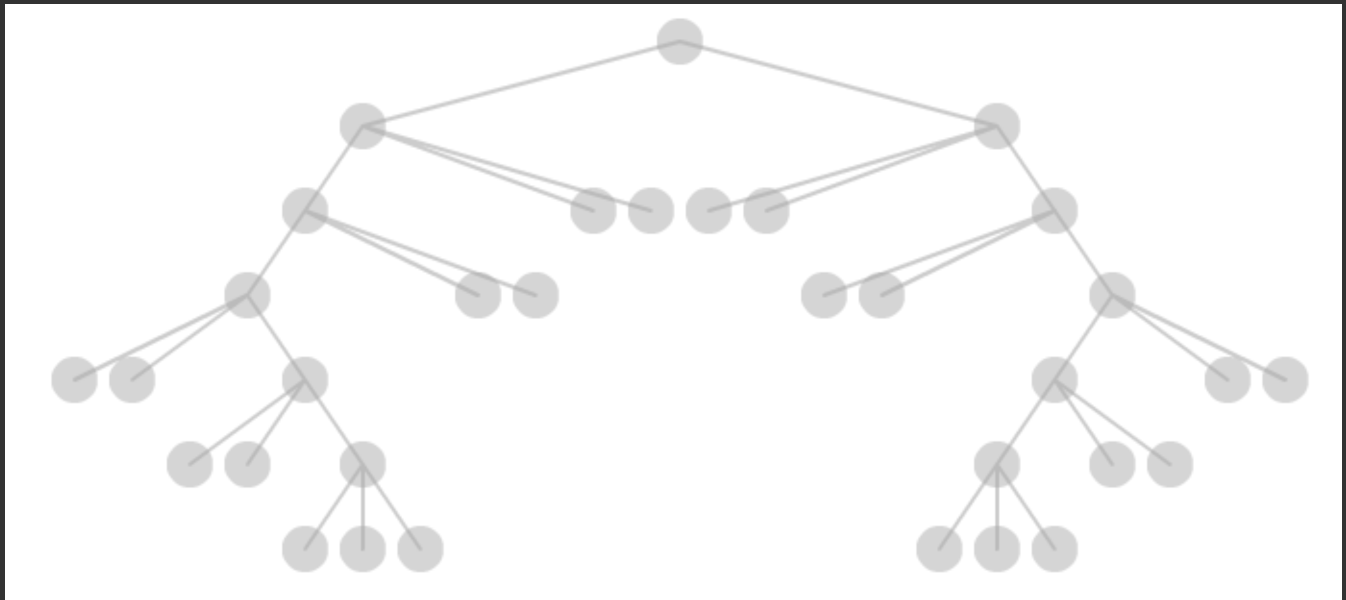
Often space is used to communicate hierarchical orientation (e.g., towards authority or generality)

# Basic Recursive Approach

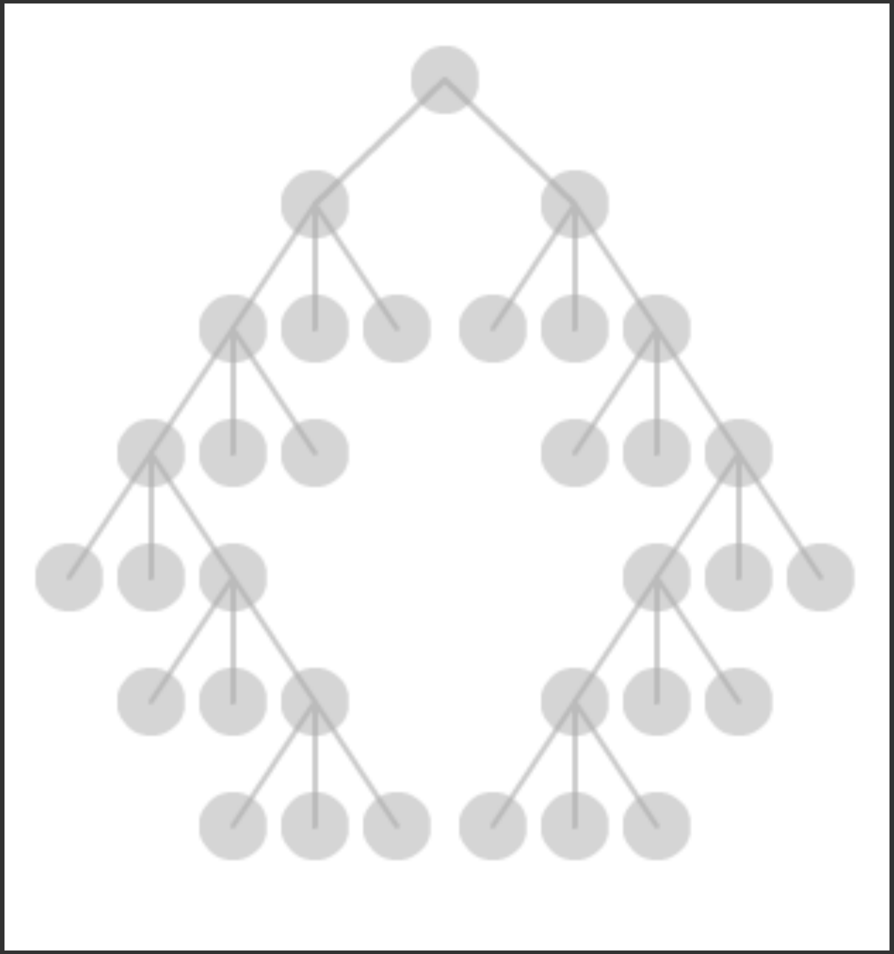
Repeatedly divide space for subtrees by leaf count

- 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

Linear algorithm – starts with bottom-up pass of the tree

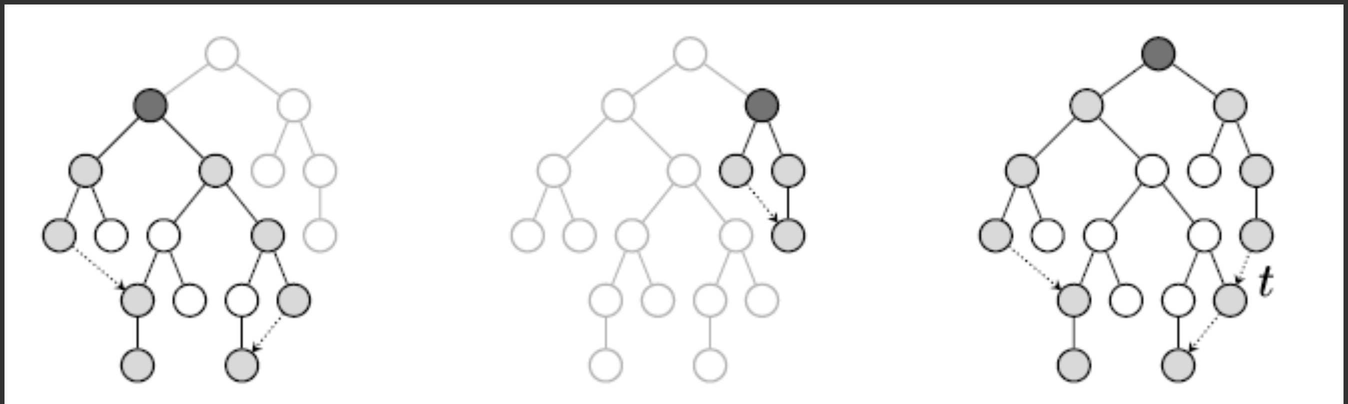
Y-coord by depth, arbitrary starting X-coord

Merge left and right subtrees

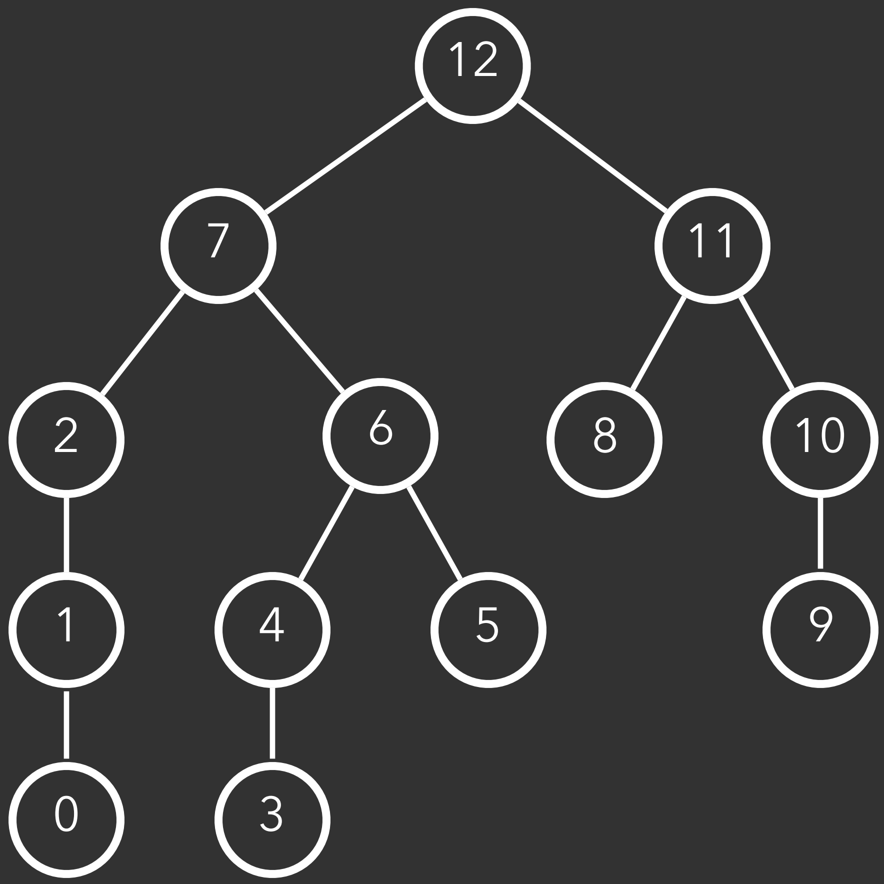
- Shift right as close as possible to left
  - Computed efficiently by maintaining subtree contours
- "Shifts" in position saved for each node as visited
- Parent nodes are centered above their children

Top-down pass for assignment of final positions

- Sum of initial layout and aggregated shifts

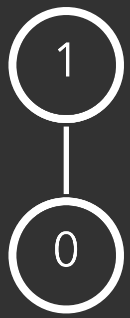


# Reingold-Tilford Layout



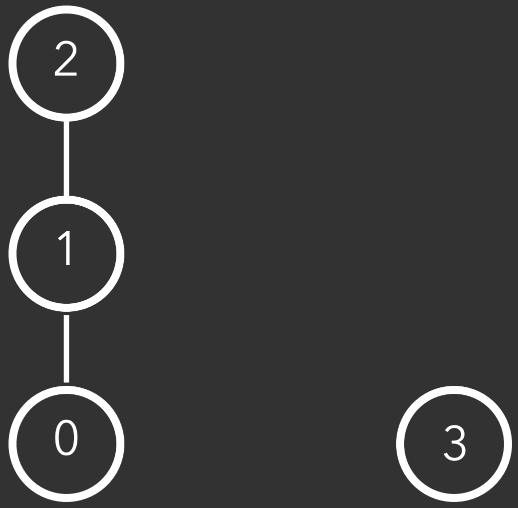
# Reingold-Tilford Layout

# Reingold-Tilford Layout

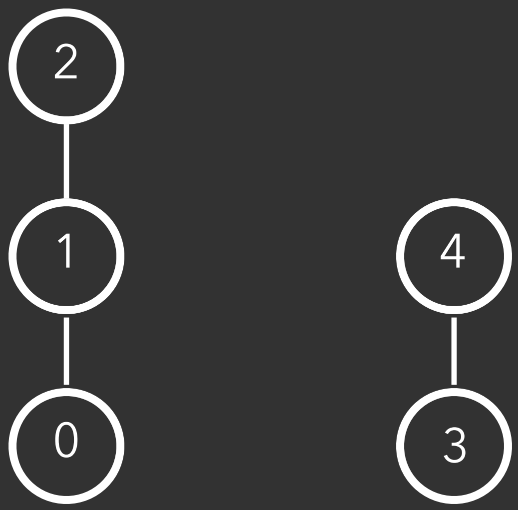


# Reingold-Tilford Layout

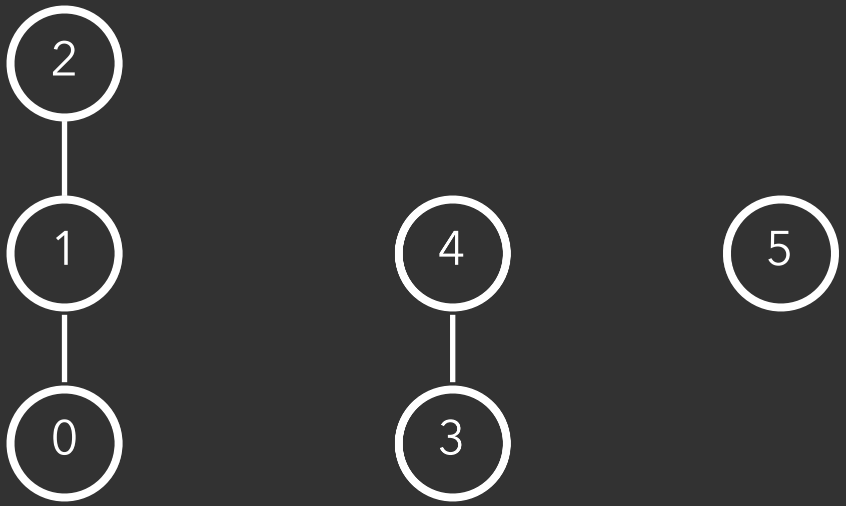
# Reingold-Tilford Layout



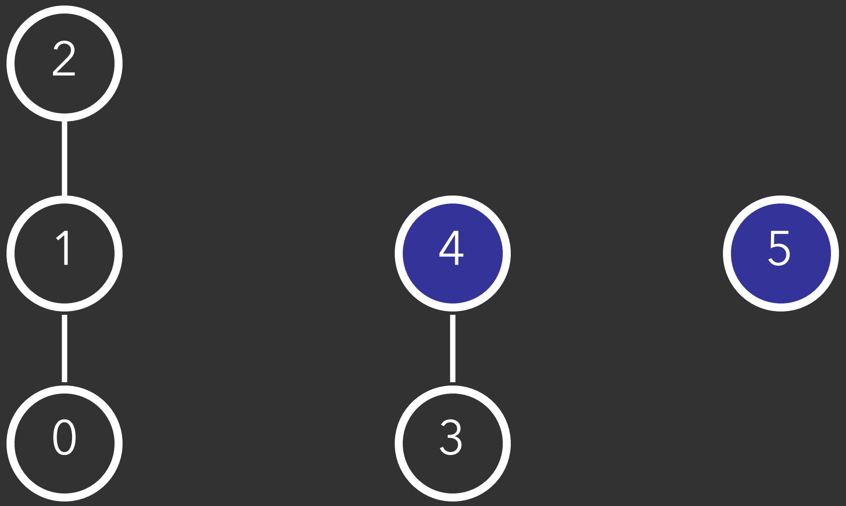
# Reingold-Tilford Layout



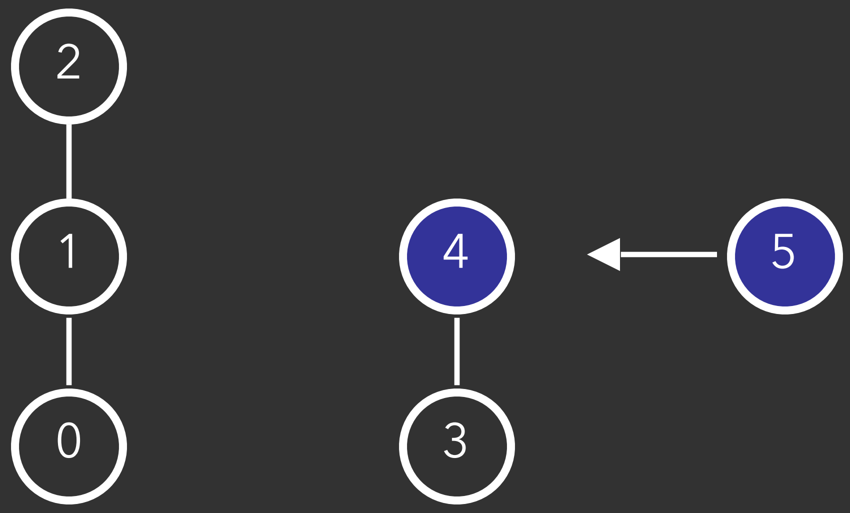
# Reingold-Tilford Layout



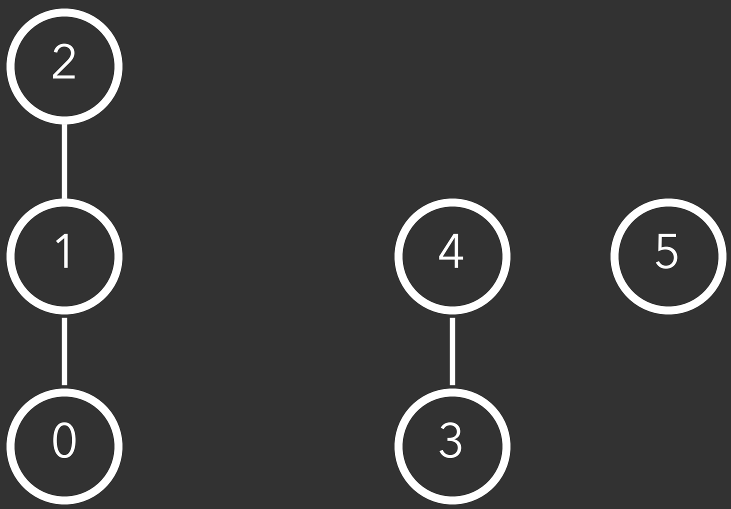
# Reingold-Tilford Layout



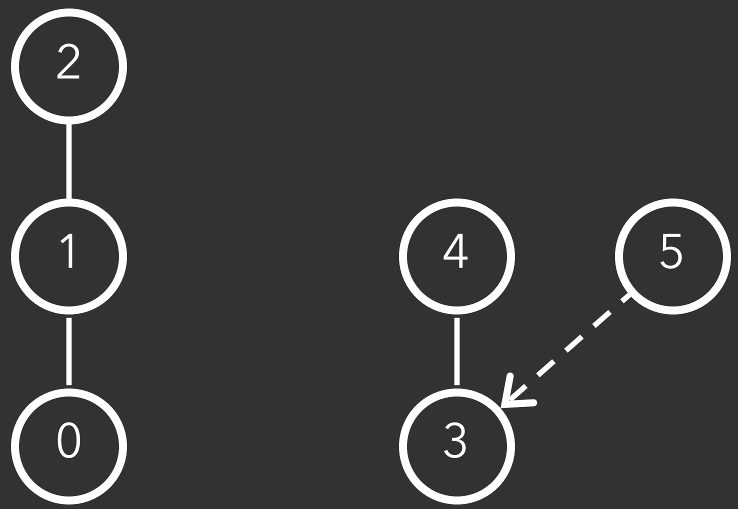
# Reingold-Tilford Layout



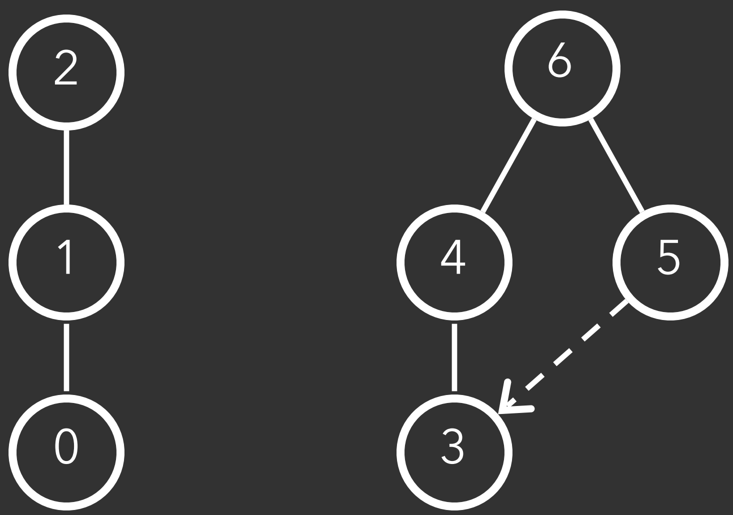
# Reingold-Tilford Layout



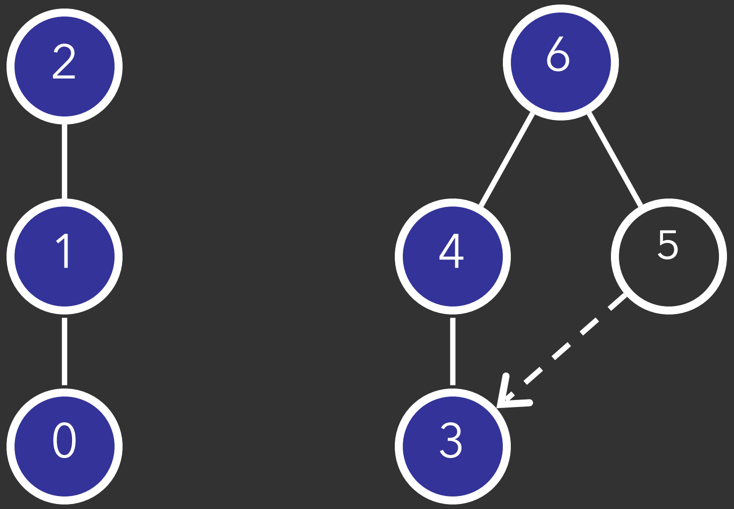
# Reingold-Tilford Layout



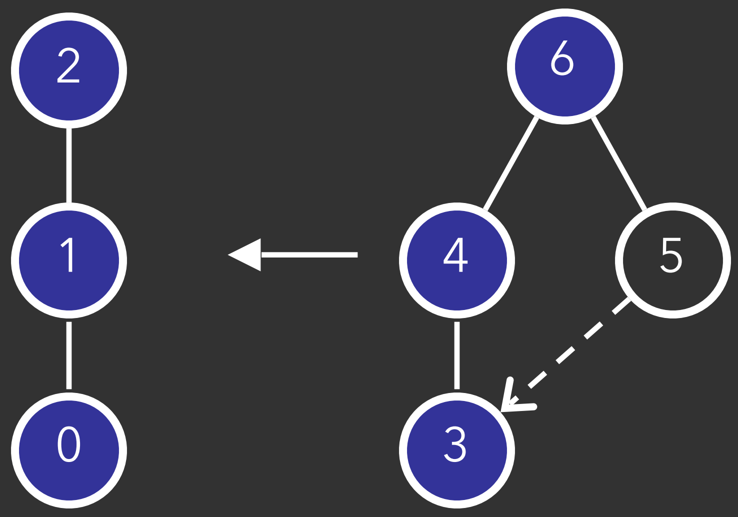
# Reingold-Tilford Layout



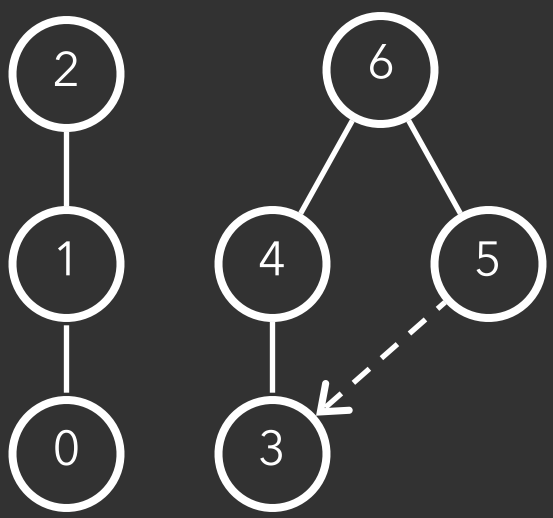
# Reingold-Tilford Layout



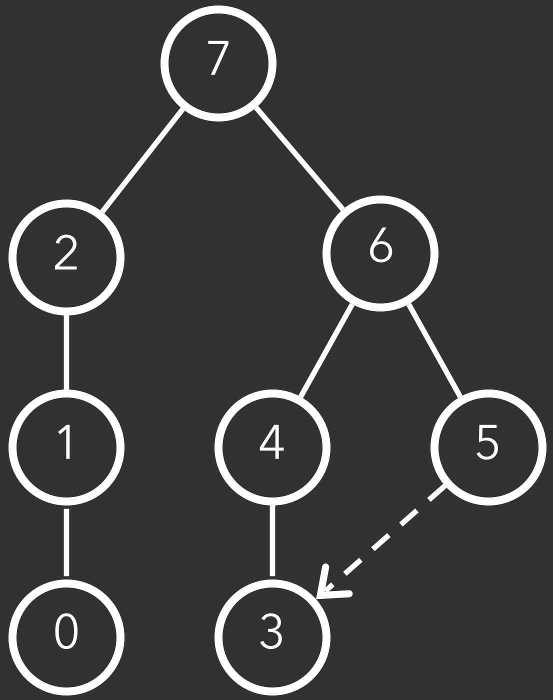
# Reingold-Tilford Layout



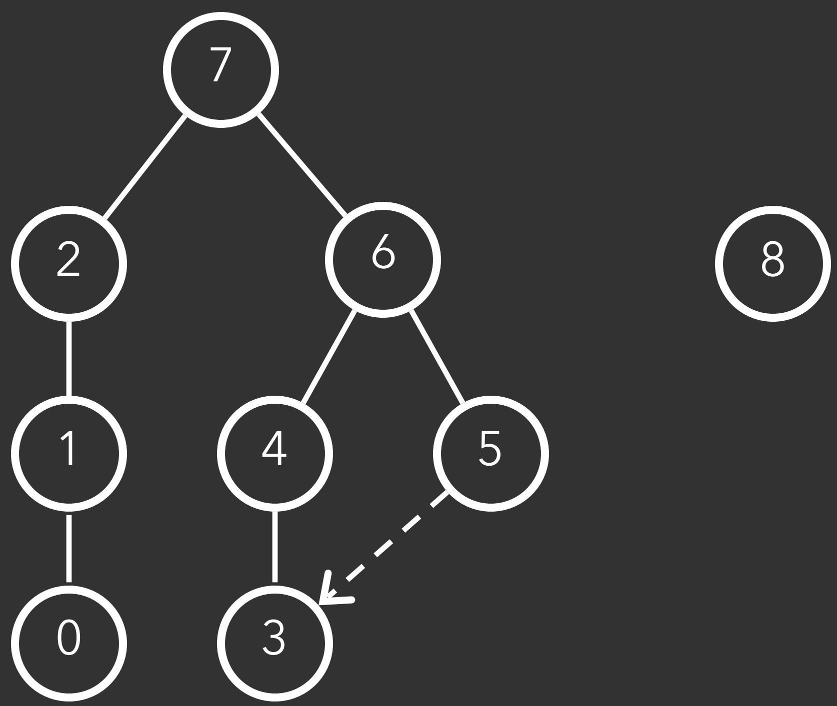
# Reingold-Tilford Layout



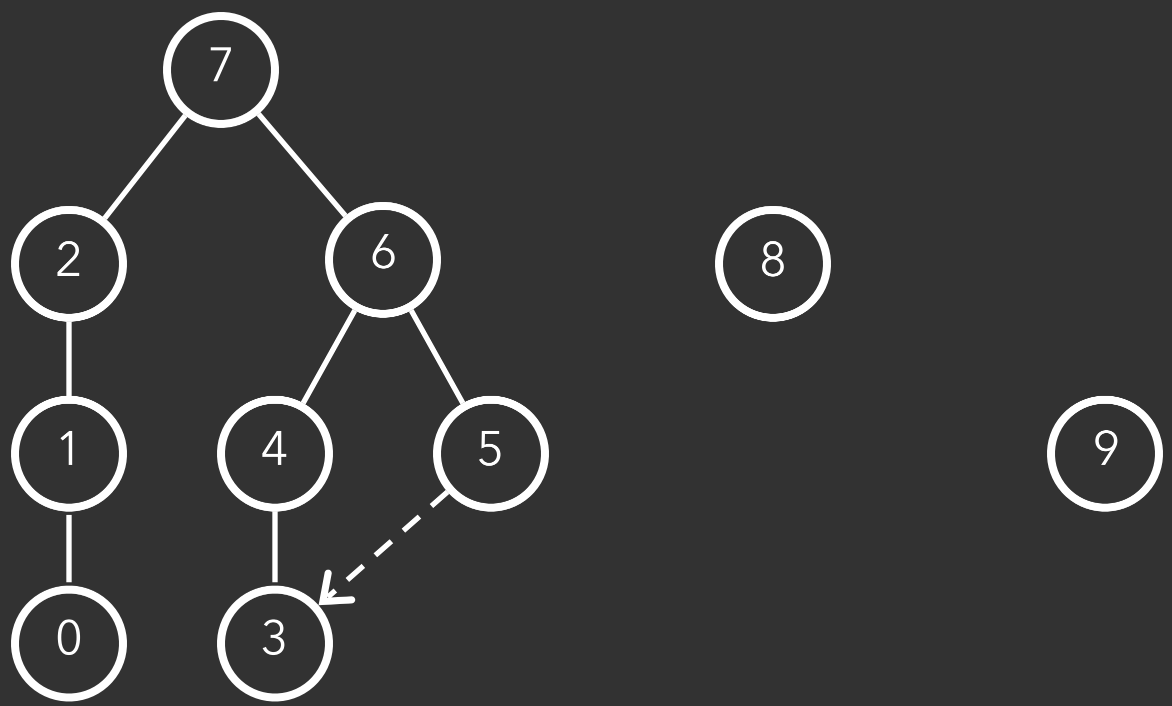
# Reingold-Tilford Layout



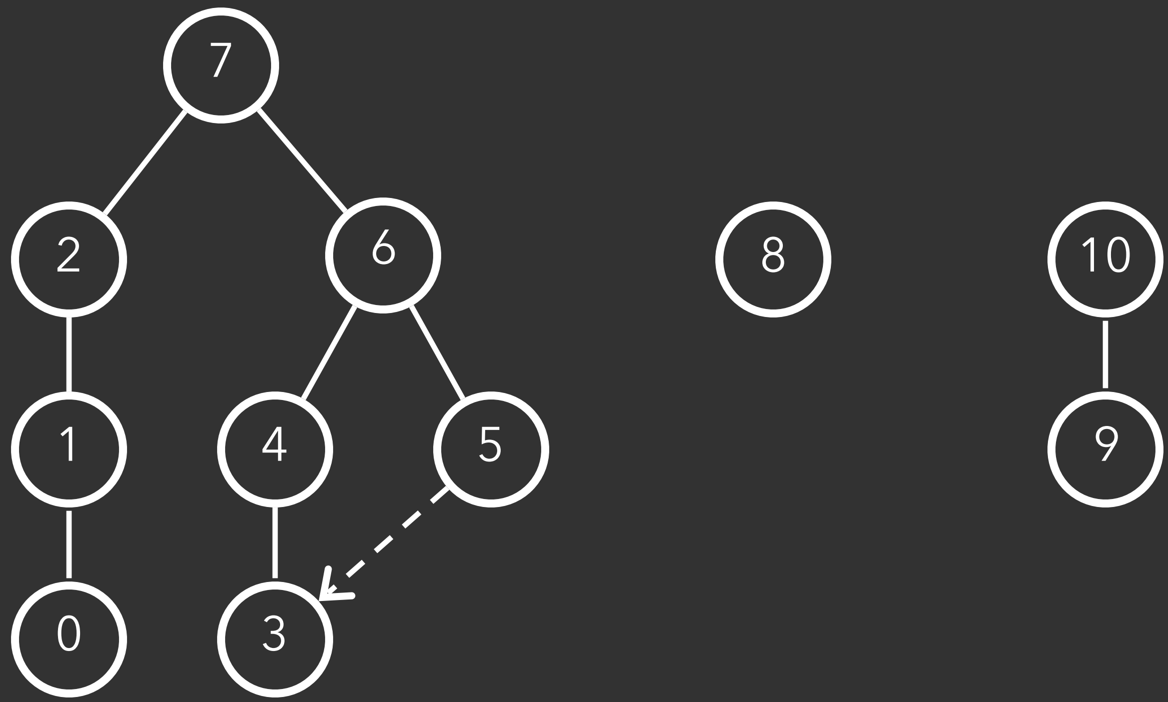
# Reingold-Tilford Layout



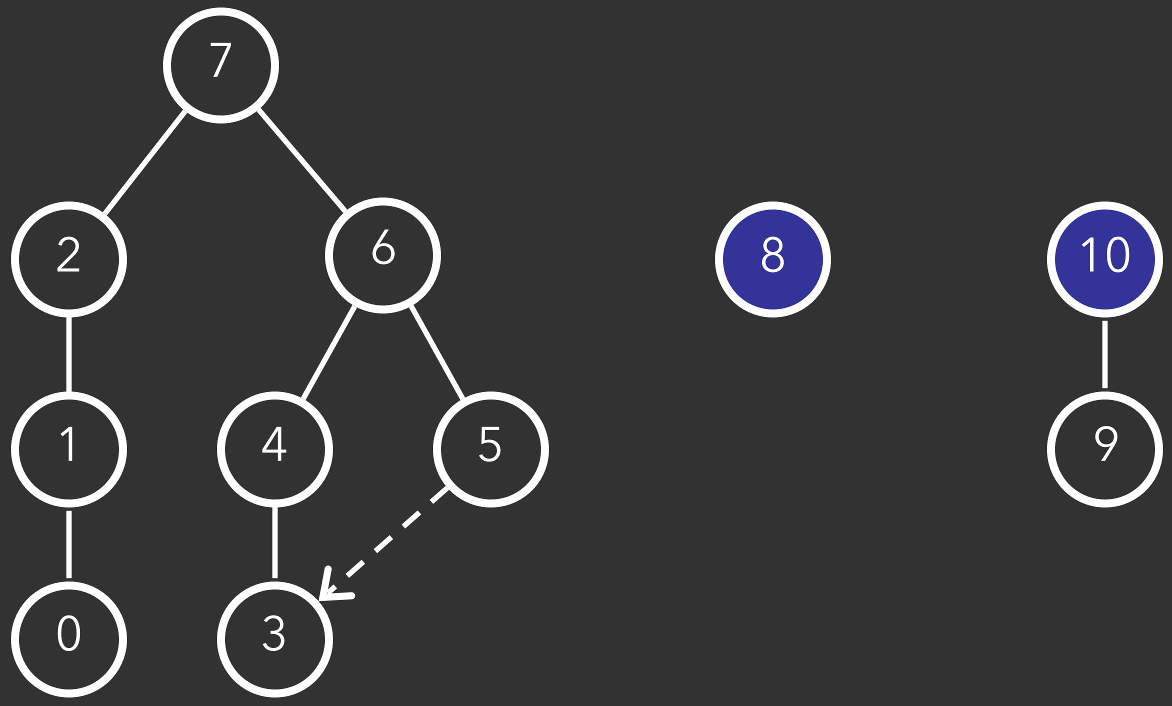
# Reingold-Tilford Layout



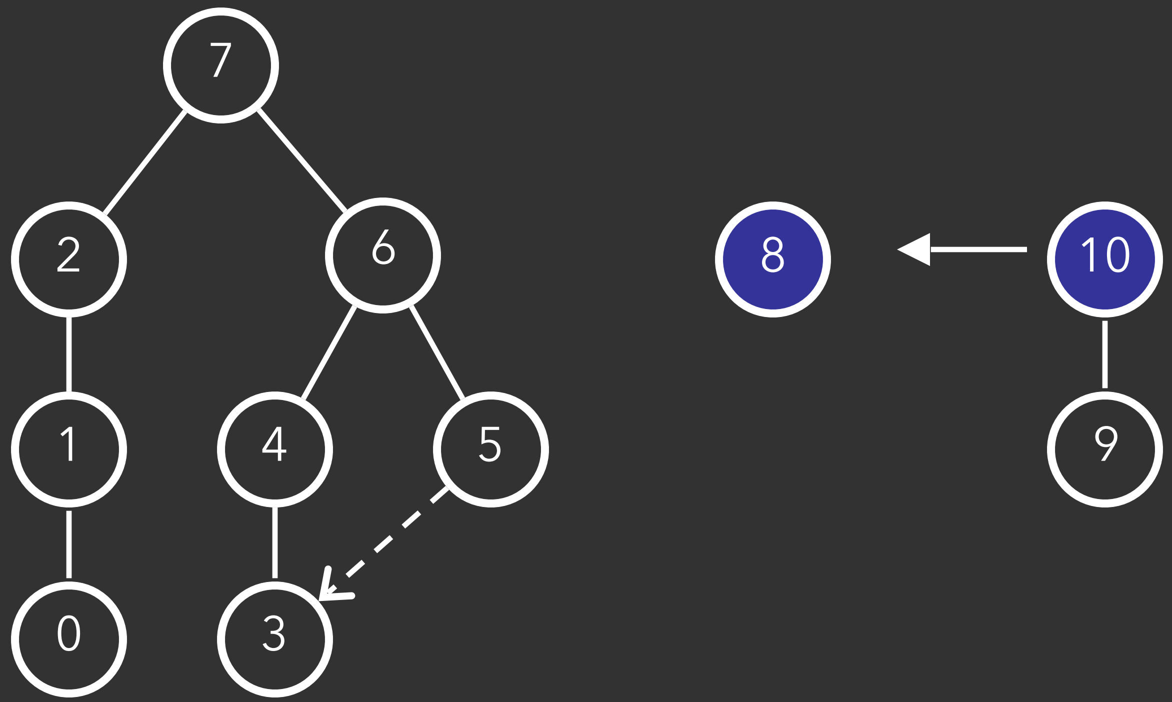
# Reingold-Tilford Layout



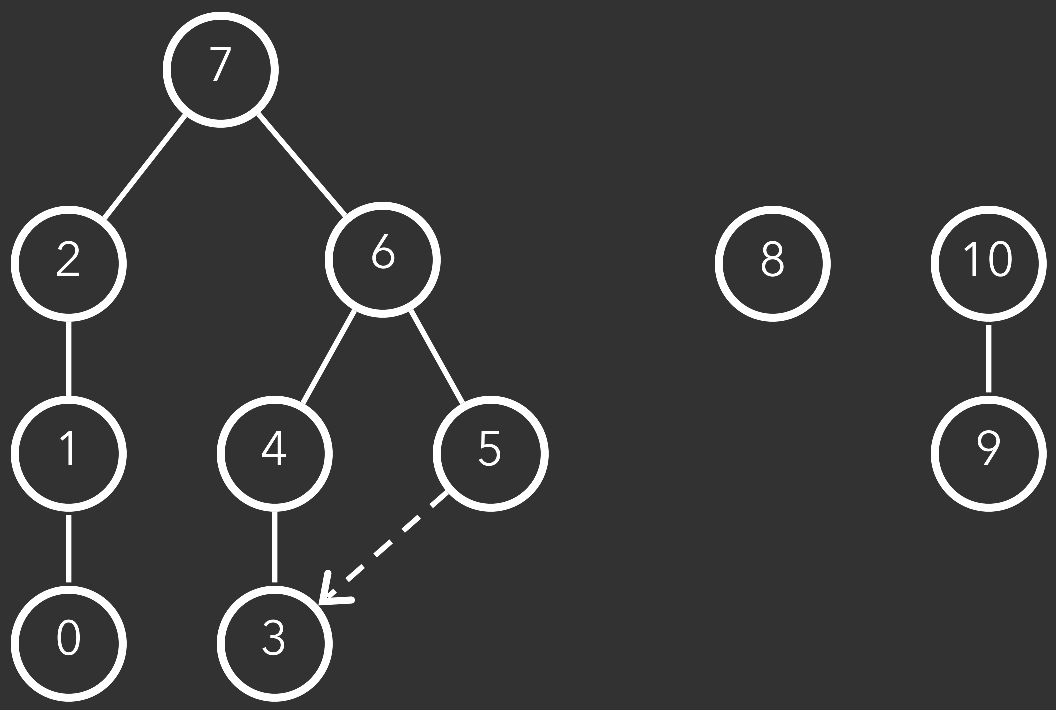
# Reingold-Tilford Layout



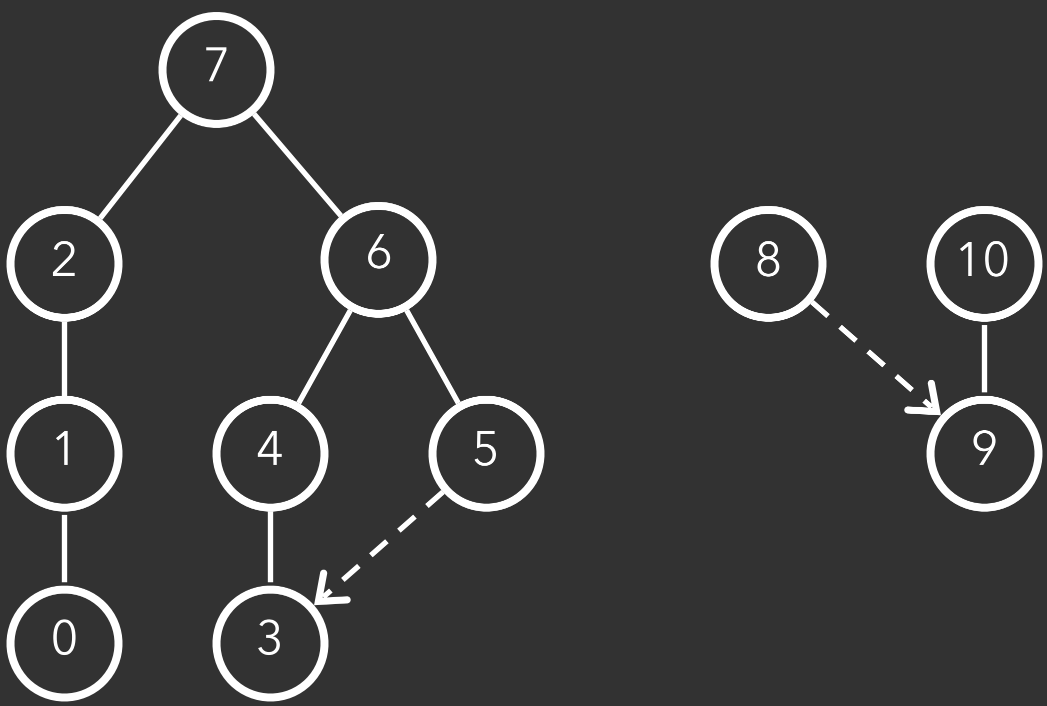
# Reingold-Tilford Layout



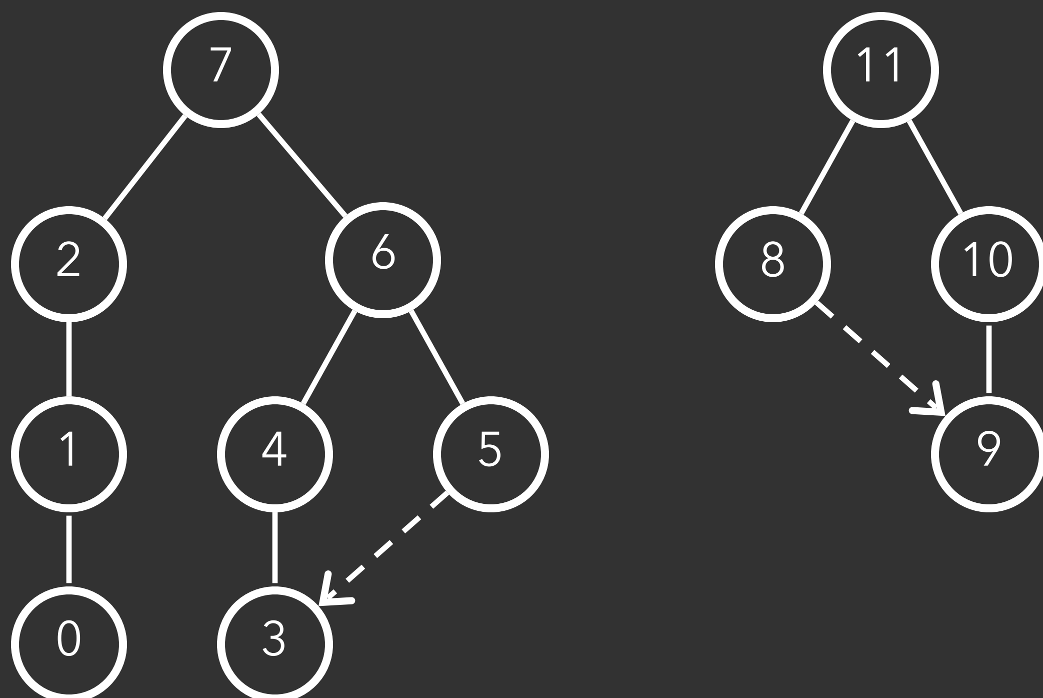
# Reingold-Tilford Layout



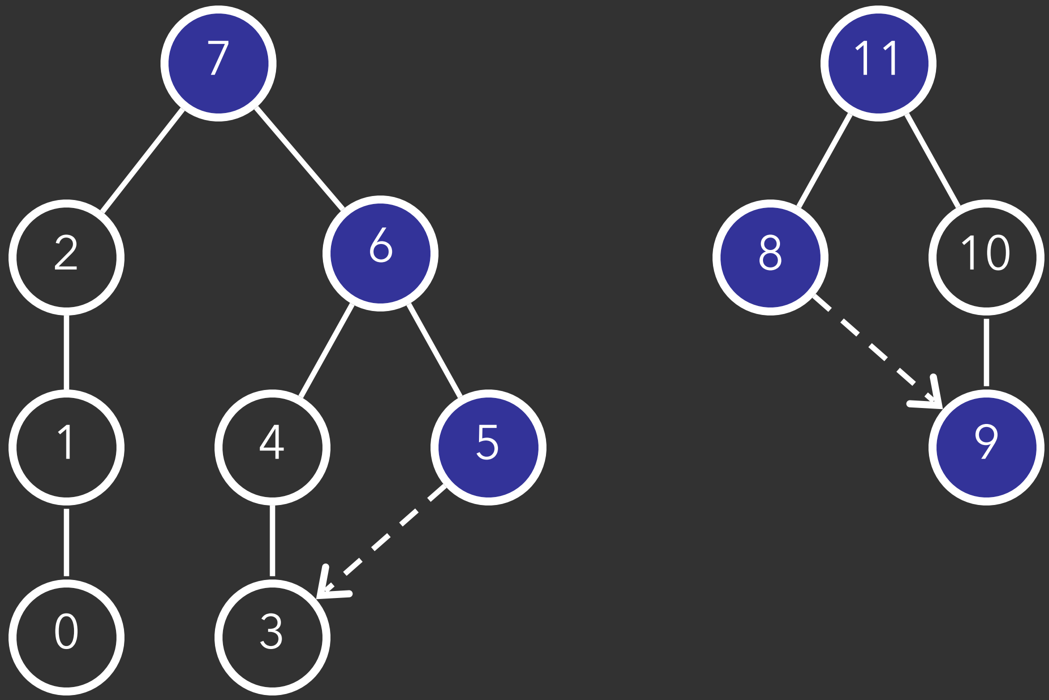
# Reingold-Tilford Layout



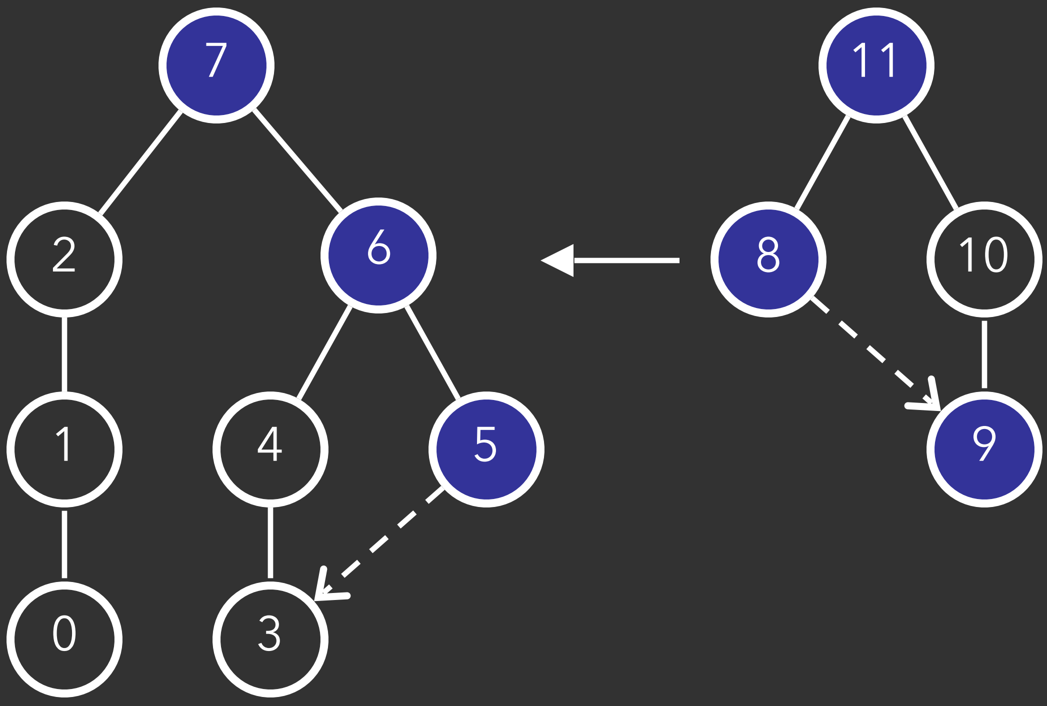
# Reingold-Tilford Layout



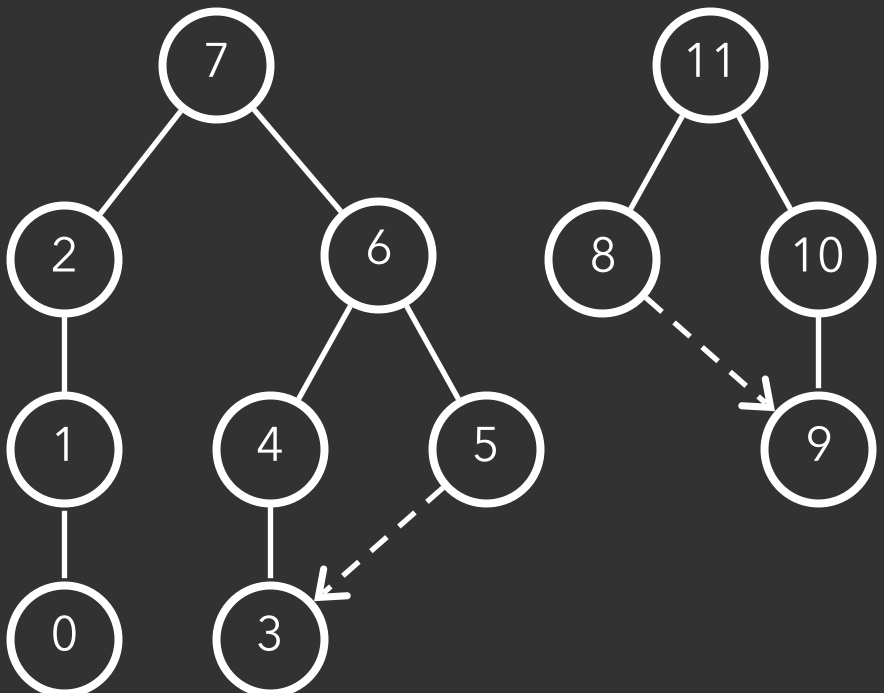
# Reingold-Tilford Layout



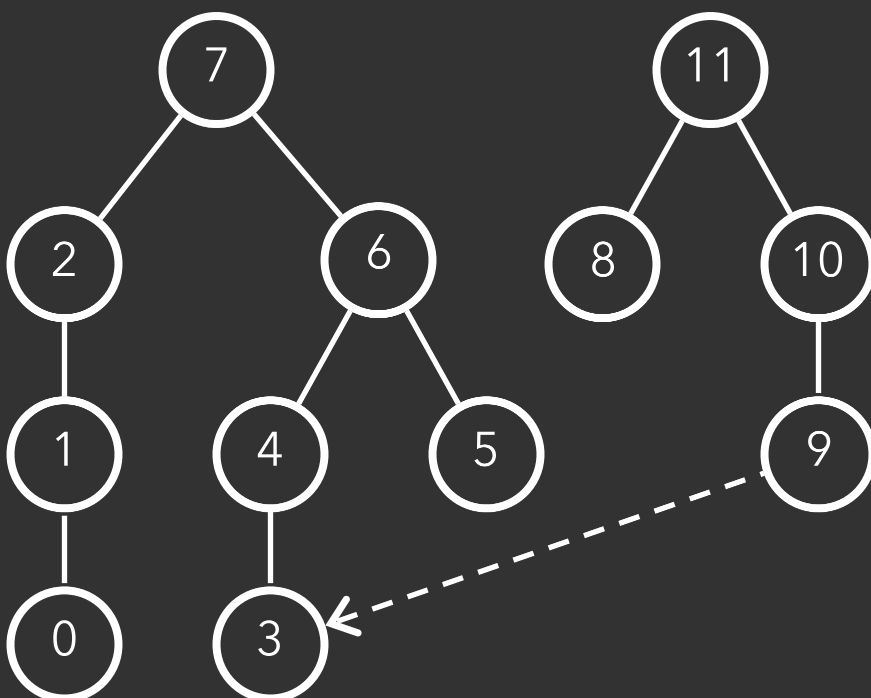
# Reingold-Tilford Layout



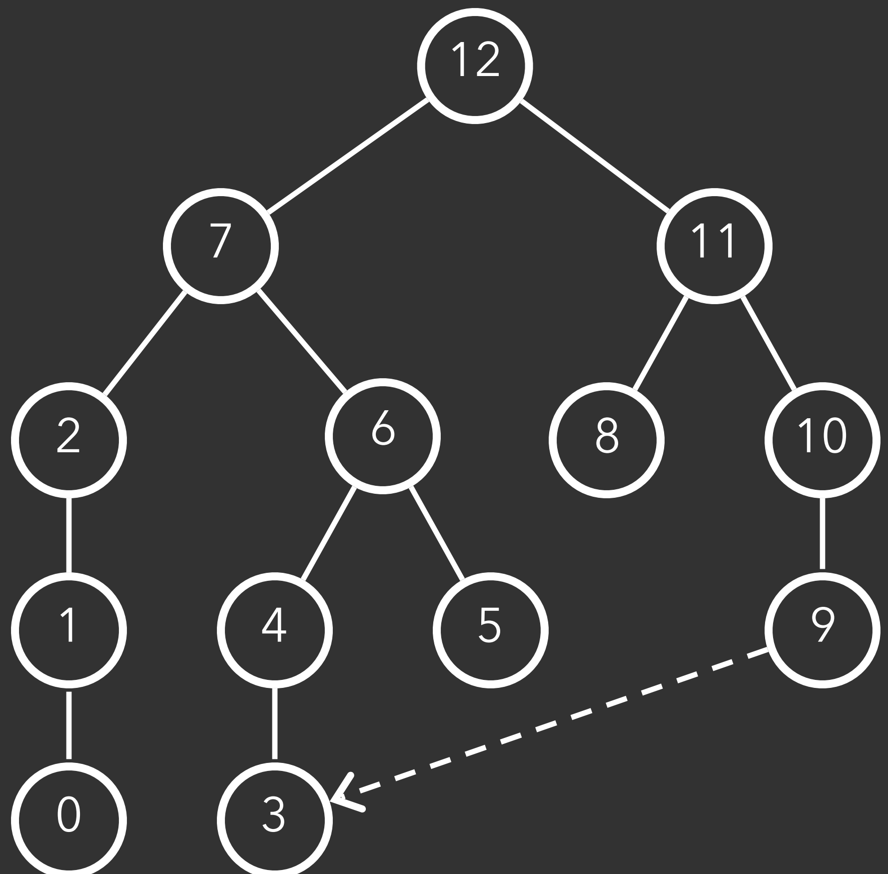
# Reingold-Tilford Layout



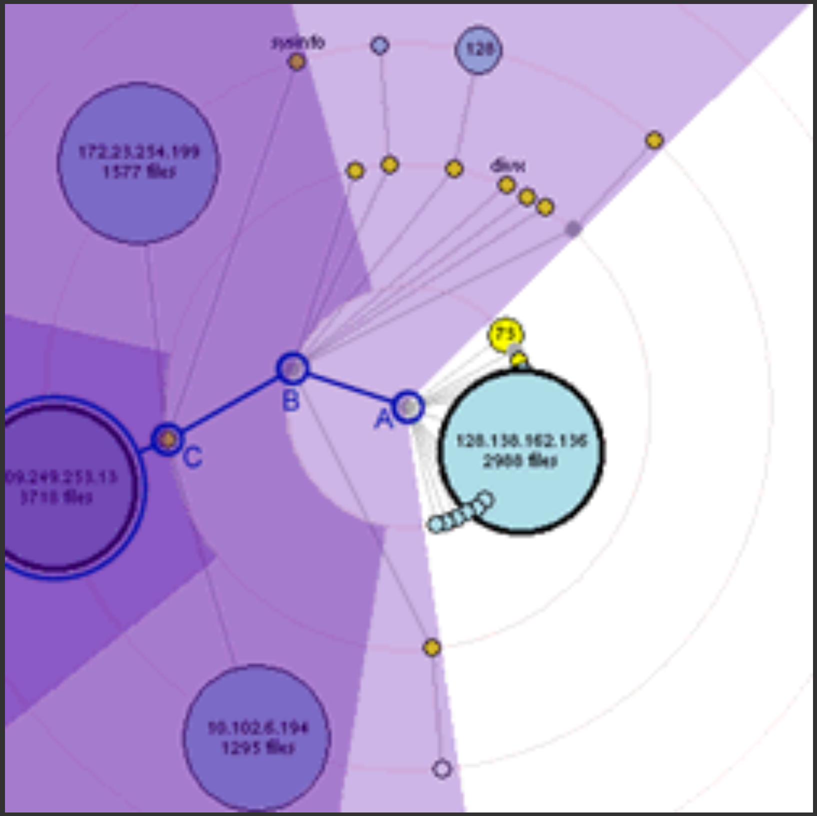
# Reingold-Tilford Layout



# Reingold-Tilford Layout



# Radial Layout



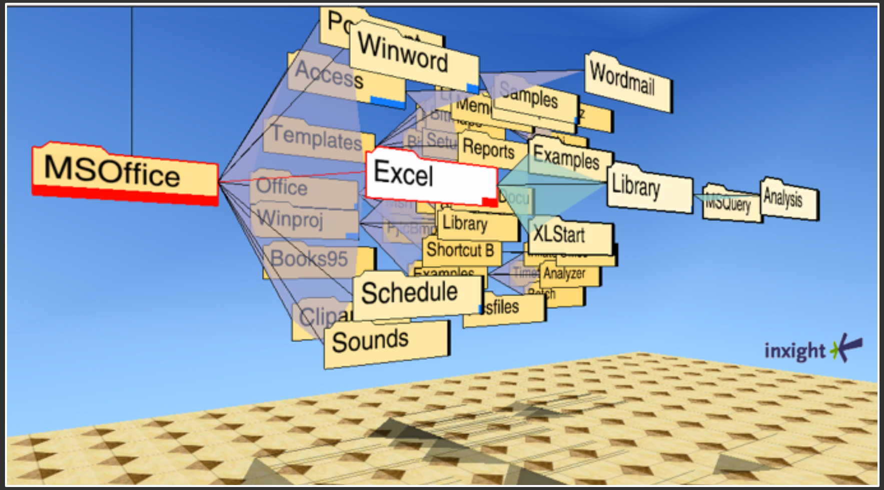

Node-link diagram in polar co-ordinates.

Radius encodes depth, with root in the center.

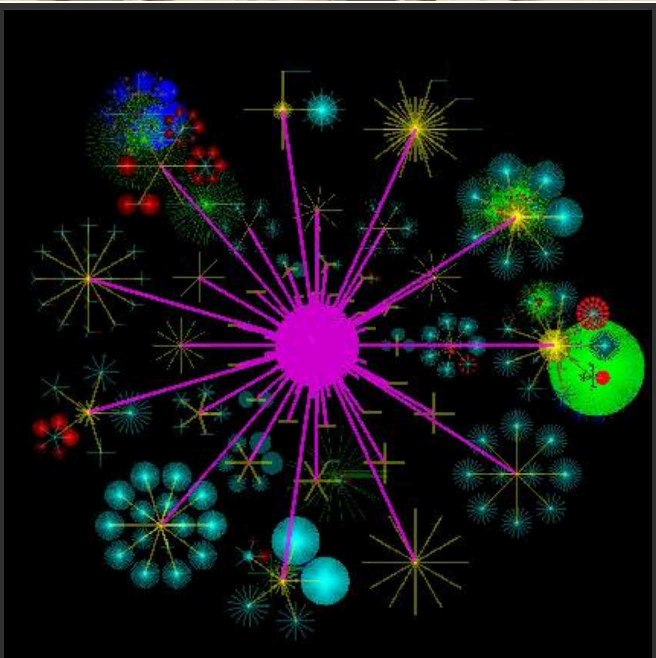
Angular sectors assigned to subtrees (typically uses recursive approach).

Reingold-Tilford method could be applied here.

# Circular Tree Layouts



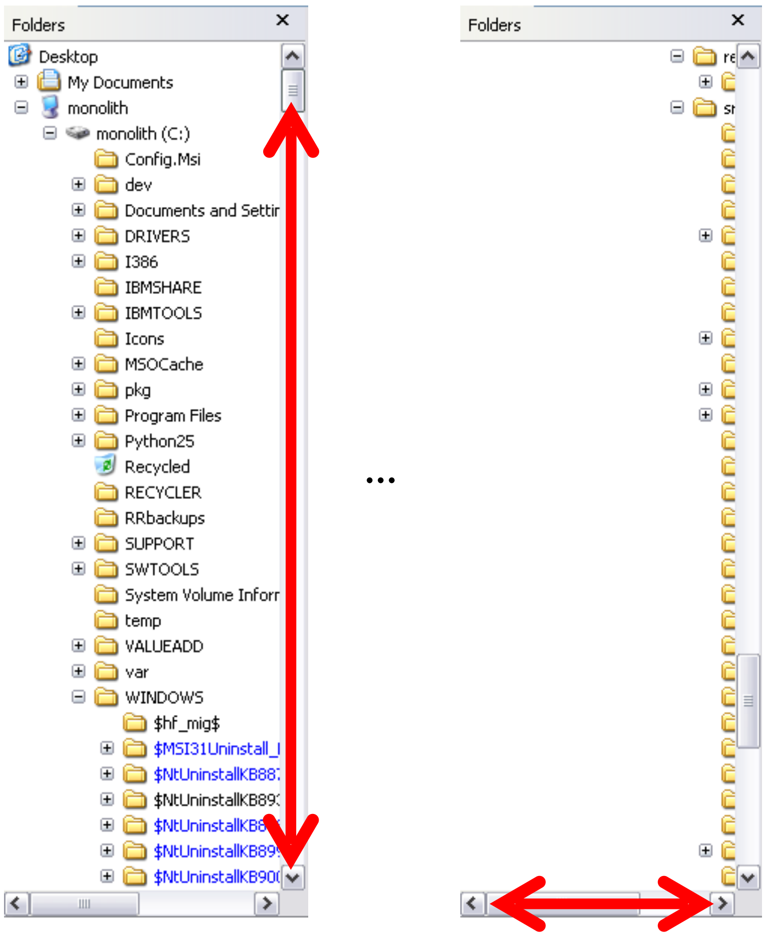

Layout in 3D to form Cone Trees.

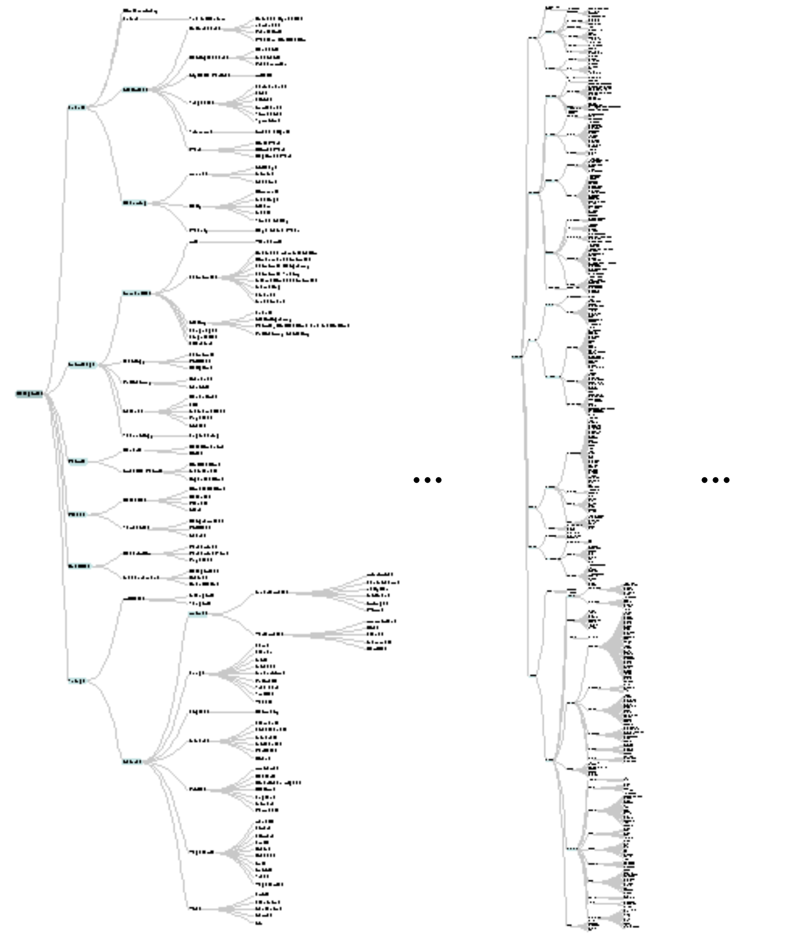


Balloon Trees can be described as a 2D variant of a Cone Tree. Not just a flattening process, as circles must not overlap.

# 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

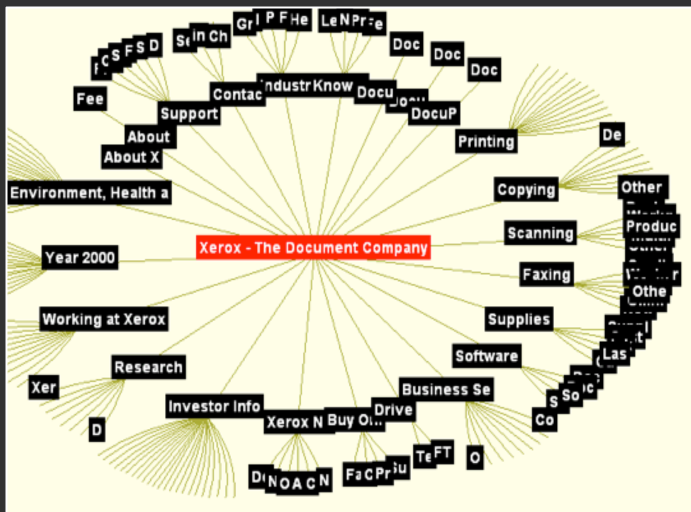
Scrolling or Panning

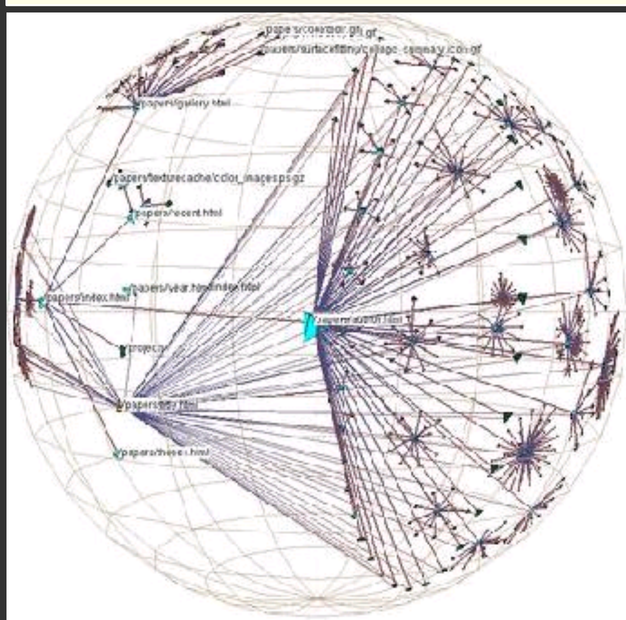
Zooming

Aggregation

MC Escher, *Circle Limit IV*

# Hyperbolic Layout



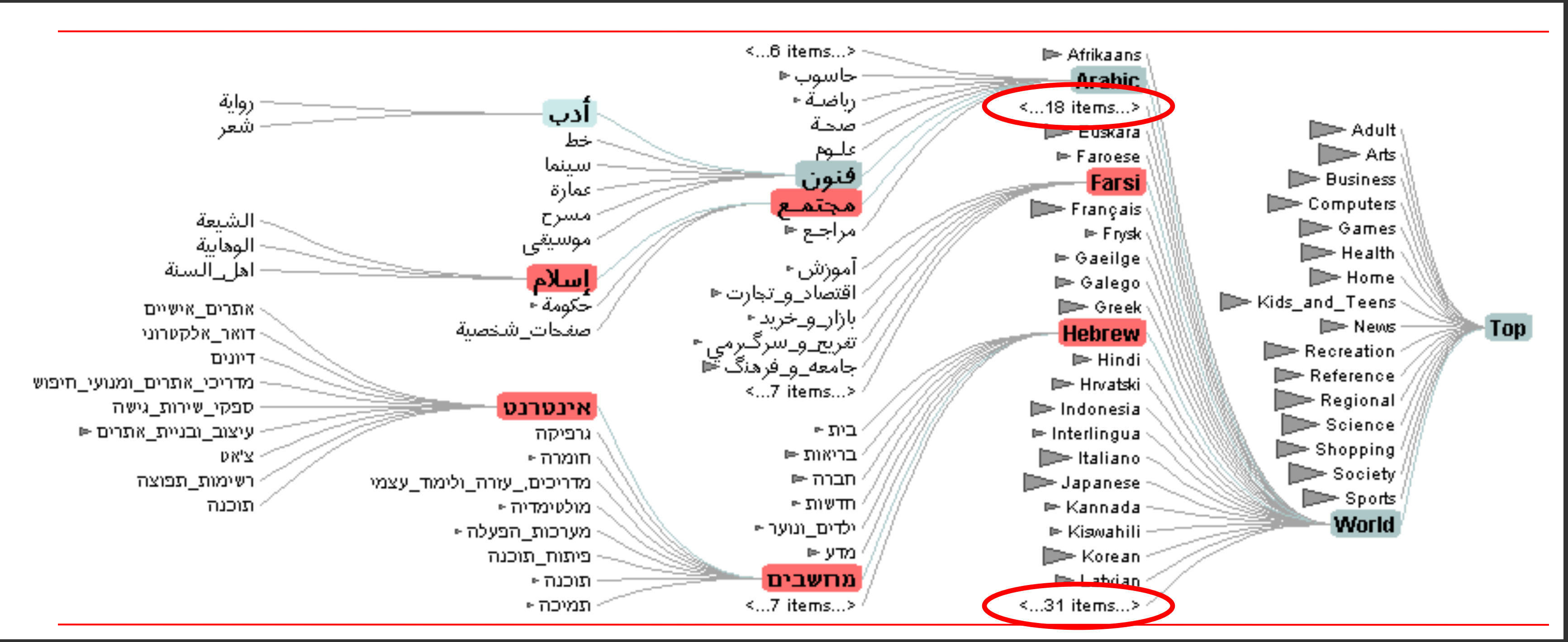



Perform tree layout in hyperbolic geometry, project the result on to the Euclidean plane.

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

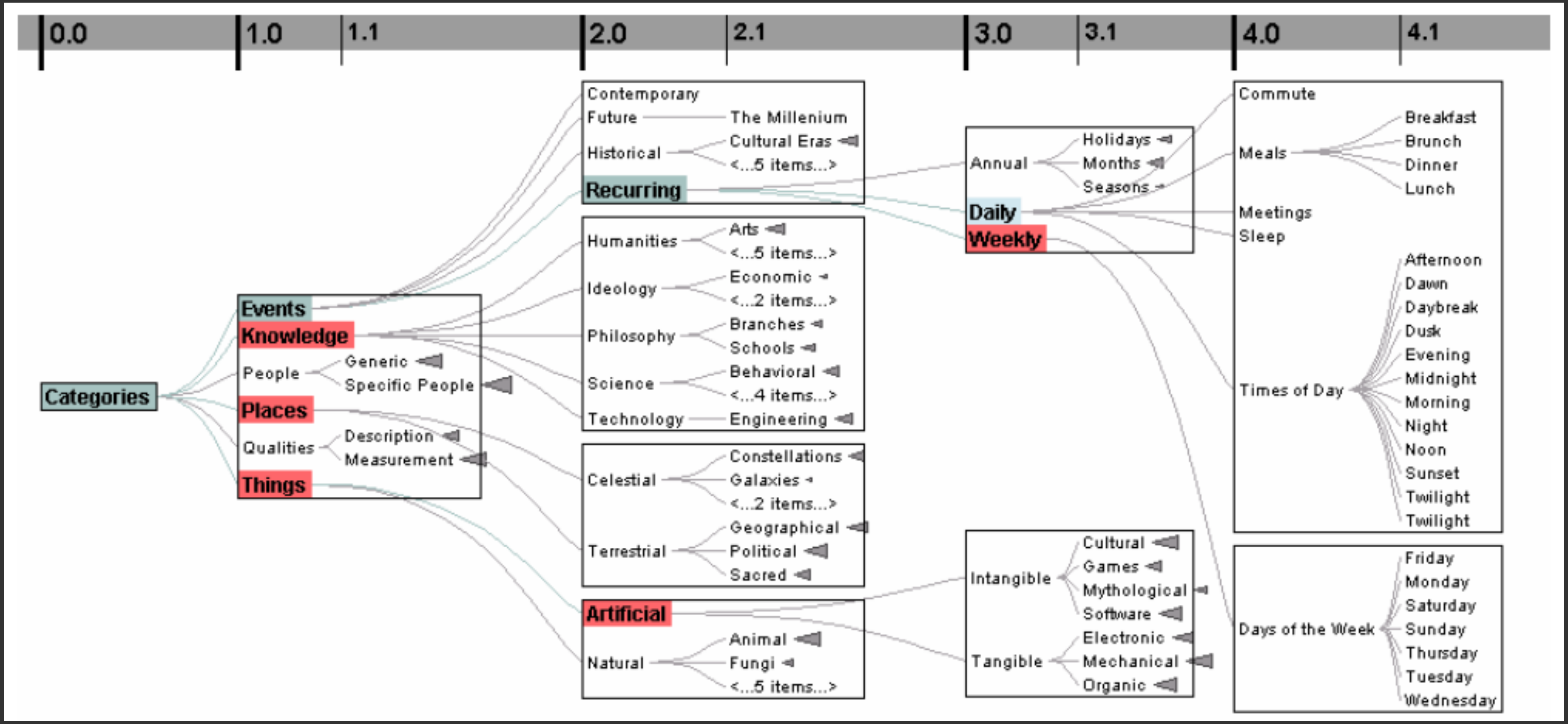
Also computable in 3D, projected into a sphere.

# Degree-of-Interest Trees




Space-constrained, multi-focal tree layout

# Degree-of-Interest Trees



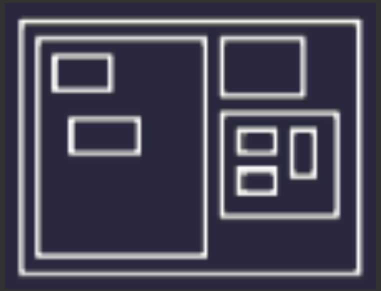

Cull “un-interesting” nodes on a per block basis until all blocks on a level fit within bounds.

Attempt to center child blocks beneath parents.

# Enclosure / Layering

# Enclosure Diagrams

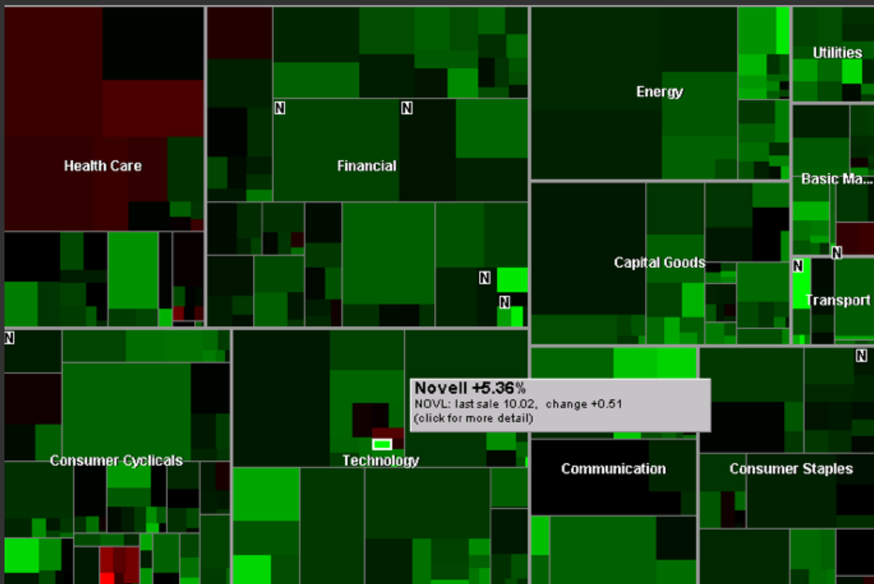
Encode structure using **spatial enclosure**
Popularly known as **treemaps**

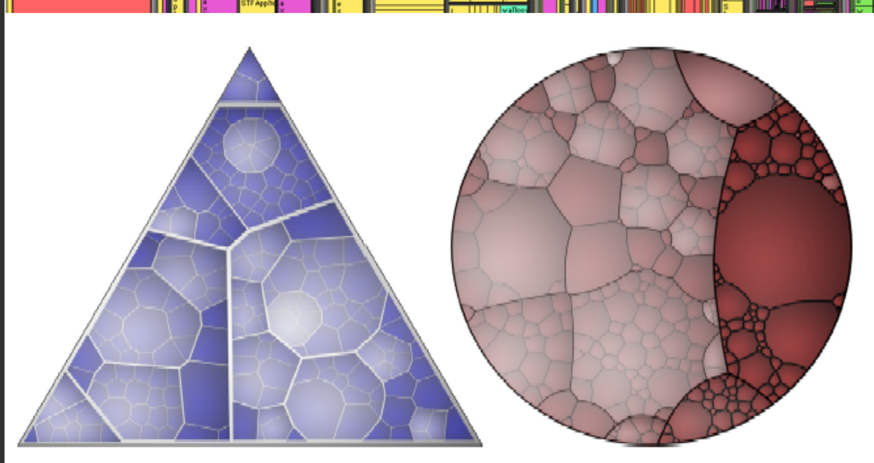


**Benefits**
Provides a single view of an entire tree
Easier to spot large/small nodes

**Problems**
Difficult to accurately read structure / depth

# Treemaps



Recursively fill space. Enclosure signifies hierarchy.

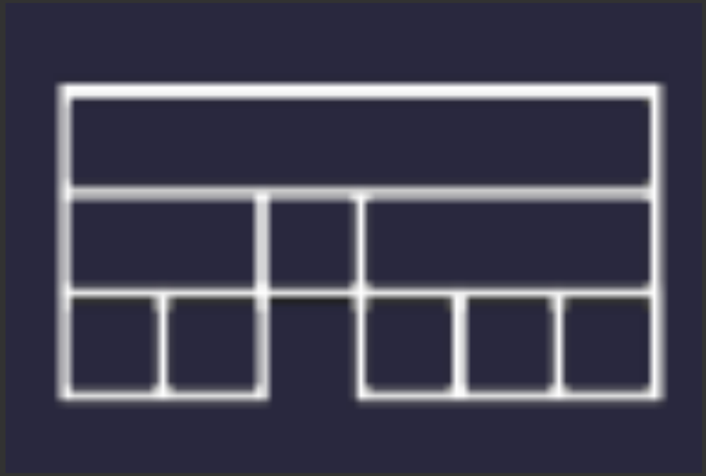
Additional measures can be taken to control aspect ratio of cells.

Often uses rectangles, but other shapes are possible, e.g., iterative Voronoi tesselation.

# Layered Diagrams

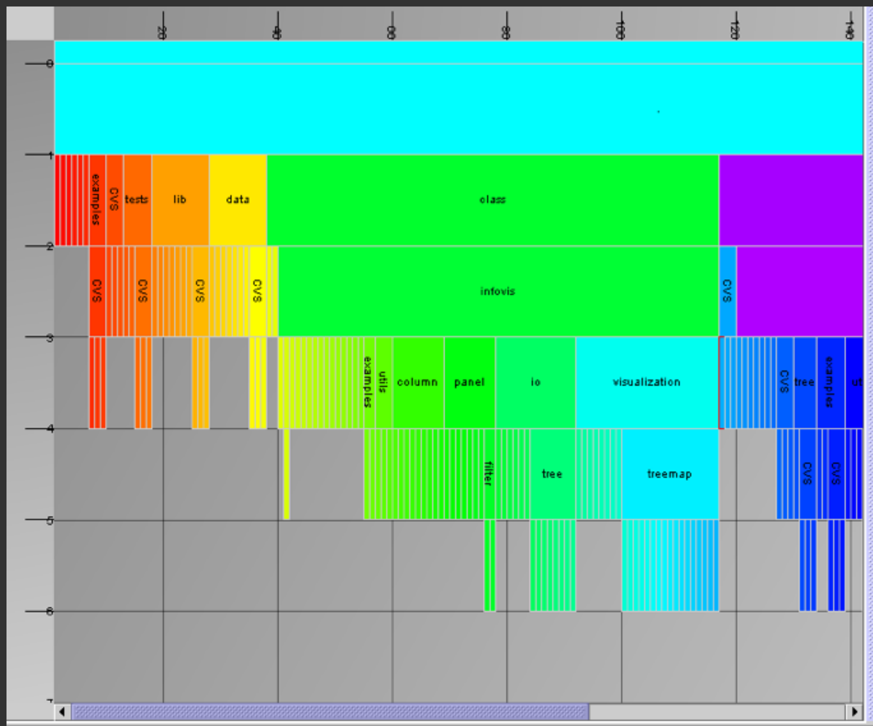
Signify tree structure using

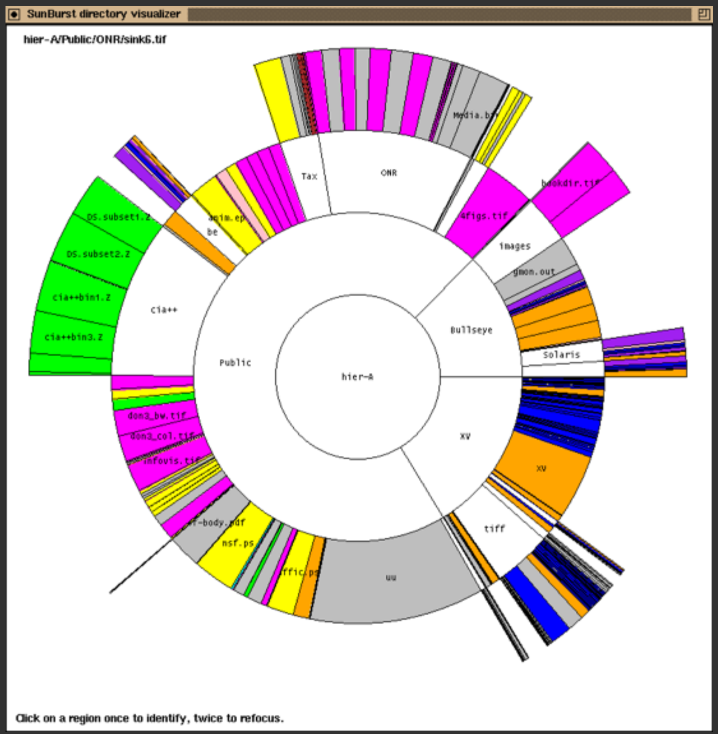
- Layering
- Adjacency
- Alignment



Involves recursive sub-division of space.

# Icicle & Sunburst Trees

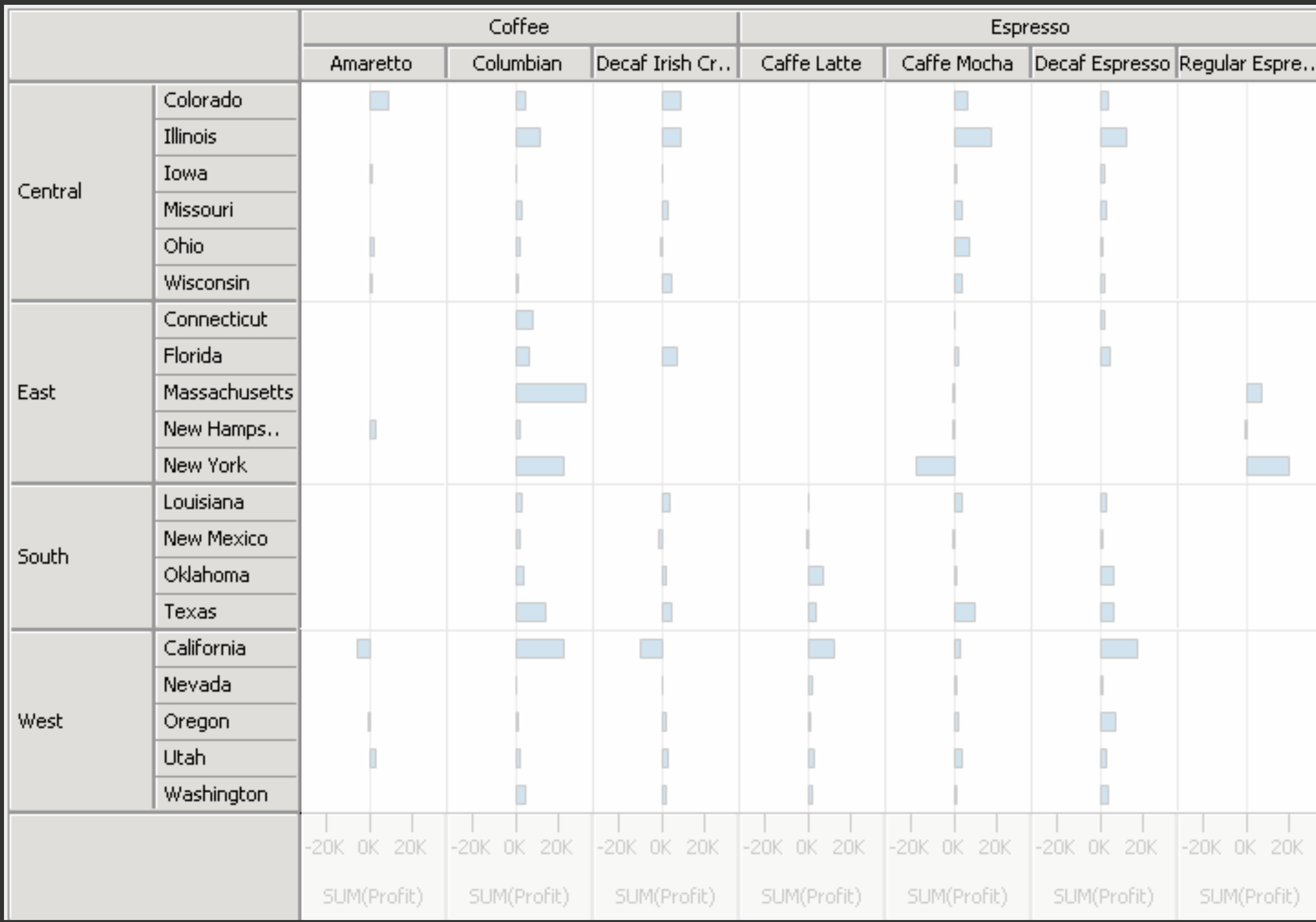


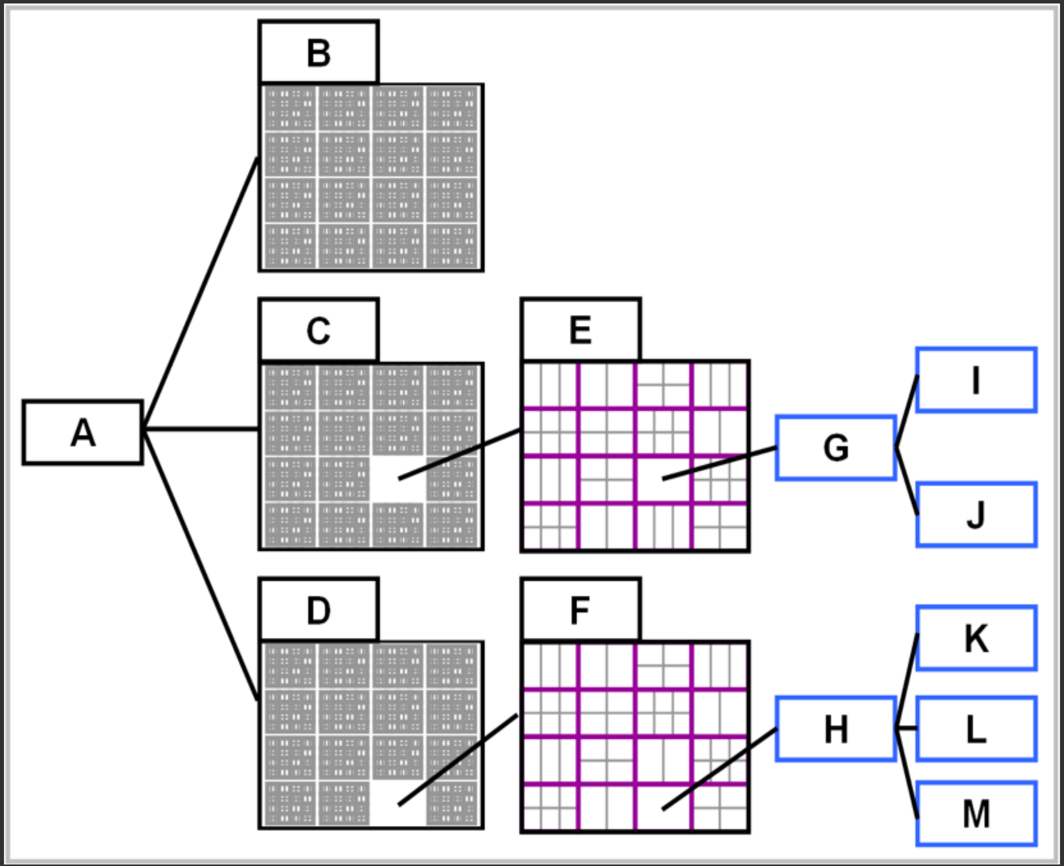

Higher-level nodes get a larger layer area, whether that is horizontal or angular extent.

Child levels are layered, constrained to parent's extent

# Layered Tree Drawing



# Hybrids are also possible…




"Elastic Hierarchies"

Node-link diagram with treemap nodes.

# Administrivia

# Final Project Schedule

| | |
|---|---|
| ***Proposal*** | **Tues, May 10 (5pm)** |
| *Presentation* | Thur, May 19 (slides: 5/18, 5pm) |
| *Poster & Demo* | Tues, Jun 7 (5-8pm) |
| *Final Paper* | Thur, Jun 9 (8am) |

**Logistics**

Groups of up to 4 people

Clearly report responsibilities of each member

# Graph Layout

# Approaches to Graph Drawing

**Calculation using Graph Structure**

Tree layout on spanning tree

Sugiyama-style (hierarchical) layout

Adjacency matrix layout

**Optimization Methods**

Constraint satisfaction

Force-directed layout

**Attribute-Driven Layout**

Layout using data attributes, not linkage

# Spanning Tree Layout

# Spanning Tree Layout

**Many graphs have useful spanning trees**

Websites, Social Networks

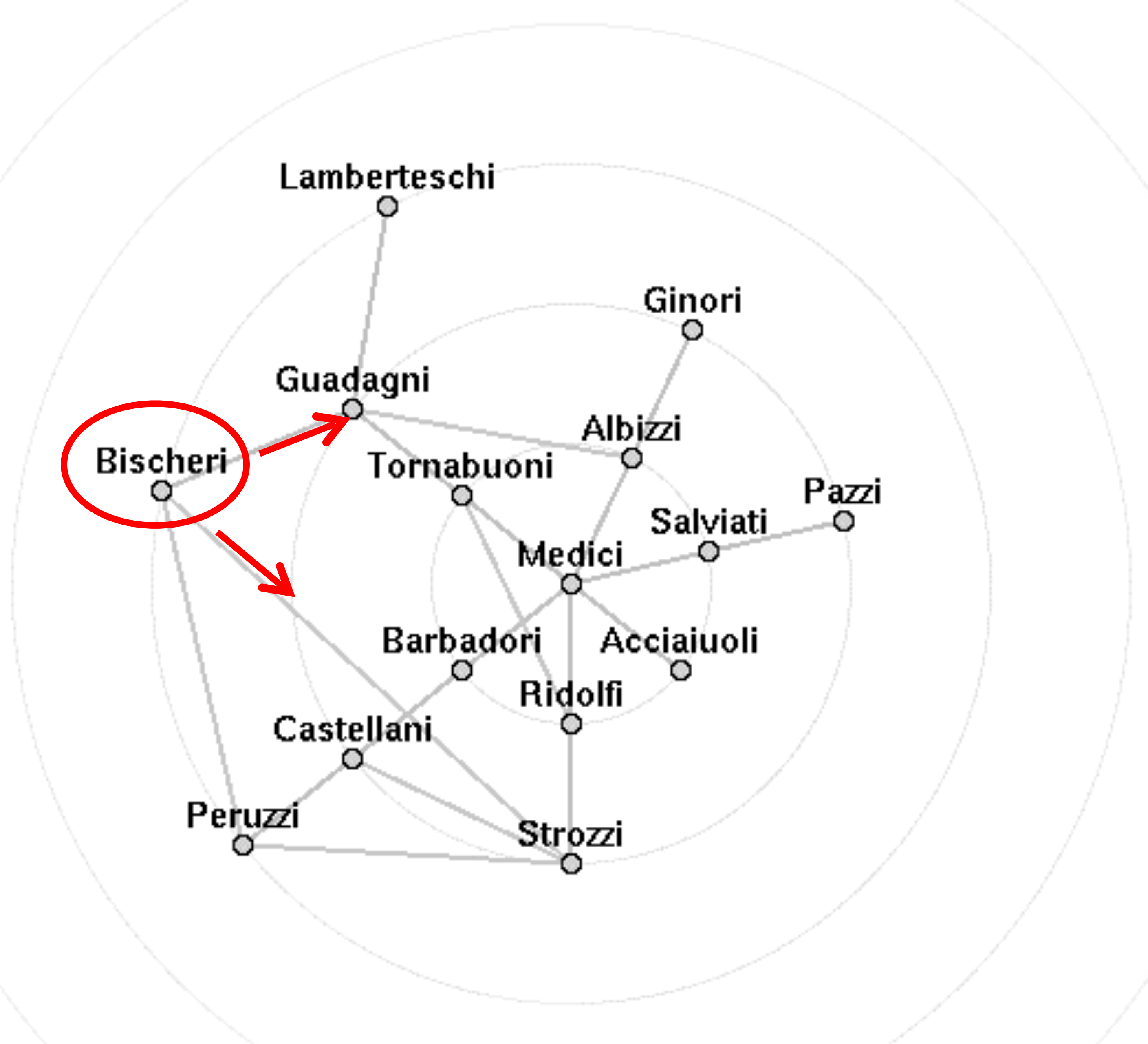
**Use tree layout on spanning tree of graph**

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



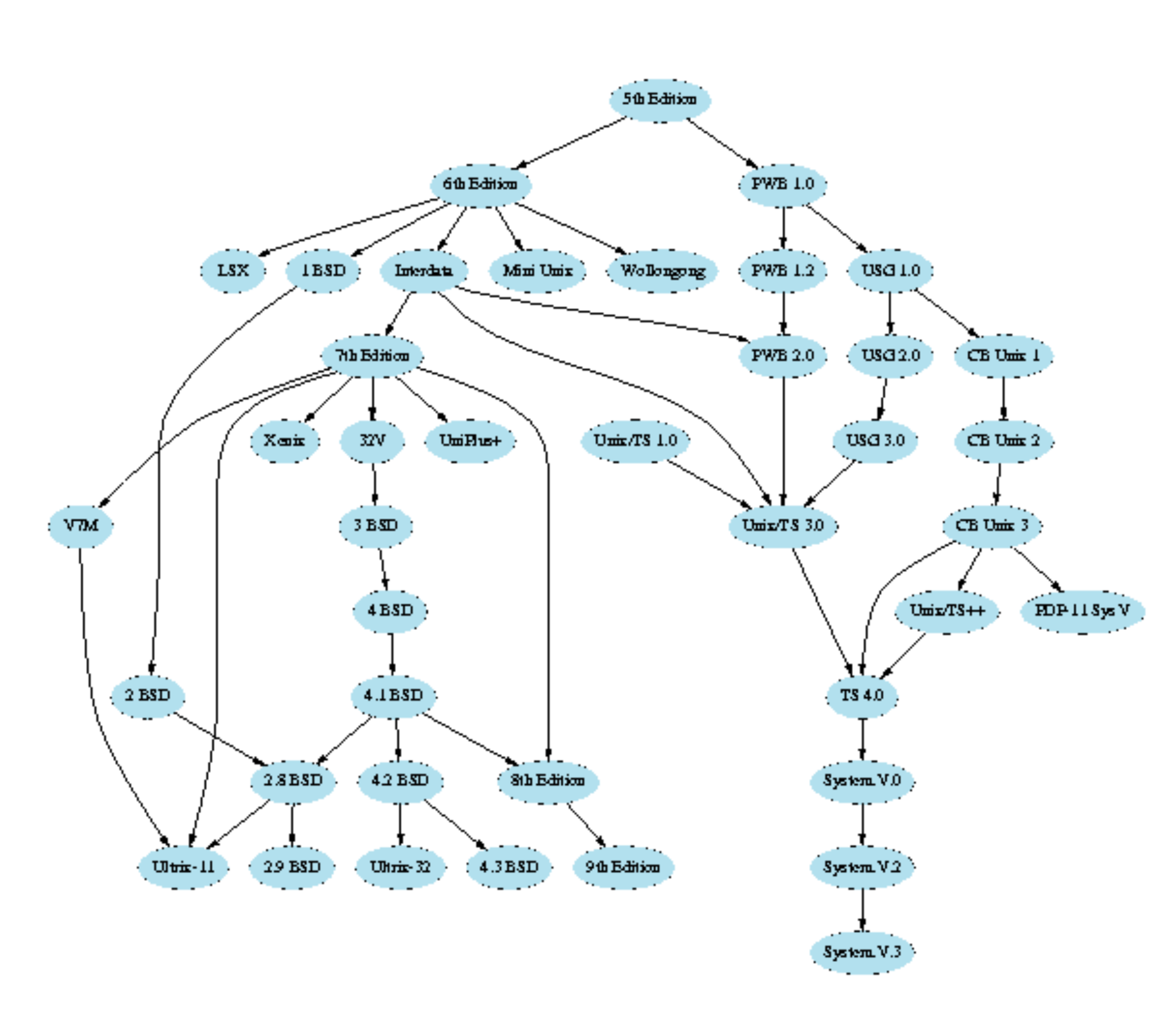

Spanning tree layout may result in arbitrary parent node

# Sugiyama-Style Layout

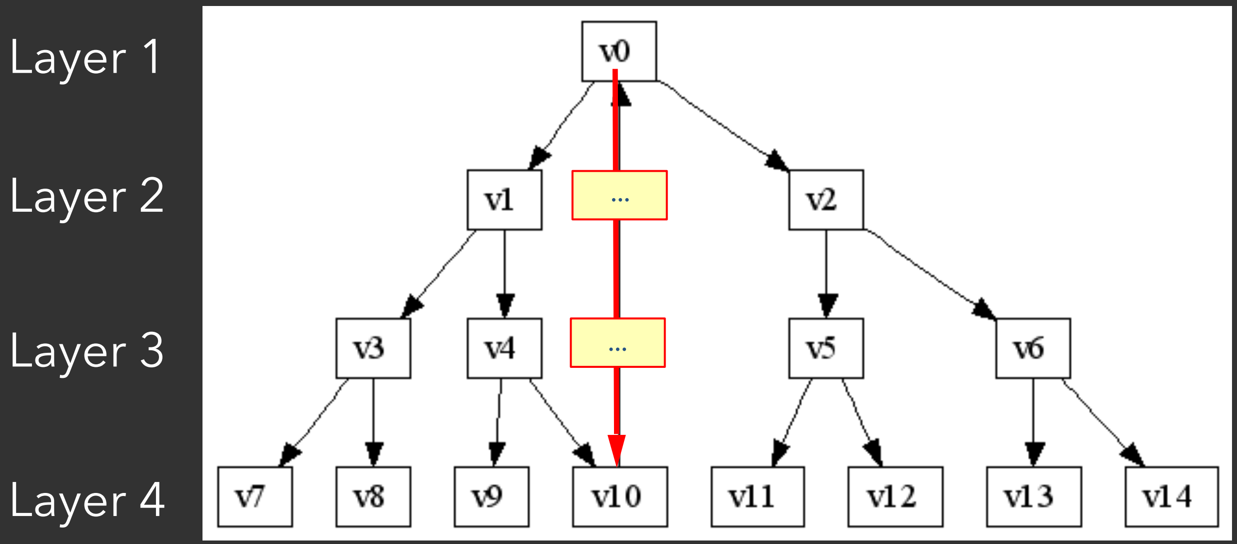
# Sugiyama-style Layout

Evolution of the UNIX operating system

Hierarchical layering based on descent



# Sugiyama-style Layout




Reverse edges to remove cycles

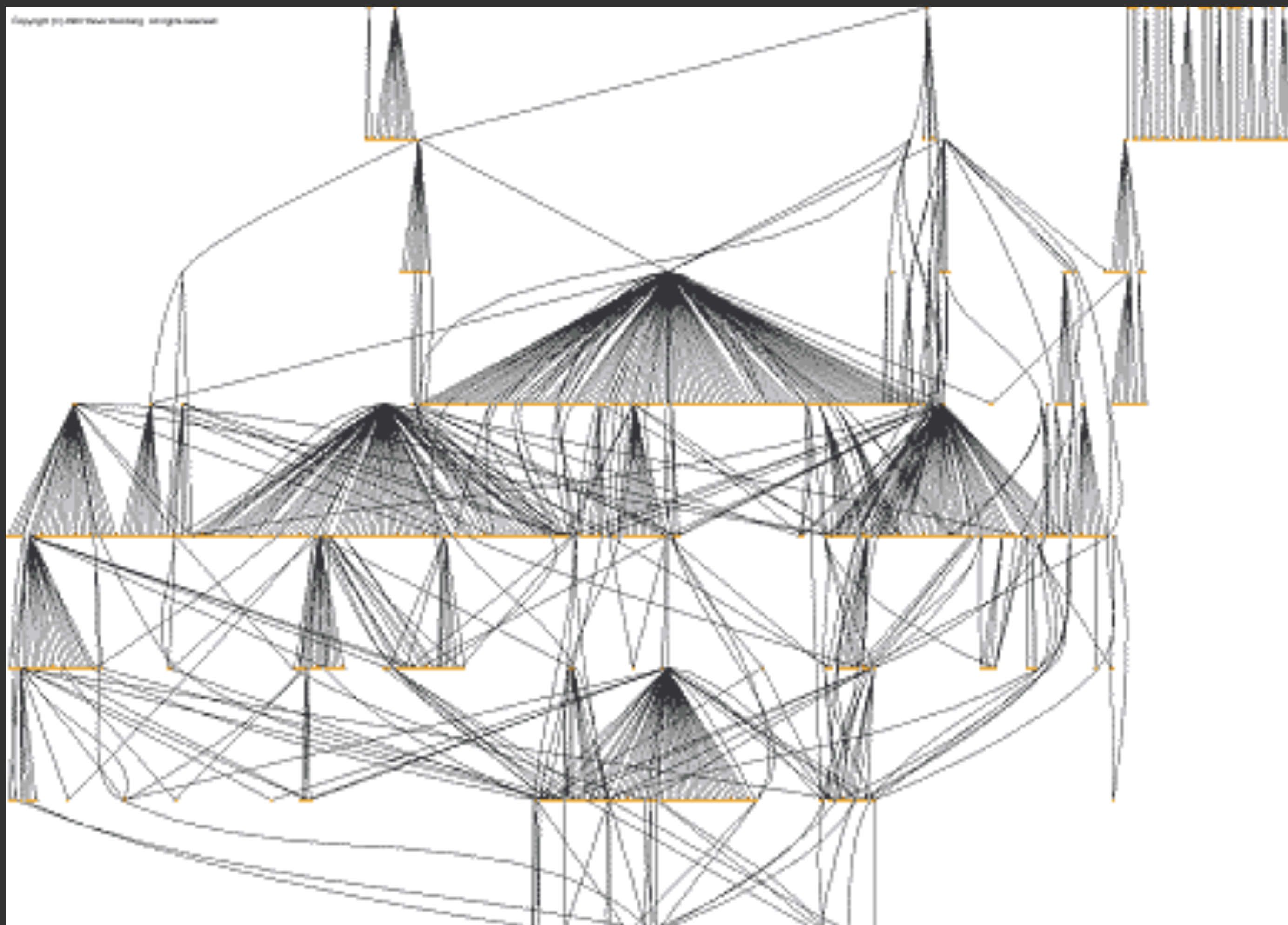
Assign nodes to hierarchy layers

Create dummy nodes to “fill in” missing layers

Arrange nodes within layer, minimize edge crossings

Route edges – layout splines if needed

# Hierarchical Layout



Gnutella network

# Force-Directed Layout

# Optimization Techniques

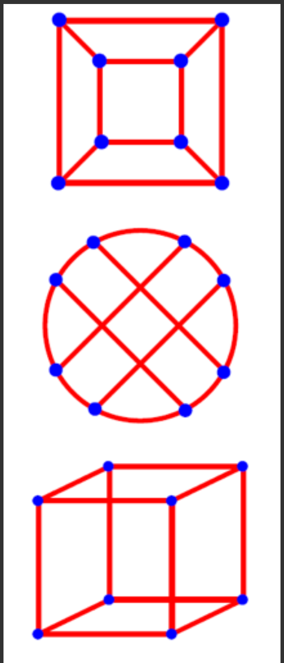
**Treat layout as an *optimization problem***
Define layout using an *energy model* along with *constraints*: equations the layout should obey.
Use optimization algorithms to solve

**Commonly posed as a physical system**
Charged particles, springs, drag force, …

**We can introduce directional constraints**
*DiG-CoLa* (Di-Graph Constr Optimization Layout) [Dwyer 05]
Iterative constraint relaxation

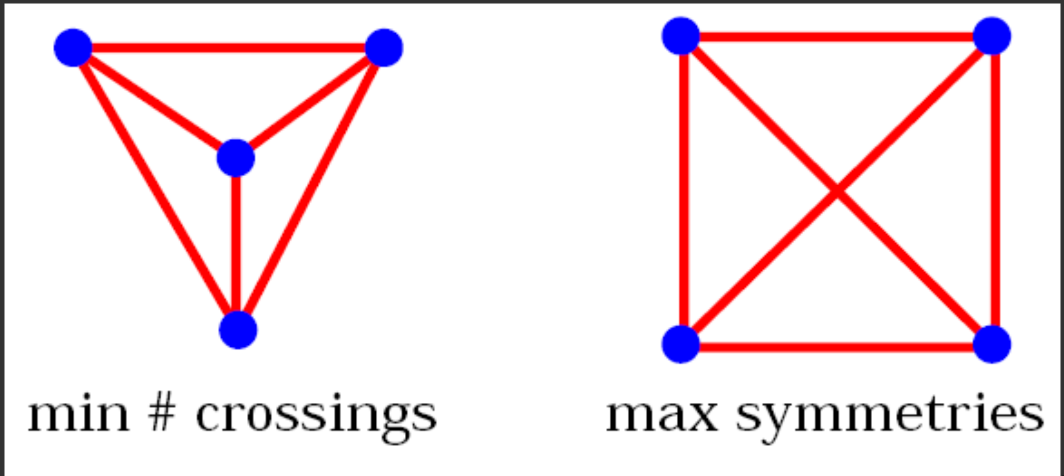
# Optimizing Aesthetic Constraints

Minimize edge crossings
Minimize area
Minimize line bends
Minimize line slopes
Maximize smallest angle between edges
Maximize symmetry



but, can't do it all.

Optimizing these criteria is often NP-Hard, requiring approximations.



# Force-Directed Layout

Nodes = charged particles $F = G*m_1*m_2 / (x_i - x_j)^2$

with air resistance $F = -b * v_i$

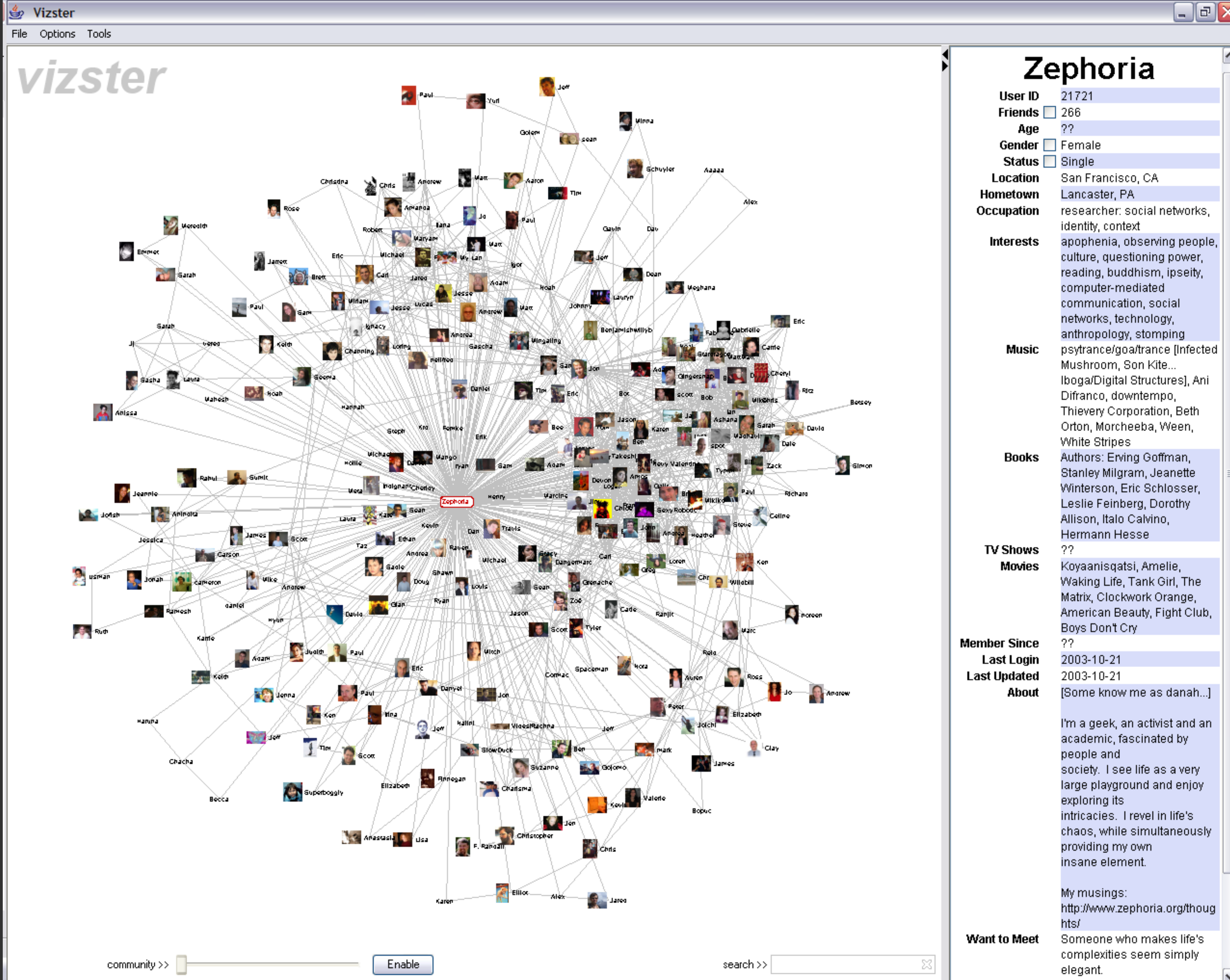
Edges = springs $F = -k * (x_i - x_j - L)$

**Iteratively calculate forces, update node positions**

Naïve n-body calculation is $O(N^2)$

O(N log N) using quadtree or k-d tree

Numerical integration of forces at each time step


Vizster
File
Options
Tools
vizster
Zephoria
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.
community >>
Enable
search >>

# Constrained Optimization

**Minimize stress function**

$\text{stress}(X) = \Sigma_{i<j}\, w_{ij}\, (\, \|X_i - X_j\| - d_{ij}\,)^2$

X: node positions, d: optimal edge length,

w: normalization constants

*Says: Try to place nodes $d_{ij}$ apart*

# Constrained Optimization

**Minimize stress function**

$\text{stress}(X) = \Sigma_{i<j}\, w_{ij} \,(\, \|X_i - X_j\| - d_{ij}\,)^2$

X: node positions, d: optimal edge length,

w: normalization constants

*Says: Try to place nodes $d_{ij}$ apart*

**Add hierarchy ordering constraints**

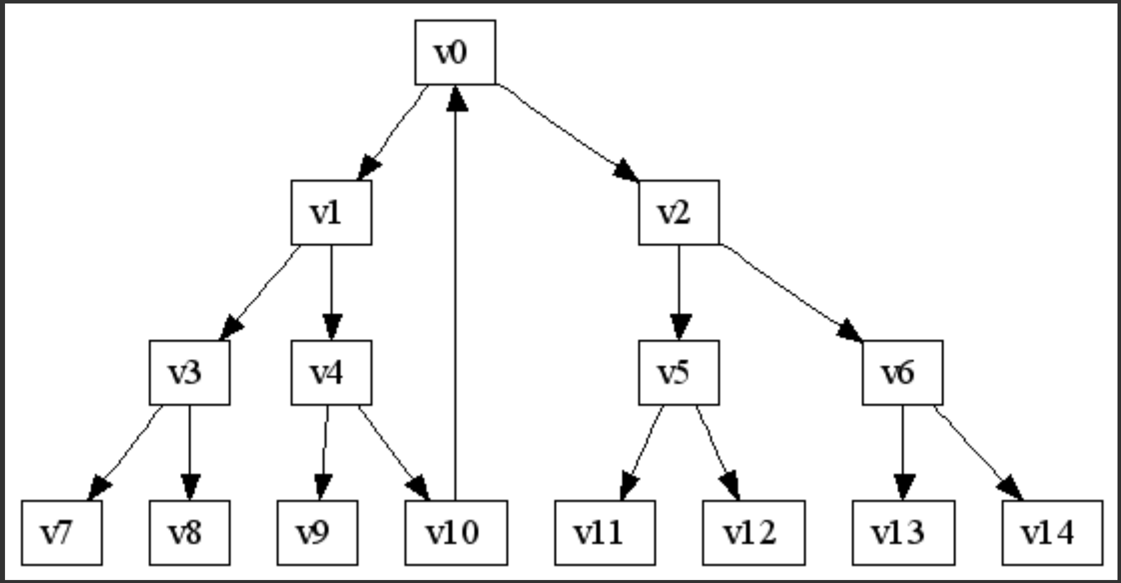
$E_H(y) = \Sigma_{(i,j)\in E}\, (\, y_i - y_j - \delta_{ij}\,)^2$

y: node y-coordinates

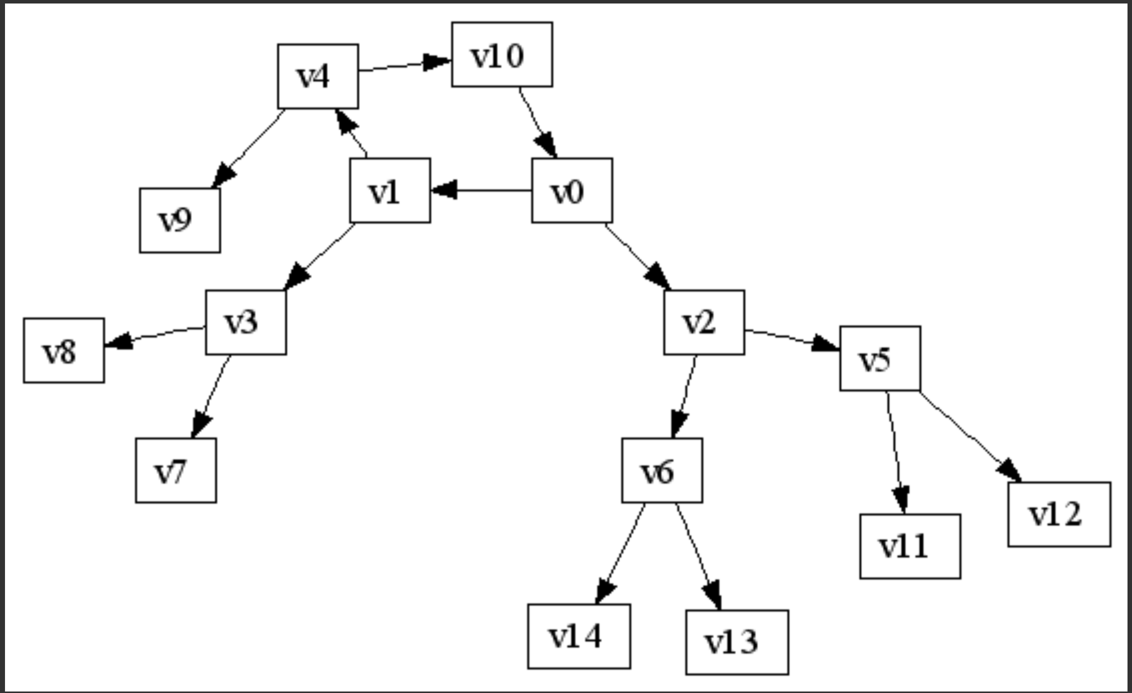
δ: edge direction (e.g., 1 for i->j, 0 for undirected)

*Says: If i points to j, it should have a lower y-value*

Sugiyama layout (dot)
Preserve tree structure

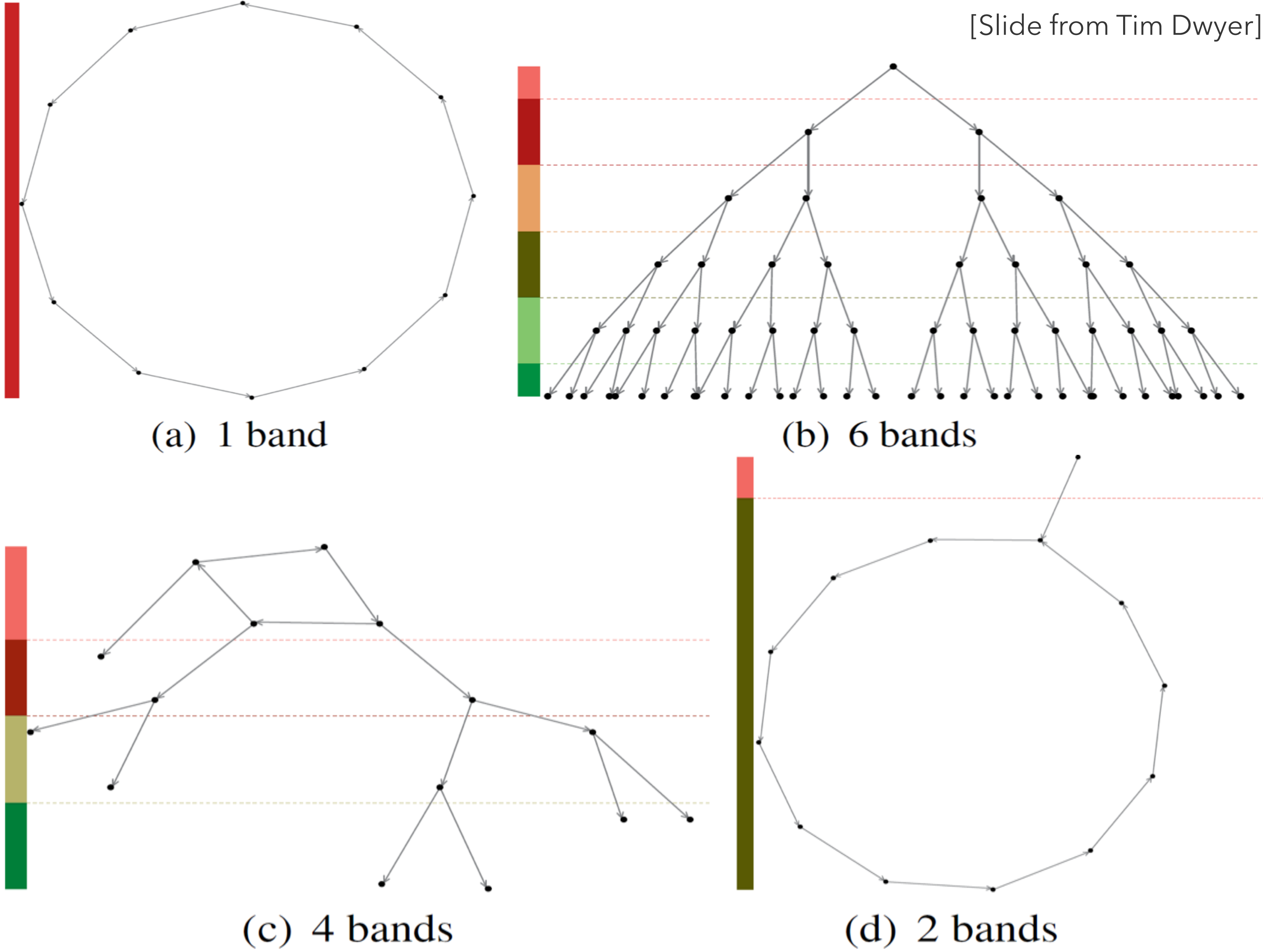



DiG-CoLa method
Preserve edge lengths




[Slide from Tim Dwyer]

[Slide from Tim Dwyer]



# Iterative Constraint Relaxation

Quadratic programming is complex to code and computationally costly. Is there a simpler way?

Iteratively **relax** each constraint [Dwyer 09]

- Given a constraint (e.g., $| x_i - x_j | = 5$)
- Simply push the nodes to satisfy!

Each relaxation may **clobber** prior results

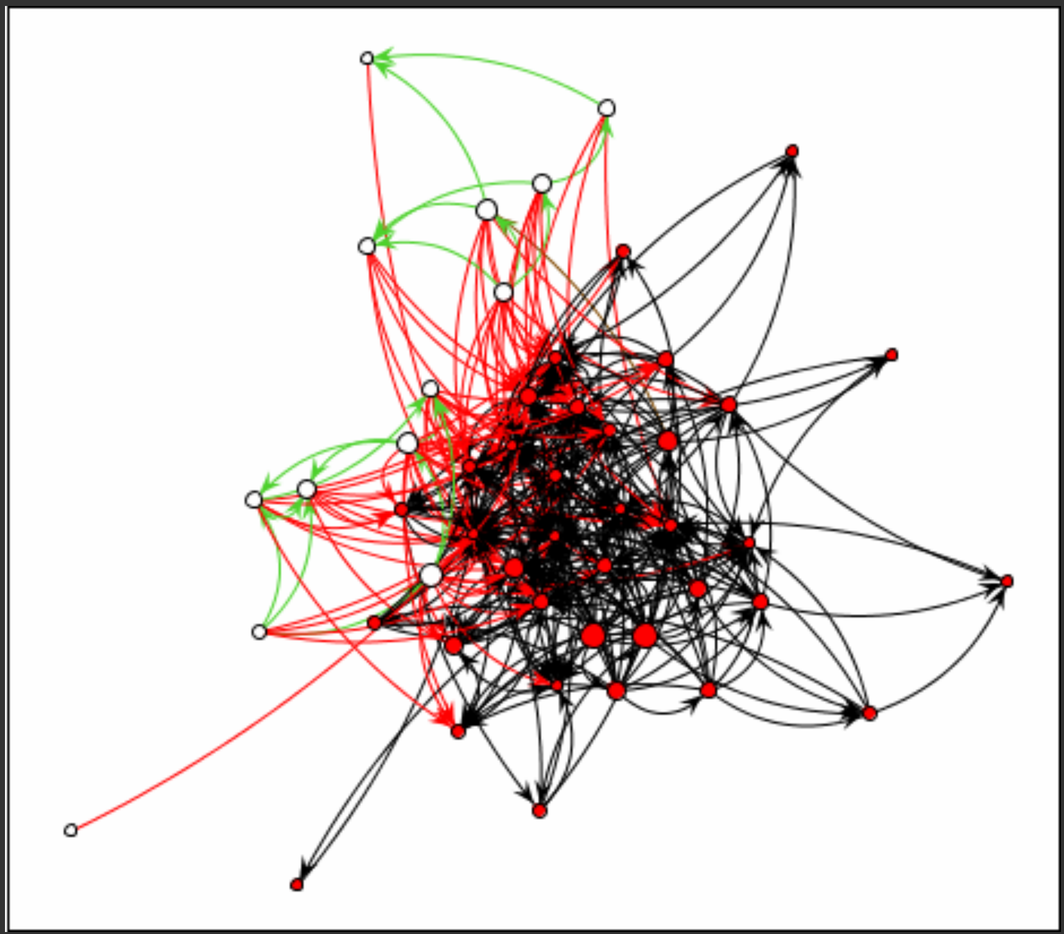
But this typically **converges quickly**

Enables **expressive constraints**!

# Use the Force!

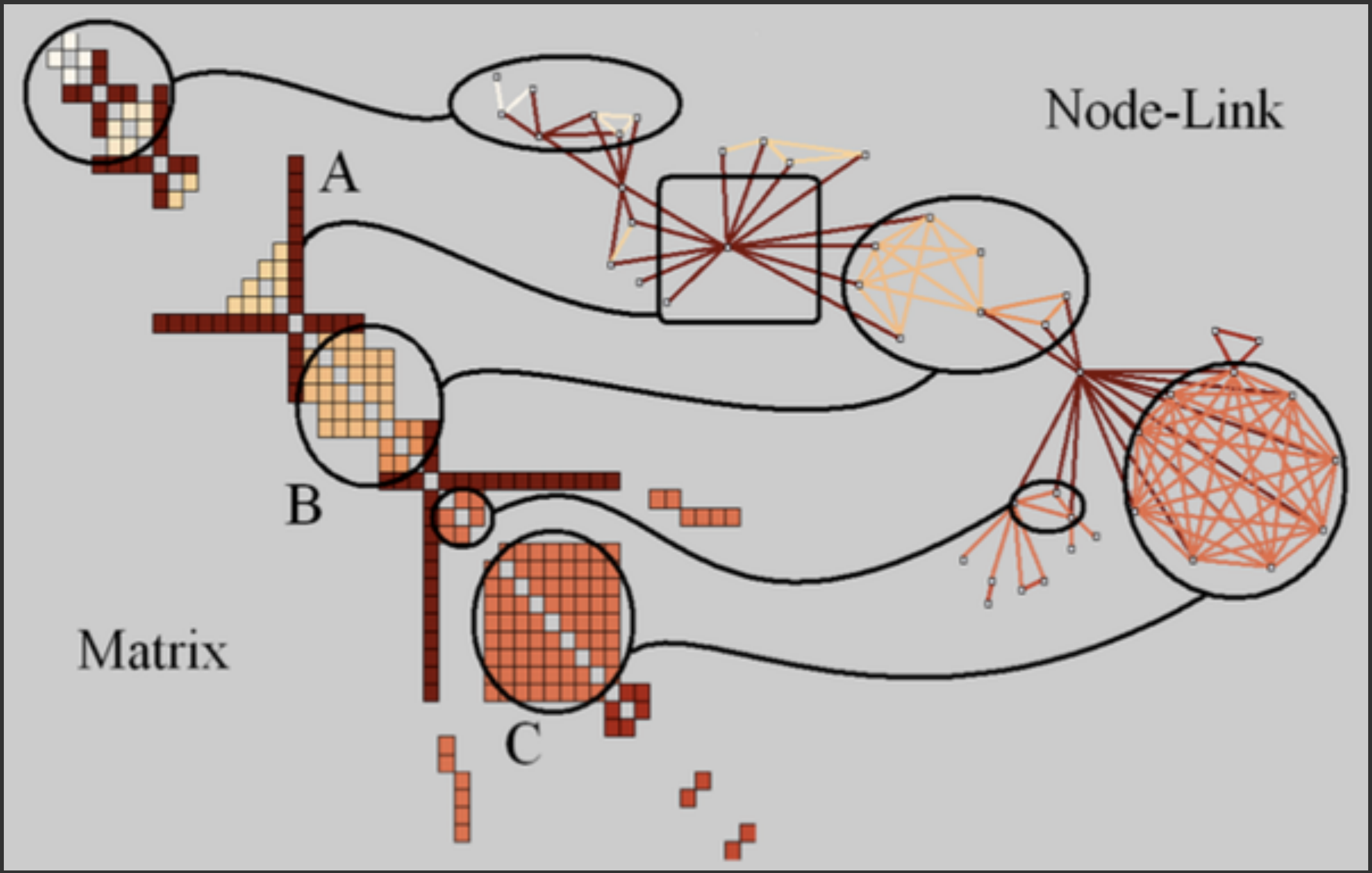
http://mbostock.github.io/d3/talk/20110921/

# Limitations of Node-Link Layout

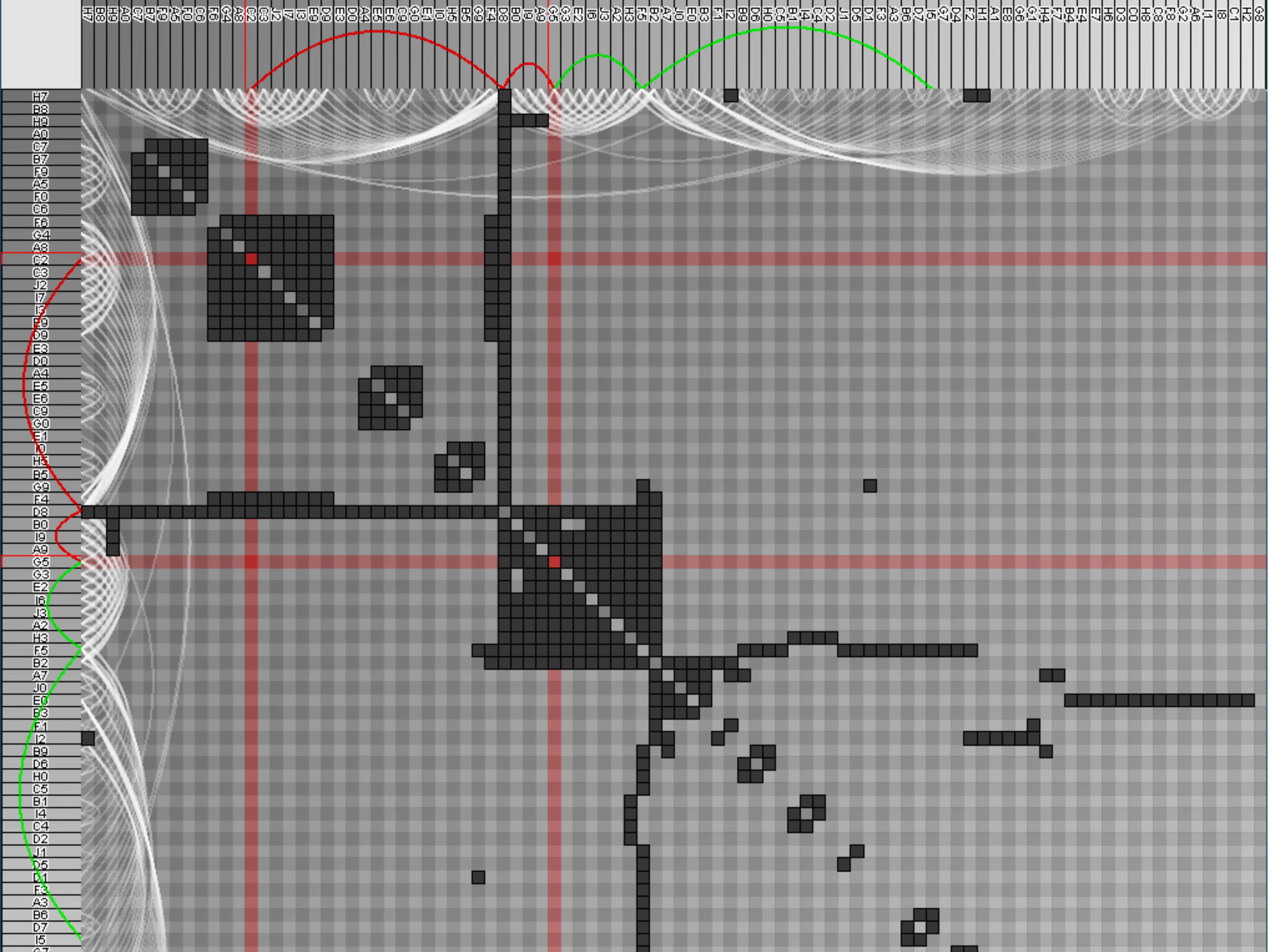


Edge-crossings and occlusion

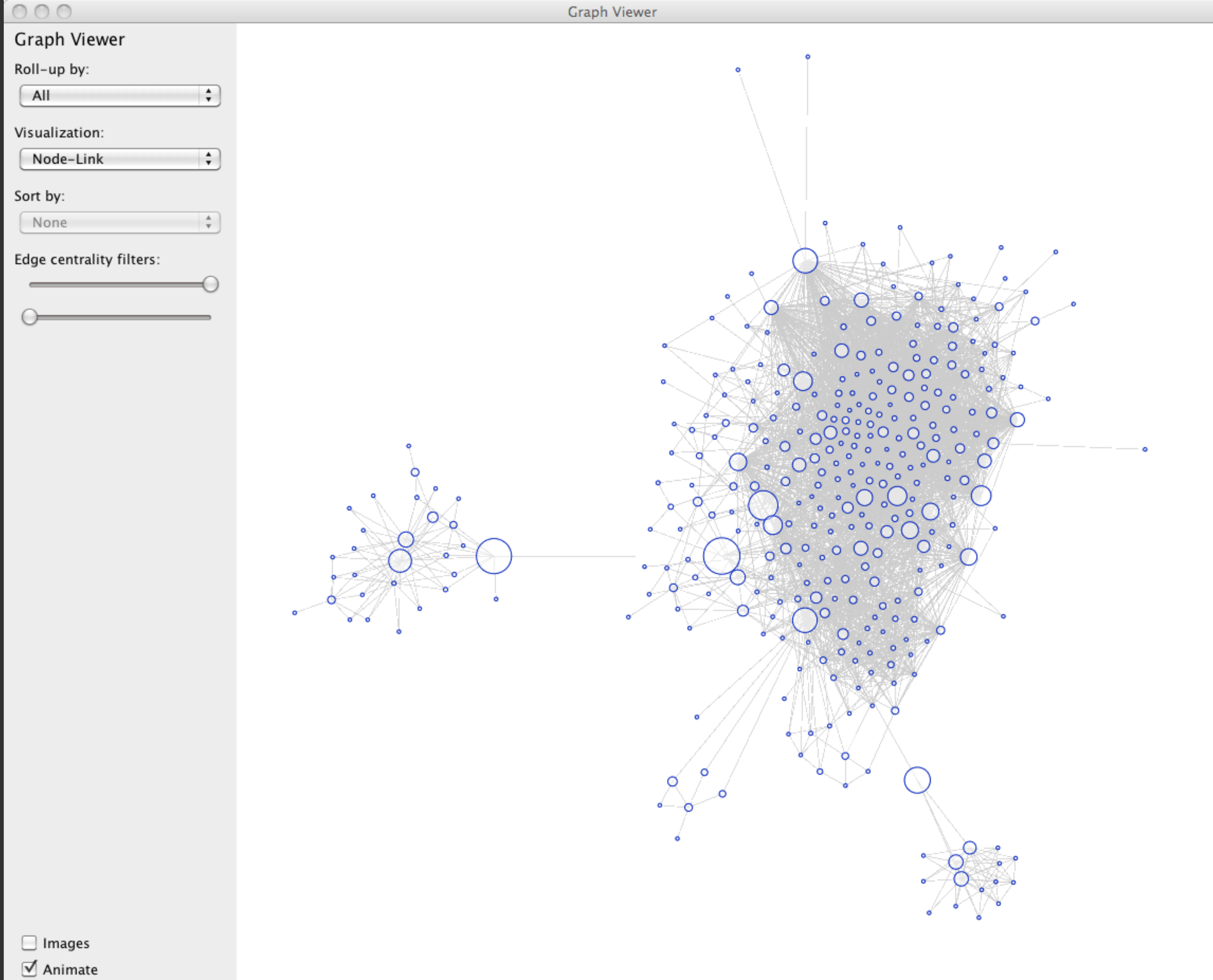
# Matrix Diagrams

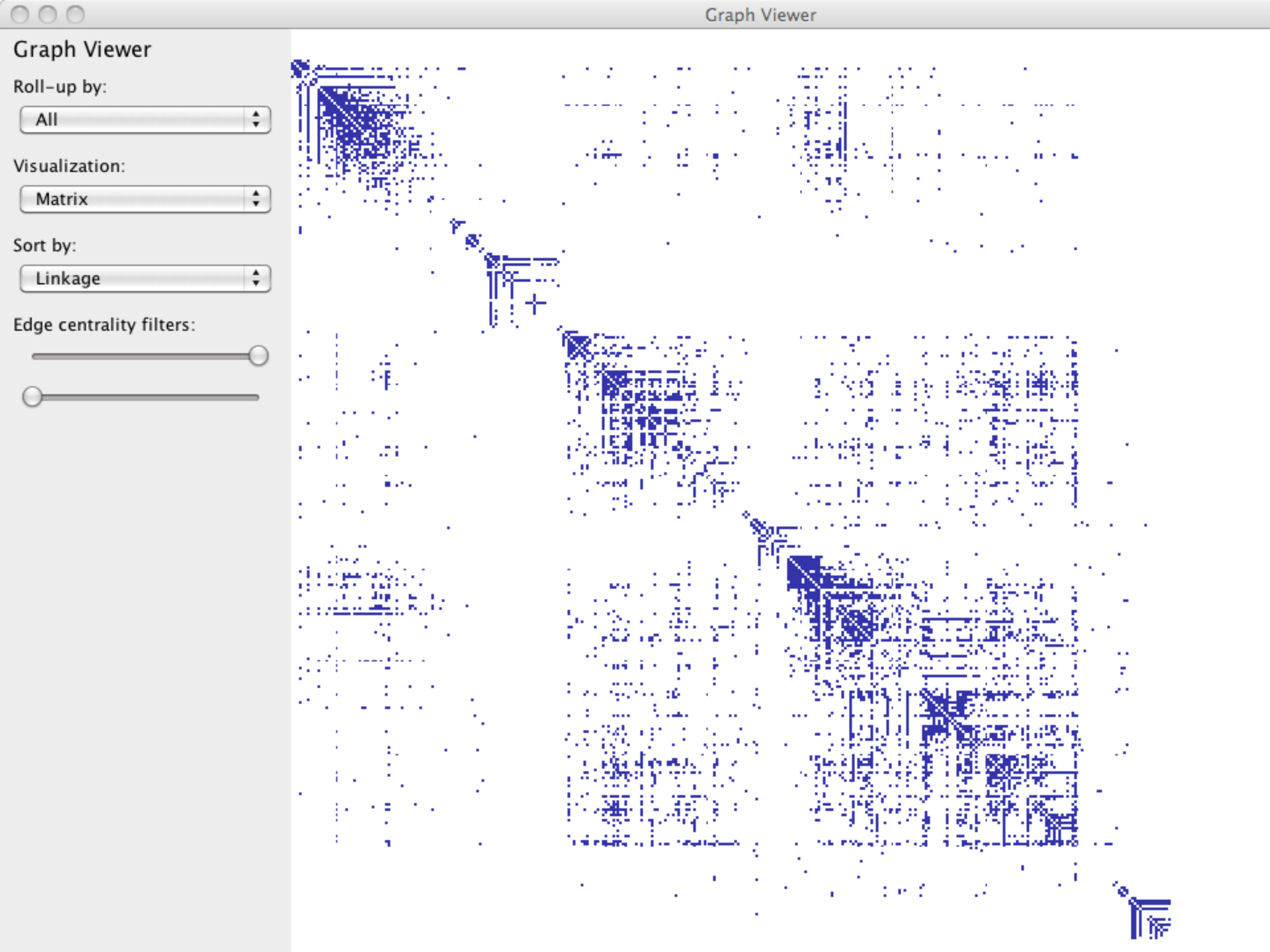


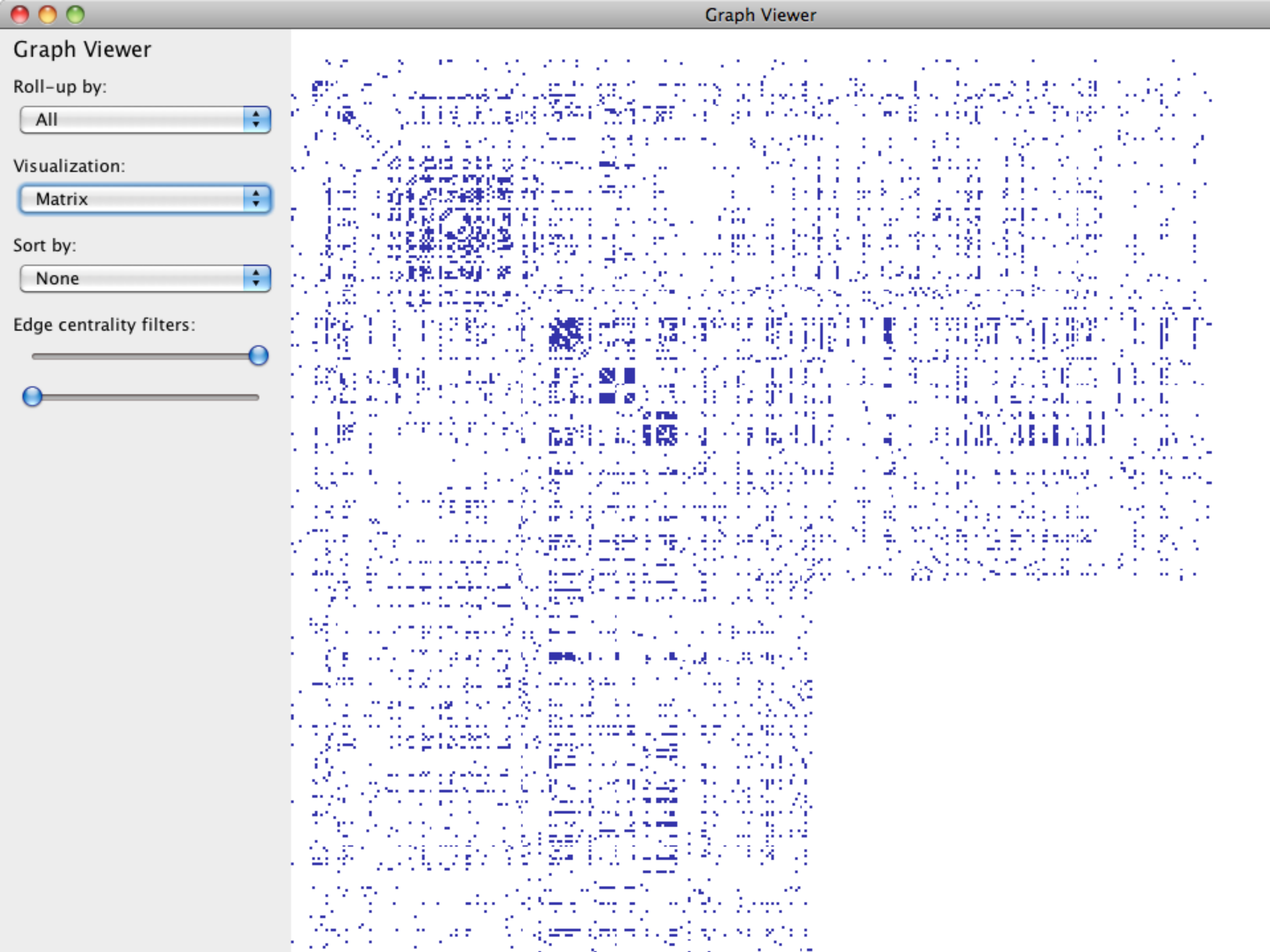

# Adjacency Matrices



H7
B8
H9
A0
C7
B7
F9
A5
F0
C6
F6
G4
A8
C2
C3
J2
I7
D9
E3
D0
A4
E5
E6
C9
G0
E1
H5
B5
G9
F4
D8
B0
I9
A9
G5
G3
E2
I6
J3
A2
H3
F5
B2
A7
J0
B3
F1
I2
B9
D6
H0
C5
B1
I4
C4
D2
J1
D5
A3
B6
D7
I5


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:

# 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

Dynamic queries / brushing to explore…

# Attribute-Driven Layout

The "Skitter" Layout

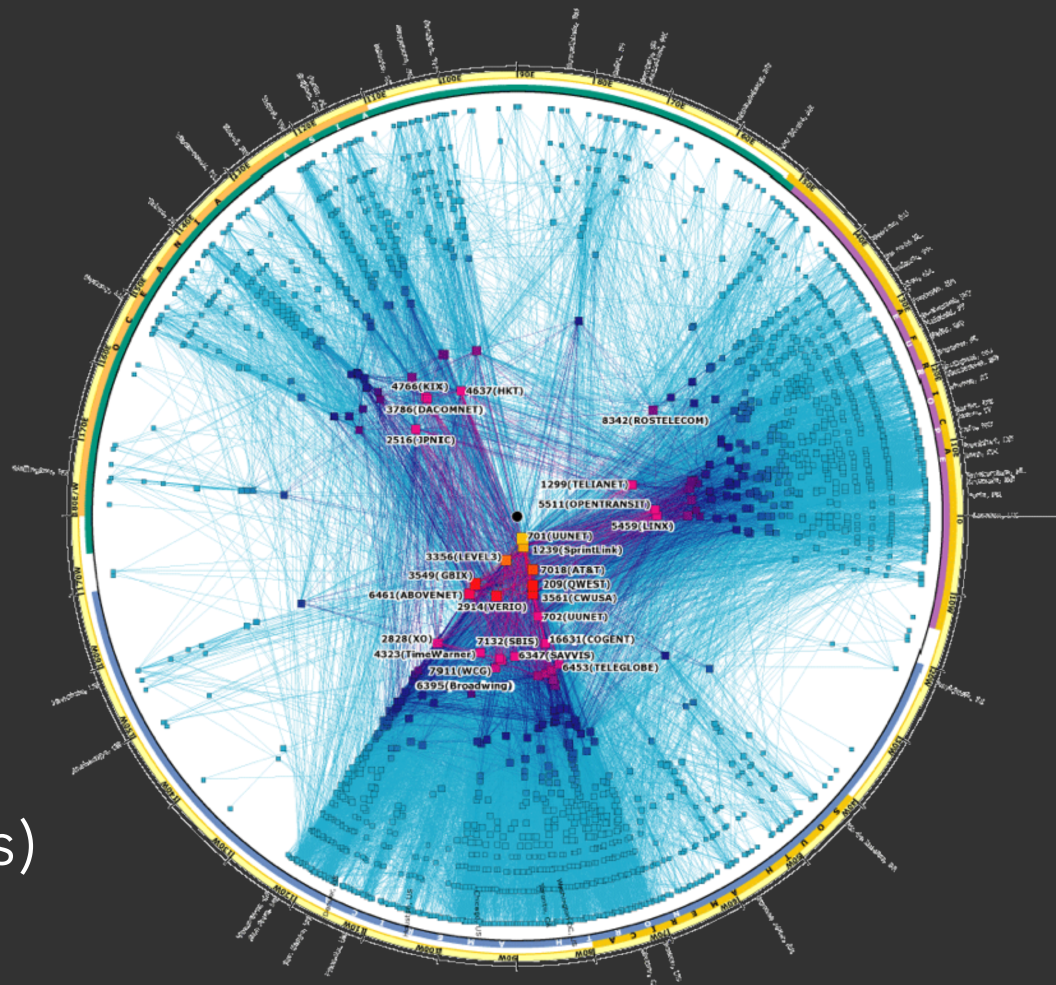
- Internet Connectivity
- Radial Scatterplot

Angle = Longitude

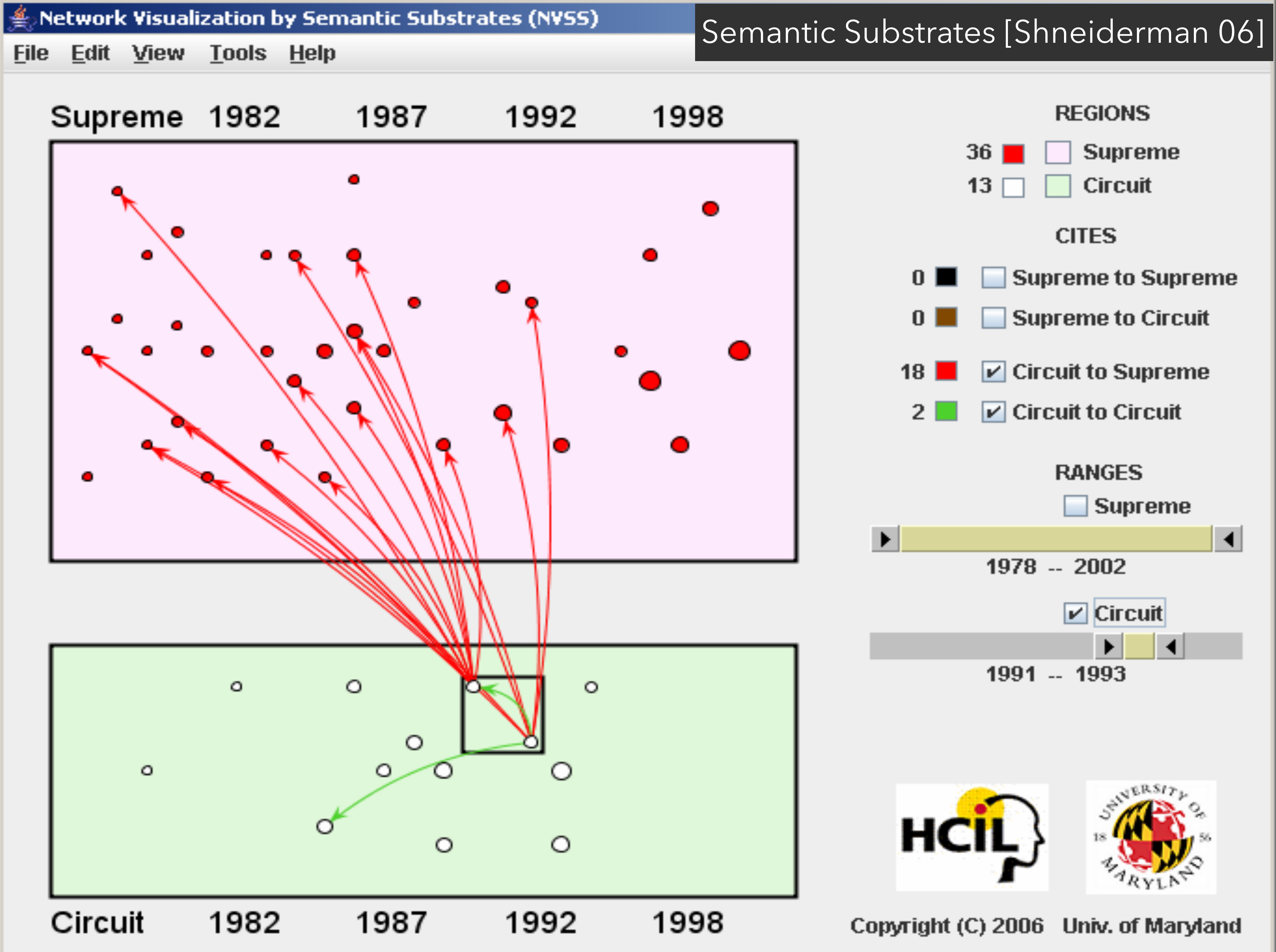
- Geography

Radius = Degree

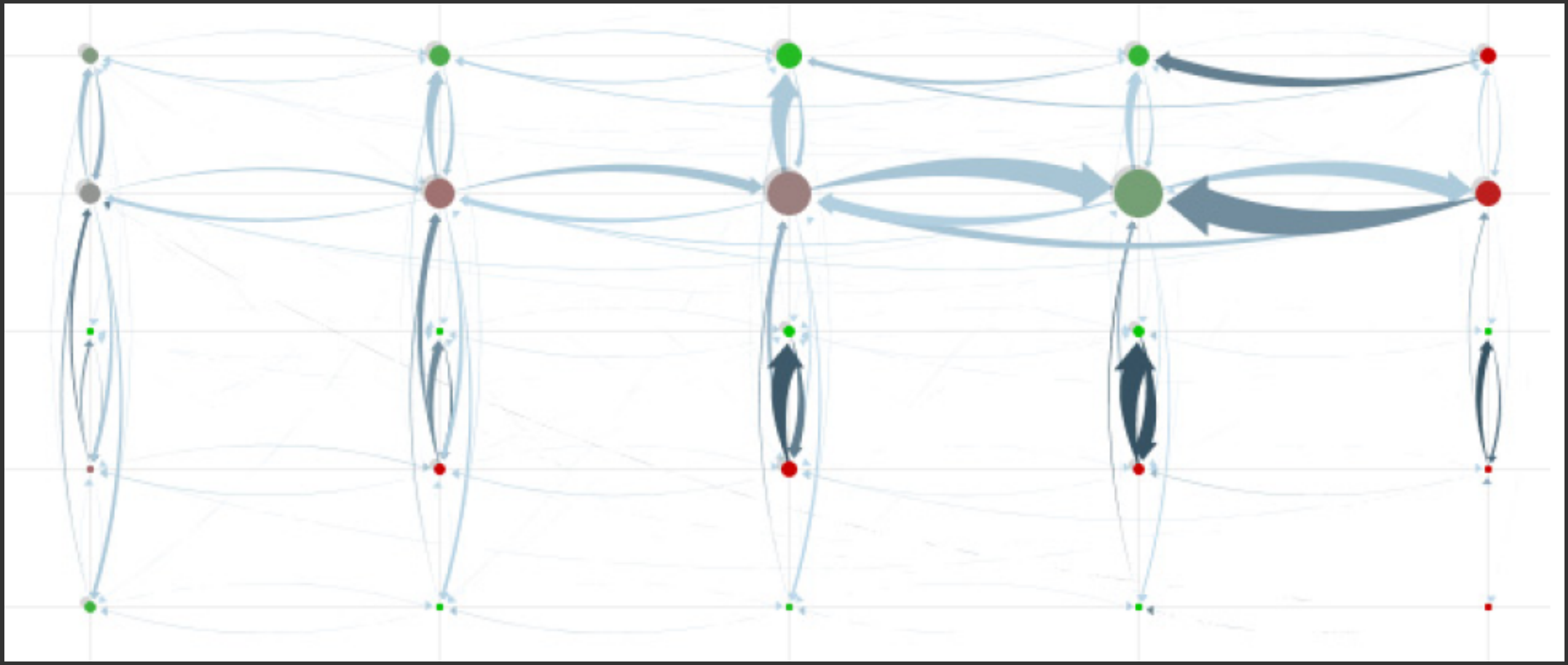
- # of connections
- (a statistic of the nodes)



Semantic Substrates [Shneiderman 06]

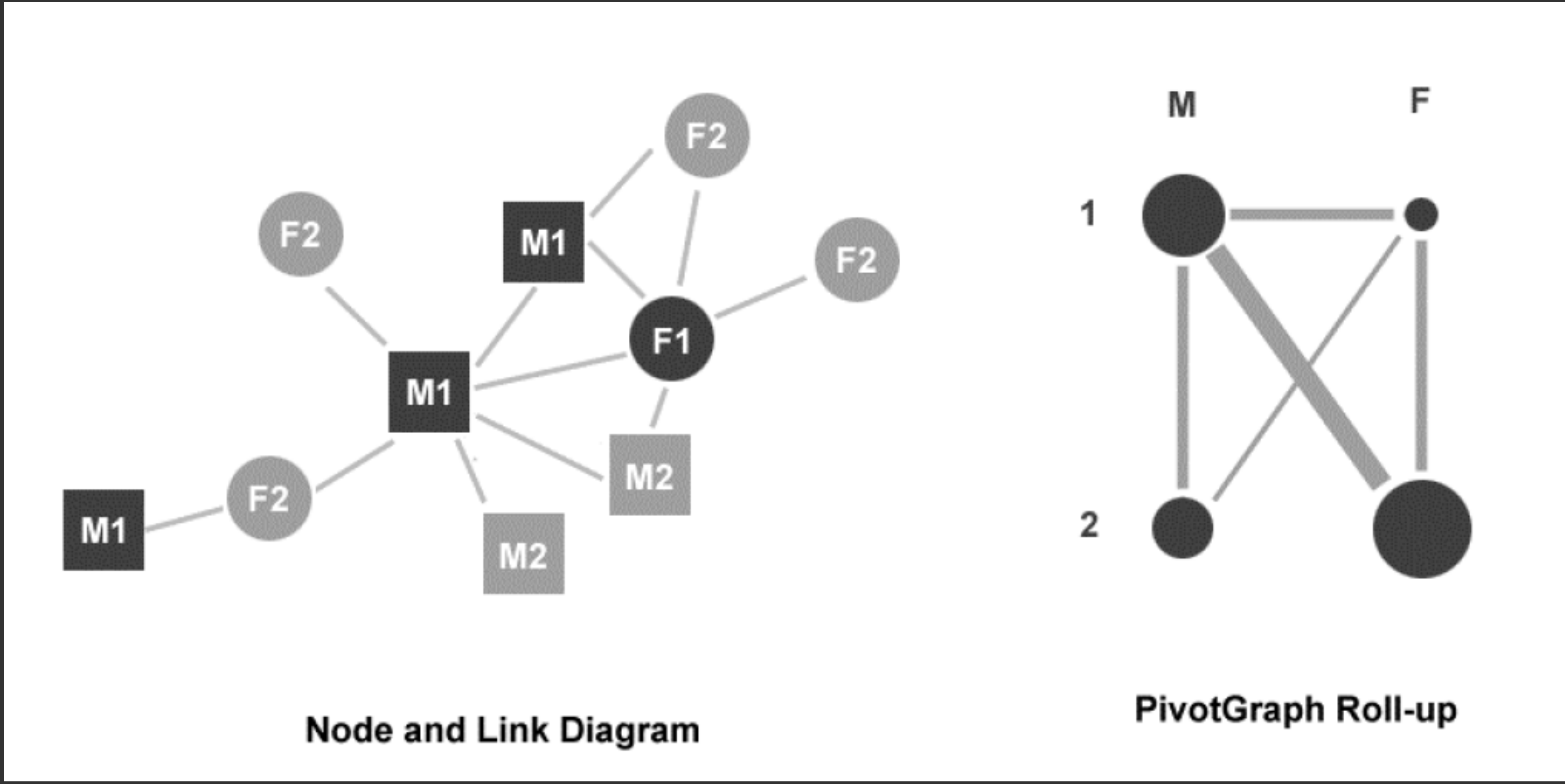


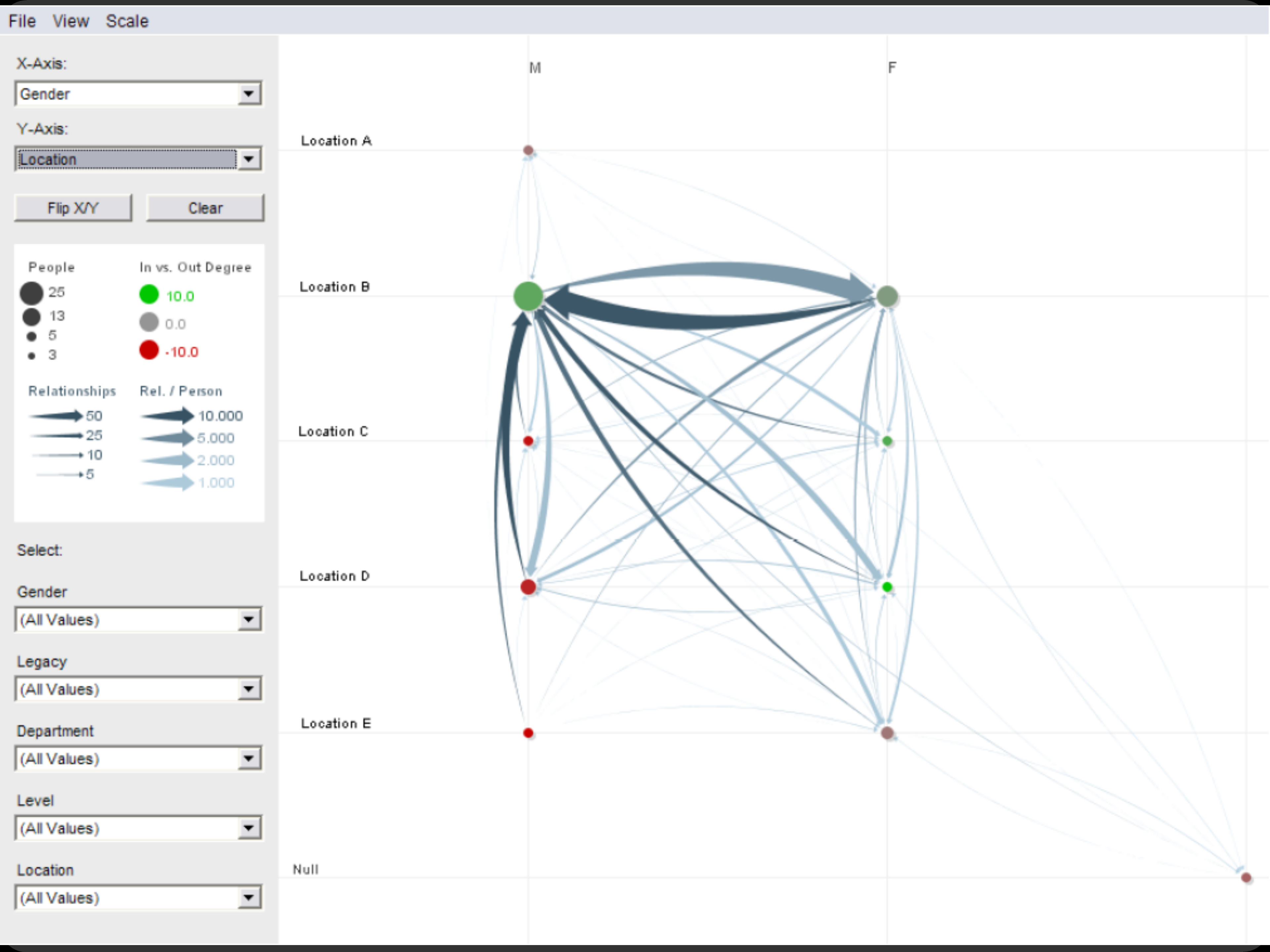
# PivotGraph [Wattenberg’06]



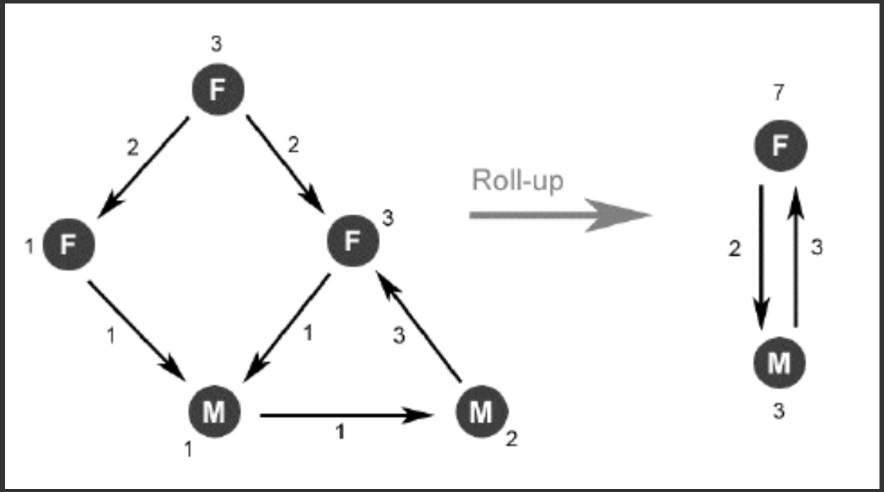
Layout aggregate graphs using node attributes.
Analogous to pivot tables and trellis display.

# PivotGraph


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

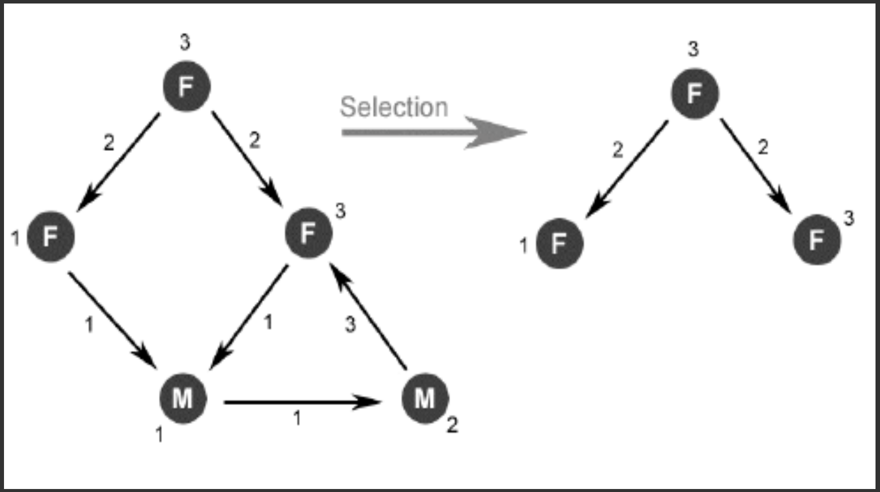

File
View
Scale
X-Axis:
Gender
Y-Axis:
Location
Flip X/Y
Clear
People
25
13
5
3
In vs. Out Degree
10.0
0.0
-10.0
Relationships
50
25
10
5
Rel. / Person
10.000
5.000
2.000
1.000
Select:
Gender
(All Values)
Legacy
(All Values)
Department
(All Values)
Level
(All Values)
Location
(All Values)
M
F
Location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

# Operators



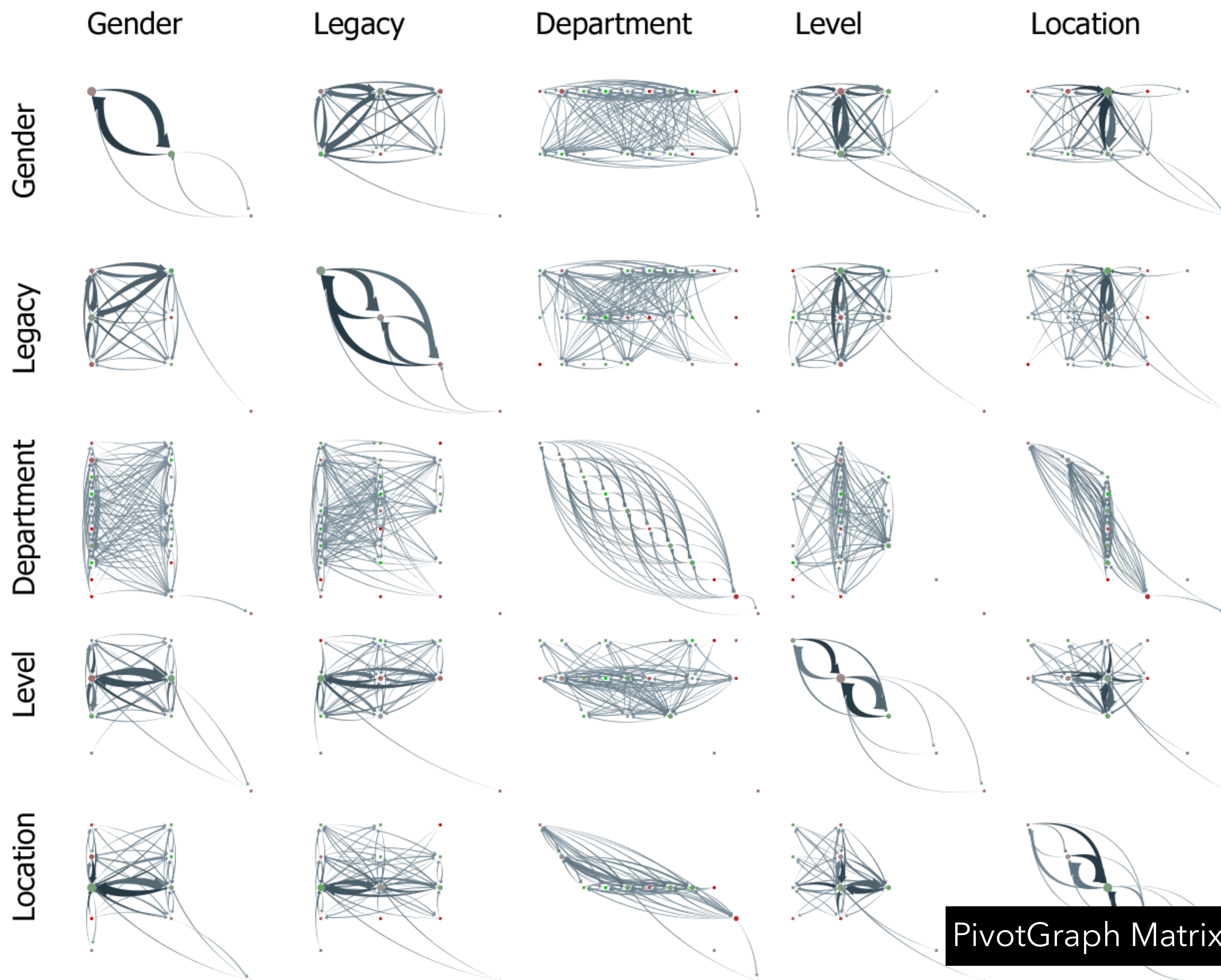

**Roll-Up**

Aggregate items with matching data values




**Selection**

Filter on data values



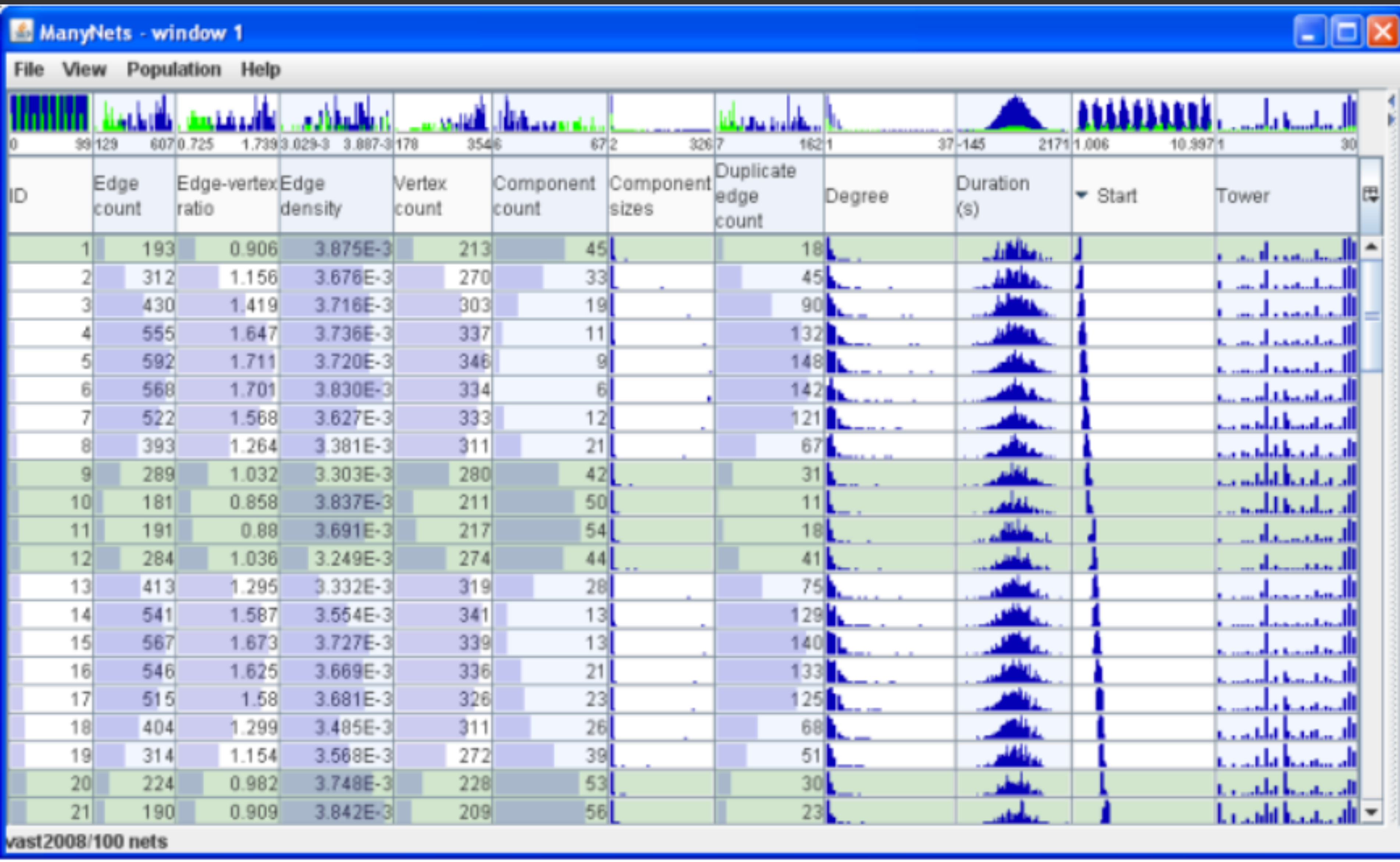

PivotGraph Matrix

# Limitations of PivotGraph

Only 2 variables (no nesting as in Tableau)

Doesn't support continuous variables

Multivariate edges?

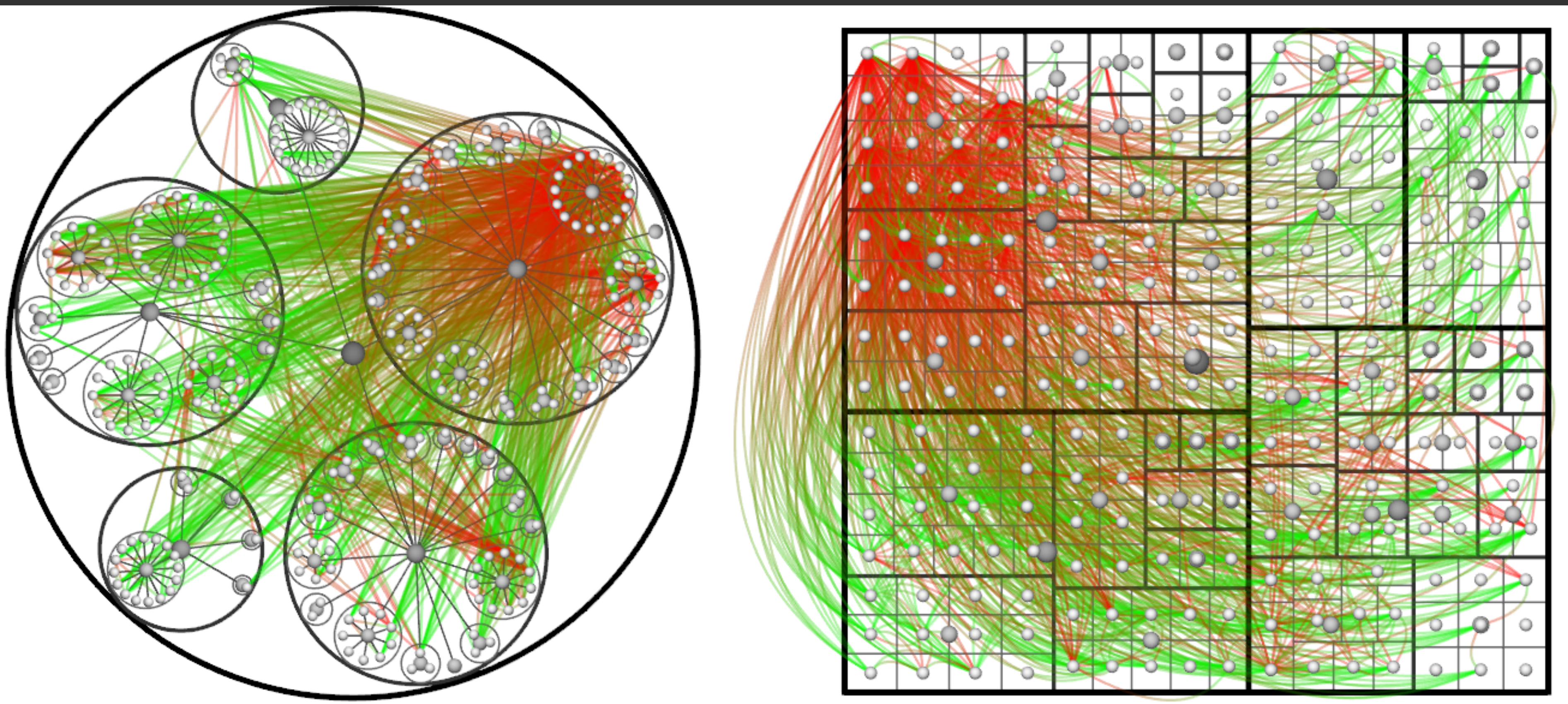



| ID | Edge count | Edge-vertex ratio | Edge density | Vertex count | Component count | Component sizes | Duplicate edge count | Degree | Duration (s) | Start | Tower |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 193 | 0.906 | 3.875E-3 | 213 | 45 | | 18 | | | | |
| 2 | 312 | 1.156 | 3.676E-3 | 270 | 33 | | 45 | | | | |
| 3 | 430 | 1.419 | 3.716E-3 | 303 | 19 | | 90 | | | | |
| 4 | 555 | 1.647 | 3.736E-3 | 337 | 11 | | 132 | | | | |
| 5 | 592 | 1.711 | 3.720E-3 | 346 | 9 | | 148 | | | | |
| 6 | 568 | 1.701 | 3.830E-3 | 334 | 6 | | 142 | | | | |
| 7 | 522 | 1.568 | 3.627E-3 | 333 | 12 | | 121 | | | | |
| 8 | 393 | 1.264 | 3.381E-3 | 311 | 21 | | 67 | | | | |
| 9 | 289 | 1.032 | 3.303E-3 | 280 | 42 | | 31 | | | | |
| 10 | 181 | 0.858 | 3.837E-3 | 211 | 50 | | 11 | | | | |
| 11 | 191 | 0.88 | 3.691E-3 | 217 | 54 | | 18 | | | | |
| 12 | 284 | 1.036 | 3.249E-3 | 274 | 44 | | 41 | | | | |
| 13 | 413 | 1.295 | 3.332E-3 | 319 | 28 | | 75 | | | | |
| 14 | 541 | 1.587 | 3.554E-3 | 341 | 13 | | 129 | | | | |
| 15 | 567 | 1.673 | 3.727E-3 | 339 | 13 | | 140 | | | | |
| 16 | 546 | 1.625 | 3.669E-3 | 336 | 21 | | 133 | | | | |
| 17 | 515 | 1.58 | 3.681E-3 | 326 | 23 | | 125 | | | | |
| 18 | 404 | 1.299 | 3.485E-3 | 311 | 26 | | 68 | | | | |
| 19 | 314 | 1.154 | 3.568E-3 | 272 | 39 | | 51 | | | | |
| 20 | 224 | 0.982 | 3.748E-3 | 228 | 53 | | 30 | | | | |
| 21 | 190 | 0.909 | 3.842E-3 | 209 | 56 | | 23 | | | | |

ManyNets

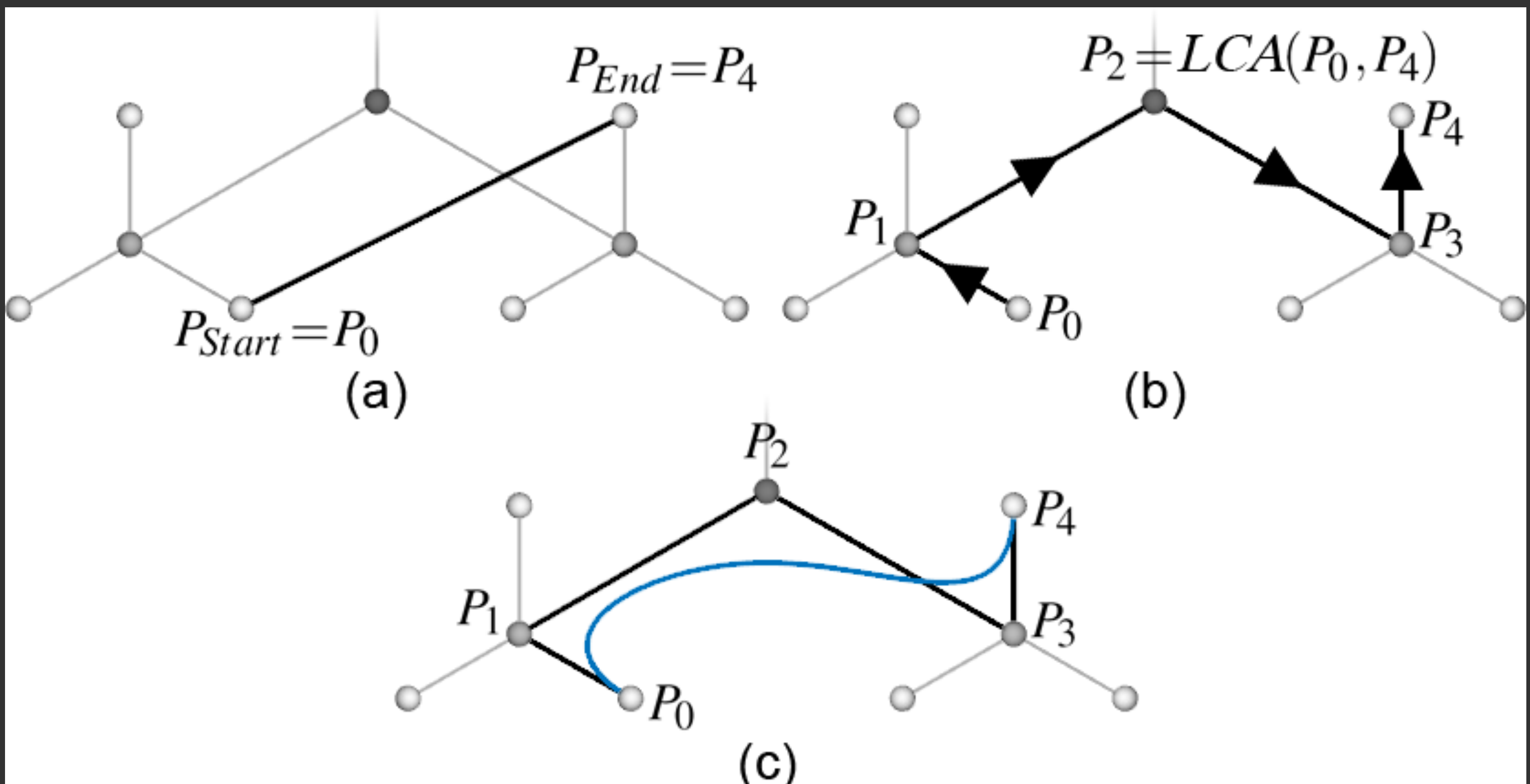
# Hierarchical Edge Bundling

# Trees with Adjacency Relations

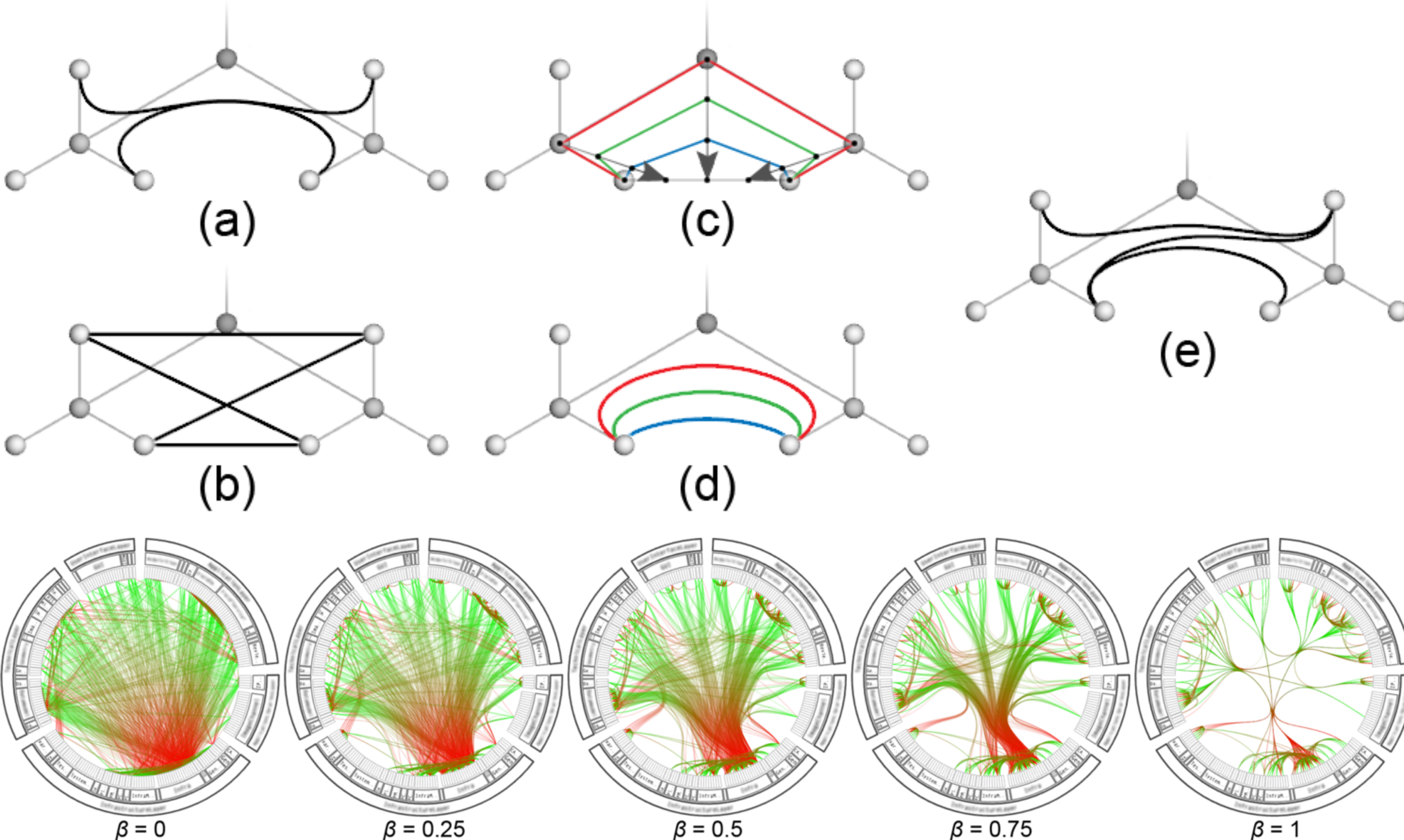


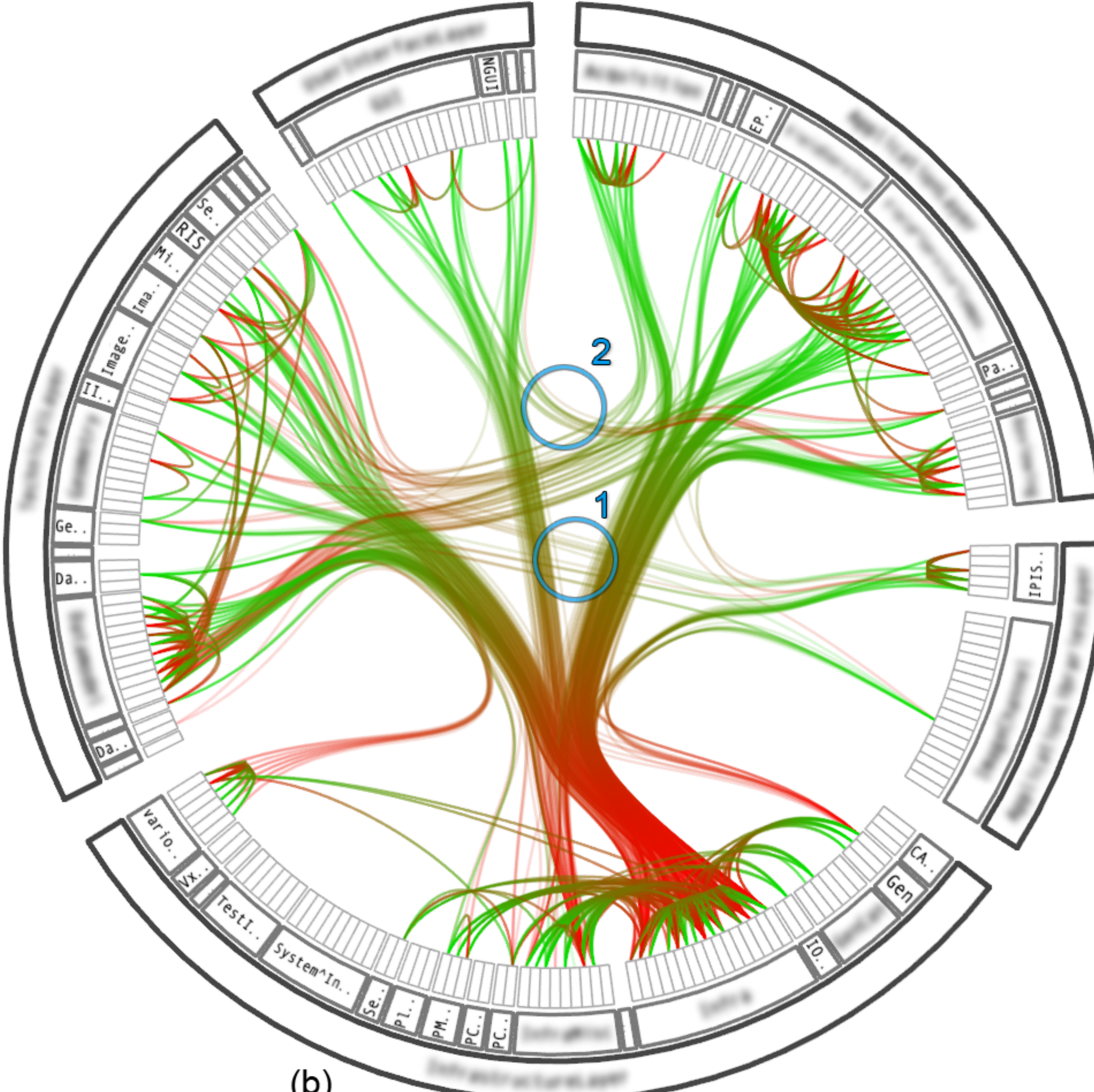
(a) (b)

# Bundle Edges Along Hierarchy



# Configuring Edge Tension




NGUI
EP..
Pa..
IPIS..
CA..
Gen
IO..
PC..
PC..
PM..
Pl..
Se..
System^In..
TestI..
Vx..
vario..
Da..
Da..
Ge..
II..
Image..
Ima..
Mi..
RIS
Se..
1
2

(b)

# Summary




**Tree Layout**

Indented / Node-Link / Enclosure / Layers

Focus+Context techniques for scale

**Graph Layout**

Spanning Tree Layout

Hierarchical “Sugiyama” Layout

Optimization (Force-Directed Layout)

Matrix Diagrams

Attribute-Driven Layout