Module 2

Visual Analytics

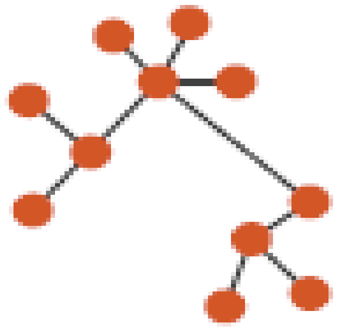
Networks and Trees

Reference Book
Tamara Munzer**, Visualization Analysis and Design** -, CRC Press 2014 .
(Chapter 9)

Arrange Networks and Trees

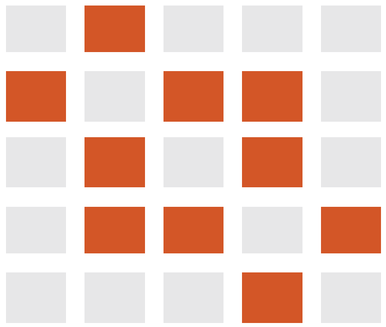
➔ Node–Link Diagrams
Connection Marks

✓ NETWORKS ✓ TREES



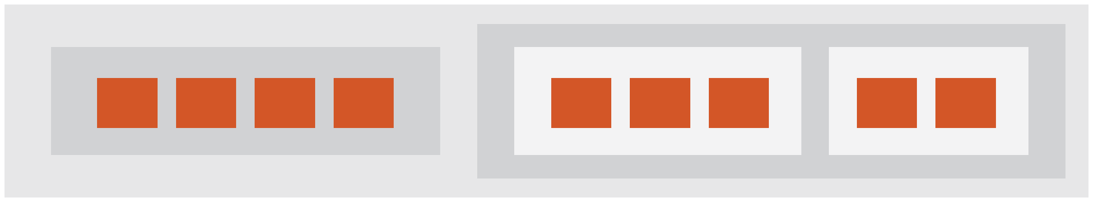
➔ Adjacency Matrix
Derived Table

✓ NETWORKS ✓ TREES



➔ Enclosure
Containment Marks

✗ NETWORKS ✓ TREES

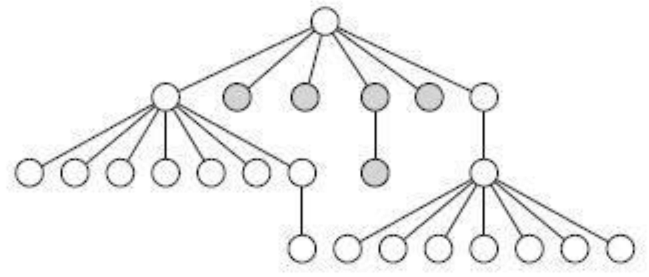


Design Choices

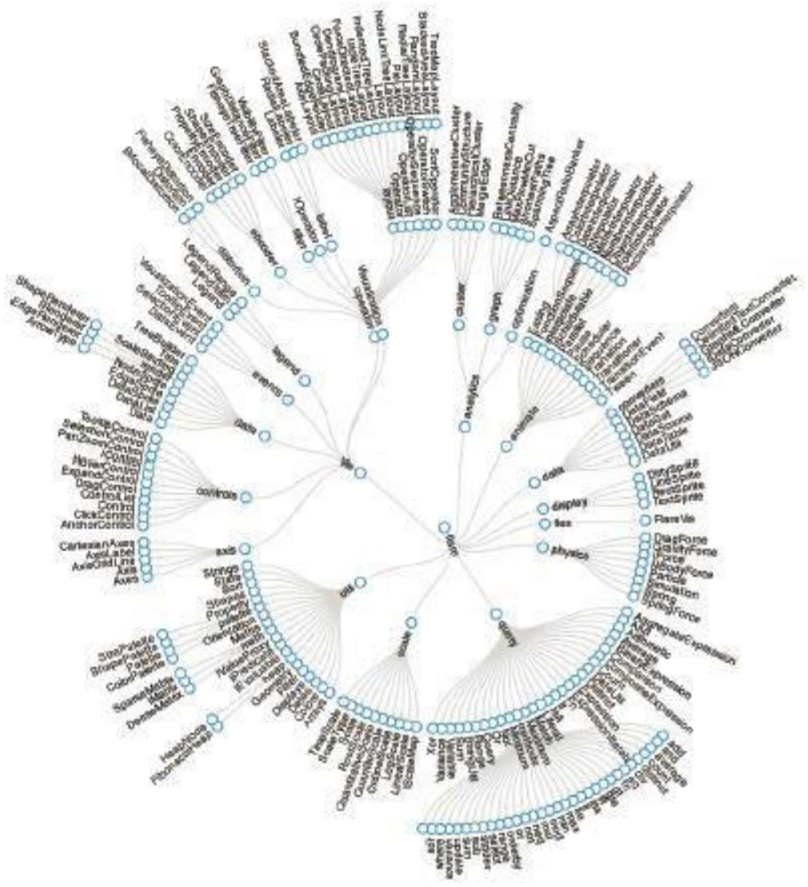
- Connectivity
  - Node-link graphs
  - Good for finding pairwise/multiway relations
  - Good for following paths through structure
  - Force-directed placement
- Containment
  - Effective at showing hierarchical structure
  - Good for finding attributes of leaf nodes
  - Treemaps, nested views
- Matrices

Node link diagrams

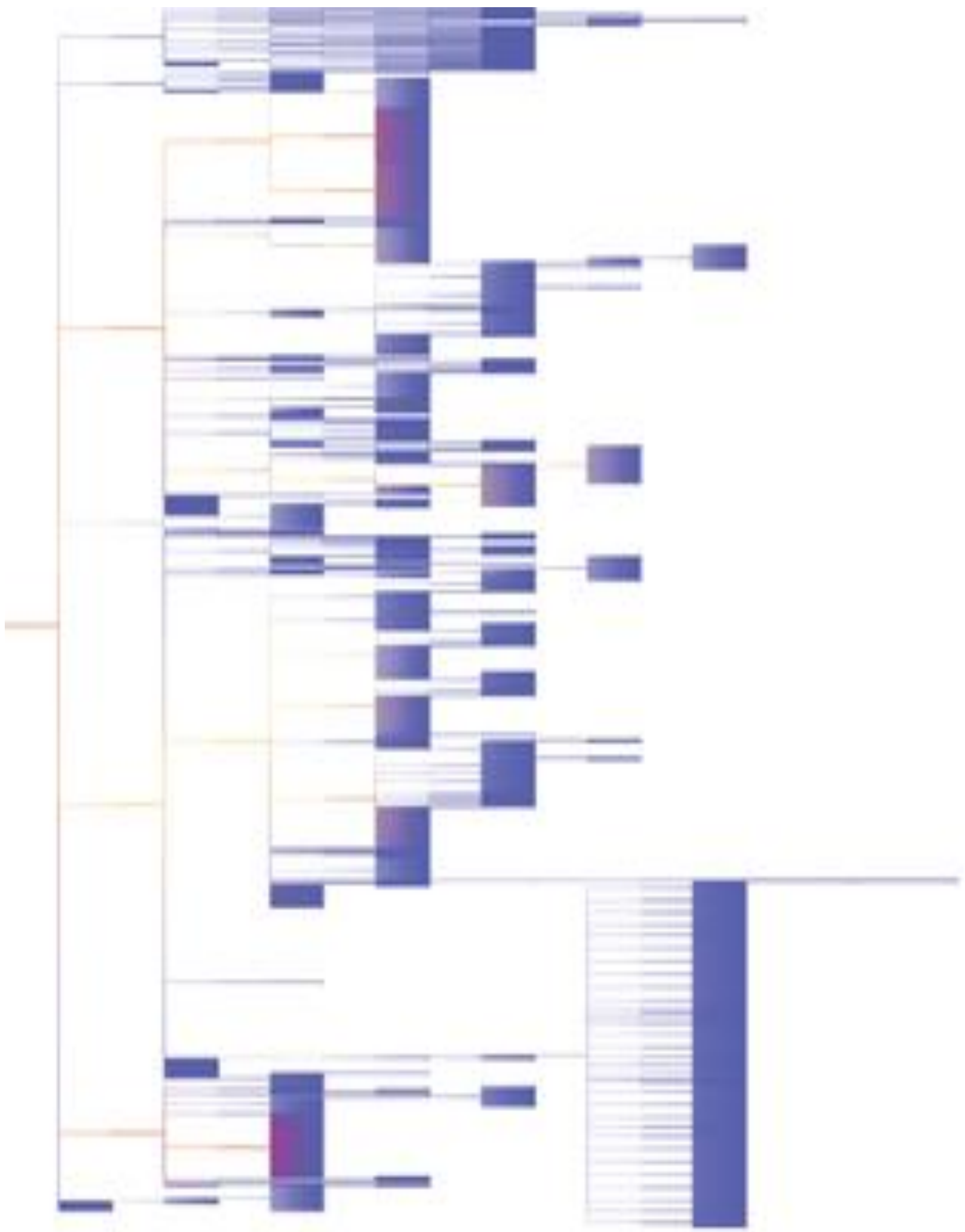
- The most common visual encoding idiom for tree and network data is with node–link diagrams, where nodes are drawn as point marks and the links connecting them are drawn as line marks.

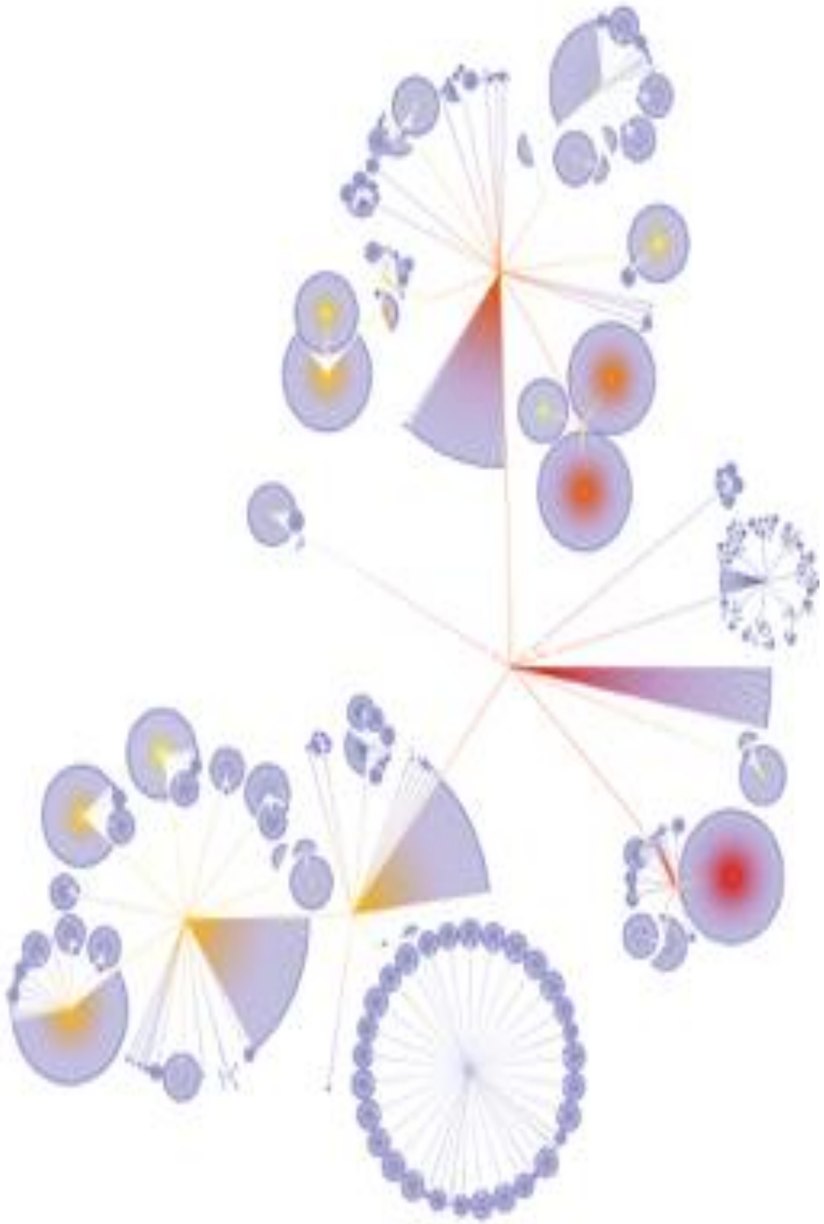


(a) **Triangular vertical**



(b) **Spline radial layout**


(a)


(b)

Node link diagrams

- Networks are also very commonly represented as node–link diagrams, using connection.
- Nodes that are directly connected by a single link are perceived as having the tightest grouping, while nodes with a long path of multiple hops between them are less closely grouped.
- The number of hops within a path - the number of individual links that must be traversed to get from one node to another - is a network-oriented way to measure distances.
- The connection marks support path tracing via these discrete hops.

Node link diagrams

- Node–link diagrams in general are well suited for tasks that involve understanding the network topology: the direct and indirect connections between nodes in terms of the number of hops between them through the set of links.
- Examples of topology tasks include finding all possible paths from one node to another, finding the shortest path between two nodes, finding all the adjacent nodes one hop away from a target node, and finding nodes that act as a bridge between two components of the network that would otherwise be disconnected.

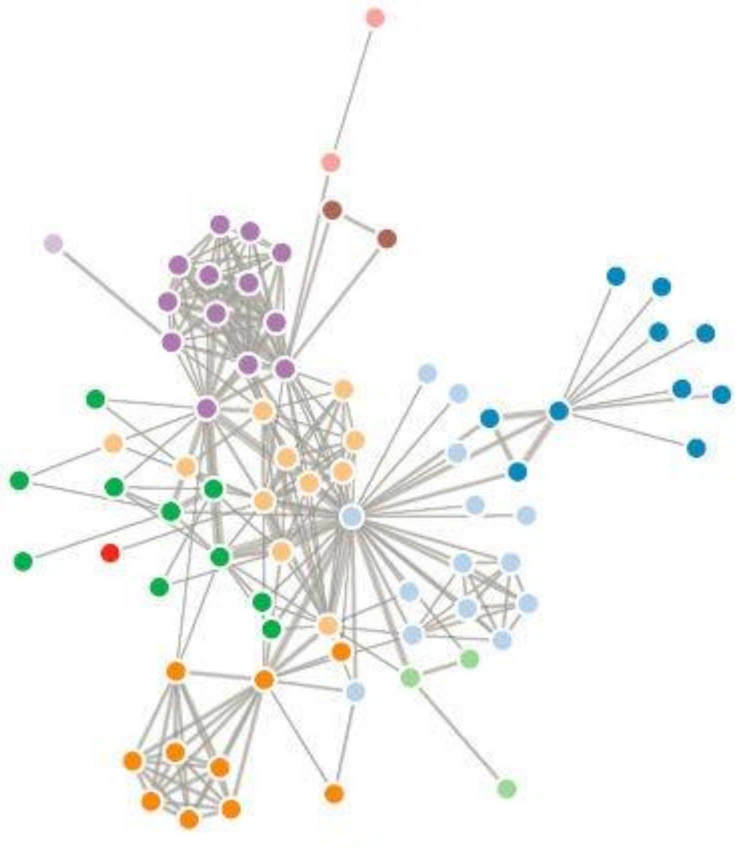
Ex: Force-Directed Placement

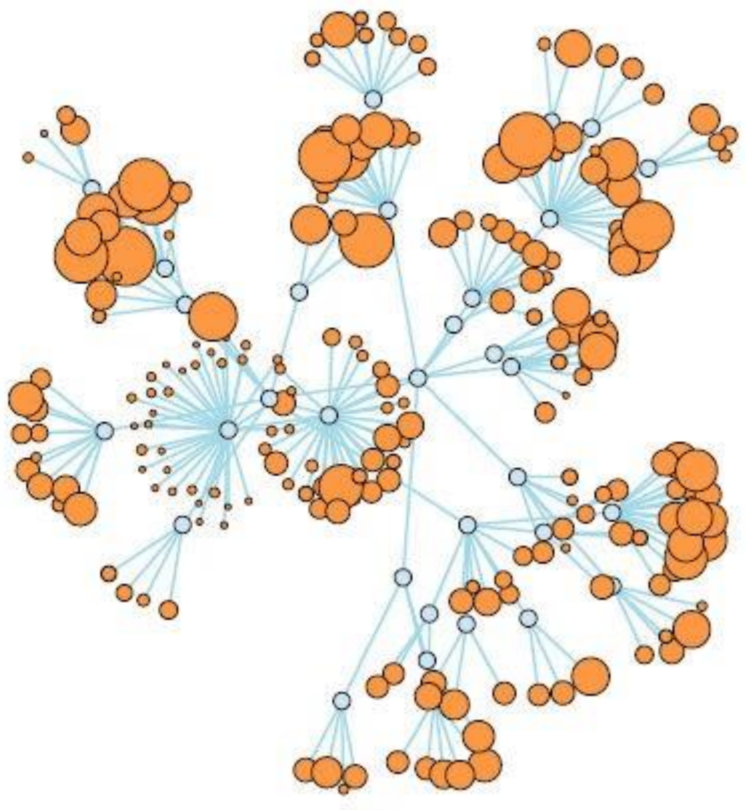
- One of the most widely used idioms for node–link network layout using connection marks is force-directed placement.
- There are many variants in the force-directed placement idiom family; in one variant, the network elements are positioned according to a simulation of physical forces where nodes push away from each other while links act like springs that draw Force-directed placement their endpoint nodes closer to each other.
- Strengths: Very easy to implement. Relatively easy to understand and explain at a conceptual level,

Ex: Force-Directed Placement

- Analyzing the visual encoding created by force-directed placement is somewhat subtle.
- Spatial position does not directly encode any attributes of either nodes or links; the placement algorithm uses it indirectly.
- A tightly interconnected group of nodes with many links between them will often tend to form a visual clump, so spatial proximity does indicate grouping through a strong perceptual cue.
- However, some visual clumps may simply be artifacts: nodes that have been pushed near each other because they were repelled from elsewhere, not because they are closely connected in the network. Thus, proximity is sometimes meaningful but sometimes arbitrary; this ambiguity can mislead the user.

Ex: Force-Directed Placement





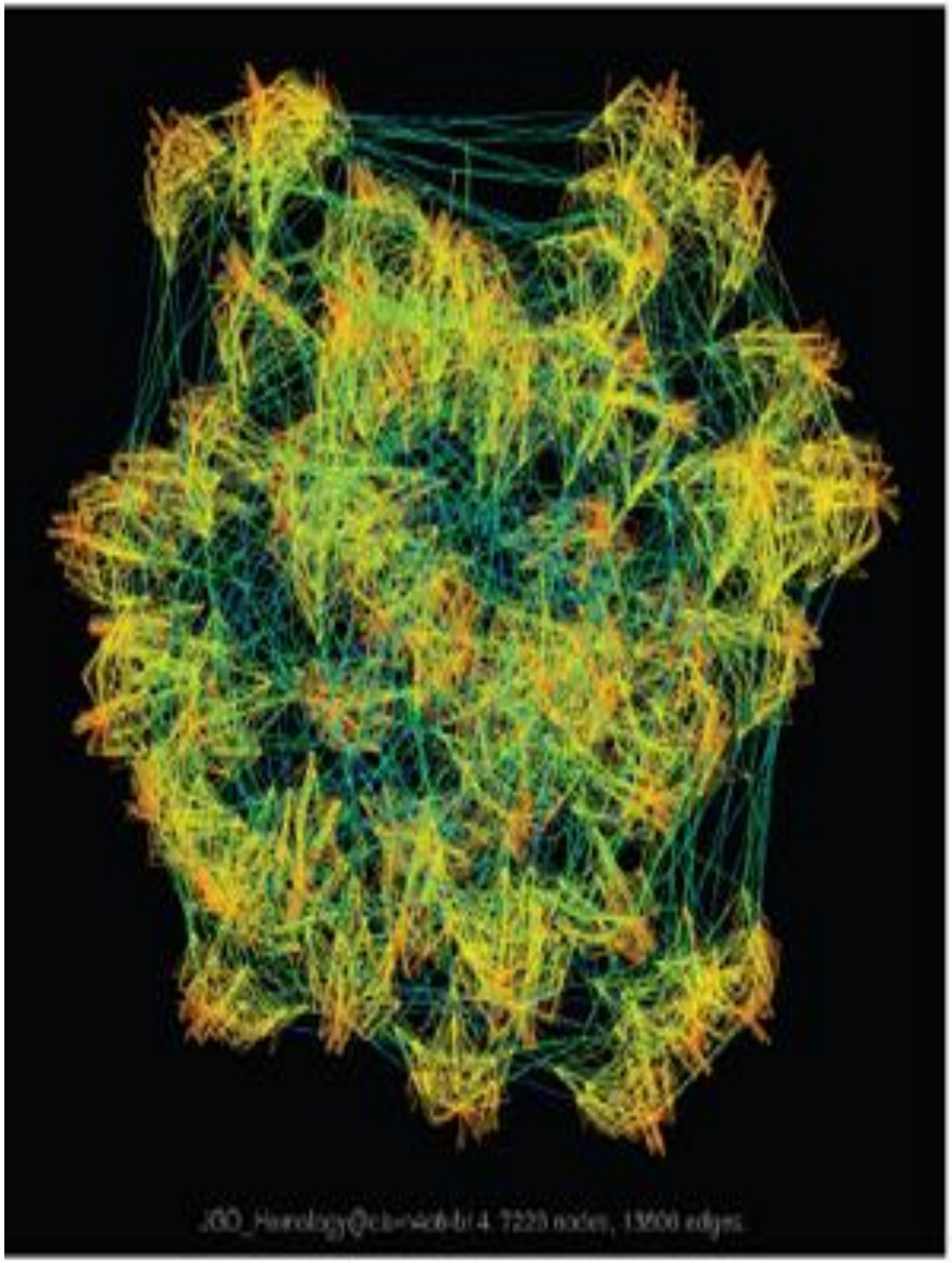
(a) with size coding for link attributes. (b) with size coding for node attributes.

Ex: Force-Directed Placement

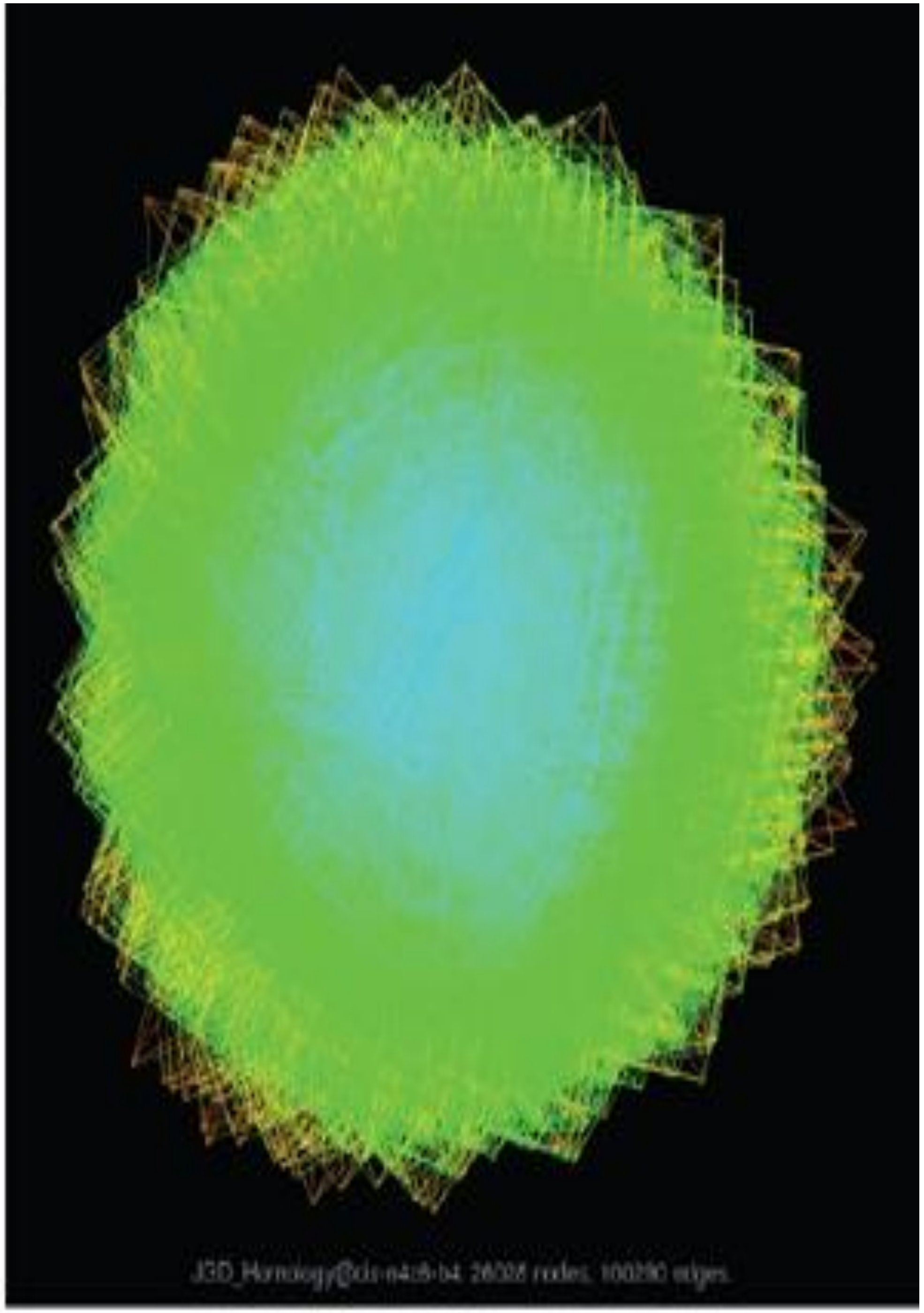
- Weekness

One weakness of force-directed placement is that the layouts are often nondeterministic, meaning that they will look different each time the algorithm is run, rather than deterministic approaches such as a scatterplot or a bar chart that yield an identical layout each time for a specific dataset.

A major weakness of force-directed placement is scalability, both in terms of the visual complexity of the layout and the time required to compute it.


JGD_Homology@cis-n4c6-b4. 7220 nodes, 13600 edges.

(a)

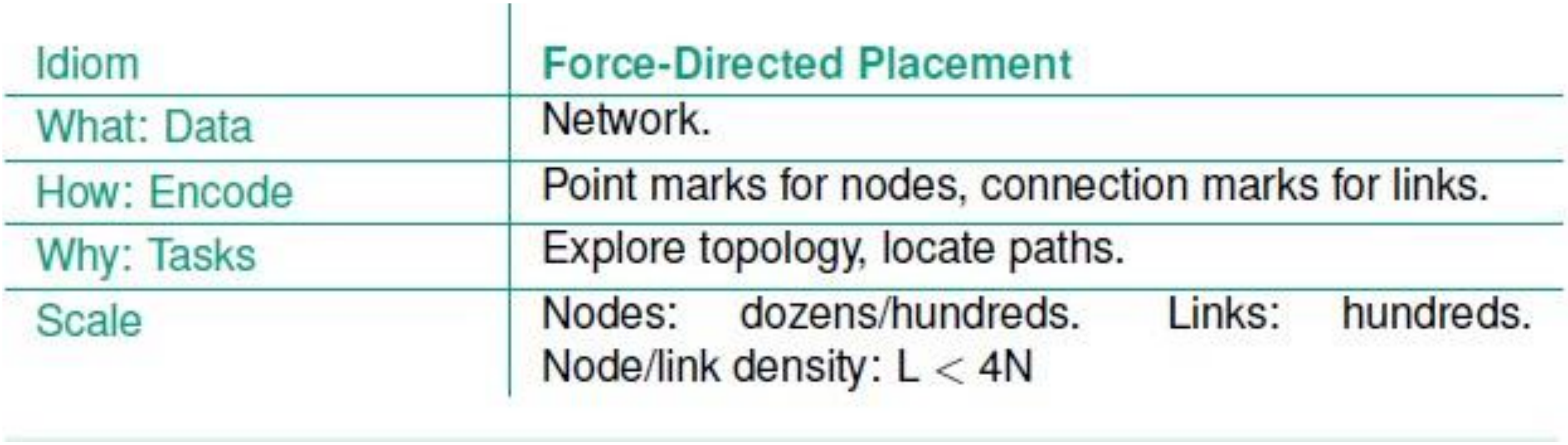

JGD_Homology@cis-n4c6-b4. 26028 nodes, 100290 edges.

(b)

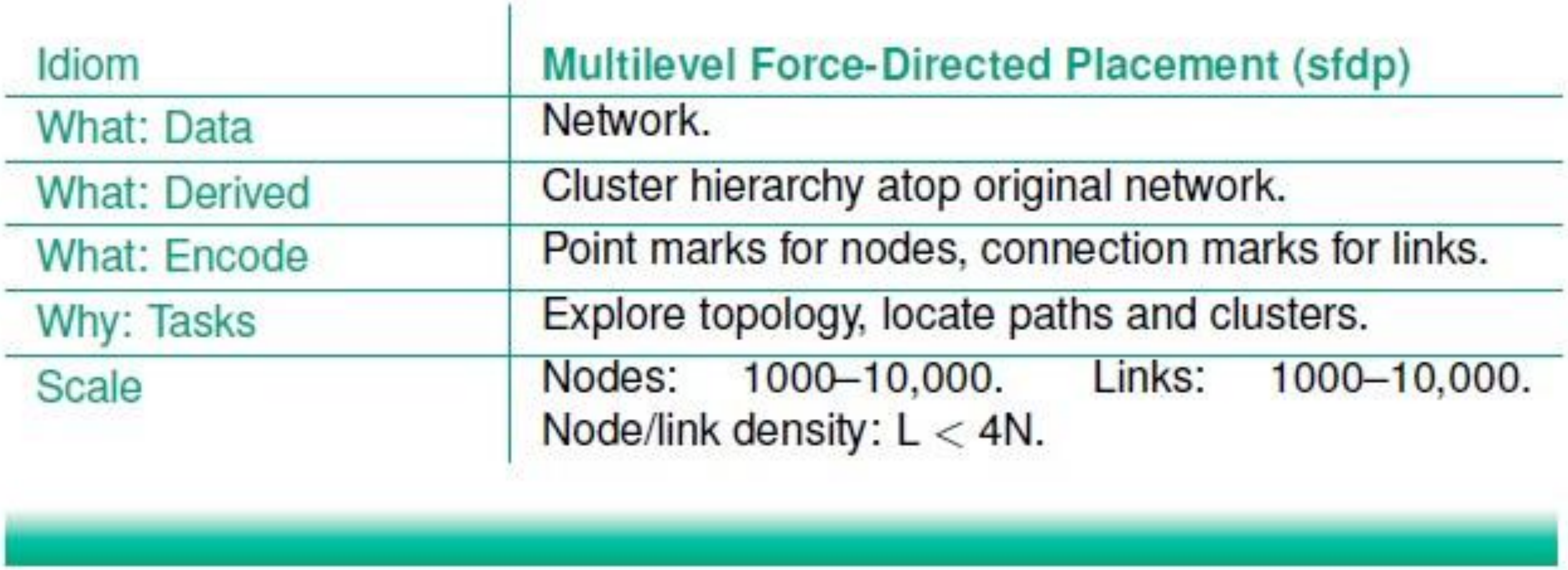
Multilevel network idioms

- Many recent approaches to scalable network drawing are multilevel network idioms, where the original network is augmented with a derived cluster hierarchy to form a compound network.
- The Cluster hierarchy is computed by coarsening the original network into successively simpler networks that nevertheless attempt to capture the most essential aspects of the original's structure.

Multilevel network idioms



Idiom	Force-Directed Placement
What: Data	Network.
How: Encode	Point marks for nodes, connection marks for links.
Why: Tasks	Explore topology, locate paths.
Scale	Nodes: dozens/hundreds. Links: hundreds. Node/link density: $L < 4N$

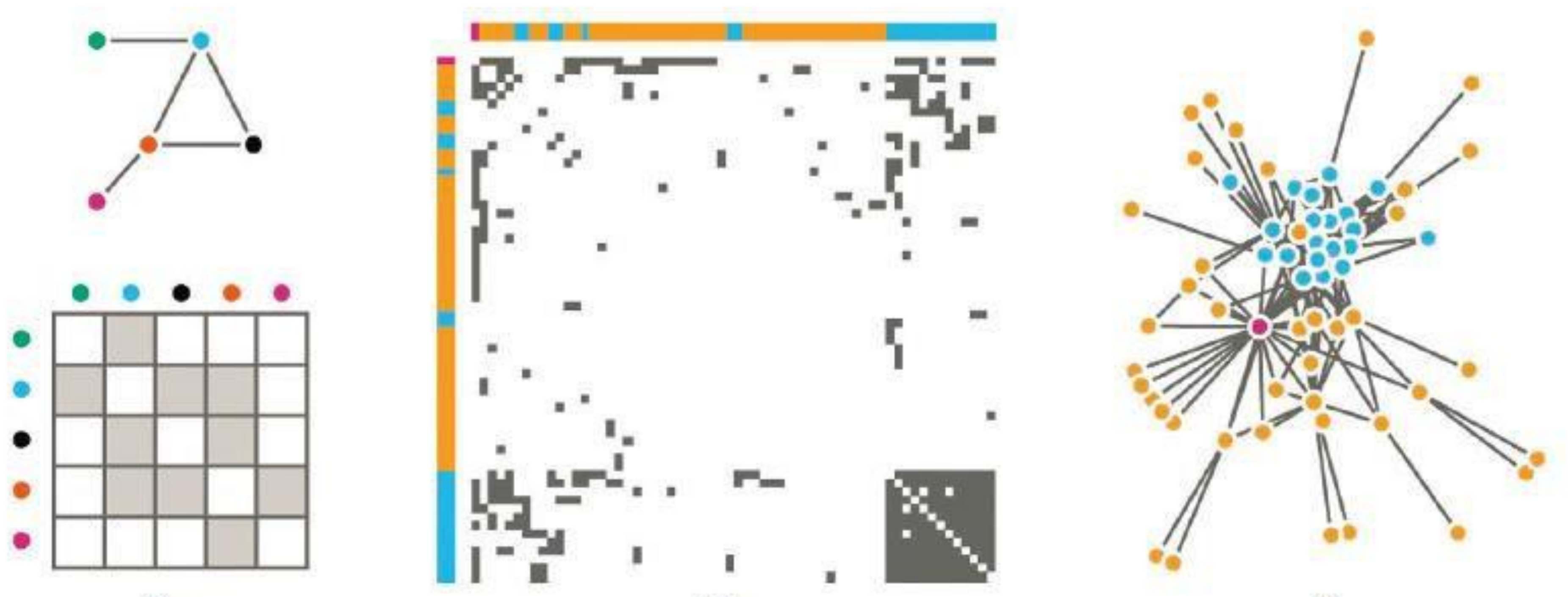


Idiom	Multilevel Force-Directed Placement (sfdp)
What: Data	Network.
What: Derived	Cluster hierarchy atop original network.
What: Encode	Point marks for nodes, connection marks for links.
Why: Tasks	Explore topology, locate paths and clusters.
Scale	Nodes: 1000–10,000. Links: 1000–10,000. Node/link density: $L < 4N$.

Matrix Views

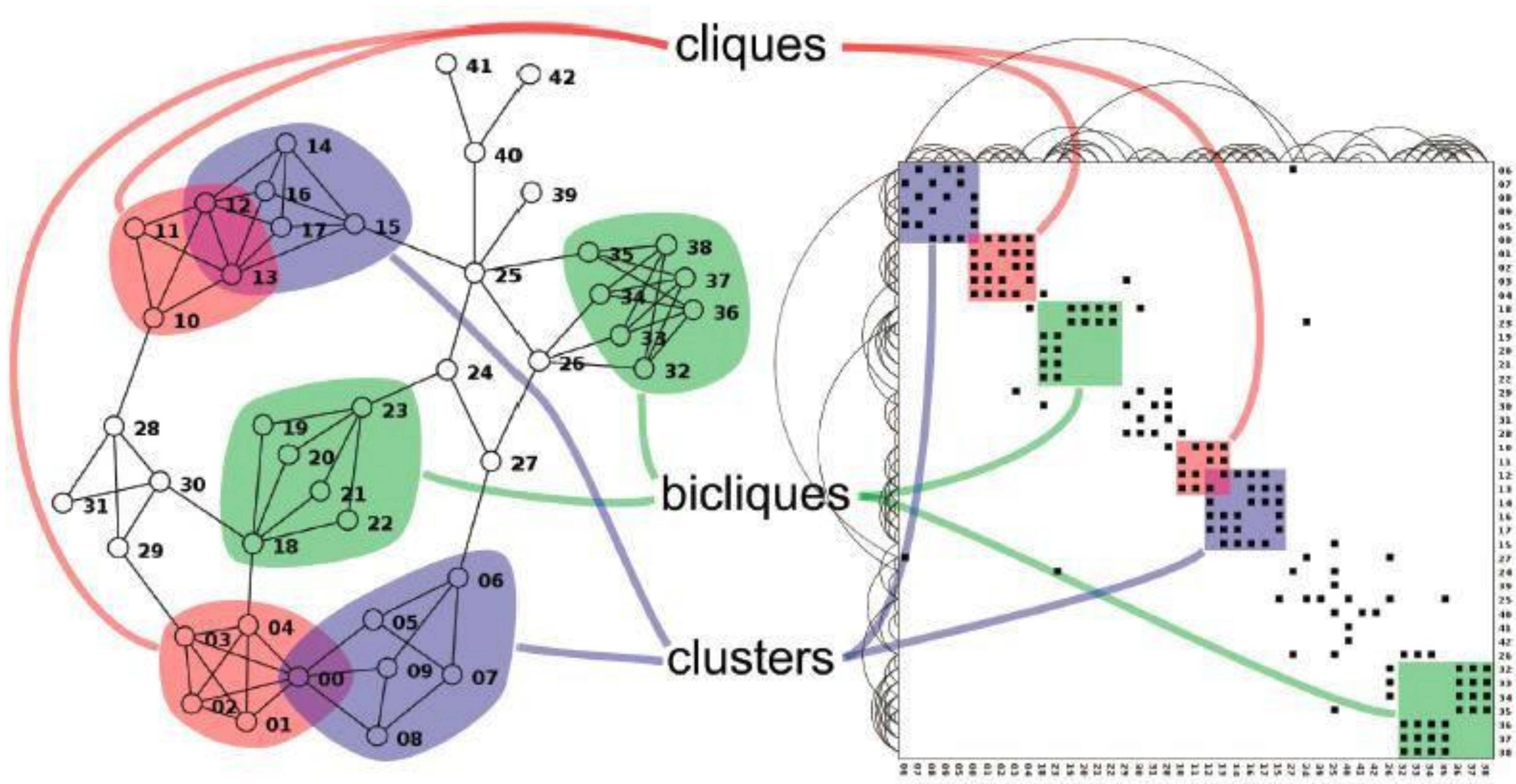
- Network data can also be encoded with a matrix view by deriving a table from the original network data.
- **Example: Adjacency Matrix View**
- where all of the nodes in the network are laid out along the vertical and horizontal edges of a square region and links between two nodes are indicated by coloring an area mark in the cell in the matrix that is the intersection between their row and column.
- That is, the network is transformed into the derived dataset of a table with two key attributes that are separate full lists of every node in the network, and one value attribute for each cell records whether a link exists between the nodes that index the cell.

Example: Adjacency Matrix View


(a) (b) (c)

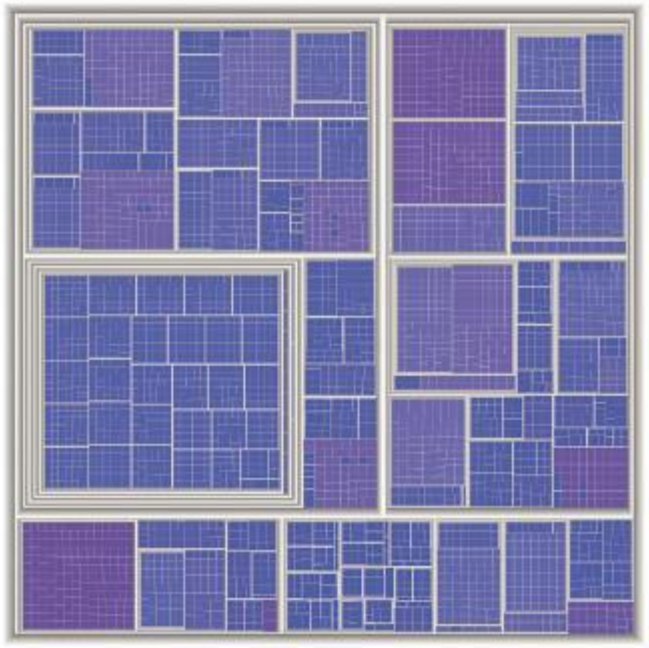
Matrix views of networks can achieve very high information density, up to a limit of one thousand nodes and one million edges, just like cluster heatmaps and all other matrix views that use small area marks.

Costs and Benefits: Connection versus Matrix



Containment: Hierarchy Marks

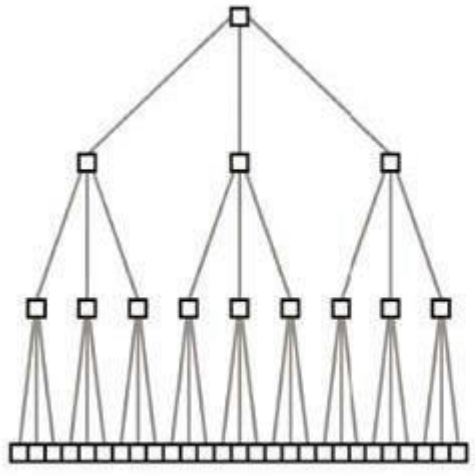
- Containment marks are very effective at showing complete information about hierarchical structure, in contrast to connection marks that only show pairwise relationships between two items at once.
- Tree Maps: The idiom of treemaps is an alternative to node–link tree drawings, where the hierarchical relationships are shown with containment rather than connection.



Seven different visual encoding idioms for tree data

Using different combinations of visual channels.

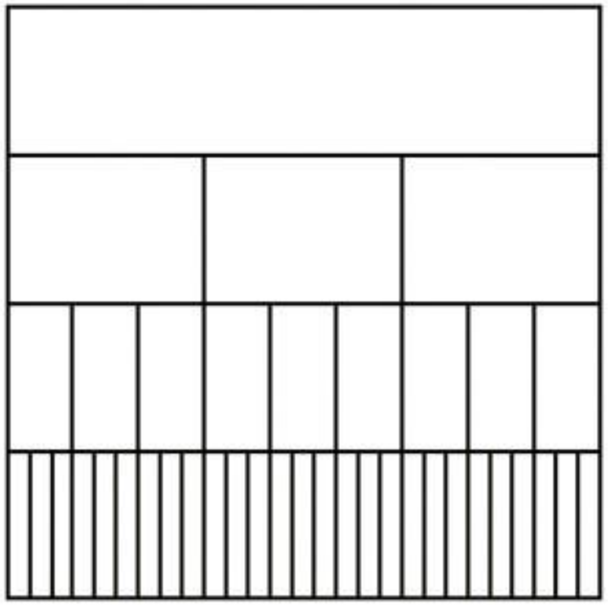
(a) Rectilinear vertical node–link, using connection to show link relationships, with vertical spatial position showing tree depth and horizontal spatial position showing sibling order.



Seven different visual encoding idioms for tree data

Using different combinations of visual channels.

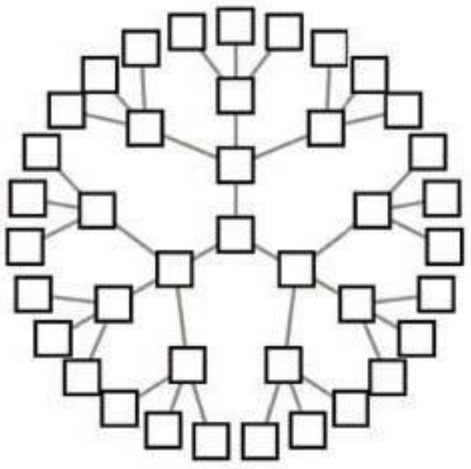
- (b) Icicle, with vertical spatial position and size showing tree depth, and horizontal spatial position showing link relationships and sibling order.



Seven different visual encoding idioms for tree data

Using different combinations of visual channels.

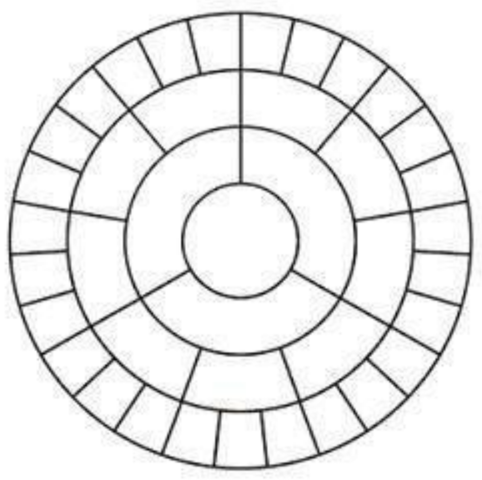
- (c) Radial node–link, using connection to show link relationships, with radial depth spatial position showing tree depth and radial angular position showing sibling order.



Seven different visual encoding idioms for tree data

Using different combinations of visual channels.

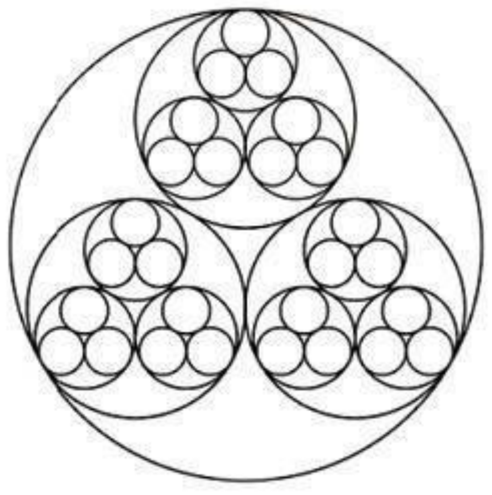
- (d) Concentric circles, with radial depth spatial position and size showing tree depth and radial angular spatial position showing link relationships and sibling order.



Seven different visual encoding idioms for tree data

Using different combinations of visual channels.

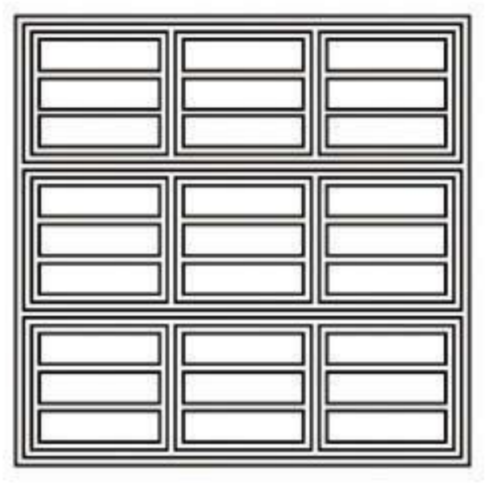
- (e) Nested circles, using radial containment, with nesting level and size showing tree depth.



Seven different visual encoding idioms for tree data

Using different combinations of visual channels.

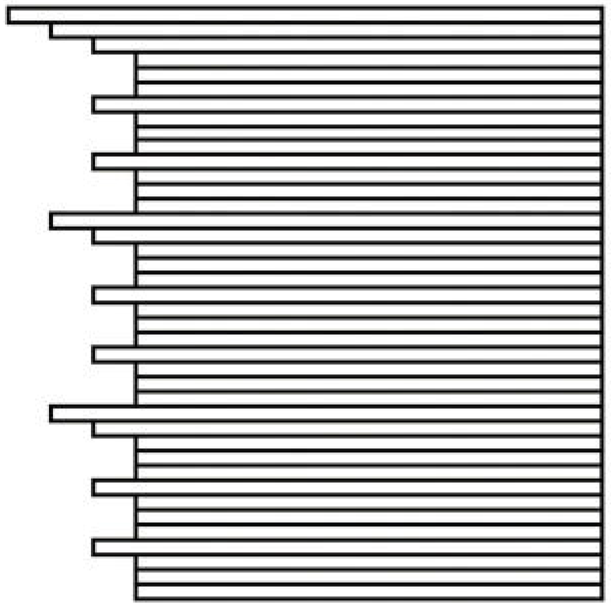
- (f) Treemap, using rectilinear containment, with nesting level and size showing tree depth.



Seven different visual encoding idioms for tree data

Using different combinations of visual channels.

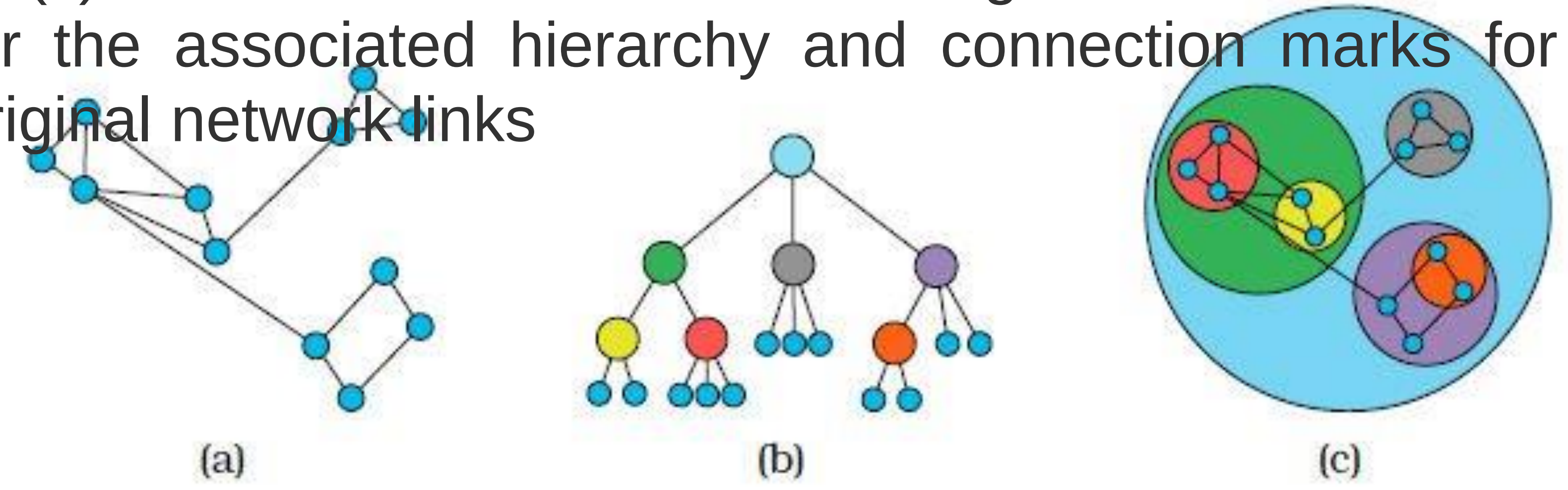
- (g) Indented outline, with horizontal spatial position showing tree depth and link relationships and vertical spatial position showing sibling order.



Compound Network

- A compound network is a combination of a **network** and a **tree** on top of it, where the nodes in the network are the leaves of the tree.
- Thus, the interior nodes of the tree encompass multiple network nodes.

(a)shows a network (b) shows a cluster hierarchy built on top of it. (c) shows a combined view using of containment marks for the associated hierarchy and connection marks for the original network links



Displaying Hierarchical Structures

Two classes of algorithms

- space-filling
- non-space-filling

Displaying Hierarchical Structures

Space-Filling Methods

- space-filling techniques make maximal use of the display space
- This is accomplished by using juxtapositioning to imply relations, as opposed to, for example, conveying relations with edges joining data objects.
- The two most common approaches to generating space-filling hierarchies
  - **Rectangular layout**
  - **Radial layouts**

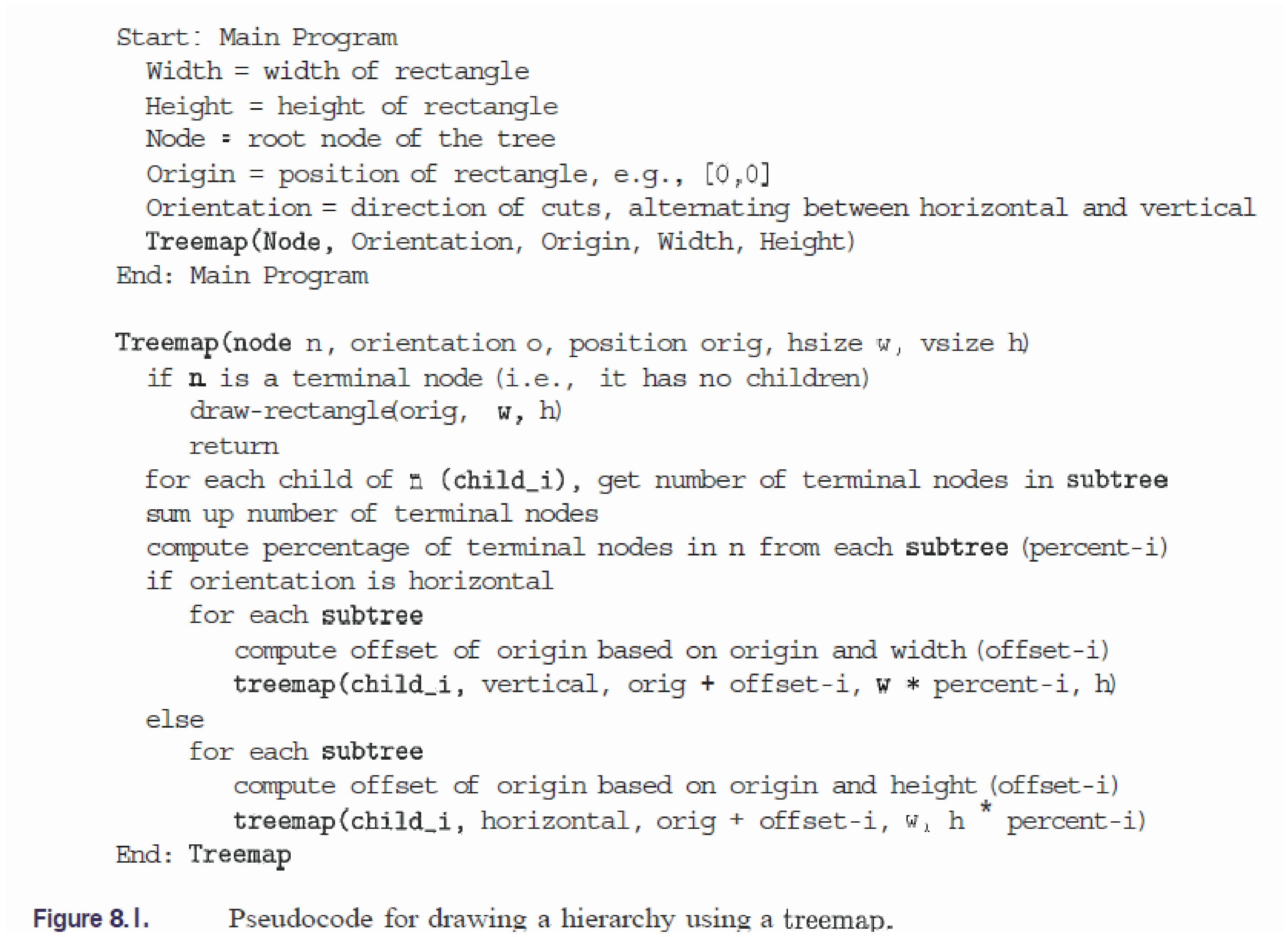
Displaying Hierarchical Structures

Space-Filling Methods

Rectangular layout

- a rectangle is recursively divided into slices, alternating horizontal and vertical slicing, based on the populations of the subtrees at a given level

Rectangular layout



```
Start: Main Program
  Width = width of rectangle
  Height = height of rectangle
  Node = root node of the tree
  Origin = position of rectangle, e.g., [0,0]
  Orientation = direction of cuts, alternating between horizontal and vertical
  Treemap(Node, Orientation, Origin, Width, Height)
End: Main Program

Treemap(node n, orientation o, position orig, hsize w, vsize h)
  if n is a terminal node (i.e., it has no children)
     draw-rectangle(orig, w, h)
     return
  for each child of n (child_i), get number of terminal nodes in subtree
  sum up number of terminal nodes
  compute percentage of terminal nodes in n from each subtree (percent-i)
  if orientation is horizontal
     for each subtree
        compute offset of origin based on origin and width (offset-i)
        treemap(child_i, vertical, orig + offset-i, w * percent-i, h)
  else
     for each subtree
        compute offset of origin based on origin and height (offset-i)
        treemap(child_i, horizontal, orig + offset-i, w, h * percent-i)
End: Treemap
```

Figure 8.1. Pseudocode for drawing a hierarchy using a treemap.

Rectangular layout

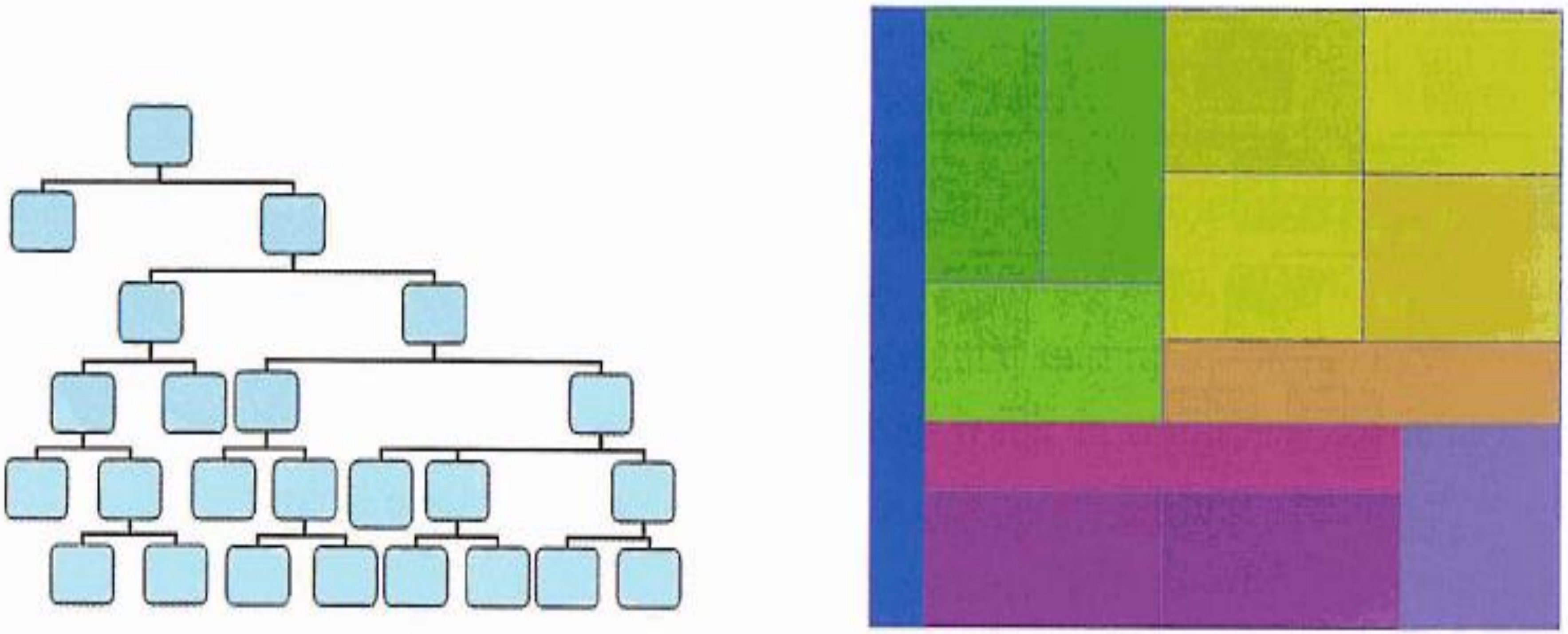
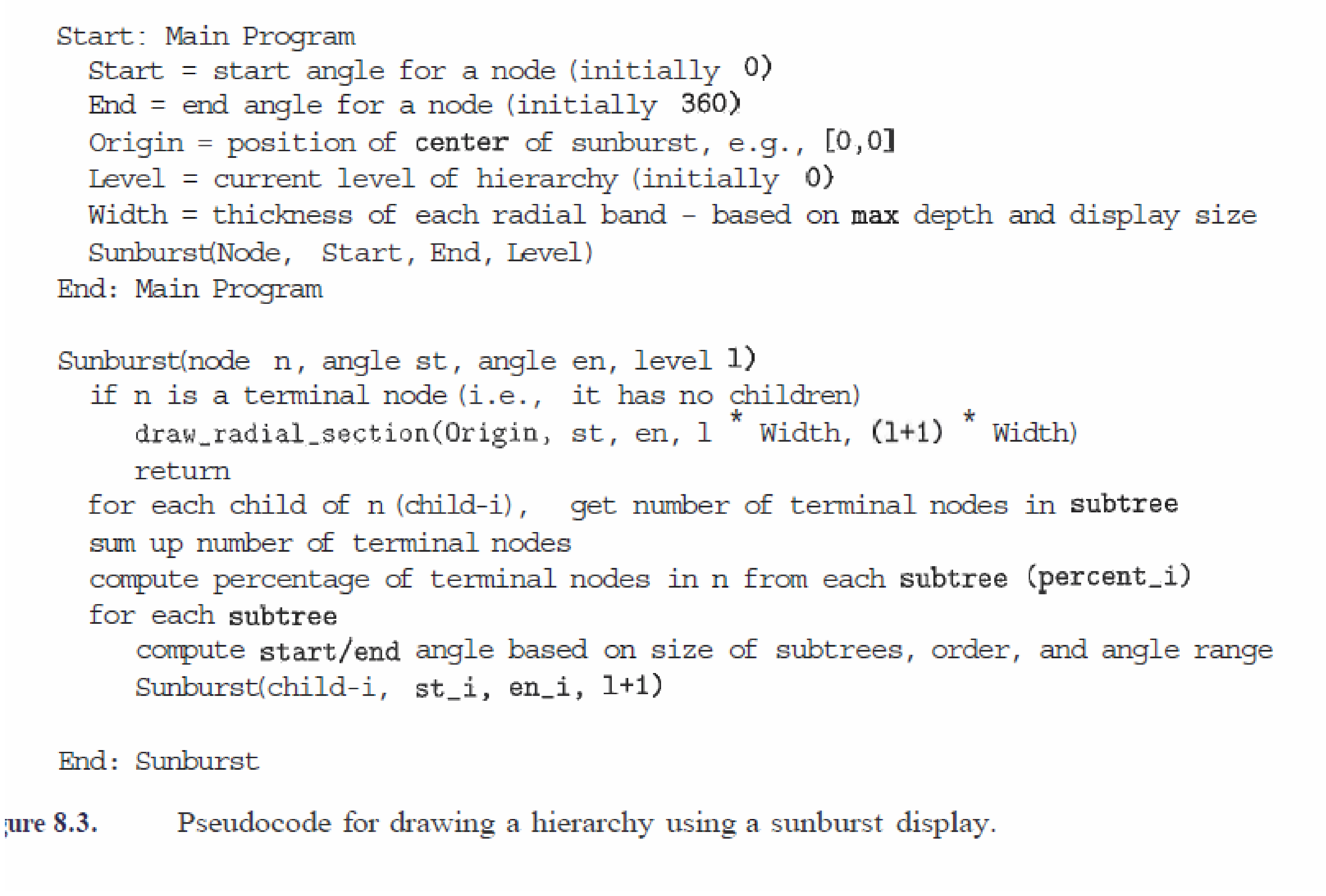


Figure. **A sample hierarchy and the corresponding treemap display.**

Radial layouts

- Radial space-filling hierarchy visualizations, sometimes referred to as sunburst displays , have the root of the hierarchy in the center of the display and use nested rings to convey the layers of the hierarchy.
- Each ring is divided based on the number of nodes at that level
- These techniques follow a similar strategy to treemaps,in that the number of terminal nodes in a subtree determines the amount of screen space that will be allocated for it.

Radial layouts



```
Start: Main Program
  Start = start angle for a node (initially 0)
  End = end angle for a node (initially 360)
  Origin = position of center of sunburst, e.g., [0,0]
  Level = current level of hierarchy (initially 0)
  Width = thickness of each radial band - based on max depth and display size
  Sunburst(Node, Start, End, Level)
End: Main Program

Sunburst(node n, angle st, angle en, level l)
  if n is a terminal node (i.e., it has no children)
     draw_radial_section(Origin, st, en, l * Width, (l+1) * Width)
     return
  for each child of n (child-i), get number of terminal nodes in subtree
  sum up number of terminal nodes
  compute percentage of terminal nodes in n from each subtree (percent_i)
  for each subtree
     compute start/end angle based on size of subtrees, order, and angle range
     Sunburst(child-i, st_i, en_i, l+1)

End: Sunburst
```

ure 8.3. Pseudocode for drawing a hierarchy using a sunburst display.

Radial layouts

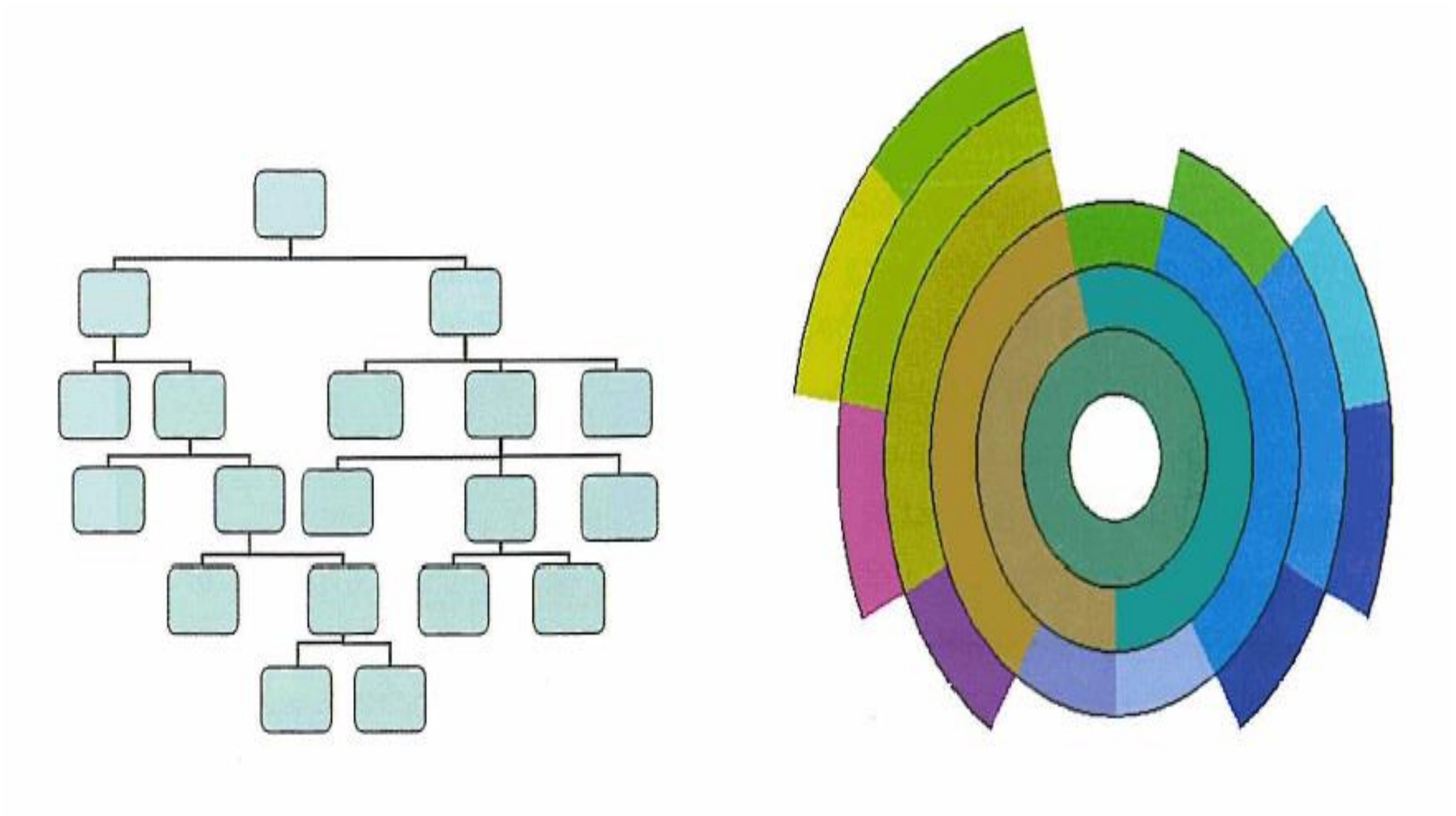


Figure. **A sample hierarchy and the corresponding sunburst display.**

Displaying Hierarchical Structures

Non-Space-Filling Methods

When designing an algorithm for drawing any node-link diagram (not just trees), one must consider three categories of often-contradictory guidelines: drawing

- Conventions: include restricting edges to be either a single straight line, a series of rectilinear lines, polygonal lines, or curves
- Constraints: include requiring a particular node to be at the center of the display, or that a group of nodes be located close to each other, or that certain links must either go from top to bottom or left to right. Each of the above guidelines can be used to drive the algorithm design.
  - Aesthetics:

Displaying Hierarchical Structures

Non-Space-Filling Methods

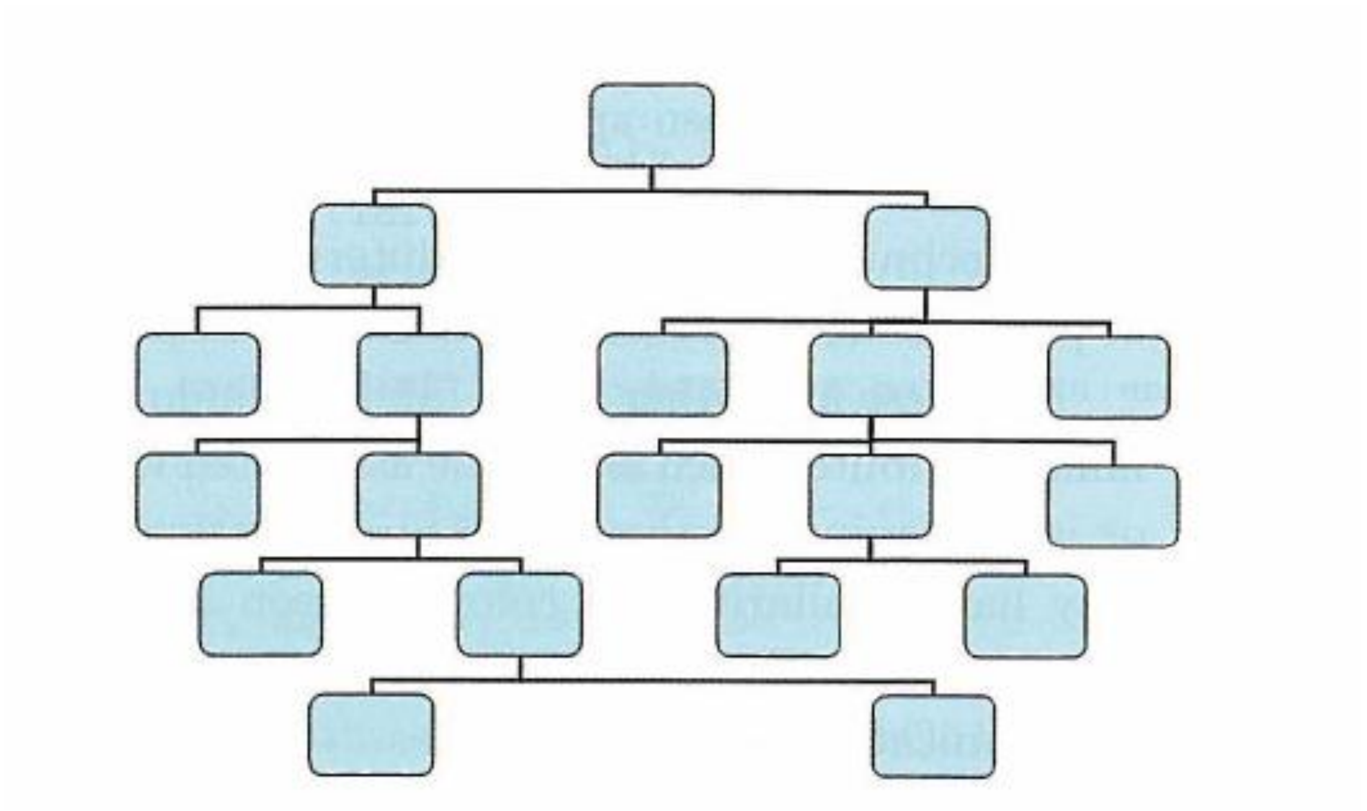
Aesthetics: often have significant impact on the interpretability of a tree or graph drawing, yet often result in conflicting guidelines. Some typical aesthetic rules include:

- minimize line crossings
- maintain a pleasing aspect ratio
- minimize the total area of the drawing
- minimize the total length of the edges
- minimize the number of bends in the edges
- minimize the number of distinct angles or curvatures used
- strive for a symmetric structure

Displaying Hierarchical Structures

Non-Space-Filling Methods

For trees, especially balanced ones, it is relatively easy to design algorithms that adhere to many, if not most, of these guidelines. For example, a simple tree drawing procedure is given below (sample output is shown in Figure:



Displaying Hierarchical Structures

Non-Space-Filling Methods

1. Slice the drawing area into equal-height slabs, based on the depth of the tree.
2. For each level of the tree, determine how many nodes need to be drawn.
3. Divide each slice into equal-sized rectangles based on the number of nodes at that level.
4. Draw each node in the center of its corresponding rectangle.
5. Draw a link between the center-bottom of each node to the center-top of its child node(s).

Displaying Hierarchical Structures

Non-Space-Filling Methods

Many enhancements can be made to this rather basic algorithm in order to improve space utilization and move child nodes closer to their parents.

Some of these include:

- Rather than using even spacing and centering, divide each level based on the number of terminal nodes belonging to each subtree.

- Spread terminal nodes evenly across the drawing area and center parent nodes above them.

Displaying Hierarchical Structures

Non-Space-Filling Methods

- Add some buffer space between adjacent nonsibling nodes to emphasize relationships.
- If possible, reorder the subtrees of a node to achieve more symmetry

  and balance.
- Position the root node in the center of the display and lay out child nodes radially, rather than vertically.

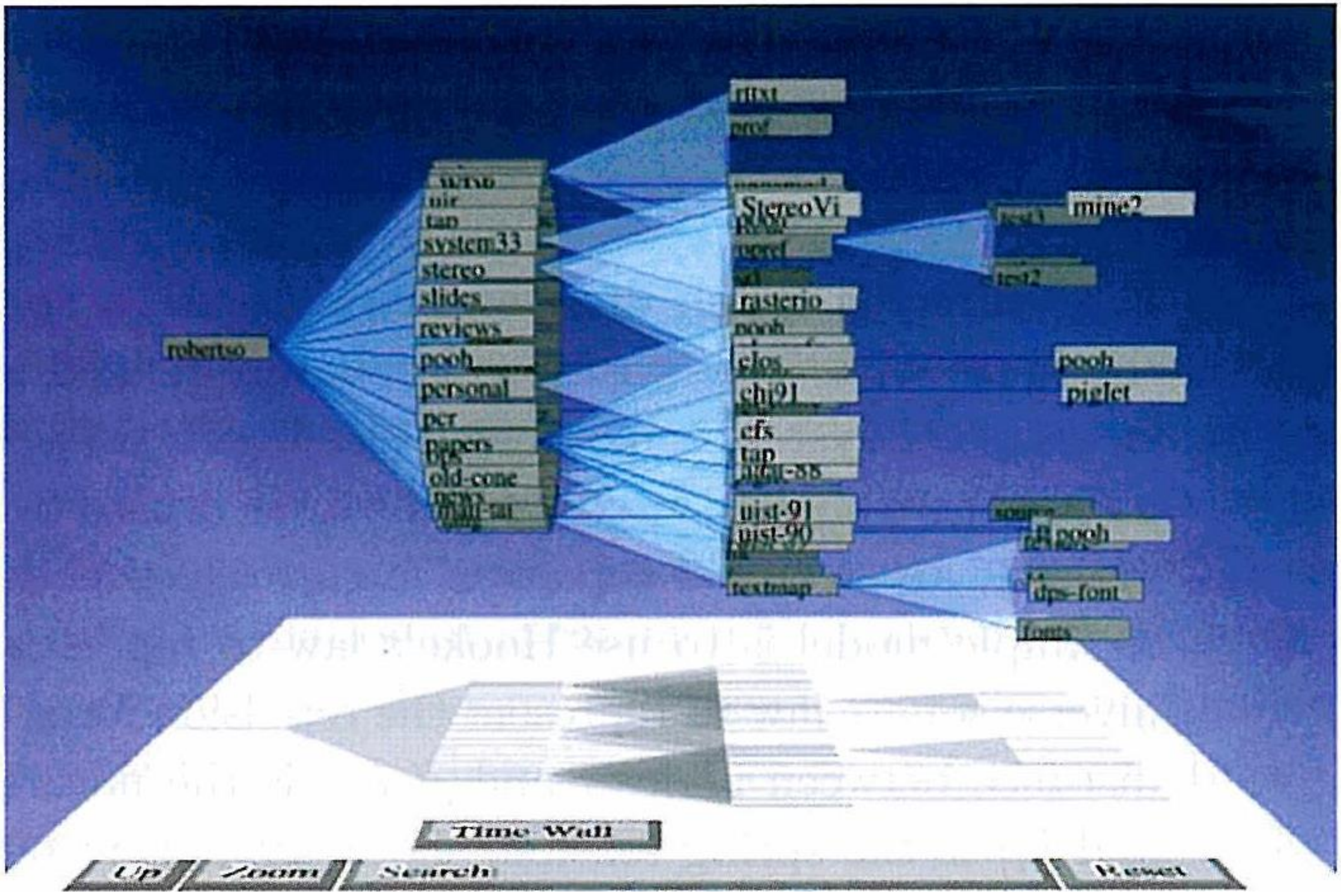
Displaying Hierarchical Structures

Non-Space-Filling Methods

- For large trees, a popular approach is to use the third dimension, supplemented with tools for rotation, translation, and zooming. Perhaps the most well-known of such techniques is called a **cone *tree***
- In this layout, the children of a node are arranged radially at evenly spaced angles and the offset perpendicular to the plane.
- The two parameters critical to this process are the radius and offset distance; varying these influences the density of the display and the level of occlusion.
- Minimally they should be set so that separate branches of the tree do not fall into the same section of 3D space.
- One method to ensure this is to have the radius inversely proportional to the depth of a node in the tree.
- In this manner, nodes close to the root are significantly separated, and those near the bottom of the tree are closer together

Displaying Hierarchical Structures

Non-Space-Filling Methods


robertso
system33
stereo
slides
reviews
pooh
personal
pcr
papers
old-cone
StereoVi
rasterio
pooh
elos
chi91
efs
tap
uist-91
uist-90
textmap
mine2
test1
test2
pooh
piglet
pooh
dps-font
fonts
Time Wall
Up
Zoom
Search
Reset