Hierarchies
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
Network Analysis Tasks  [Pretorius '13]

Structure-based: relationships and connectivity

Attribute-based: specific node/link attributes

Browsing: understand paths in the data

Estimation: summarization and temporal changes
Network Analysis Tasks [Pretorius '13]

**Structure-based:** relationships and connectivity

Find all of the friends of friends for Taylor.
Find all of the people who are friends with Jordan and Alex.
Six degrees of separation: shortest path between two individuals.

**Attribute-based:** specific node/link attributes

**Browsing:** understand paths in the data

**Estimation:** summarization and temporal changes
Network Analysis Tasks [Pretorius '13]

**Structure-based:** relationships and connectivity
Find all of the friends of friends for Taylor.
Find all of the people who are friends with Jordan and Alex.
Six degrees of separation: shortest path between two individuals.

**Attribute-based:** specific node/link attributes
Find all "students" attending CSE412.
Find all the "friends" and "family" of Alex.

**Browsing:** understand paths in the data

**Estimation:** summarization and temporal changes
Network Analysis Tasks [Pretorius '13]

**Structure-based**: relationships and connectivity
Find all of the friends of friends for Taylor.
Find all of the people who are friends with Jordan and Alex.
Six degrees of separation: shortest path between two individuals.

**Attribute-based**: specific node/link attributes
Find all "students" attending CSE412.
Find all the "friends" and "family" of Alex.

**Browsing**: understand paths in the data
Find Alex's friend Taylor, and then Taylor's friend Jordan.

**Estimation**: summarization and temporal changes
Network Analysis Tasks  [Pretorius '13]

**Structure-based:** relationships and connectivity
Find all of the friends of friends for Taylor.
Find all of the people who are friends with Jordan and Alex.
Six degrees of separation: shortest path between two individuals.

**Attribute-based:** specific node/link attributes
Find all "students" attending CSE412.
Find all the "friends" and "family" of Alex.

**Browsing:** understand paths in the data
Find Alex's friend Taylor, and then Taylor's friend Jordan.

**Estimation:** summarization and temporal changes
How does Jordan's friend group change over the course of the year?
Topics

**TODAY** - Tree Visualization

**Wed** - Graph Layout: Node-Link Diagrams

**Wed** - Alternative Visualizations and Techniques

Select an image to jump to those slides.
Tree Visualization
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
Indentation
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.
What tasks are these good for?

Benefits:
Navigation + Browsing, Parent-Child Relationships

Disadvantages:
Estimation, Comparison, Network Overview
Node-Link Diagrams
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
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’s “Tidy” Layout
Reingold-Tilford Layout

Design Considerations
- Clearly encode depth
- No edge crossings
- Draw isomorphic subtrees identically (same shape)
- Preserve layout ordering and symmetry
- Compact, space-saving 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
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.
Cone Trees [Robertson 91]
Balloon Trees

Described as a 2D variant of a Cone Tree.

Not just a flattening process: circles must not overlap.
Analysis Tasks: 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
Hyperbolic Layout

Perform tree layout in hyperbolic geometry, project the result onto 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
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.
What tasks are supported/missing?
Indentation & Node-Link Diagrams

Encode structure in **2D space** (breadth/depth)

**Benefits**
Clearly depicts node relationships / structure
Structure-based or browsing tasks

**Problems**
Even with tidy layout, quickly run out of space

**Missing**
Attribute-based encodings
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.
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.  
   \textit{Mathematically true!}

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

3. Similar aspect ratios are easier to compare.  
   \textit{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?*
Position is generally more effective than area, but...
What happens when the element count gets high? What happens when comparing groups of elements, such as leaf values vs. internal node values?
At low densities (< 4k elements), bar charts more accurate than treemaps for leaf-node comparisons.

At higher density, treemaps led to faster judgments. Treemaps better for group-level comparisons.

Treemaps vs. Bar Charts [Kong et al. ’10]
Interactive Example...
Cushion Treemaps [van Wijk & Wetering ’99]

Uses shading to emphasize hierarchical structure.
Cascaded Treemaps  [Lü & Fogarty ’08]

Uses 2.5D effect to emphasize hierarchy relations.
Voronoï Treemaps [Balzer et al. ‘05]

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

Use iterative, weighted Voronoï tessellations to achieve cells with value-proportional areas.
Iterative Voronoi Tessellations  [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

2.00% of visits begin with this sequence of pages
Layered Trees Useful Elsewhere...

<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>Central</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Colorado</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Illinois</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iowa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missouri</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ohio</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wisconsin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>East</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connecticut</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Florida</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Massachusetts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Hampshire</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New York</td>
<td></td>
<td></td>
</tr>
<tr>
<td>South</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Louisiana</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Mexico</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oklahoma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Texas</td>
<td></td>
<td></td>
</tr>
<tr>
<td>West</td>
<td></td>
<td></td>
</tr>
<tr>
<td>California</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nevada</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oregon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utah</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Washington</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SUM(Profit) -20K 0 20K | SUM(Profit) -20K 0 20K | SUM(Profit) -20K 0 20K | SUM(Profit) -20K 0 20K | SUM(Profit) -20K 0 20K | SUM(Profit) -20K 0 20K | SUM(Profit) -20K 0 20K | SUM(Profit) -20K 0 20K
Topics

**TODAY** - Tree Visualization

**Wed** - Graph Layout: Node-Link Diagrams

**Wed** - Alternative Visualizations and Techniques

Select an image to jump to those slides.