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

0
1
2

3
4
5
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.
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
Focus + Context
Visualizing Large Hierarchies

Indented Layout

Reingold-Tilford 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 tessellation.
Layered Diagrams

Signify tree structure using

- Layering
- Adjacency
- Alignment

Involves recursive sub-division of space.
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

<table>
<thead>
<tr>
<th></th>
<th>Coffee</th>
<th>Espresso</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Amaretto</td>
<td>Columbian</td>
</tr>
<tr>
<td></td>
<td>Decaf Irish Cr..</td>
<td>Caffe Latte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Caffe Mocha</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decaf Espresso</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Regular Espresso</td>
</tr>
<tr>
<td>Central</td>
<td>Colorado</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Illinois</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Iowa</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Missouri</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ohio</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wisconsin</td>
<td></td>
</tr>
<tr>
<td>East</td>
<td>Connecticut</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Florida</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Massachusetts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New Hamps..</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New York</td>
<td></td>
</tr>
<tr>
<td>South</td>
<td>Louisiana</td>
<td></td>
</tr>
<tr>
<td></td>
<td>New Mexico</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Oklahoma</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Texas</td>
<td></td>
</tr>
<tr>
<td>West</td>
<td>California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nevada</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Oregon</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Utah</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Washington</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SUM(Profit)</th>
<th>SUM(Profit)</th>
<th>SUM(Profit)</th>
<th>SUM(Profit)</th>
<th>SUM(Profit)</th>
<th>SUM(Profit)</th>
<th>SUM(Profit)</th>
<th>SUM(Profit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-20K 0K 20K</td>
<td>-20K 0K 20K</td>
<td>-20K 0K 20K</td>
<td>-20K 0K 20K</td>
<td>-20K 0K 20K</td>
<td>-20K 0K 20K</td>
<td>-20K 0K 20K</td>
<td>-20K 0K 20K</td>
</tr>
</tbody>
</table>
Hybrids are also possible...

“Elastic Hierarchies”
Node-link diagram with treemap nodes.
Administrivia
Final Project Schedule

Proposal  Tues, May 12 (5pm)
Presentation  Thur, May 21 (slides: 5/20, 5pm)
Poster & Demo  Mon, Jun 8 (5-8pm)
Final Paper  Thur, Jun 11 (8am)

Logistics
Groups of up to 4 people
Clearly report responsibilities of each member
Graph Layout
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
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
with air resistance
Edges = springs

\[ F = G \cdot m_1 \cdot m_2 / (x_i - x_j)^2 \]
\[ F = -b \cdot v_i \]
\[ F = -k \cdot (x_i - x_j - L) \]

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
Zephoria

User ID: 21721
Friends: 256
Age: 24
Gender: Female
Status: Single
Location: San Francisco, CA
Hometown: Lancaster, PA
Occupation: researcher, social networks, identity, context
Interests: schizophrenia, observing people, culture, questioning power, reading, Buddhism, inquiry, computer-mediated communication, social networks, technology, anthropology, stoning
Music: portamento,Mushroom, Sun Kilu, blog:Digital Structure, Aoi, DrDranco, downtempo, Thievery Corporation, Beth Orton, Manchuks, Veeva, White Stripes
TV Shows: ??
Movies: Clockwork Orange, American Beauty, Fight Club, Boys Don't Cry
Member Since: ??
Last Login: 2003-12-21
Last Updated: 2003-12-21
About: 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... enjoyable.
A partner in craziness, oh...
Constrained Optimization

Minimize stress function

\[
\text{stress}(X) = \sum_{i<j} w_{ij} ( \|X_i - X_j\| - d_{ij})^2
\]

- \(X\): node positions, \(d\): optimal edge length,
- \(w\): normalization constants

Says: *Try to place nodes \(d_{ij}\) apart*
Constrained Optimization

Minimize stress function

\[ \text{stress}(X) = \sum_{i<j} w_{ij} \left( \|X_i - X_j\| - d_{ij} \right)^2 \]

- \(X\): node positions, \(d\): optimal edge length,
- \(w\): normalization constants

Says: *Try to place nodes \(d_{ij}\) apart*

Add hierarchy ordering constraints

\[ E_H(y) = \sum_{(i,j) \in E} \left( y_i - y_j - \delta_{ij} \right)^2 \]

- \(y\): node y-coordinates
- \(\delta\): 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
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
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)
Layout aggregate graphs using 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 Bundling
Trees with Adjacency Relations
Bundle Edges Along Hierarchy

(a) $P_{\text{Start}} = P_0$

(b) $P_{\text{End}} = P_4$

(c) $P_2 = \text{LCA}(P_0, P_4)$
Configuring Edge Tension
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