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
Topics

Tree Visualization
Graph Layout: Node-Link Diagrams
  Sugiyama-Style Layout
  Force-Directed Layout
Alternatives to Node-Link Diagrams
  Matrix Diagrams
  Attribute-Driven Layout & Hive Plots
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
Tree 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
Single-Focus (Accordion) List

Separate breadth & depth along 2D. Focus on a single path at a time.
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
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)*

![Diagram of Reingold-Tilford Layout](image)
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
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout

Diagram: Nodes representing numbers 2, 1, 0, and 3
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

Diagram showing a tree with nodes labeled 0 to 7 and additional nodes labeled 8 and 9.
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout
Reingold-Tilford Layout
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.
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
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
Degree-of-Interest Trees

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.
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
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
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.

Interactive Example...
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.  
   *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?*
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?*
Cushion Treemaps [van Wijk & Wetering ’99]

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

Uses 2.5D effect to emphasize hierarchy relations.
Instead of rectangles, create treemaps with arbitrary polygonal shapes and boundary.

Use iterative, weighted Voronoi tessellations to achieve cells with value-proportional areas.
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>Region</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>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>Region</th>
<th>Coffee</th>
<th>Espresso</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SUM(Profit)</td>
<td>SUM(Profit)</td>
</tr>
<tr>
<td>Central</td>
<td>-20K</td>
<td>0K</td>
</tr>
<tr>
<td>East</td>
<td>-20K</td>
<td>0K</td>
</tr>
<tr>
<td>South</td>
<td>-20K</td>
<td>0K</td>
</tr>
<tr>
<td>West</td>
<td>-20K</td>
<td>0K</td>
</tr>
</tbody>
</table>
Node-Link Graph Layout
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
GraphViz package!
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
Sugiyama-style layout emphasizes hierarchy. However, cycles in the graph may mislead. Long edges can impede perception of proximity.
Force-Directed Layout
Interactive Example: Configurable Force Layout
Zephoria

User ID: 21721
Friends: 286
Age: ??
Gender: Male
Status: Single
Location: San Francisco, CA
Hometown: Lancaster, PA
Occupation: researcher, social networks, identity, context
Interests: apophasia, observing people, culture, questioning power, reading, Buddhism, games, computer-mediated communication, social networks, technology, anthropology, stonking
Music: psytrance/goa trance [Infected Mushroom, Enki:... Ingles/Digital Structures], Anthrax, downtempo, Thievery Corporation, Beth Orson, Massive, Ween, White Stripes
Books: Authors: Erving Goffman, Stanley Milgram, Jeanette Winterson, Eric Schlosser, Leslie Faint, Dorothy Allison, Italo Calvino, Hermann Hesse
TV Shows: ??
??
Member Since: 2003-9-21
Last Login: 2003-9-21
Last Updated: 2003-9-21

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

community >>> 
Enable

search >>>
Use the Force!

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

7,922 nodes
11,881 edges

[Kai Chang]
Force-Directed Layout

Nodes = charged particles \[ F = q_i \cdot q_j / d_{ij}^2 \]
with air resistance \[ F = -b \cdot v_i \]
Edges = springs \[ F = k \cdot (L - d_{ij}) \]

At each timestep, calculate forces acting on nodes. Integrate for updated velocities and positions.

D3’s force layout uses \textbf{velocity Verlet} integration. Assume uniform mass \( m \) and timestep \( \Delta t \):
\[ F = ma \rightarrow F = a \rightarrow F = \Delta v / \Delta t \rightarrow F = \Delta v \]

Forces simplify to velocity offsets!
N-Body Force

Naïve calculation of repulsive force doesn’t scale! Comparing all pairs of nodes is $O(V^2)$.

We can approximate force calculations using a spatial index (e.g., quadtree) to achieve $O(V \log V)$.

One such approach is the **Barnes-Hut algorithm**, originally created for astronomical simulations.

The key idea is to approximate forces from distant nodes by comparing to aggregate centers of charge rather than individual nodes.
Naive calculation of forces at a point uses sum of forces from all other n-1 points.
For fast approximate calculation, we build a spatial index (here, a quadtree) and use it to compare with distant groups of points instead.
The Barnes-Hut $\theta$ parameter controls when to compare with an aggregate center of charge.

$w_{\text{quadnode}} / d_{ij} < \theta$ ?

$\theta = 0.5$
\[ \theta = 0.9 \] (default setting)
Different forces can be composed to create an expressive space of custom layouts.

A **beeswarm plot** can be made by combining:
- Attractive $X$ and $Y$ forces to draw nodes of a certain category to a desired point
- **Collide** force to detect collision & remove overlap
Matrix Diagrams
Limitations of Node-Link Layouts

Edge-crossings and occlusion! Poor scalability....
Adjacency Matrices
Seriation / Ordination / Permutation

Goal: Ensure similar items placed near each other. E.g., minimize sum of distances of adjacent items.

Requires combinatorial optimization: NP-Hard!

Instead, approximate / heuristic approaches used:
- Perform hierarchical clustering, sort cluster tree.
- Apply approximate traveling salesman solver.

Seriation initially used in archaeology for relative dating of artifacts based on observed properties.
 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

Attributes may be associated with nodes or edges
or may be statistical properties of the graph.

Use 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)
Drawing all edges is not particularly useful here...
Node layout determined by geographic location. Adjacent edges shown on node selection.
PivotGraph [Wattenberg ’06]

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
PivotGraph Matrices

Gender | Legacy | Department | Level | Location
--- | --- | --- | --- | ---

Gender

Legacy

Department

Level

Location

PivotGraph Matrix
Limitations of PivotGraph

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

[Freire et al. ’10](#)

## Table:

<table>
<thead>
<tr>
<th>ID</th>
<th>Edge count</th>
<th>Edge-vertex ratio</th>
<th>Edge density</th>
<th>Vertex count</th>
<th>Component count</th>
<th>Component sizes</th>
<th>Duplicate edge count</th>
<th>Degree</th>
<th>Duration (s)</th>
<th>Start</th>
<th>Tower</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>193</td>
<td>0.906</td>
<td>3.87E-3</td>
<td>213</td>
<td>45</td>
<td></td>
<td></td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>312</td>
<td>1.156</td>
<td>3.67E-3</td>
<td>270</td>
<td>33</td>
<td></td>
<td></td>
<td>45</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>430</td>
<td>1.119</td>
<td>3.71E-3</td>
<td>303</td>
<td>19</td>
<td></td>
<td></td>
<td>90</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>555</td>
<td>1.647</td>
<td>3.73E-3</td>
<td>337</td>
<td>11</td>
<td></td>
<td></td>
<td>132</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>592</td>
<td>1.711</td>
<td>3.72E-3</td>
<td>346</td>
<td>9</td>
<td></td>
<td></td>
<td>148</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>568</td>
<td>1.701</td>
<td>3.83E-3</td>
<td>334</td>
<td>6</td>
<td></td>
<td></td>
<td>142</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>522</td>
<td>1.568</td>
<td>3.62E-3</td>
<td>333</td>
<td>12</td>
<td></td>
<td></td>
<td>121</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>393</td>
<td>1.264</td>
<td>3.38E-3</td>
<td>311</td>
<td>21</td>
<td></td>
<td></td>
<td>67</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>289</td>
<td>1.032</td>
<td>3.30E-3</td>
<td>280</td>
<td>42</td>
<td></td>
<td></td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>181</td>
<td>0.858</td>
<td>3.83E-3</td>
<td>211</td>
<td>50</td>
<td></td>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>191</td>
<td>0.88</td>
<td>3.69E-3</td>
<td>217</td>
<td>54</td>
<td></td>
<td></td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>284</td>
<td>1.036</td>
<td>3.24E-3</td>
<td>274</td>
<td>44</td>
<td></td>
<td></td>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>413</td>
<td>1.295</td>
<td>3.33E-3</td>
<td>319</td>
<td>28</td>
<td></td>
<td></td>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>541</td>
<td>1.587</td>
<td>3.55E-3</td>
<td>341</td>
<td>13</td>
<td></td>
<td></td>
<td>129</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>567</td>
<td>1.673</td>
<td>3.72E-3</td>
<td>339</td>
<td>13</td>
<td></td>
<td></td>
<td>140</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>546</td>
<td>1.625</td>
<td>3.66E-3</td>
<td>336</td>
<td>21</td>
<td></td>
<td></td>
<td>133</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>515</td>
<td>1.58</td>
<td>3.68E-3</td>
<td>326</td>
<td>23</td>
<td></td>
<td></td>
<td>125</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>404</td>
<td>1.299</td>
<td>3.49E-3</td>
<td>311</td>
<td>26</td>
<td></td>
<td></td>
<td>68</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>314</td>
<td>1.154</td>
<td>3.56E-3</td>
<td>272</td>
<td>51</td>
<td></td>
<td></td>
<td>51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>224</td>
<td>0.982</td>
<td>3.74E-3</td>
<td>228</td>
<td>53</td>
<td></td>
<td></td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>190</td>
<td>0.909</td>
<td>3.84E-3</td>
<td>209</td>
<td>56</td>
<td></td>
<td></td>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HivePlots
[Krzywinski ’11]

Nodes (dots) may be replicated.

Nodes sorted on radial axes by network statistics (e.g., by degree).

Different axes may contain different subsets of nodes.

egweb.bcgsc.ca
Summary: Hierarchies & Networks

Tree Layout
Indented / Node-Link / Enclosure / Layers
Focus+Context techniques for scale

Graph Layout
.Spanning Tree Layout, “Sugiyama” Layout
Arc Diagrams
Force-Directed Layout, Optimization Methods
Matrix Diagrams
Attribute-Driven Layout