

CS448B :: 4 Nov 2010

Graph and Tree Layout



Jeffrey Heer Stanford University

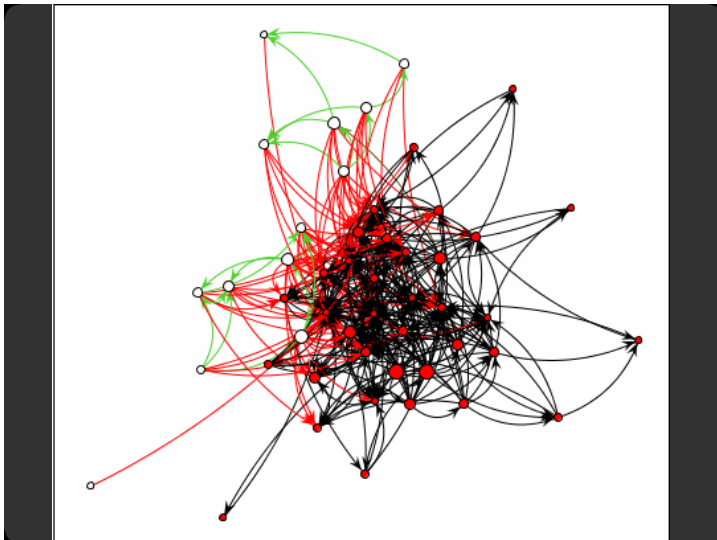
Topics

Graph and Tree Visualization

- Tree Layout
- Graph Layout

Goals

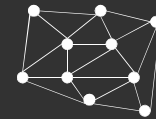
- Overview of layout approaches and their strengths and weaknesses
- Insight into implementation techniques



Graphs and Trees

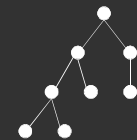
Graphs

- 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

The primary concern of graph drawing is the spatial layout 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 of Tree / Graph Layout

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

Tree Visualization

Indentation

- Linear list, indentation encodes depth

Node-Link diagrams

- Nodes connected by lines/curves

Enclosure diagrams

- Represent hierarchy by enclosure

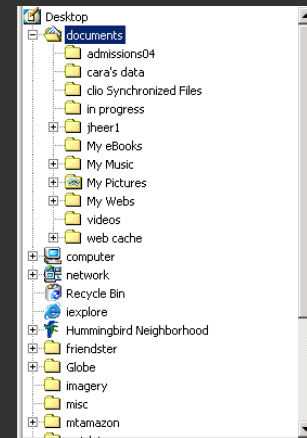
Layering

- Layering and alignment

Tree layout is fast: $O(n)$ or $O(n \log n)$, enabling real-time layout for interaction.



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 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 (typically towards authority or generality)

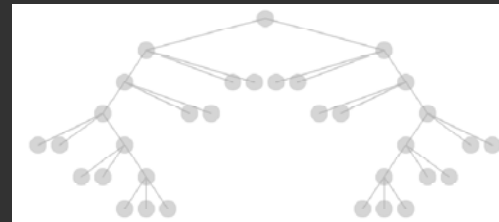


Basic Recursive Approach

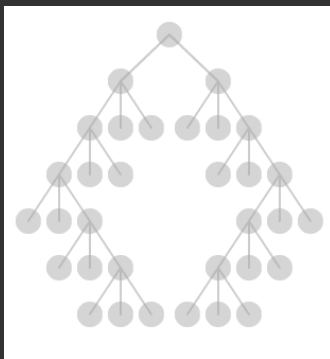
Repeatedly divide space for subtrees by leaf count

- Breadth of tree along one dimension
- Depth along the other dimension

Problem: exponential growth of breadth



Reingold & Tilford's Tidier Layout



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

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

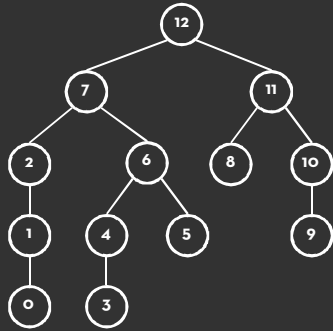
This extension was corrected by Buchheim et al to achieve a linear time algorithm.

Reingold-Tilford Layout

Design concerns

- Clearly encode depth level
- No edge crossings
- Isomorphic subtrees drawn identically
- Ordering and symmetry preserved
- *Compact layout (don't waste space)*

Reingold-Tilford Algorithm



Reingold-Tilford Algorithm



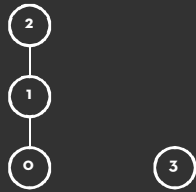
Reingold-Tilford Algorithm



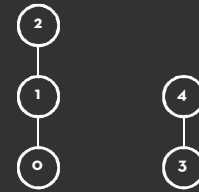
Reingold-Tilford Algorithm



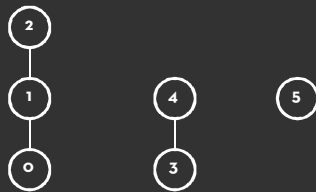
Reingold-Tilford Algorithm



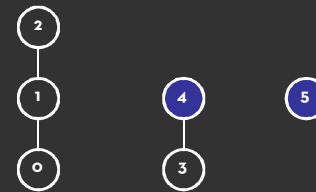
Reingold-Tilford Algorithm



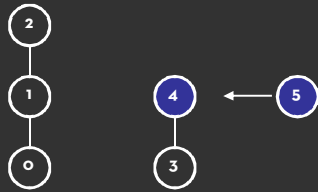
Reingold-Tilford Algorithm



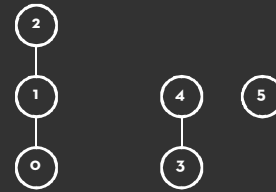
Reingold-Tilford Algorithm



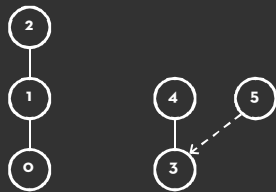
Reingold-Tilford Algorithm



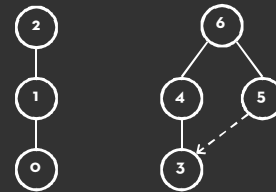
Reingold-Tilford Algorithm



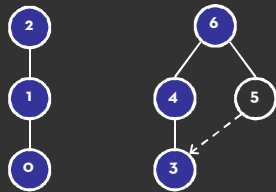
Reingold-Tilford Algorithm



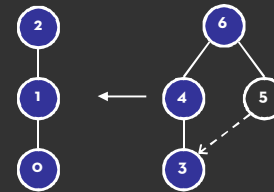
Reingold-Tilford Algorithm



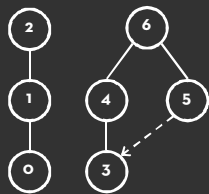
Reingold-Tilford Algorithm



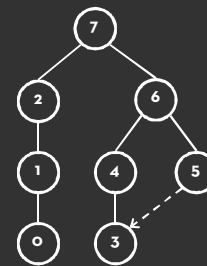
Reingold-Tilford Algorithm



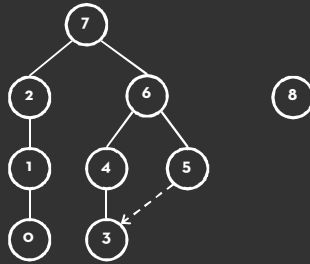
Reingold-Tilford Algorithm



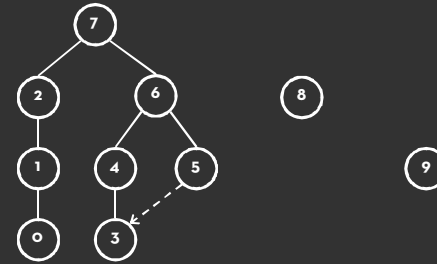
Reingold-Tilford Algorithm



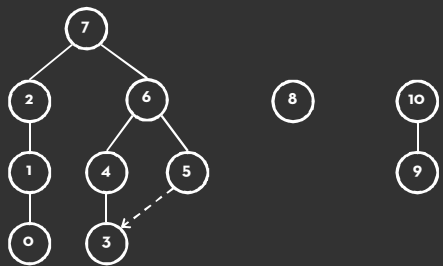
Reingold-Tilford Algorithm



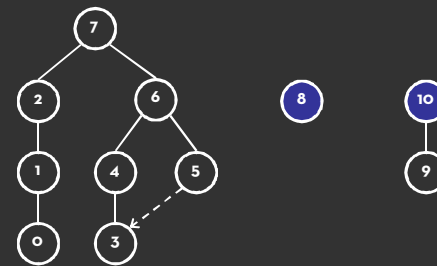
Reingold-Tilford Algorithm



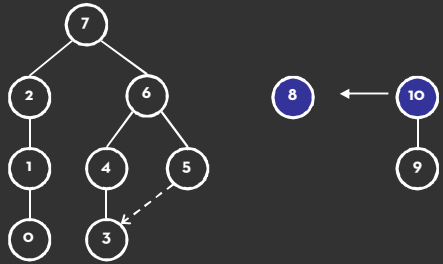
Reingold-Tilford Algorithm



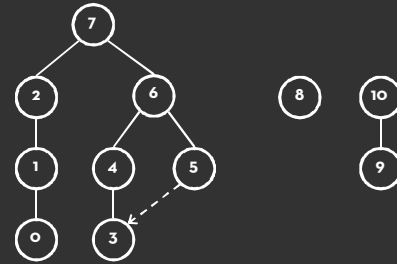
Reingold-Tilford Algorithm



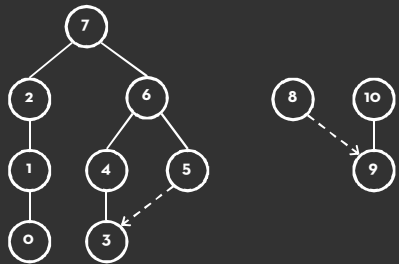
Reingold-Tilford Algorithm



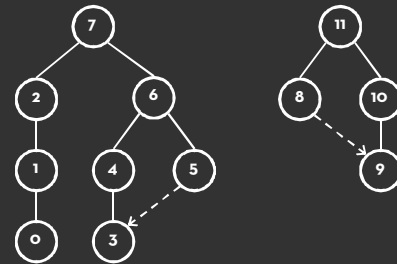
Reingold-Tilford Algorithm



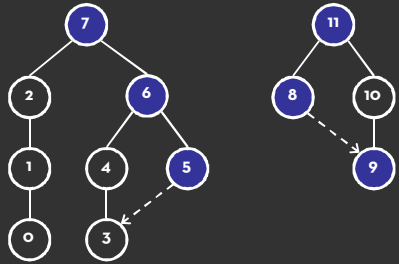
Reingold-Tilford Algorithm



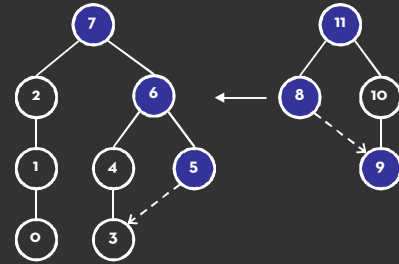
Reingold-Tilford Algorithm



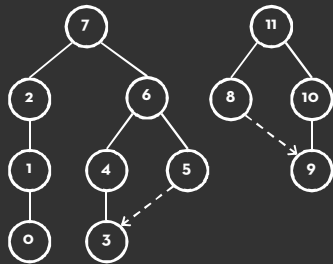
Reingold-Tilford Algorithm



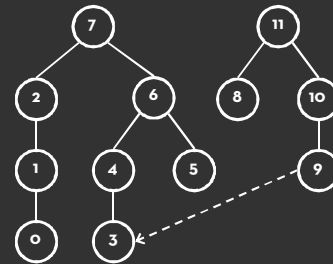
Reingold-Tilford Algorithm



Reingold-Tilford Algorithm



Reingold-Tilford Algorithm



Problems with Node-Link Diagrams

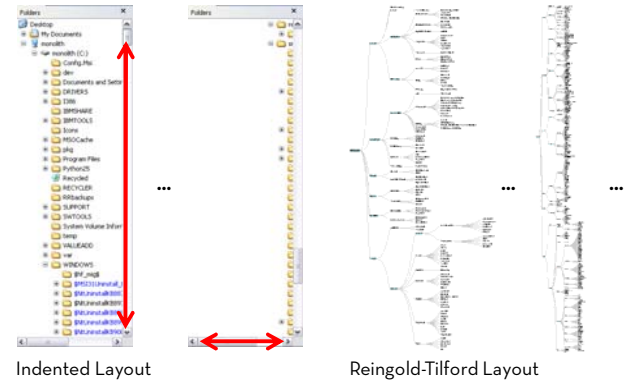
Scale

- Tree breadth often grows exponentially
- Even with tidier layout, quickly run out of space

Possible solutions

- Filtering
- Focus+Context
- Scrolling or Panning
- Zooming
- Aggregation

Visualizing Large Hierarchies



MC Escher, *Circle Limit IV*

Hyperbolic Layout

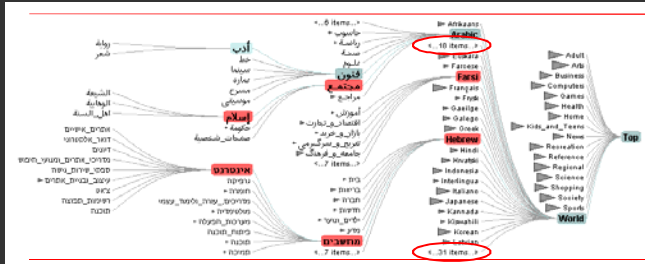


Perform tree layout in hyperbolic geometry, then project the result on to the Euclidean plane.

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

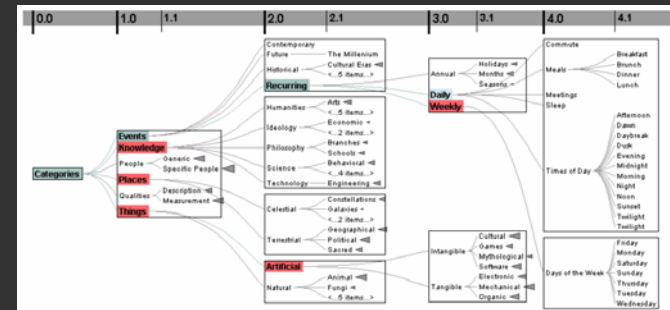
Also computable in 3D, projected into a sphere.

Degree-of-Interest Trees [AVI 04]



Space-constrained, multi-focal tree layout

Degree-of-Interest Trees



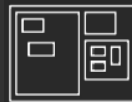
Cull “un-interesting” nodes on a per block basis until all blocks on a level fit within bounds.

Attempt to center child blocks beneath parents.

Enclosure 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

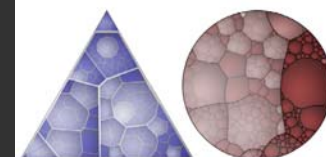
- Difficult to accurately read depth

TreeMaps



Recursively fill space based on a size metric for nodes. 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 tessellation.

Layered Diagrams

Signify tree structure using

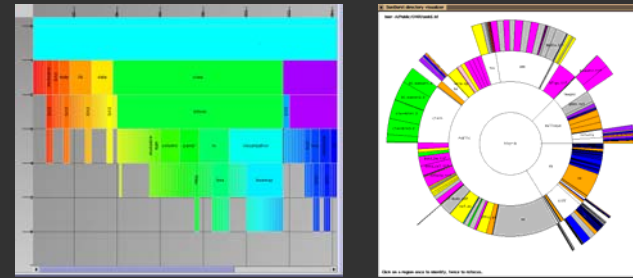
- Layering
- Adjacency
- Alignment



Involves recursive sub-division of space

We can apply the same set of approaches as in node-link layout.

Icicle and Sunburst Trees



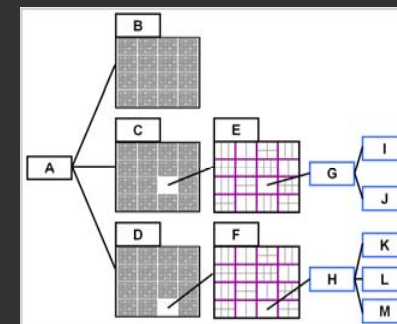
Higher-level nodes get a larger layer area, whether that is horizontal or angular extent.

Child levels are layered, constrained to parent's extent

Layered Tree Drawing

		Coffee			Espresso			
		Amaretto	Columbian	Decaf Irish Cr..	Caffe Latte	Caffe Mocha	Decaf Espresso	Regular Espresso
Central	Colorado							
	Illinois							
	Iowa							
	Missouri							
	Ohio							
East	Wisconsin							
	Connecticut							
	Florida							
	Massachusetts							
	New Hamp...							
South	New York							
	Louisiana							
	New Mexico							
	Oklahoma							
	Texas							
West	California							
	Nevada							
	Oregon							
	Utah							
	Washington							
		SUMMARY	SUMMARY	SUMMARY	SUMMARY	SUMMARY	SUMMARY	SUMMARY

Hybrids are also possible...



“Elastic Hierarchies”
Node-link diagram
with treemap nodes.

Administrivia

Final Project

Design a new visualization technique or system

Many options: new system, interaction technique, design study
6-8 page paper in conference paper format
2 Project Presentations

Schedule

Project Proposal: **Tuesday, Nov 9** (*end of day*)
Initial Presentation: **Tuesday, Nov 16 & Thursday, Nov 18**
Poster Presentation: **Tuesday, Dec 7** (4-6pm)
Final Papers: **Friday, Dec 10** (*end of day*)

Logistics

Groups of up to 3 people, graded individually
Clearly report responsibilities of each member

Final Project Ideas

Read the Final Project Wiki Page ASAP!

Also follow the links for suggested projects. A number of domain experts have provided project ideas for which they would like to collaborate with you!

Collaborators include folks from: Stanford Biology, Stanford Computer Science, Willow Garage, Human Rights Data Analysis Group, The Encyclopedia of Life

We **strongly** encourage you to consider working in a partnership with a domain expert, particularly if you have difficulty formulating a project idea.

Graph Visualization

Approaches to Graph Drawing

Direct Calculation using Graph Structure

- Tree layout on spanning tree
- Hierarchical layout
- Adjacency matrix layout

Optimization-based Layout

- Constraint satisfaction
- Force-directed layout

Attribute-Driven Layout

- Layout using data attributes, not linkage

Spanning Tree Layout

Many graphs are tree-like or have useful spanning trees

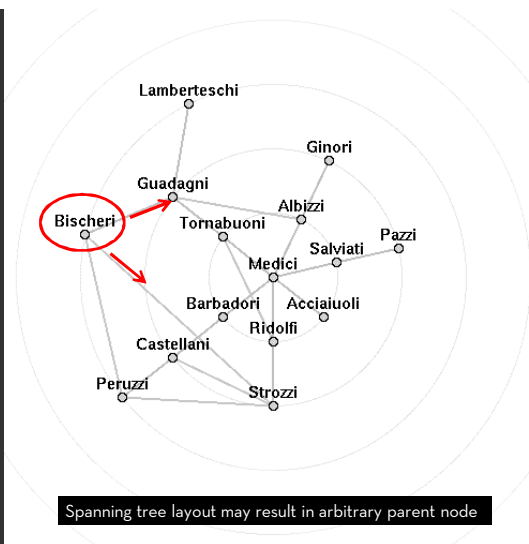
- Websites, Social Networks

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

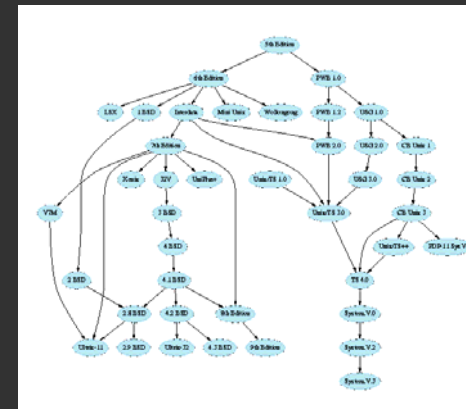
Heuristics may further improve layout



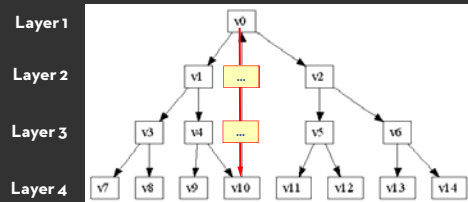
Sugiyama-style graph layout

Evolution of the UNIX operating system

Hierarchical layering based on descent

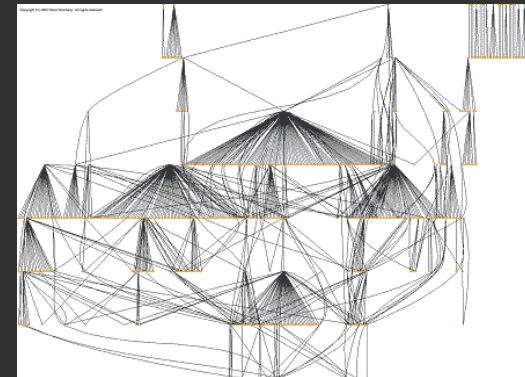


Sugiyama-style graph layout



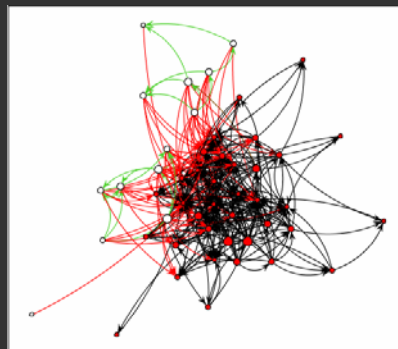
- Reverse edges to remove cycles
- Create dummy nodes to “fill in” missing layers
- Assign nodes to hierarchy layers
- Arrange nodes within layer, minimize edge crossings
- Route edges - layout splines if needed

Hierarchical graph layout



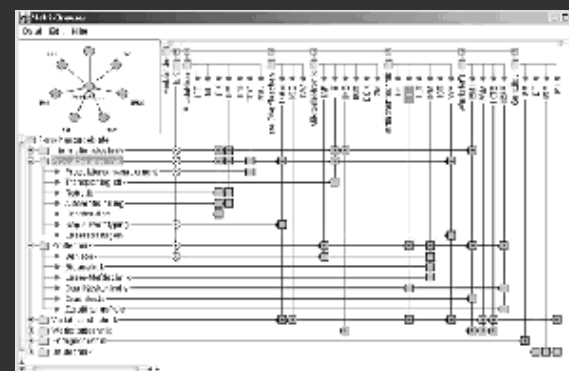
Gnutella network

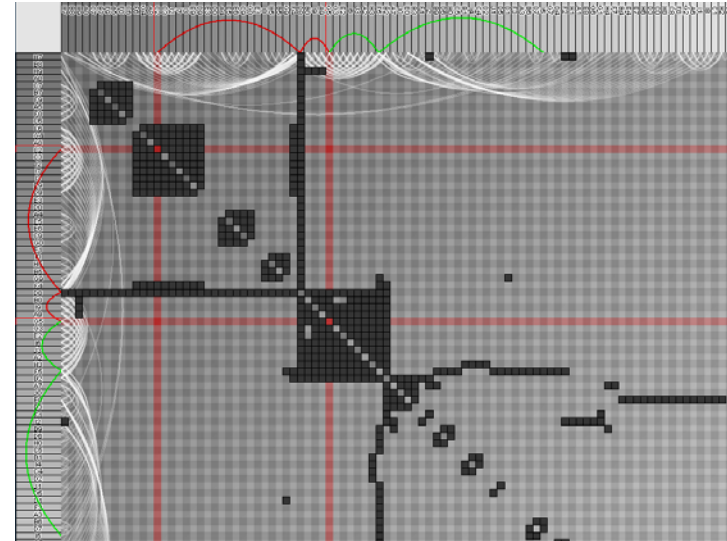
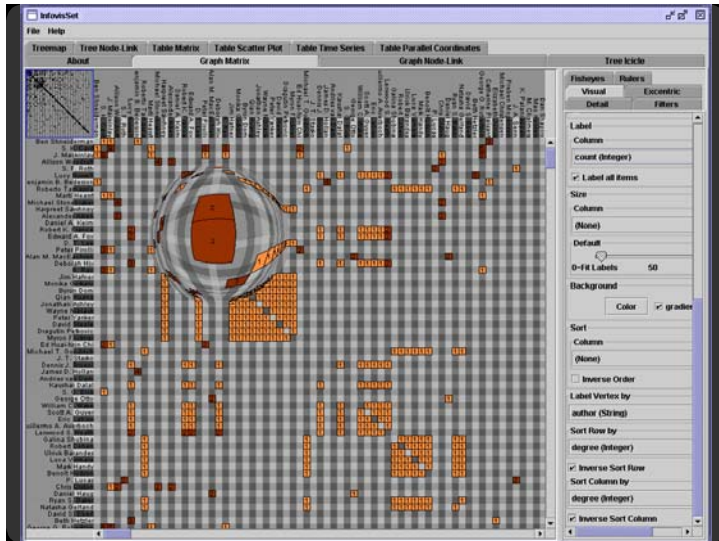
Limitations of Node-Link Layout



Edge-crossings and occlusion

Adjacency Matrices





Optimization Techniques

Treat layout as an *optimization problem*

- Define layout using a set of *constraints*: equations the layout should try to obey
- Use optimization algorithms to solve

Common approach for undirected graphs

- *Force-Directed Layout* most common

We can introduce directional constraints

- *DiG-CoLa* (Di-Graph Constrained Optimization Layout) [Dwyer 05]

Optimizing “Aesthetic” Constraints

Minimize edge crossings

Minimize area

Minimize line bends

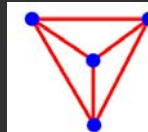
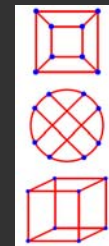
Minimize line slopes

Maximize smallest angle between edges

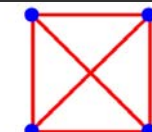
Maximize symmetry

but, can’t do it all.

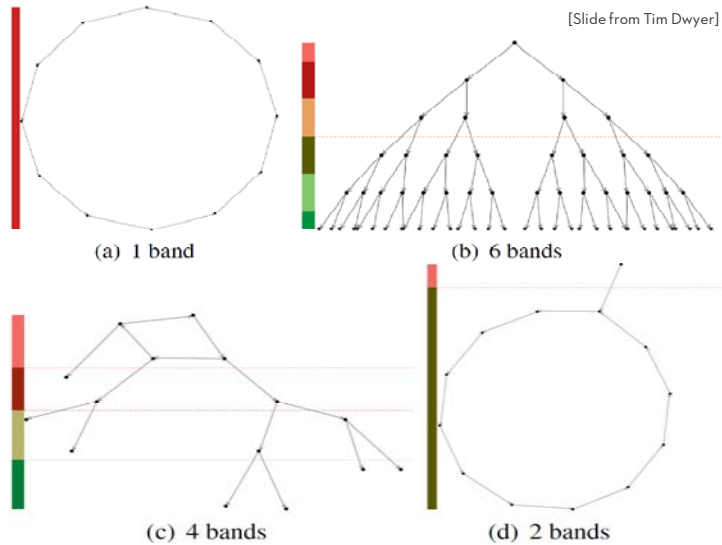
Optimizing these criteria is often NP-Hard, requiring approximations.



min # crossings



max symmetries



Attribute-Driven Layout

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

Idea: Use data attributes to perform layout

- e.g., scatter plot based on node values

Dynamic queries and/or brushing can be used to explore connectivity

Attribute-Driven Layout

The “Skitter” Layout

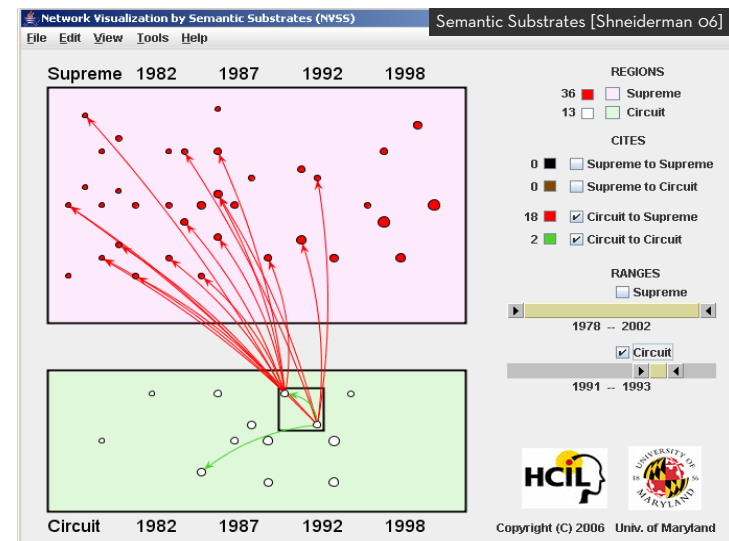
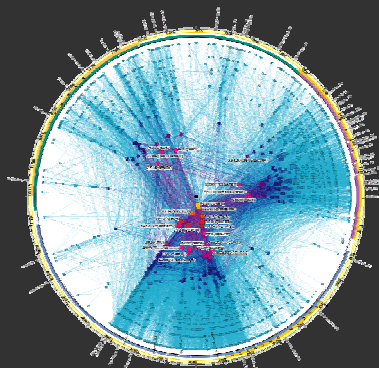
- Internet Connectivity
- Radial Scatterplot

Angle = Longitude

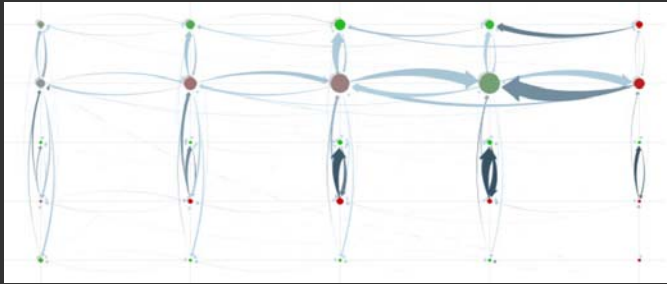
- Geography

Radius = Degree

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

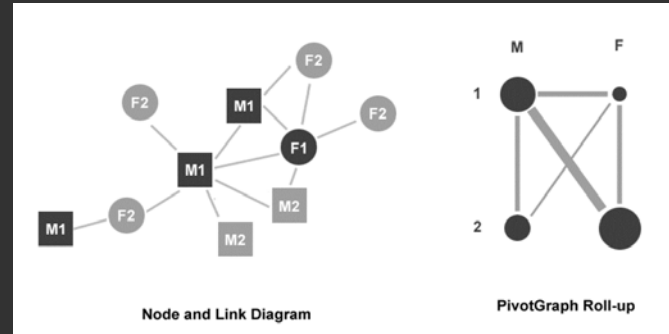


PivotGraph [Wattenberg 2006]



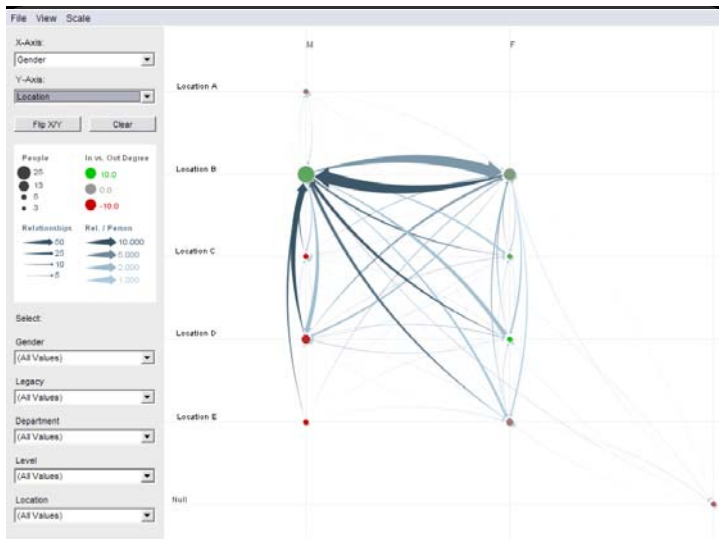
Layout aggregated graphs according to node attributes. Analogous to pivot tables and trellis display.

PivotGraph

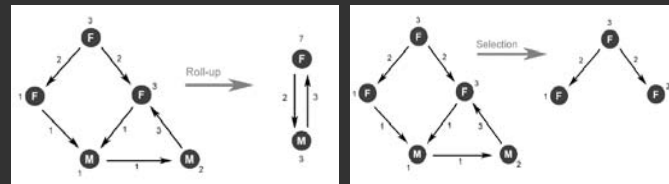


Node and Link Diagram

PivotGraph Roll-up

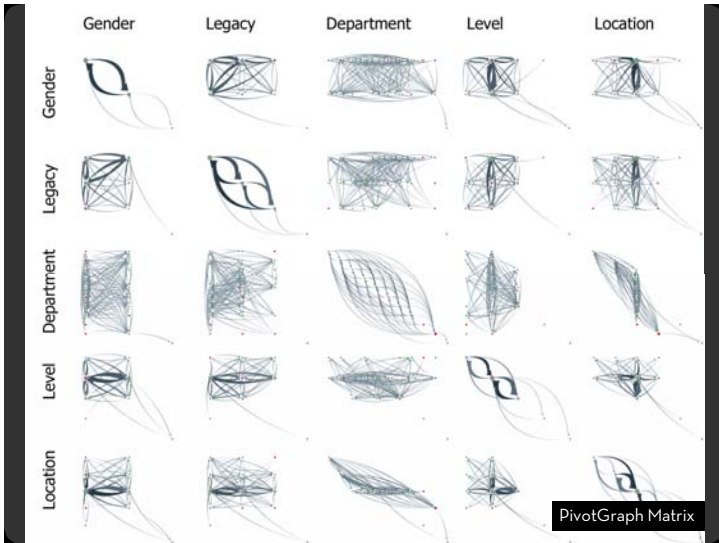


Operators



Roll-Up
Aggregate items with matching data values

Selection
Filter on data values

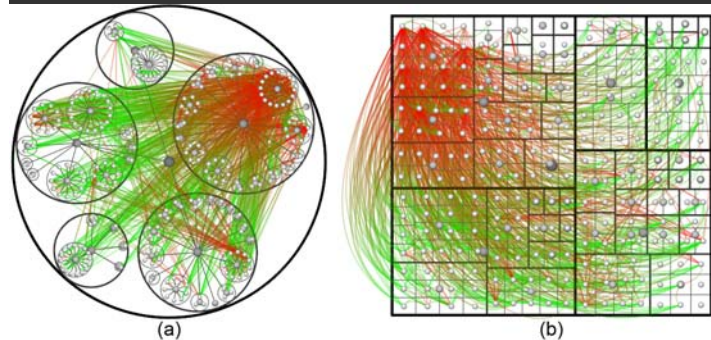


Limitations of PivotGraph

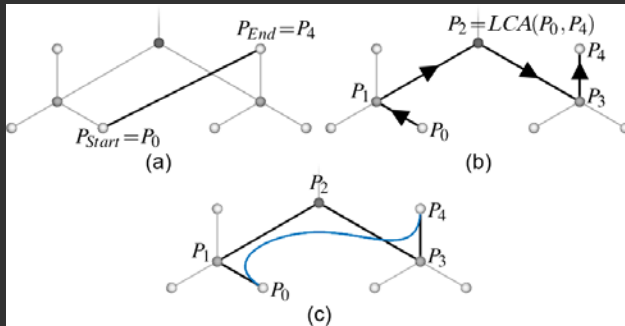
Only 2 variables (no nesting as in Tableau)
 Doesn't support continuous variables
 Multivariate edges?

Hierarchical Edge Bundles

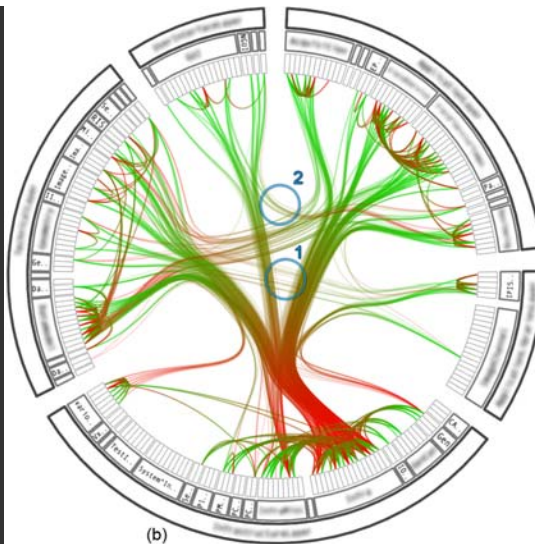
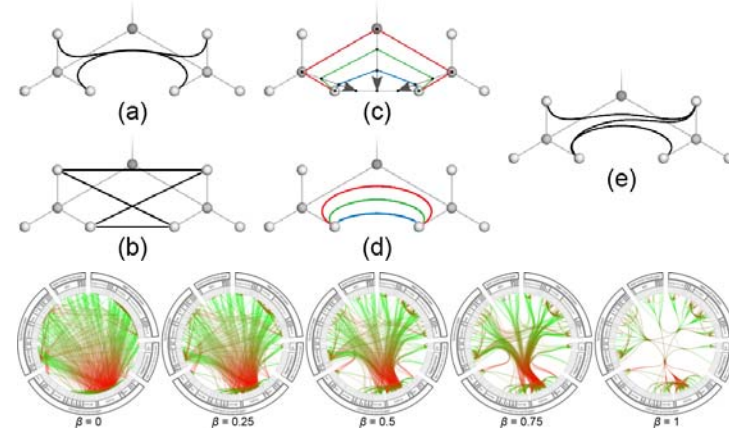
Trees with Adjacency Relations



Bundle Edges along Hierarchy



Configuring Edge Tension



Summary



Tree Layout

- Indented / Node-Link / Enclosure / Layers
- How to address issues of scale?
 - Filtering and Focus + Context techniques

Graph Layout

- Tree layout over spanning tree
- Hierarchical "Sugiyama" Layout
- Optimization (Force-Directed Layout)
- Attribute-Driven Layout