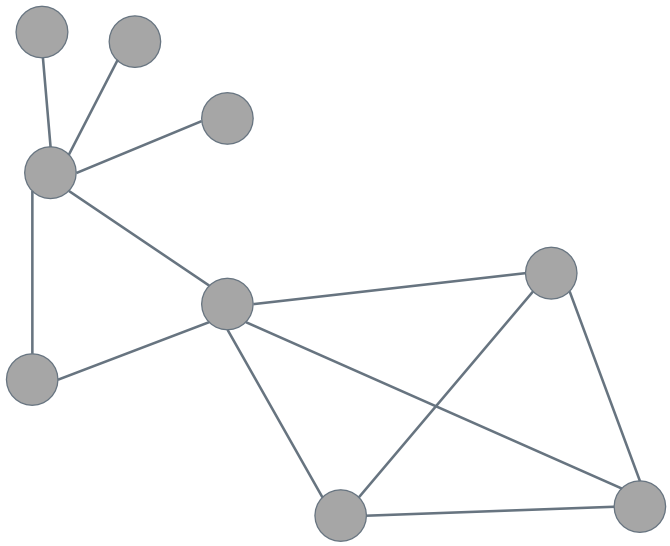
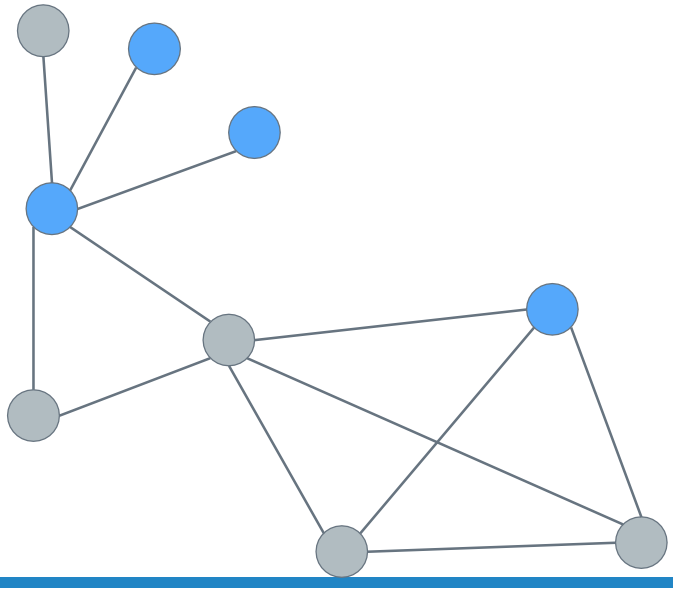
# Data Visualization

## Graph Visualization

### What's in a graph?

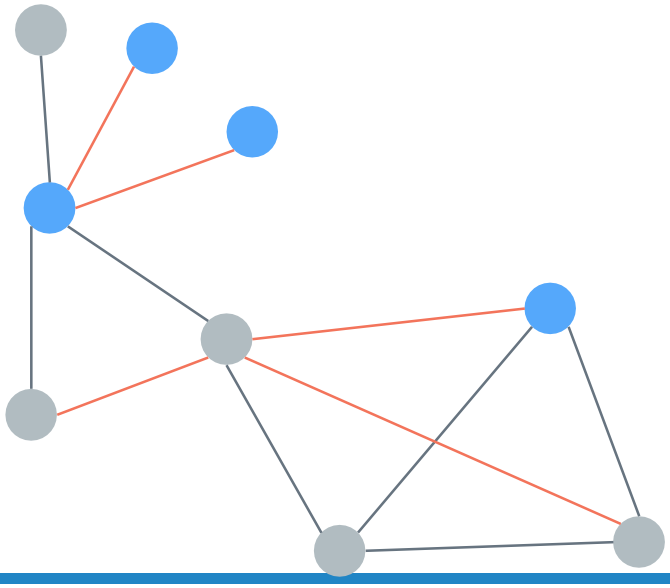


# What's in a graph?

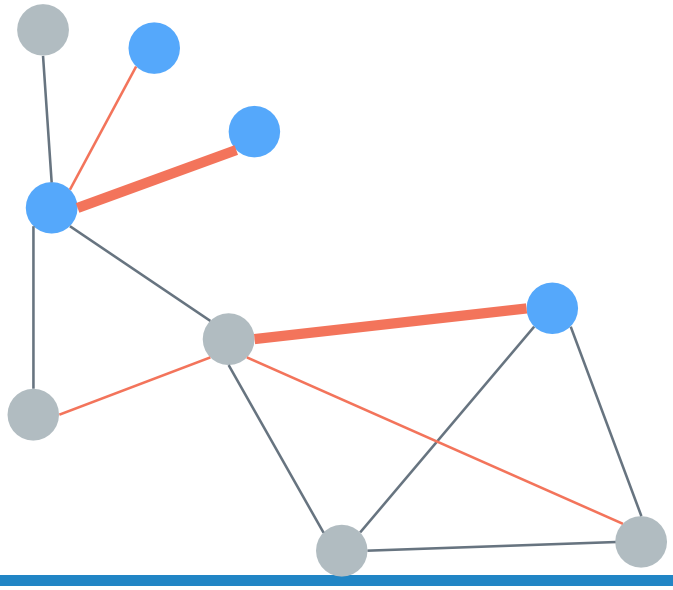




# What's in a graph?

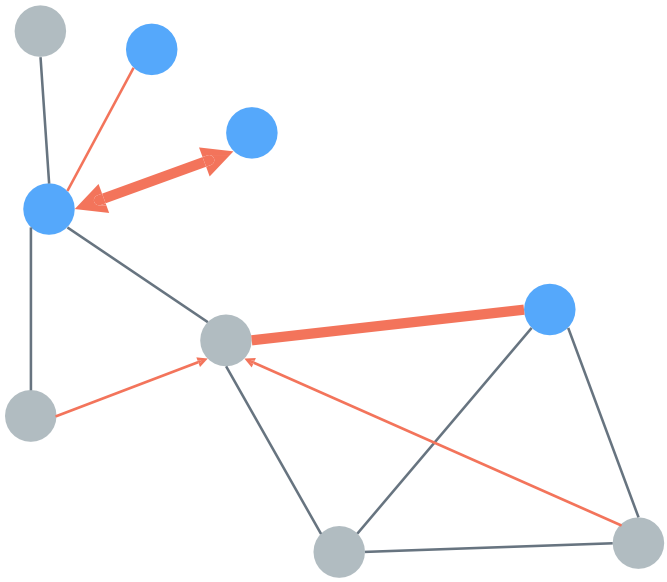


# What's in a graph?

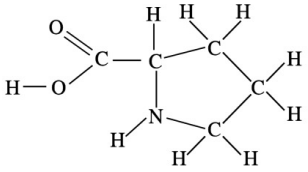


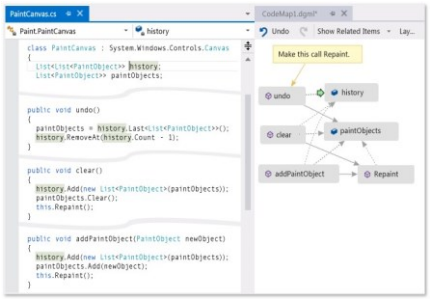


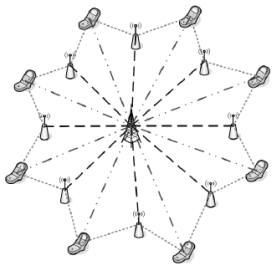
# What's in a graph?

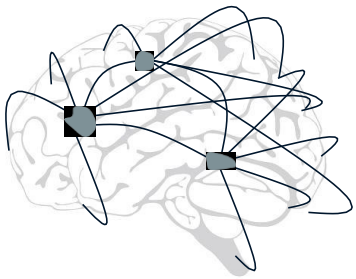


# Everything can be a graph!





# What questions might we ask?

- How does the brain organize itself to achieve a function?
- How does knowledge disseminate in online communities?
- How are two graphs similar?
- Which entities in a social network might be terrorists?

# Graph Drawing

- The primary concern of graph drawing is:
  - The spatial arrangement of nodes and links
- Often (but not always) the goal is to effectively depict the graph structure:
  - Connectivity patterns
  - Partitions / Clusters
  - Outliers

9

# Vertex Choices

- **Shape**: Symbols for vertices could show distinguishable objects.
  - Or could show people's pictures in boxes (unique but distinguishable only at small scale).
- **Color:** categorical colorings.
- **Size**: could scale vertex to show ordinal/interval/continuous variables.
  - But difficult to tell difference.
- **Location**: could pull together "similar" vertices (separate from edge effects) to group.
- **Label**: could add labels to vertices, but layout problems.

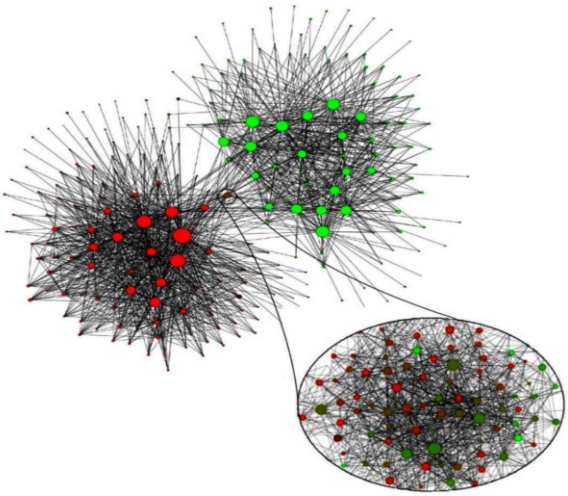
10

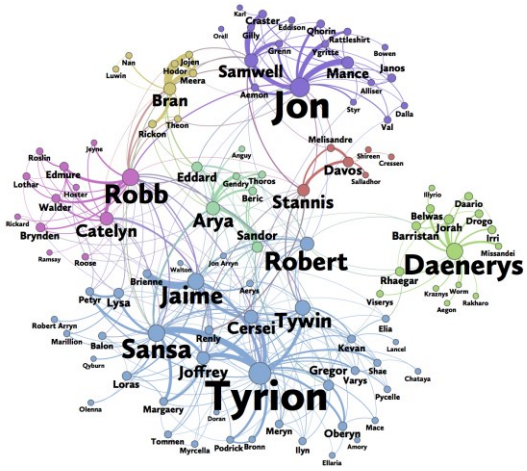
# Edge Choices

- **Thickness** (width): ordinal/interval/continuous.
  - Effective, natural, and works up until very large scale.
- **Length:** distance between nodes shows strength.
- **Color**: use colors to denote strength of relationship
- **Label**: can label, but not immediately perceived (cognitive processing) and increases clutter significantly.
- **Form**: to show different types of relationships (dotted, dashed, etc.)
- **Direction**: can have arrows at end to indicate direction

11

# Putting things into perspective





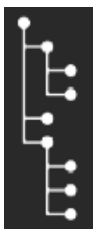

12

# Tree Visualization

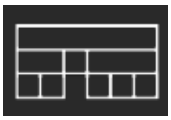


## Four major tree visualizations

Indented lists

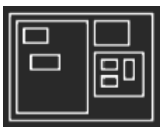


Layered diagrams

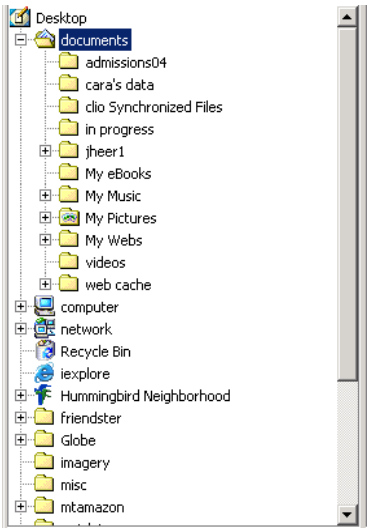


Node-link trees

Treemaps



# Indented List

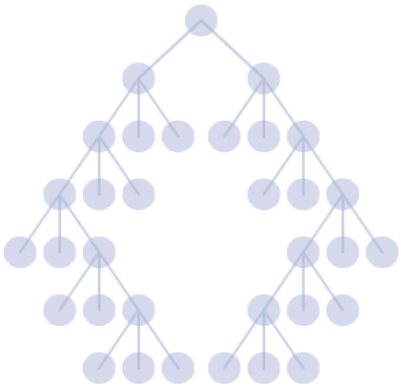



- Places all items along vertically spaced rows
- Indentation used to show parent/child relationships
- Commonly used as a component in an interface
- Breadth and depth contend for space
- Often requires a great deal of scrolling

15

# Node-Link Trees

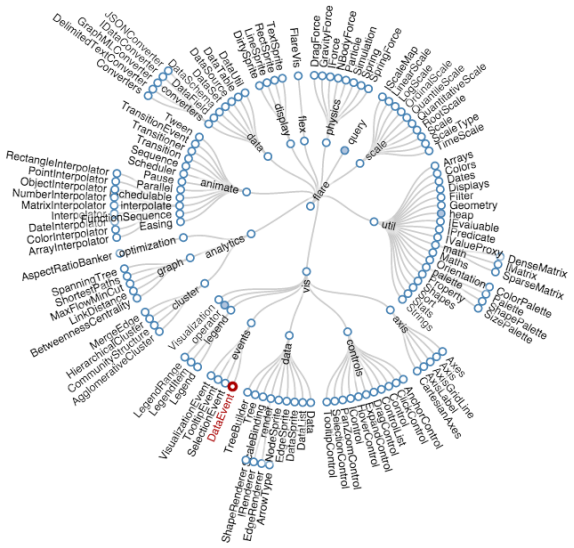
- Nodes are distributed in space, connected by straight or curved lines.
- Typical approach is to use 2D space to break apart breadth and depth.
- Reingold-Tilford algorithm achieves linear time.



16

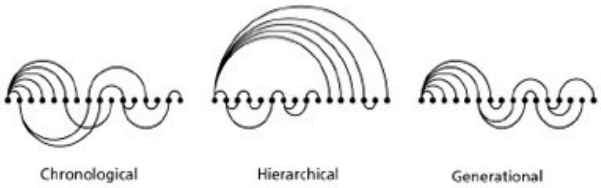
# Node-Link Trees

- Radial layout places the root in the center.
- The radius encodes the depth.





# Other node-Link trees




**ThreadArcs**

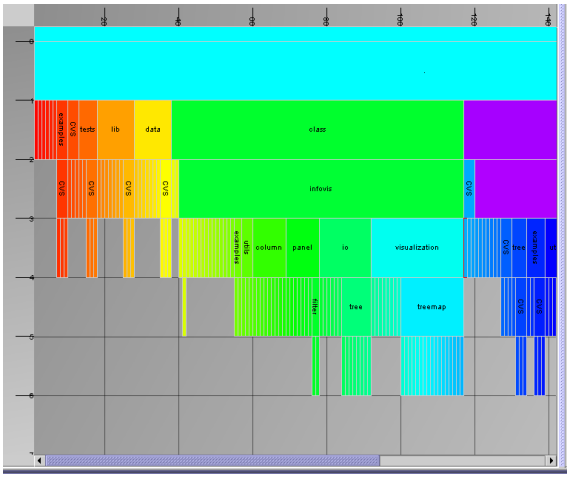
**PhylloTrees**

# Layered Diagrams

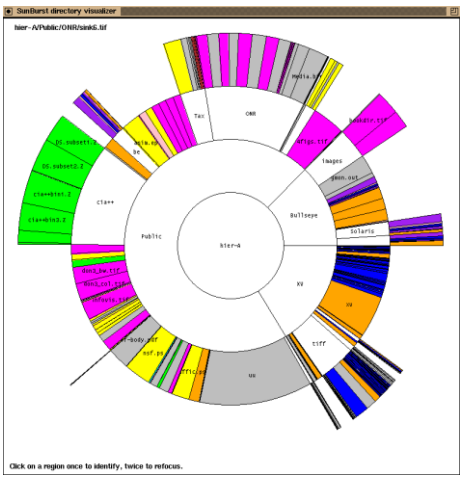
- Signify tree structure using
  - Layering
  - Adjacency
  - Alignment
- Involves recursive sub-division of space
- We can apply the same set of approaches as in node-link layout.



# Layered Diagrams



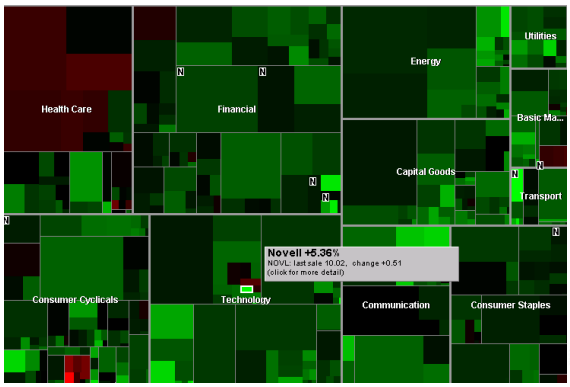

**Icicle Trees**

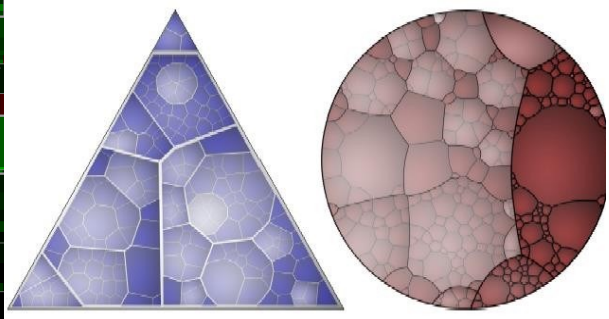



**SunBurst**

# Treemaps

- Encode hierarchy using spatial enclosure
- Space-filling technique



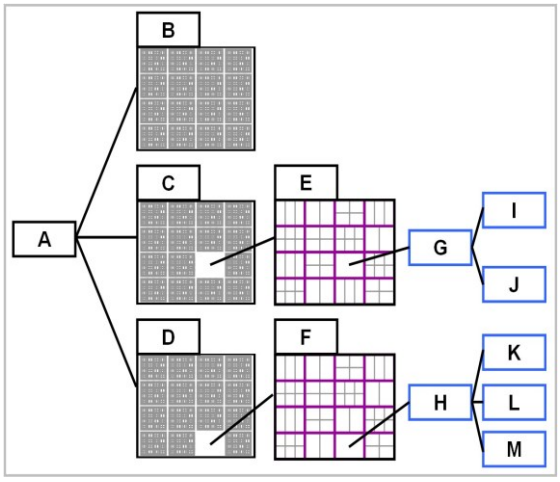





# Treemaps

- **Benefits**
  - Provides a single view of an entire tree
  - Easier to spot large/small nodes
- **Problems**
  - Difficult to accurately read depth

# Hybrids

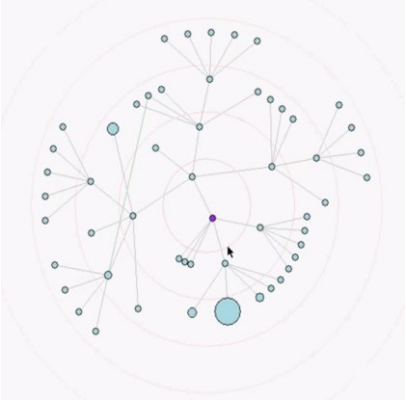



**Elastic Hierarchies**

23

# See the tree in this graph?

- Many graphs are tree-like or have useful spanning trees
- Spanning trees lead to arbitrary roots
- Fast tree layouts allow graph layouts to be recalculated at interactive rates



24

# Graph Layouts

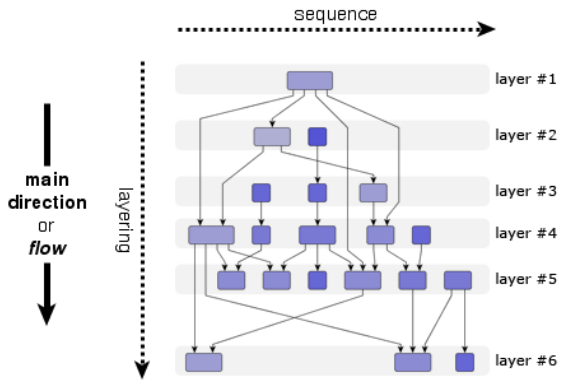


## Common Layout Techniques

- **Hierarchical:** work well for trees and directed acyclic graphs (DAGs).
- **Force-directed:** gravity "pulls" vertices together
- **Circular:** laid out on a circle, or a sphere, or the perimeter of a circle
- **Geographic-based**: spatial coordinates, or location in space based on similarity
- **Clustered**: cluster by similarity
- **Attribute-based:** positioned by attribute values, for example clustered together
- **Matrix:** row/column orientation of adjacency matrix

# Hierarchical Layout

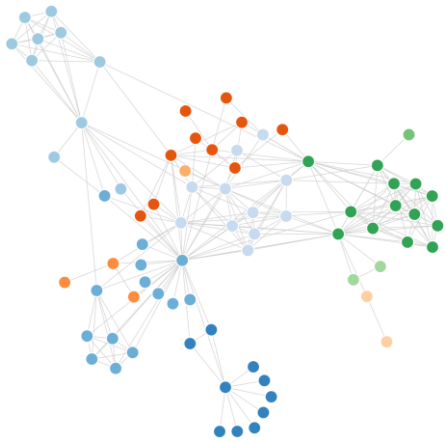
- Sugiyama-style or layered graph drawing
- Layout of a Direct Acyclic Graph
- Hierarchical layering based on descent




27

# Force-directed Layout

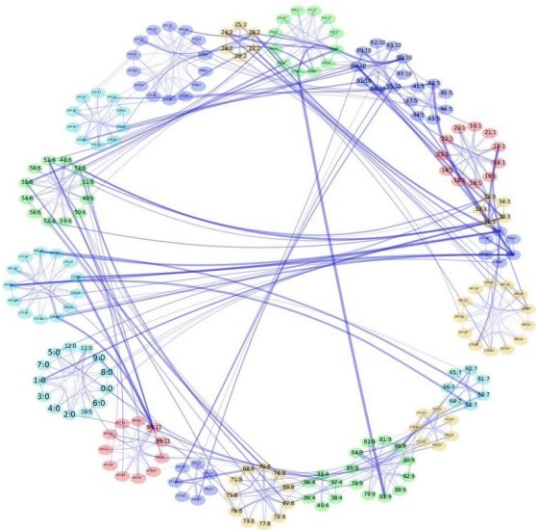
- A force-directed layout highlights the underlying topology of the graph.
- You can visually identify clusters, cliques, and bridges.
- If the graph is too highly connected or too unstructured, then a force-directed graph will tend to produce a hairball.



28

# Circular Layout

- Circular Layout is usually used with edge bundling
- *Edge bundling* refers to bundling together multiple edges so that their common parts are, to some degree, merged into a bundled part.
- It is useful for increasing the readability of graph drawings with a high number of edges



29

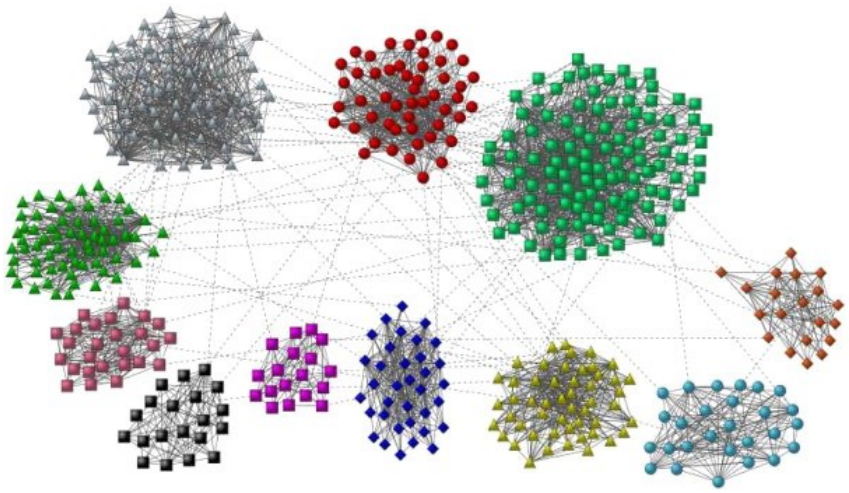
# Geographic-based Layout


30

# Geographic-based Layout



# Cluster/attribute based Layout



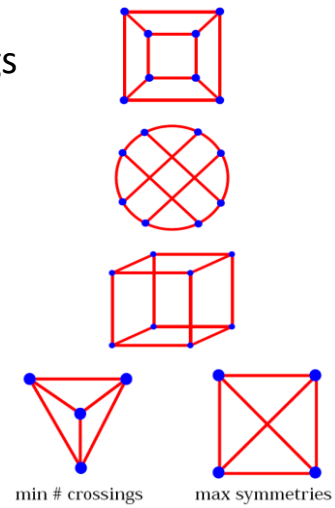
# Layout Recommendations

- Planar in most cases.
- Hierarchical for trees and DAGs.
- Force-directed to capture magnitude of edge relation OR another variable.
- Cluster/attribute based can be good choice to show similarity of nodes.
- Circles Perimeters are good to emphasize connections between equal things (on perimeter).
- Spheres are good for large scale large number of nodes to provide interactive focus zoom.

33

# Aesthetic Constraints

- Minimize edge crossings
- Minimize line bends
- Minimize line slopes
- Maximize symmetry
- but, can't do it all.



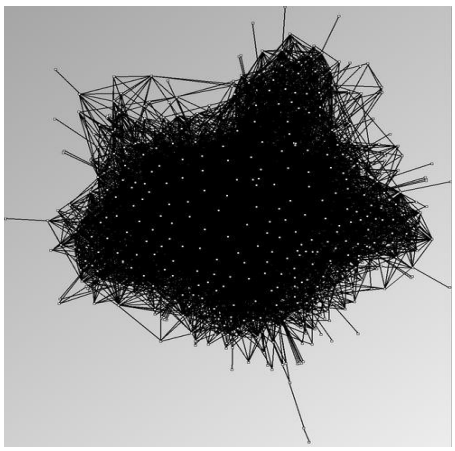

34

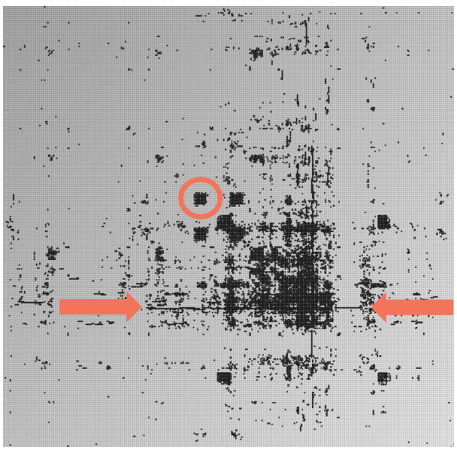
# Matrices



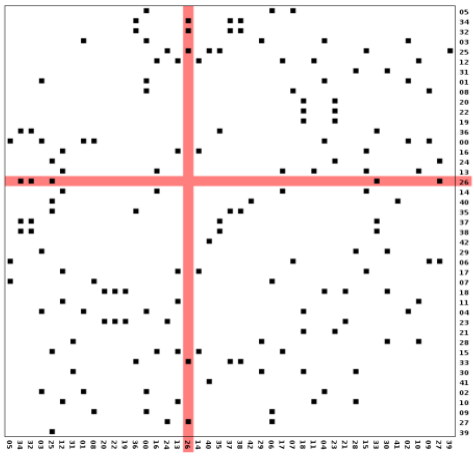
## Matrices

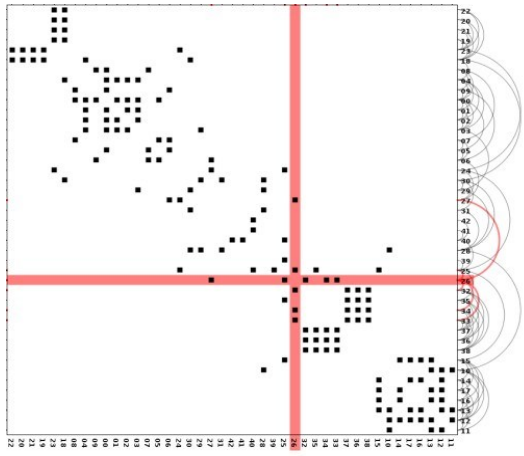
- One year of email between ~500 researchers





# Revealing patterns





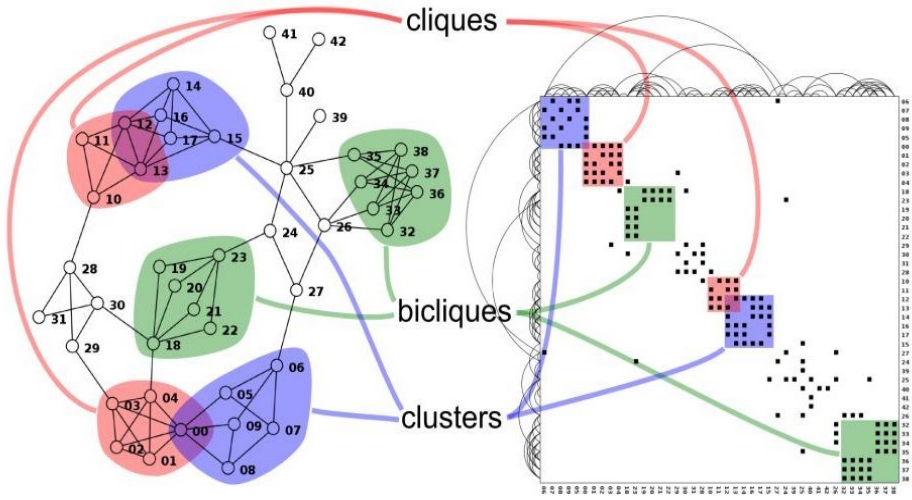
37

# Matrix vs. Node-Link

| | |
|---|---|
| × Require learning | ✓ Familiar |
| ✓ No overlap | × Node overlap |
| ✓ No crossings | × Link crossing |
| × Use a lot of space | ✓ More compact |
| ✓ Dense graphs | × Dense graphs |
| × Sparse graphs | ✓ Sparse graphs |

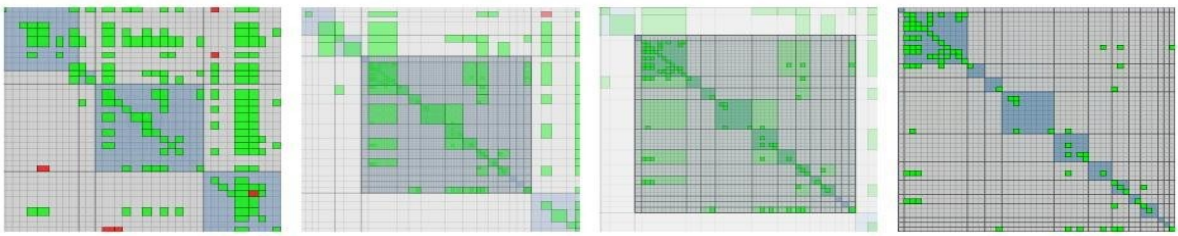
38

# Matrix + Node-Link






# Hierarchical Aggregation



# Graph Visualization Challenges

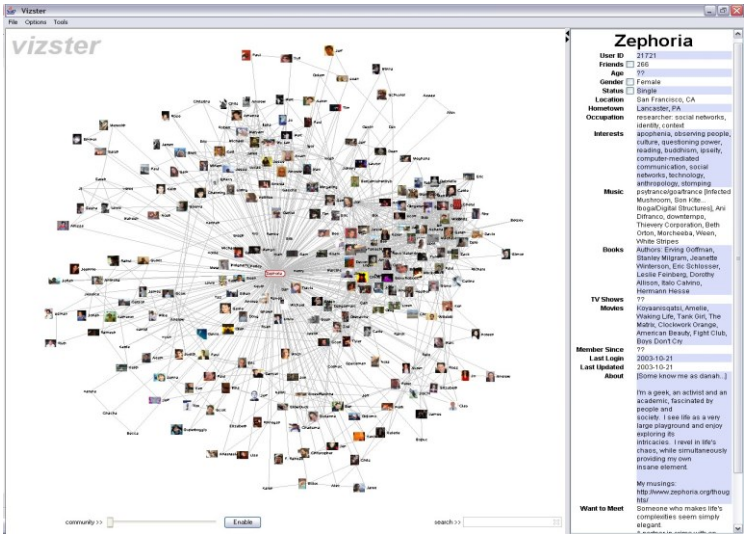
- Graph layout and positioning
  - Make a concrete rendering of abstract graph
- Navigation/Interaction
  - How to support user changing focus and moving around the graph
- Scale
  - Above two issues not too bad for small graphs, but large ones are much tougher



# Scale Challenge

- May run out of space for vertices and edges (turns into “ball of string”)
- Very large datasets can reduce rendering speeds to less than real-time.
- Often use clustering to help
  - Extract highly connected sets of vertices
  - Collapse some vertices together

# Ego-Centered Networks






# Filtered Networks