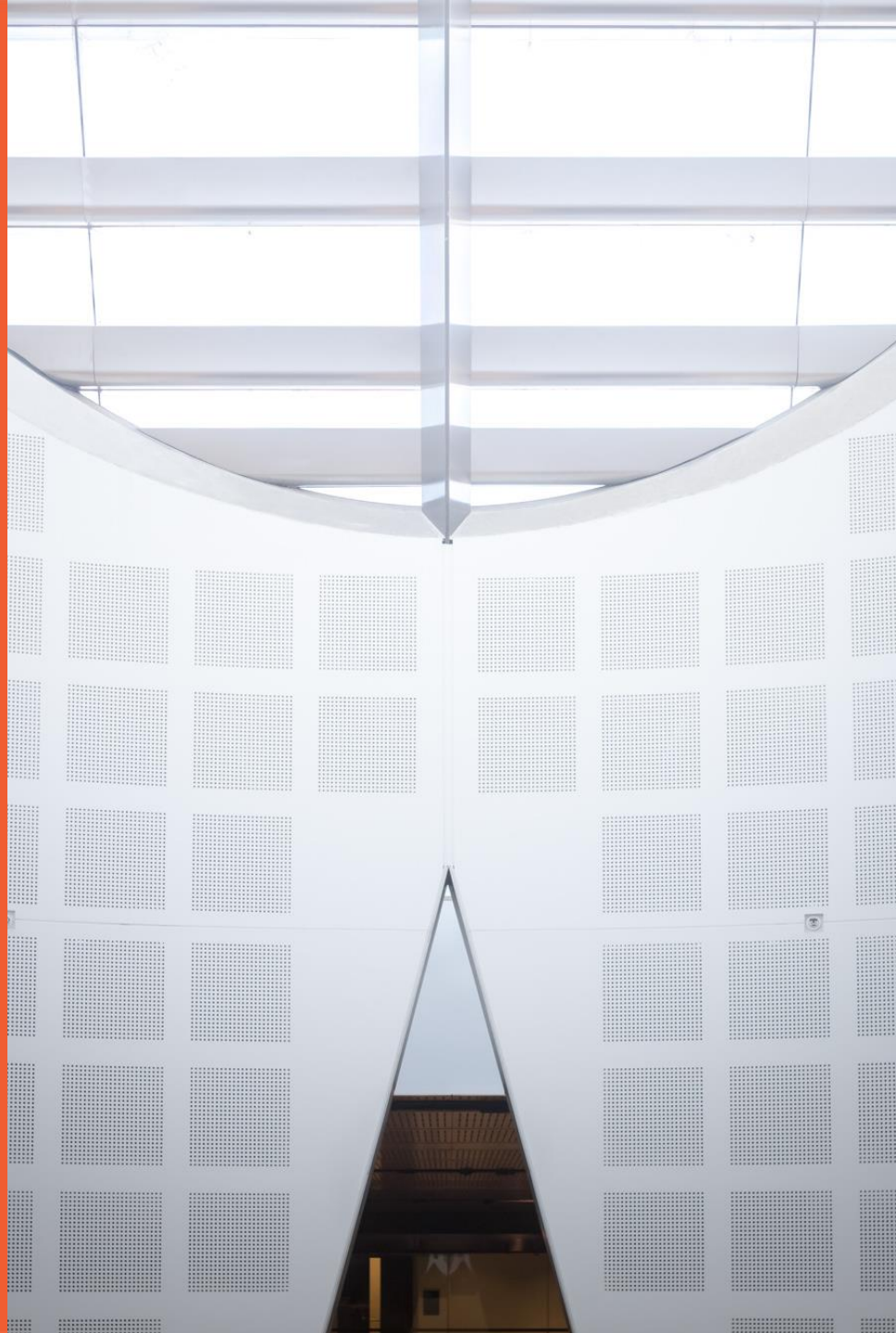
COMP5048
Visual Analytics

Week 2: Complex Data Visualisation

Professor Seokhee Hong
School of Information Technologies

THE UNIVERSITY OF SYDNEY



Copyright warning

Data Types

Table data:

Multi-dimensional/Muti-variate/Multi-attribute Data:

Types of Attributes:

-> Nominal value, categorical value, text, image, time stamps, spatial information etc

eg. Student data =(Degree, Year, Academic records, International/local, Address, Phone number, email)

Graph data:

G=(V, E), V: vertex (node) set , E: edge set

Social network, Twitter, facebook, linked-in etc

Complex Data Visualisation

1. Multi-dimensional/Multi-variate Data
   - Multiple attributes: ordinal, nominal, categorical, image etc
   - (eg) matrix with many columns/rows

2. Spatial Data
   - Data with geometry (map, longitude/latitude)

3. Temporal/Dynamic Data
   - Data with time stamps: changing over time

4. Relational Data with Constraints (Week 2-4)
   - Relations, Hierarchy, Clusters, Directions
   - *Tree* (Hierarchical Relational Data): Week 2

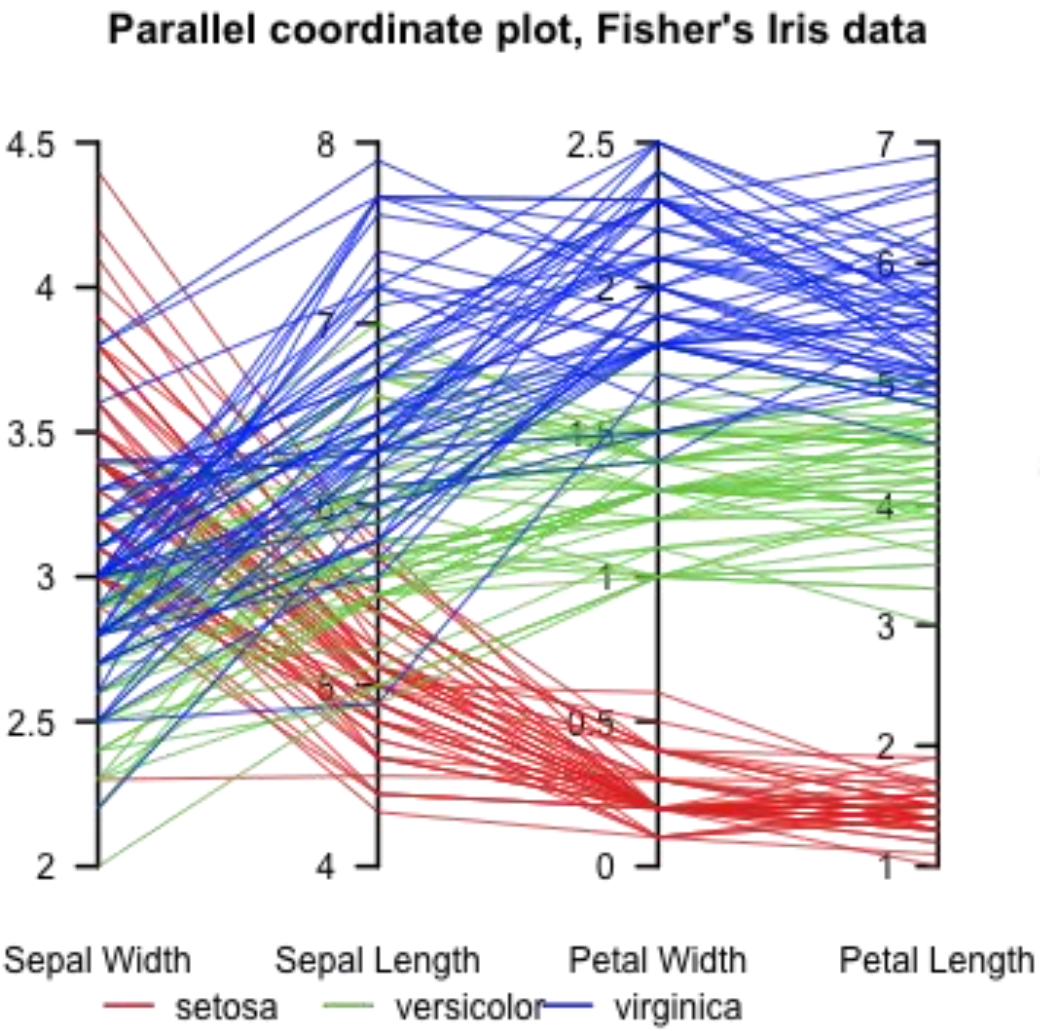
(1) Multi-dimensional/Multi-variate Data Visualisation

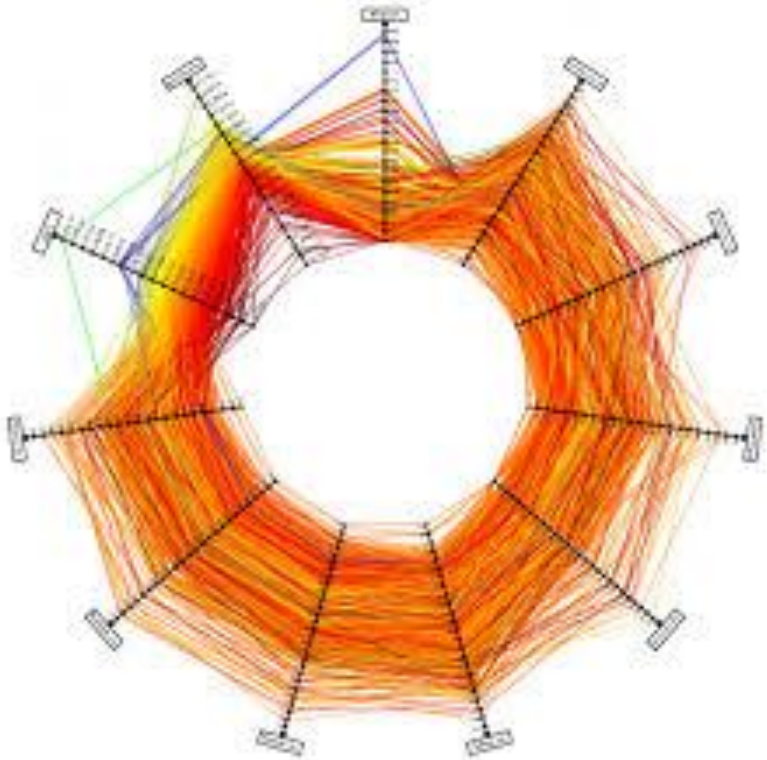
1. Parallrel Coordinates

2. MDS (Multi-Dimensional Scaling)

3. Glyph (Visual variables)

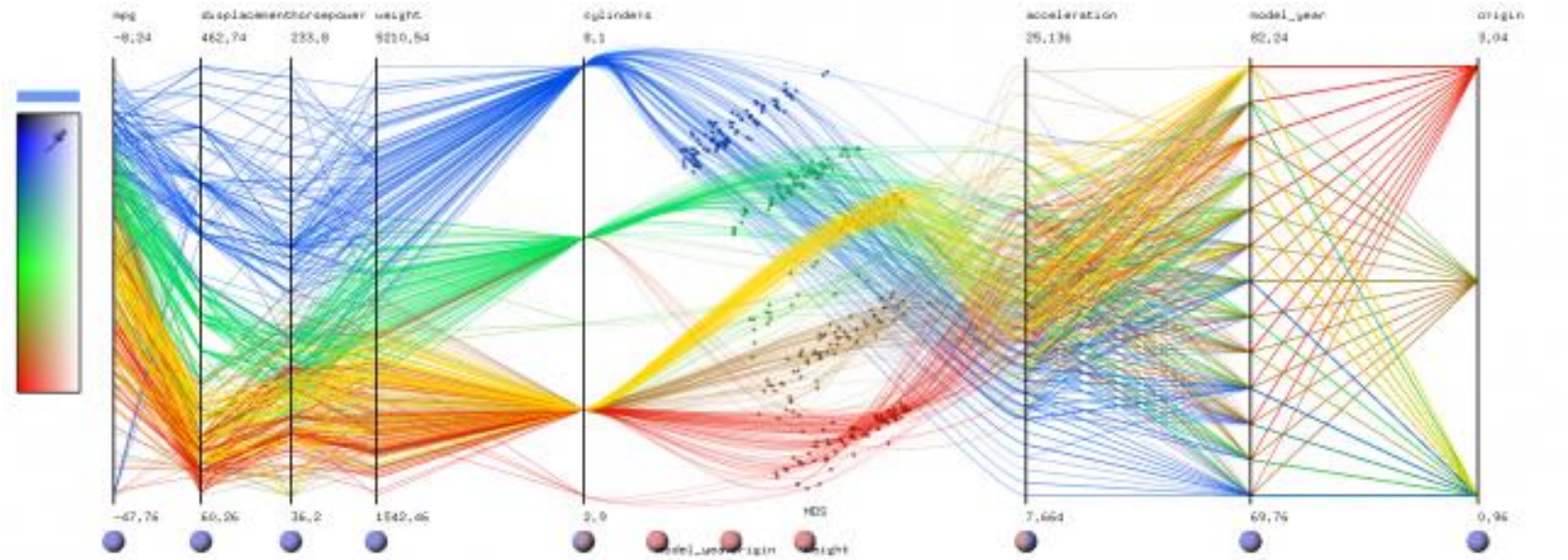
1. Parallel Coordinates [Inselburg]

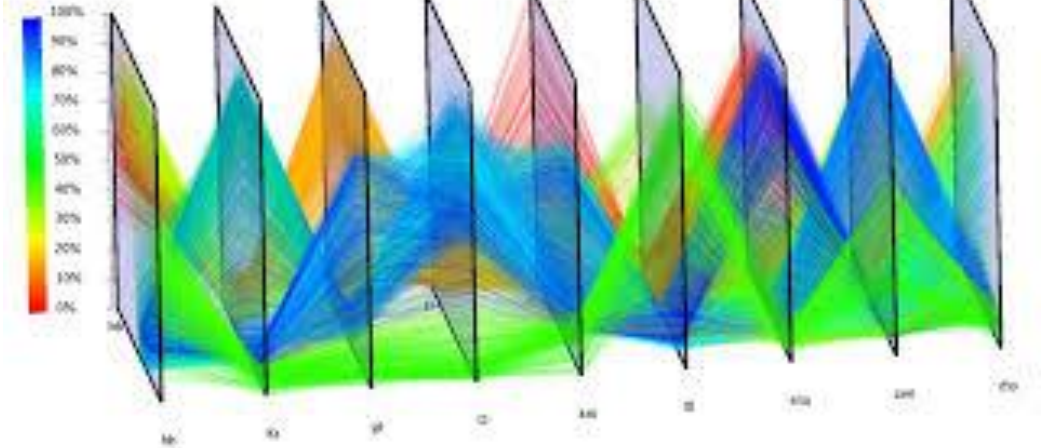


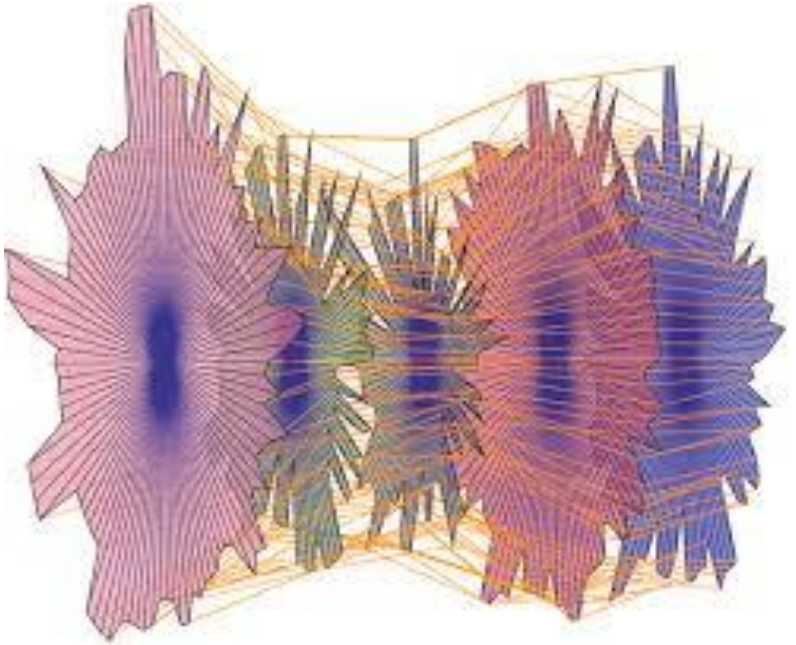


- Nominal values
- Change ordering to see patterns

Variations

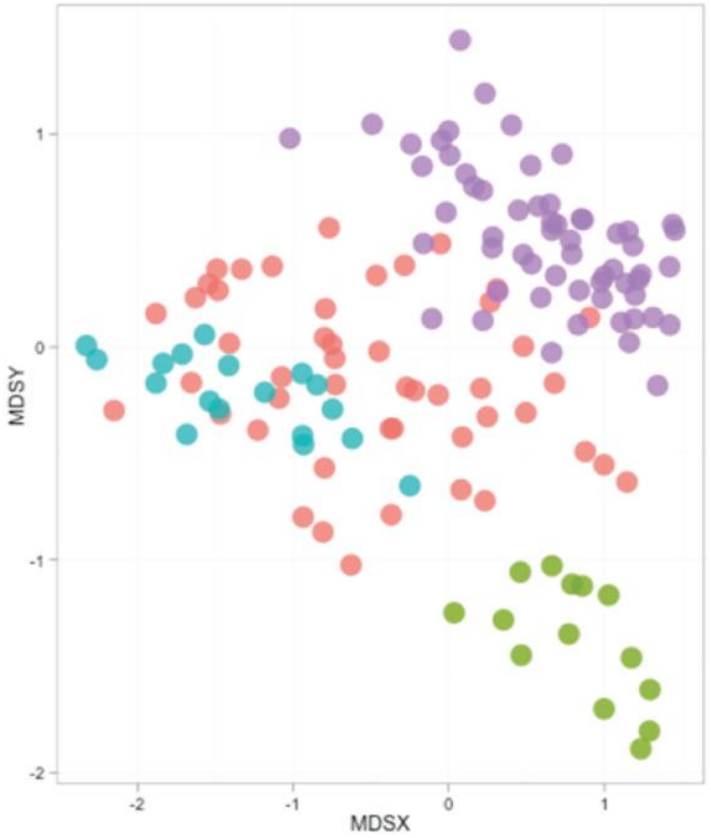

mpg
displacement
horsepower
weight
cylinders
acceleration
model_year
origin

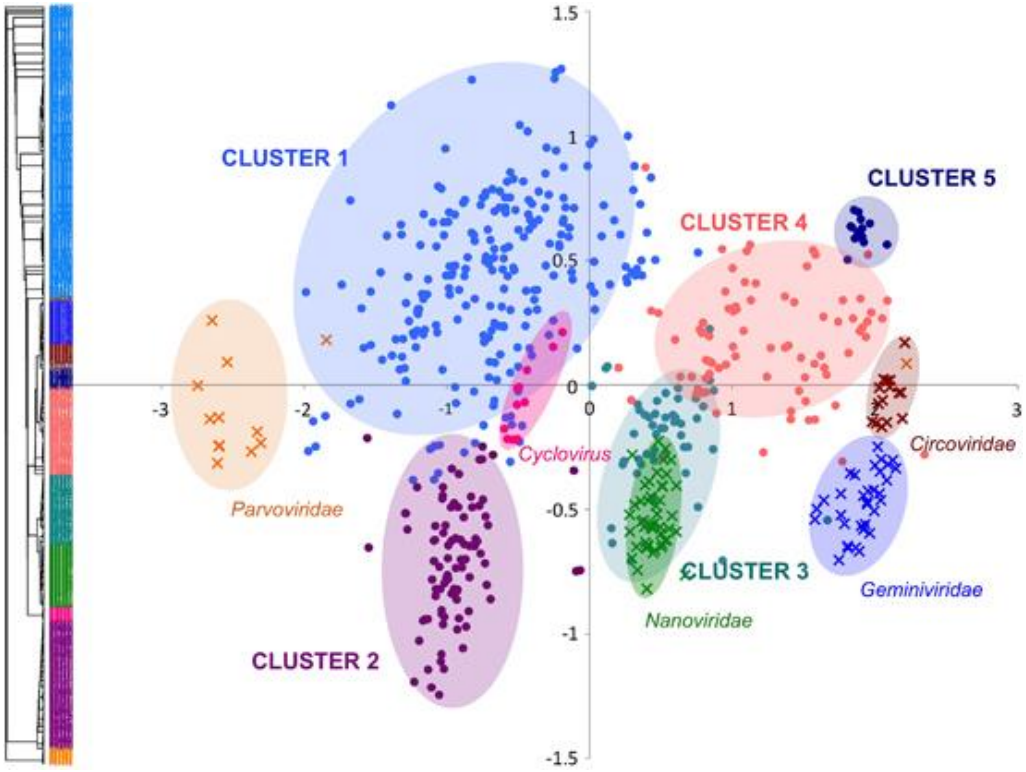




2. MDS (Multi-Dimensional Scaling)

- Compute similarity between the data using distance matrix (n dimension)
- (e.g) **PCA** (Principal Component Analysis)
  - Eigenspace decomposition using largest eigenvalues
  - Project into 2 or 3 dimensions
- Visualisation: 2D scatter plot



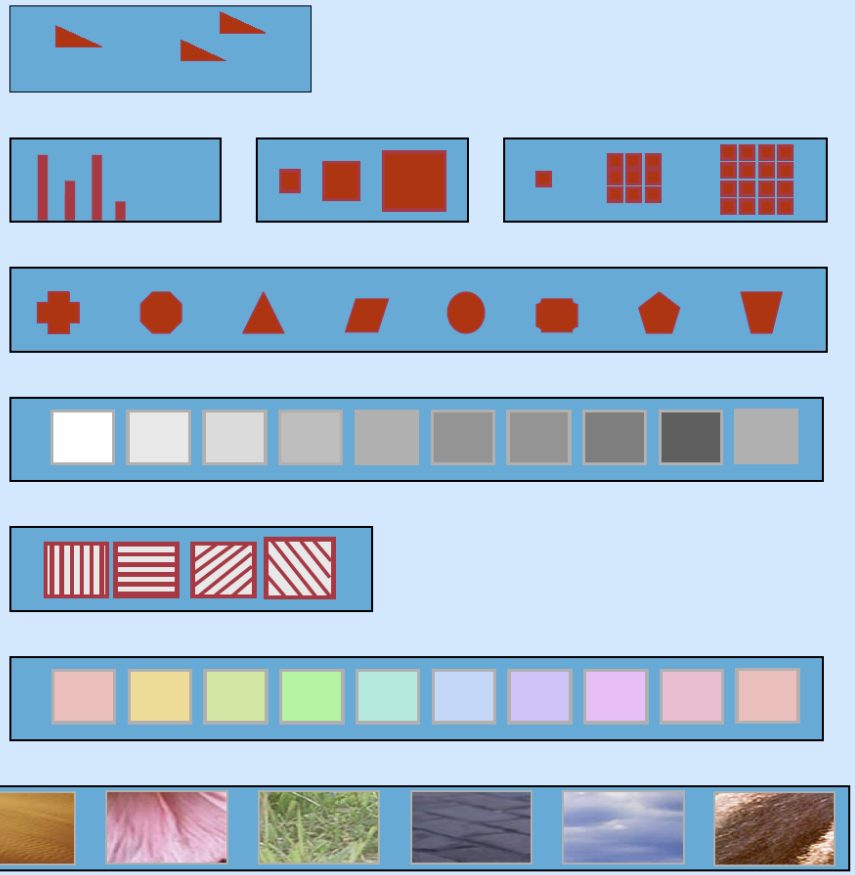


Seokhee Hong

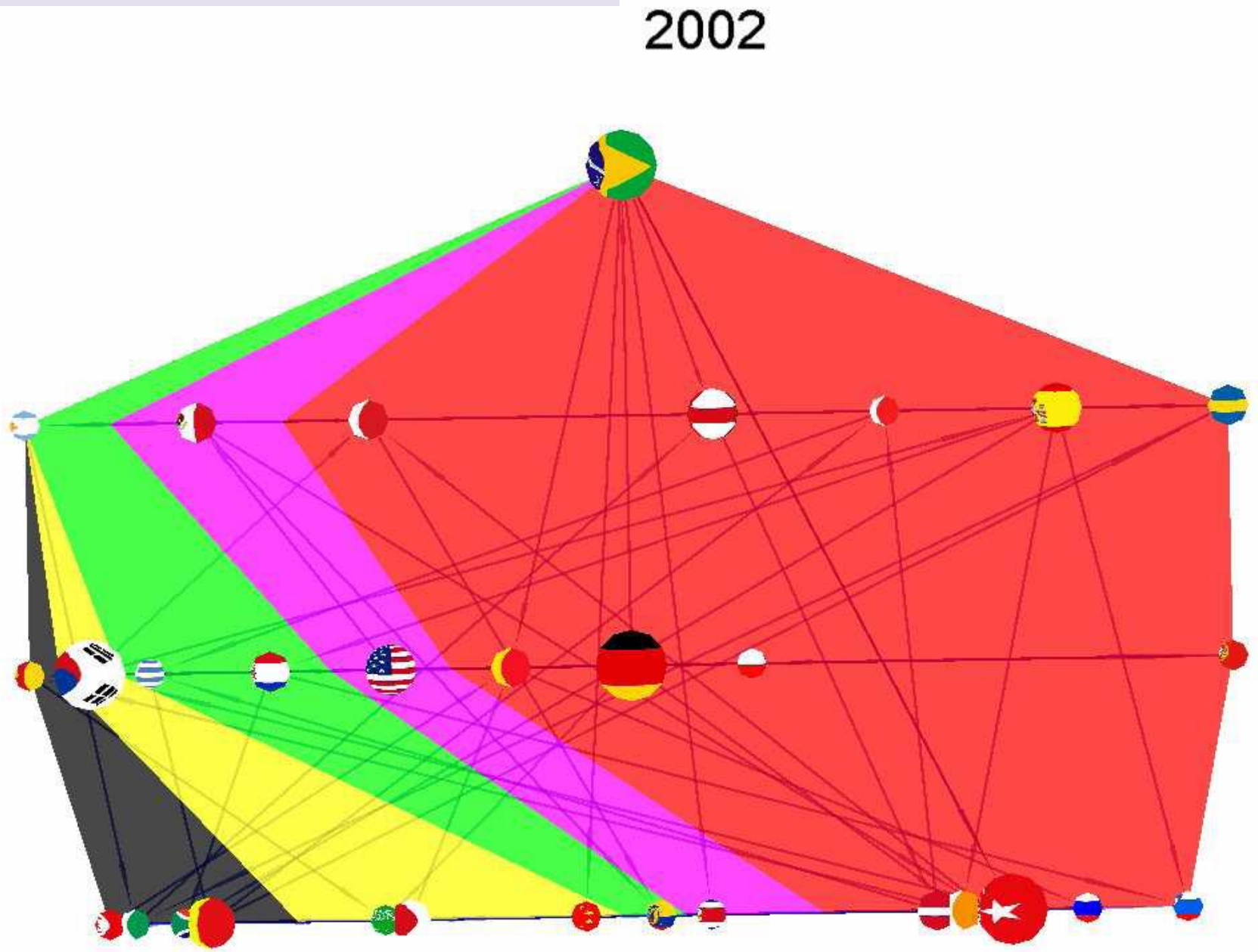
3. Glyph

- ✓ **Data-Ink-Ratio** [Tufte]: minimise ink to represent information
- ✓ **Marks (signs): line, point, area**
- ✓ **Visual Variables** [Bertin]

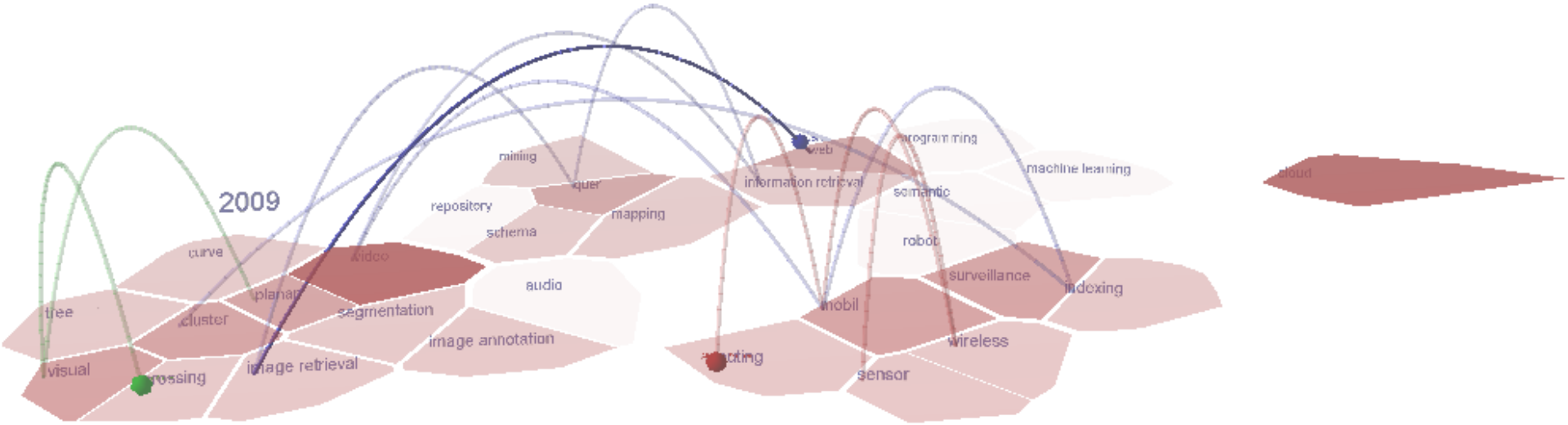
- **position**
  - changes in the x, y, (z) location
- **size**
  - change in length, area or repetition
- **shape**
  - infinite number of shapes
- **value**
  - changes from light to dark
- **orientation**
  - changes in alignment
- **colour**
  - changes in hue at a given value
- **texture**
  - variation in pattern
- **motion**



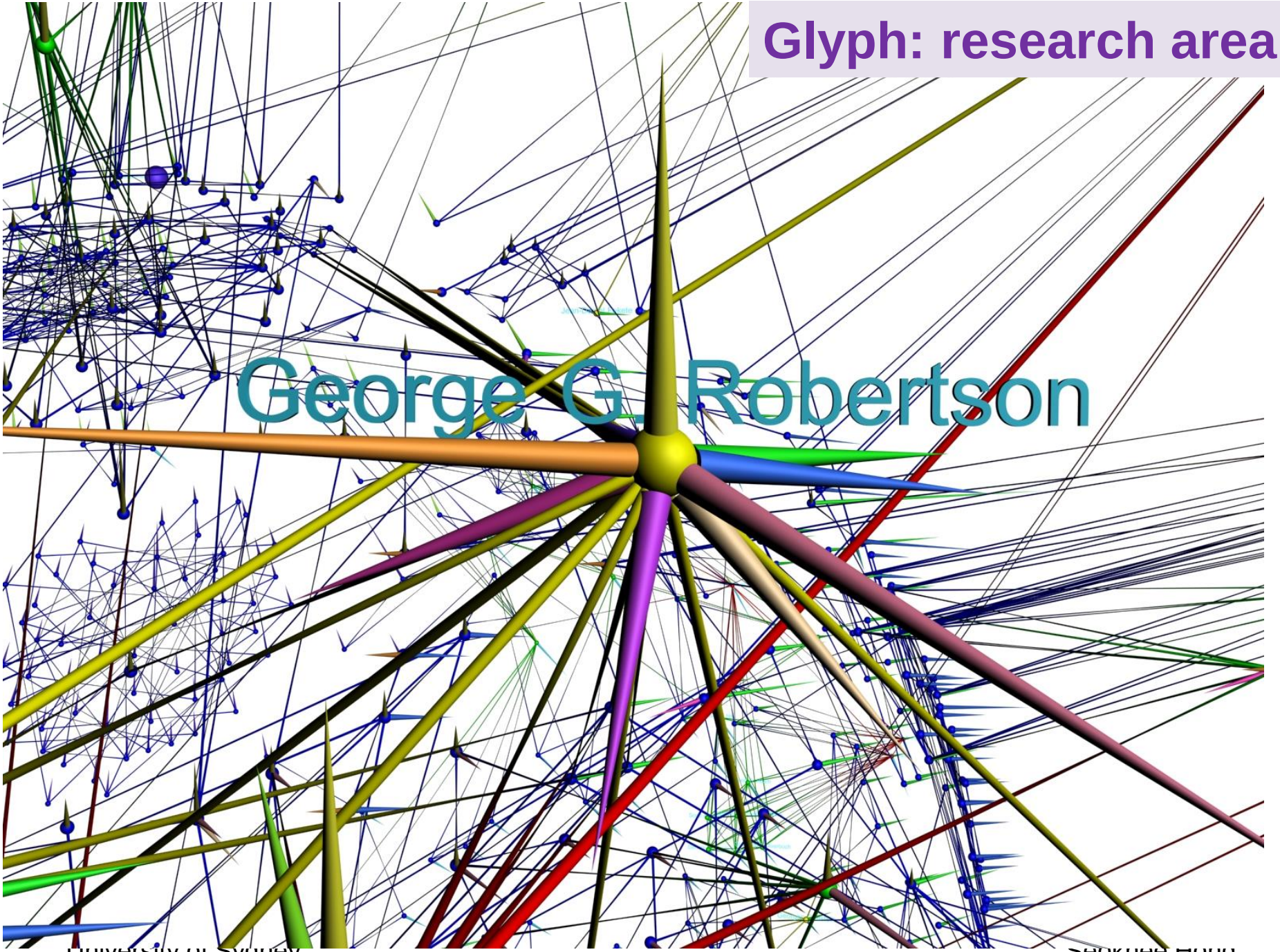
History of World Cup



Comparison of Trajectories



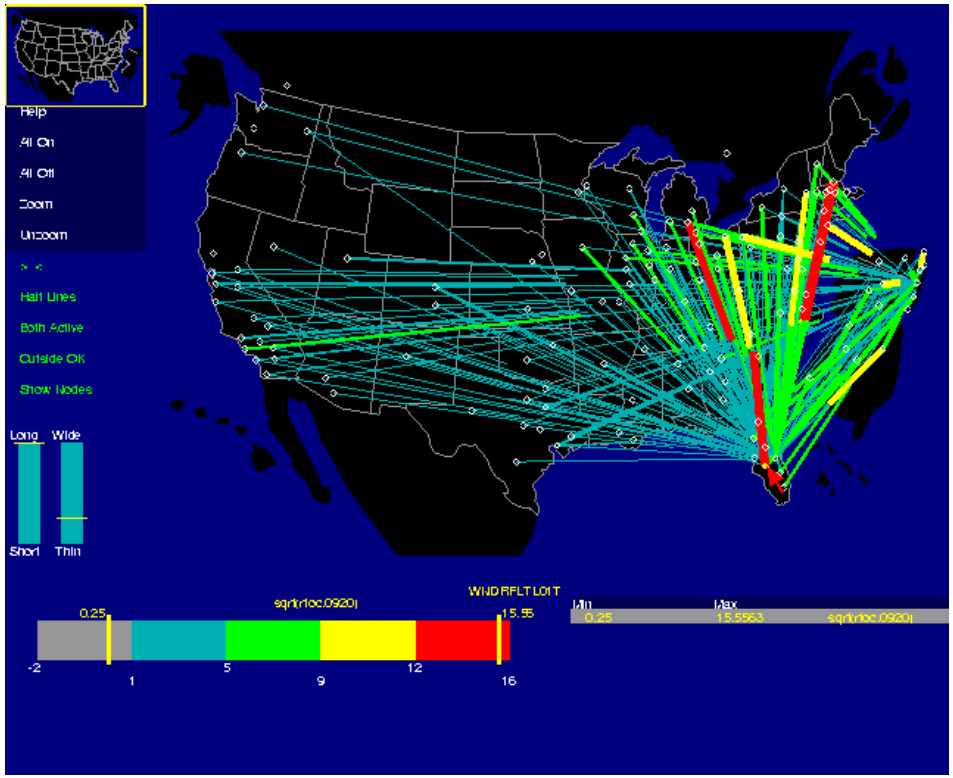
Glyph: research area

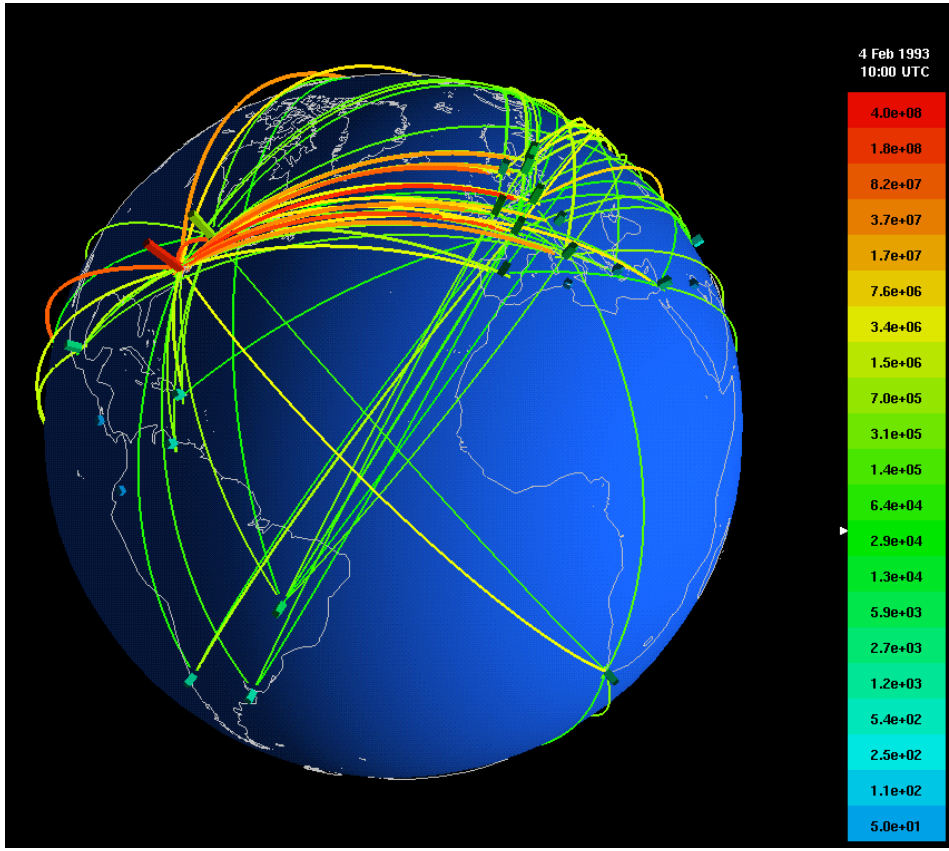


(2) Spatial Data Visualisation

Geo Visualisation (map)
Trajectory visualisation

Geographic/Spatial Data

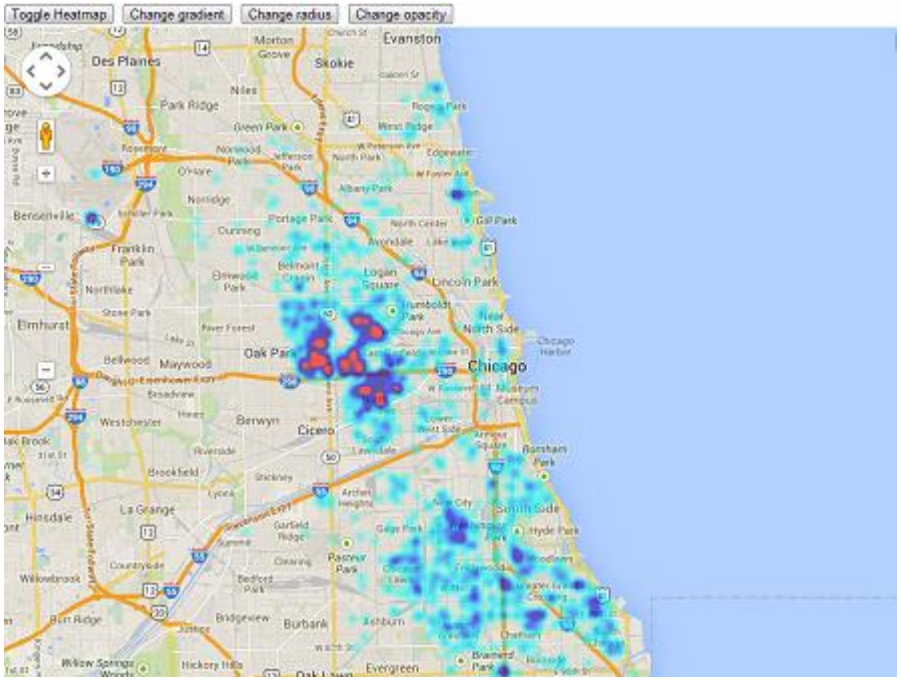




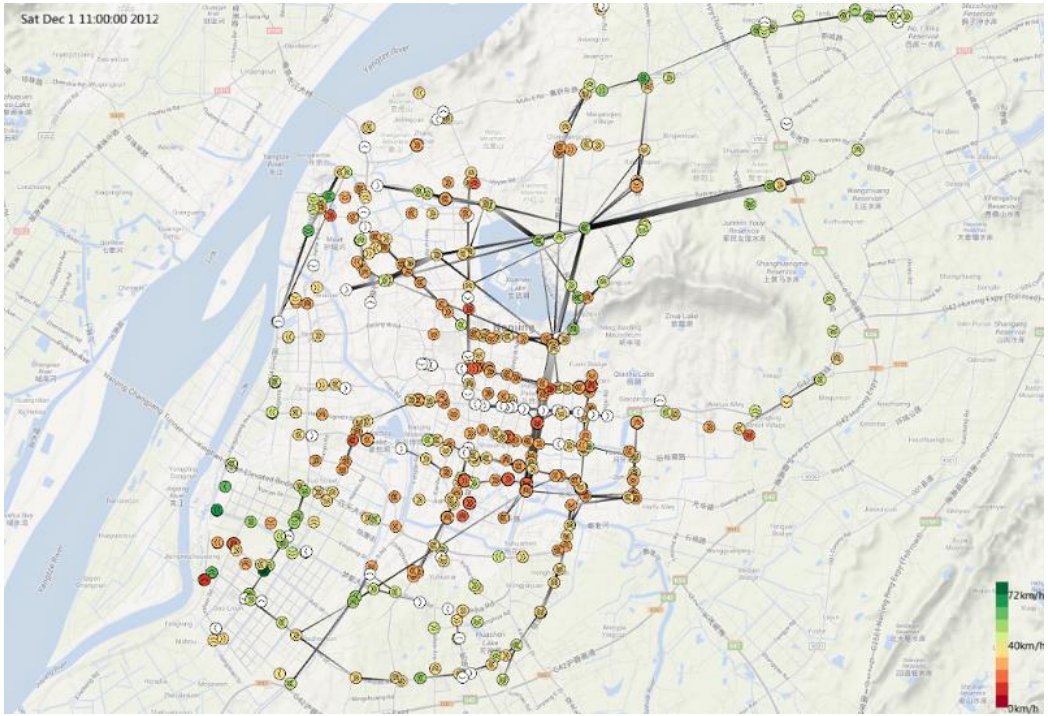
- **[Becker, Eick, Wilks 95]**
- **SeeNet**

[Cox, Eick 95] 3D Displays of Network Traffic

SeeNet3D



Crime Data Visualisation



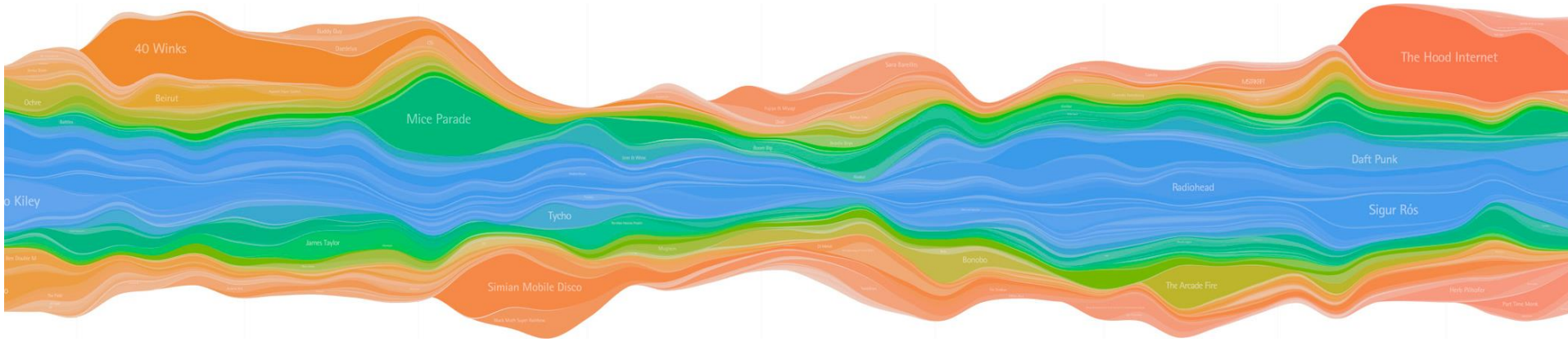
Traffic analysis using Trajectory

Seokhee Hong

(3) Temporal/Dynamic Data Visualisation

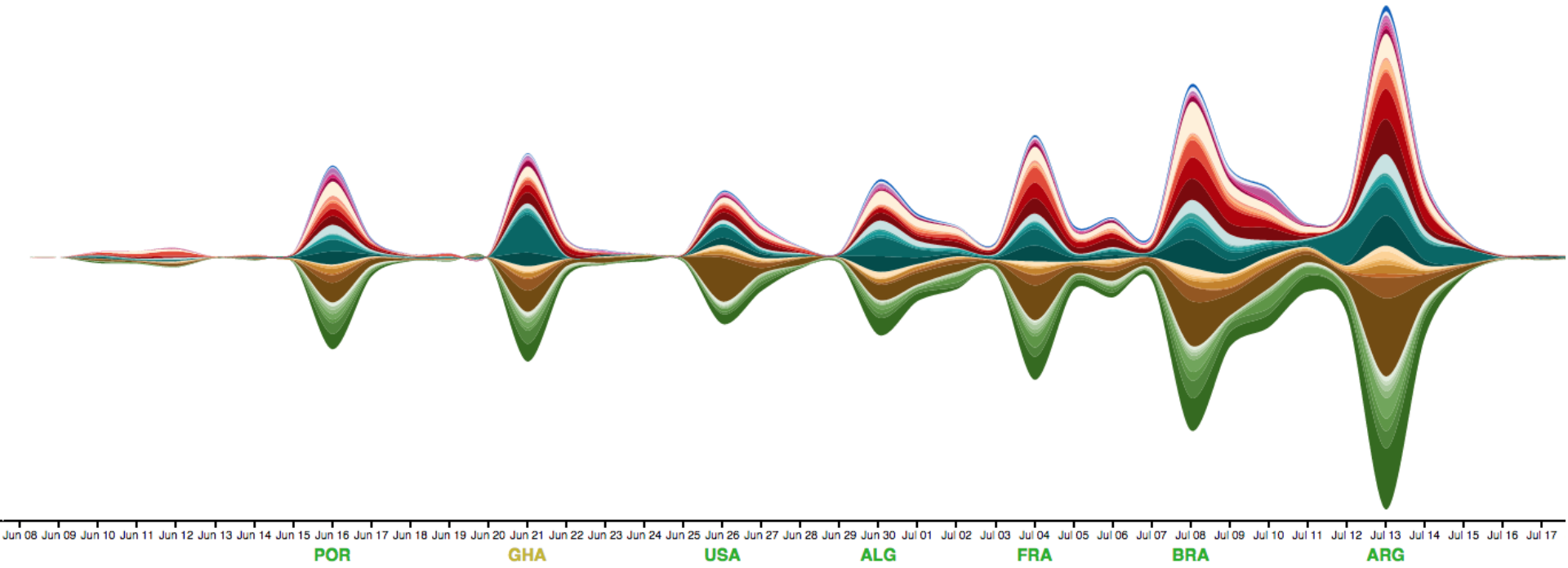
1. Stream Graph
2. Theme River
3. Storyline

1. Stream Graph



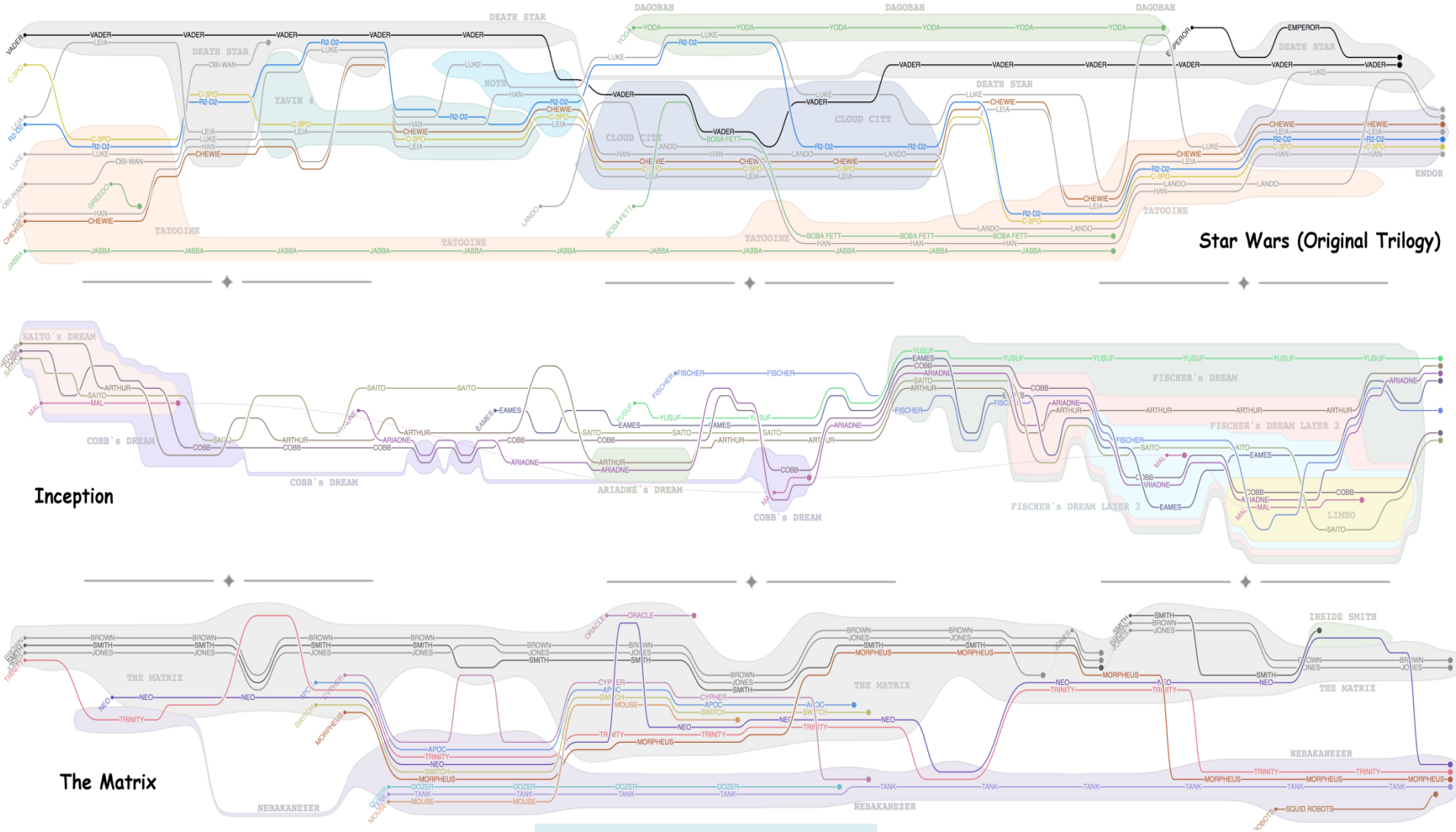
Shows overview
Trend analysis

2. Theme River



Shows overview
Trend analysis

3. Storyline



Shows overview
Dynamic clustering

(4) Relational Data Visualisation:

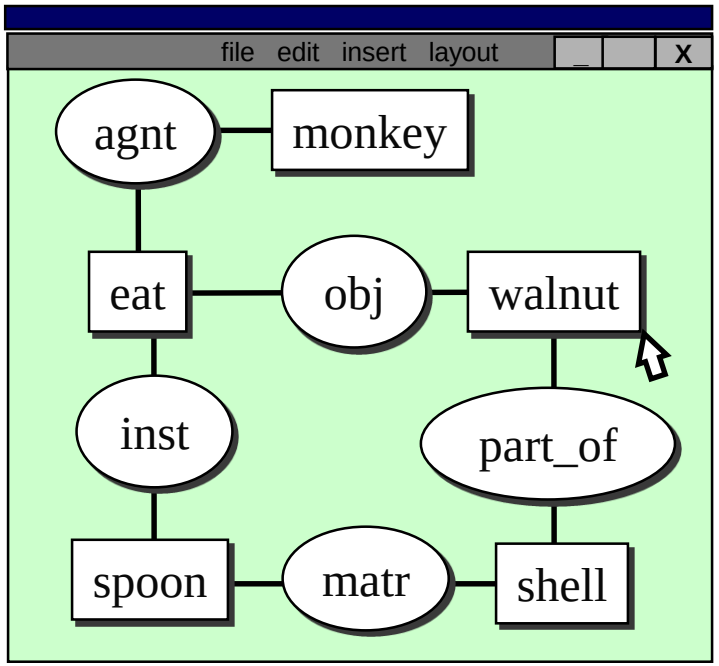
Graph Drawing
(Network Visualisation)

(4A) Graph Drawing

The input graph is usually some relational description of a software system.

The output picture is used in a system design/analysis tool.

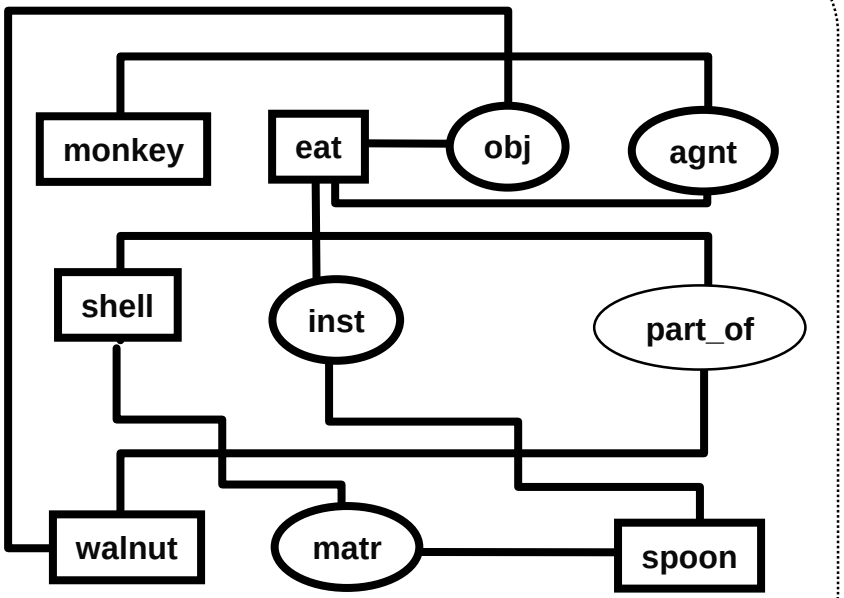
agnt(monkey, eat).
inst(eat, spoon).
obj(eat, walnut).
part_of(walnut, shell).
matr(spoon, shell).



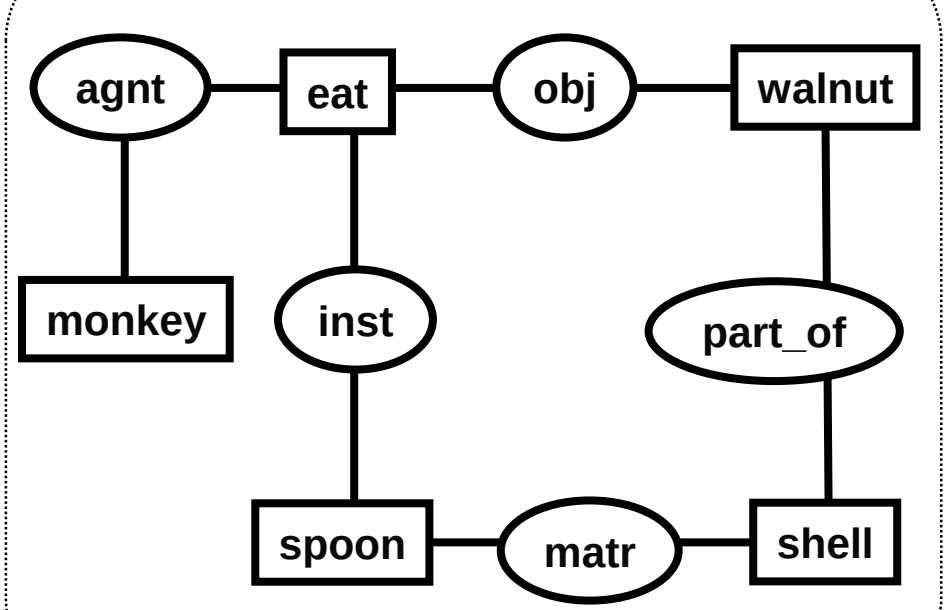
The graph drawing problem is to design methods to give good drawings of graphs.

Aesthetics

readability: **the drawing should be easy to read, easy to understand, easy to remember, beautiful.**



less readable



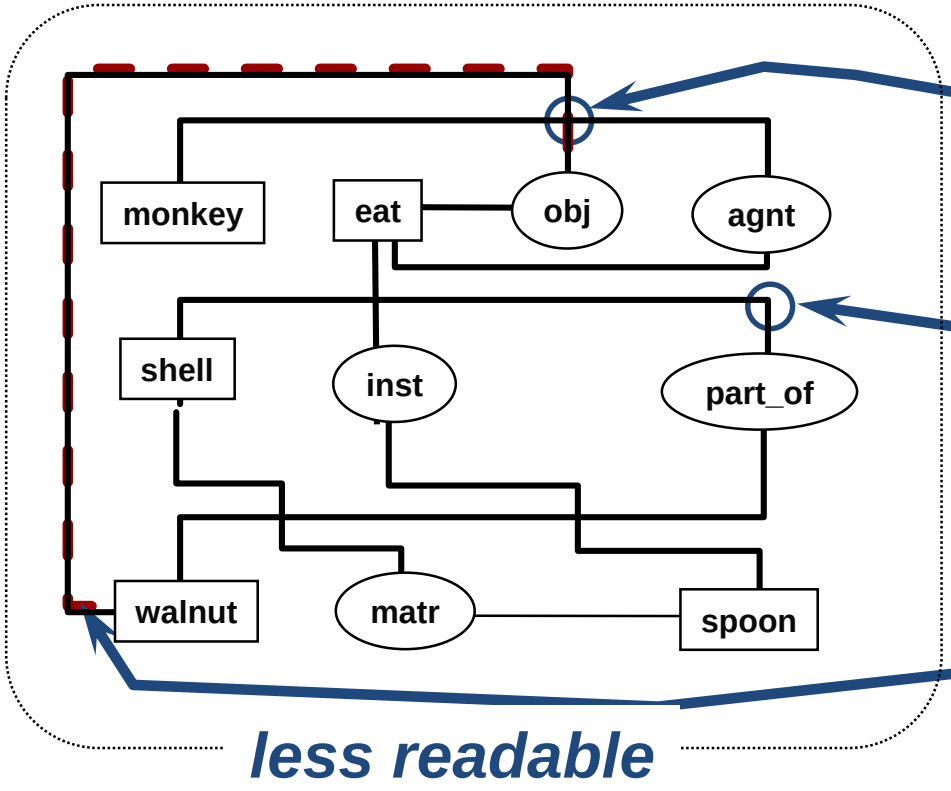
more readable

Aesthetics (quality metrics)

Readability is sometimes measured by ***aesthetic criteria***

- crossings
- area
- symmetry
- edge length
  - total edge length, maximum edge length, uniform edge length
- bends
  - total bends, maximum bends, uniform bends
- angular resolution
- aspect ratio

Crossings and bends

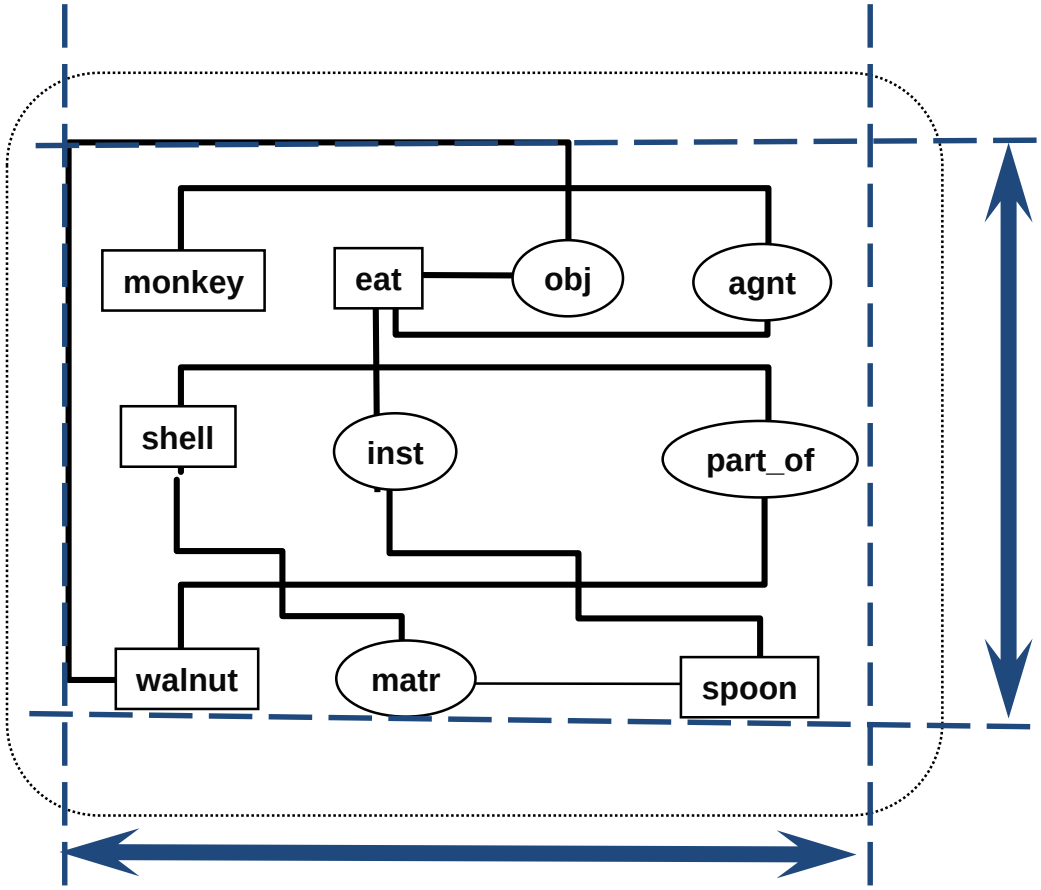


Avoid
edge ***crossings***

Avoid
edge ***bends***

Avoid
long edges

Area and resolution



One should spread the nodes evenly over the page. This can be measured:

- minimise ***area*** (for fixed size nodes)

or equivalently

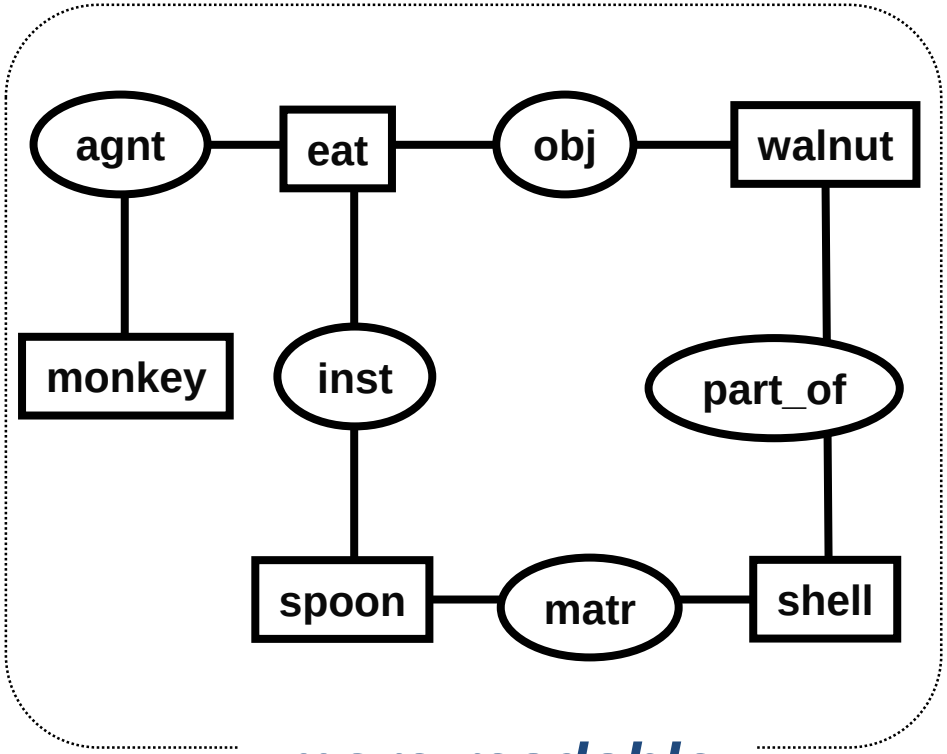
- maximise ***resolution*** (for a fixed size screen).

Aesthetics (Quality Metrics)

There are many aesthetic criteria for good diagrams:

- minimum **edge crossings**,
- minimum **bends**,
- minimum **edge lengths**,
- maximum **resolution**,

and **many more**.



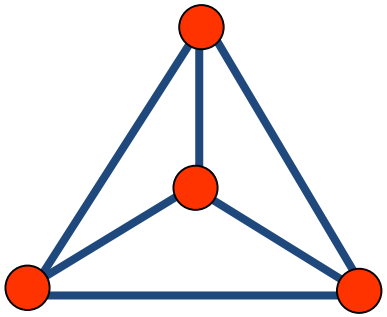
more readable

Optimisation Problem

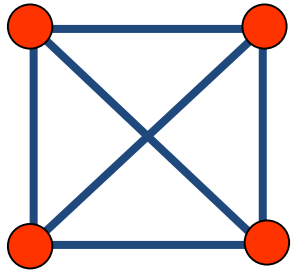
- minimize crossings
- minimize area
- maximize symmetry
- minimize total edge length
- minimize number of bends
- maximize angular resolution

-> ***NP-hard*** probelm

Conflicts

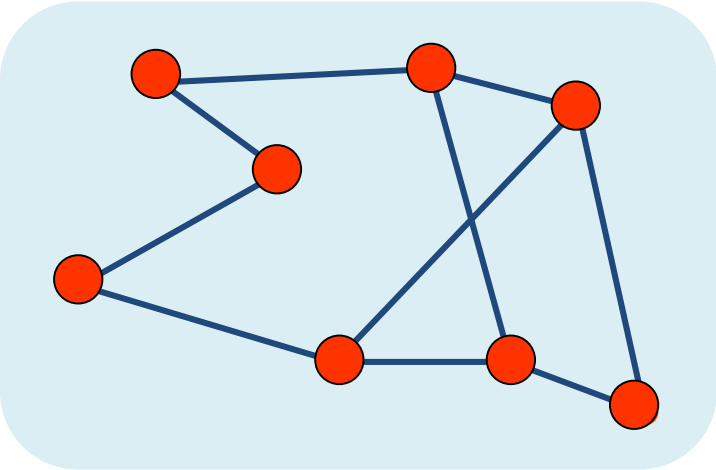


Minimize edge crossings

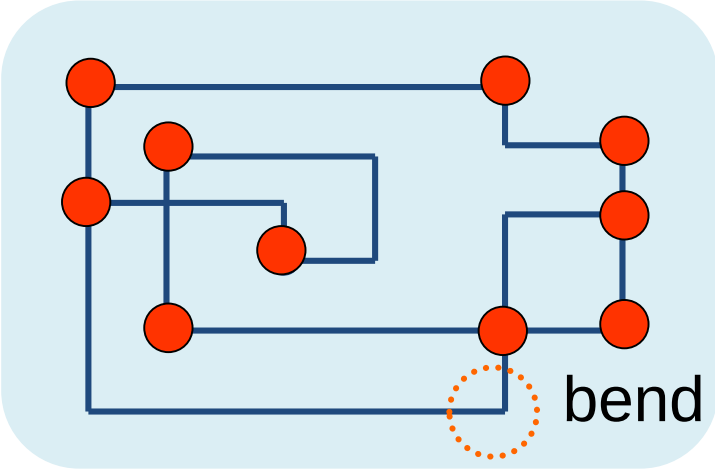


Maximize symmetry

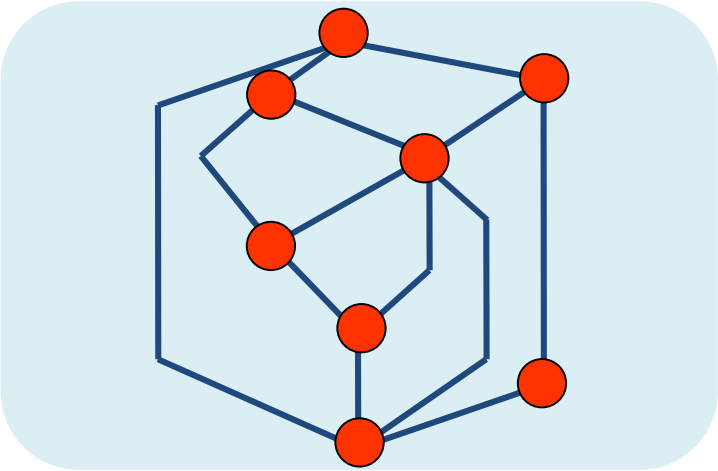
Drawing conventions



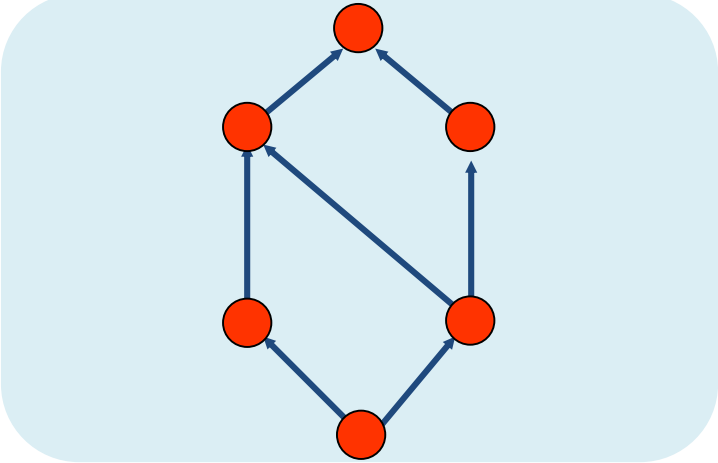
Straight-line drawing



Orthogonal drawing



Polyline drawing



Upward drawing

Graph Classes

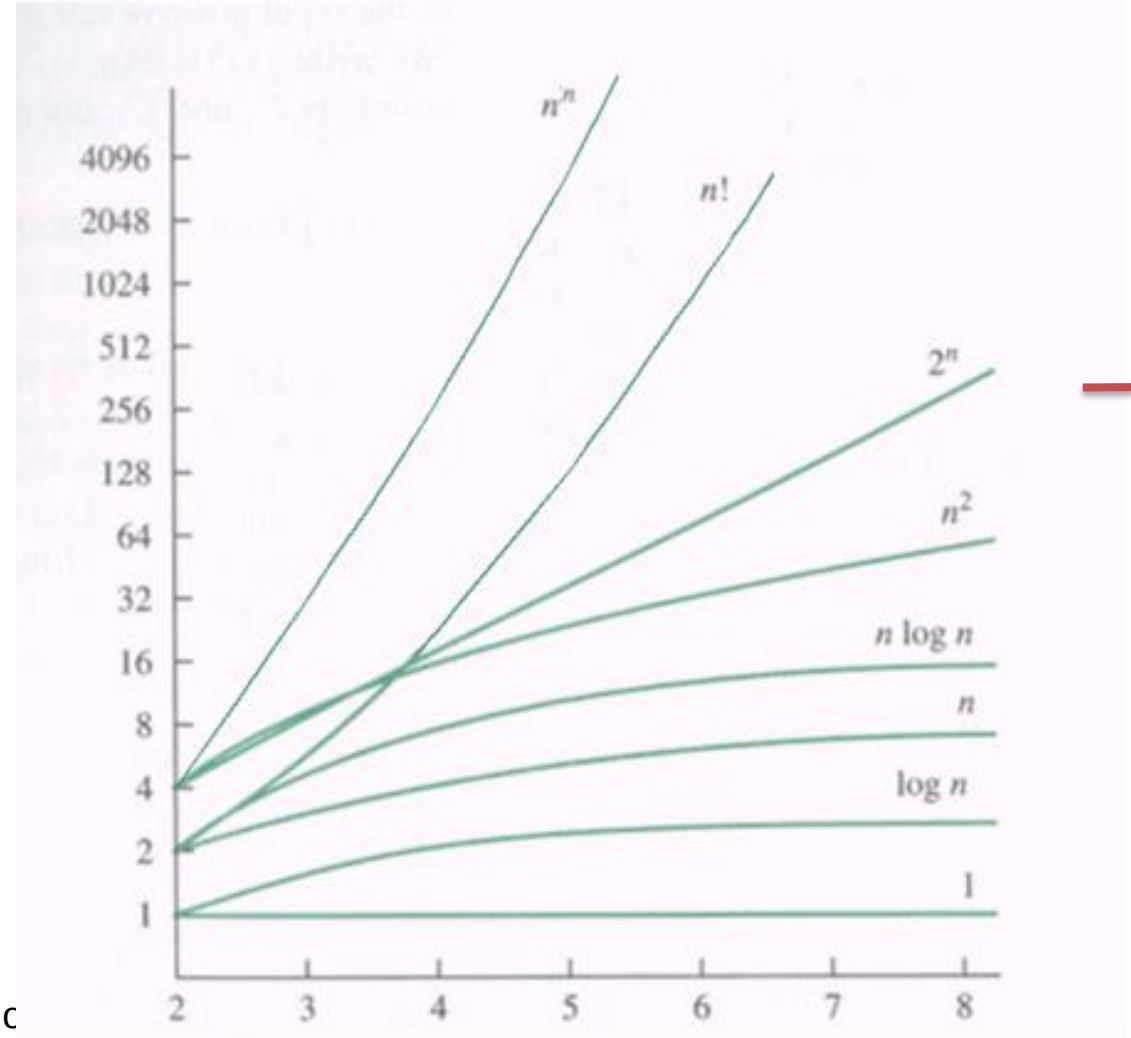
- ***tree (Week 2)***
  - **free tree**
  - **binary tree**
  - **rooted tree**
- **planar graphs**
- ***general graphs (Week 3)***
- ***directed/hierarchical graphs (Week 4)***
- ***clustered graphs (Week 5)***
  - **hyper graphs/ higraphs**

(4B) Tree Drawing Algorithm

1. Terminology
2. Layered Drawing
3. Radial Drawing
4. HV-Drawing
5. Inclusion Drawing

Big O Notation

- Measure Time complexity (Efficiency) of algorithm: run-time
- Asymptotic: eg. 5n+10 => O(n), 10000n => O(n), n: size of input data



Exponential time

Polynomial time

Algorithm: Terminology

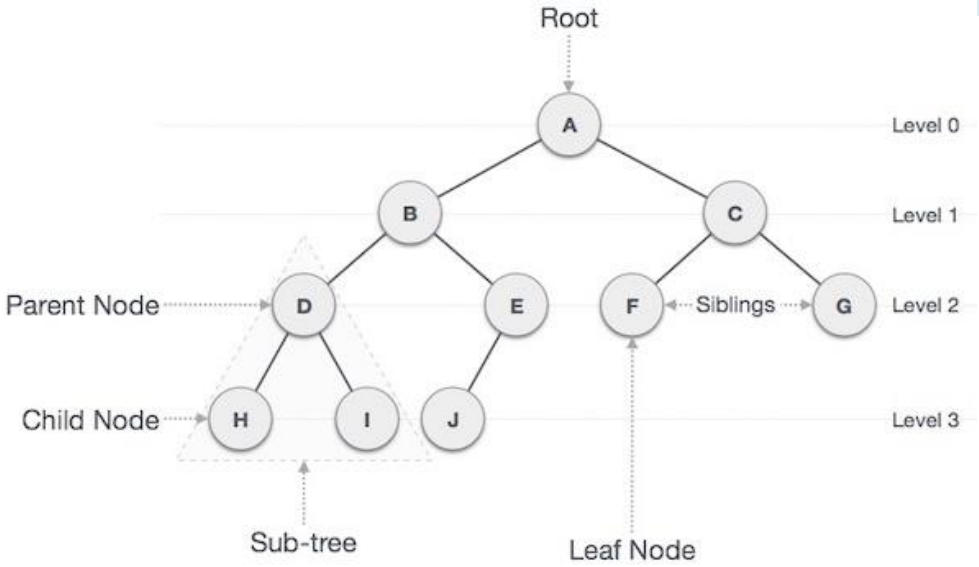
- **Divide and Conquer algorithm**
  - Divide into smaller subproblems/instances recursively
  - Solve solutions for subproblems
  - Merge solutions to obtain a solution to the original problem

- **Dynamic Programming algorithm**
  - breaking it down into a collection of simpler subproblems
  - solve each subproblem, and store the solution in a table.
  - look up previously solved subproblems and use the solutions to compute the solution for the original problem.

- **Recursion:**
  - a method where the solution to a problem depends on solutions to smaller instances of the same problem

1. Tree: terminology

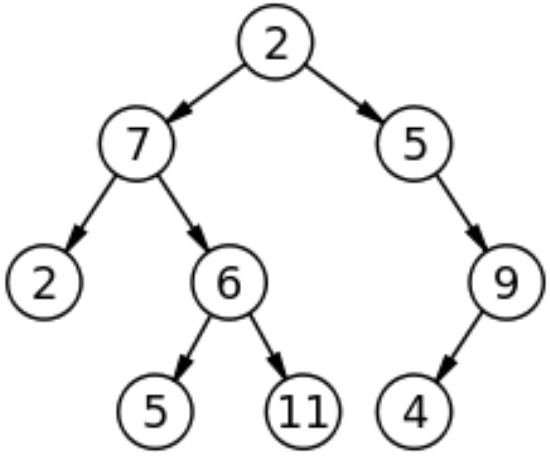
- Data structure to represent hierarchical information (no cycle)
- **Rooted** tree
  - **Root**: a distinguished vertex in a tree
  - directed edge *u->v* *(u* : *parent* of *v, v* : *child* of *u)*
  - Leaf node: no child
- **Subtree** rooted at *v* : subgraph induced by all “descendants” of *v*
- **Depth** (level) of a vertex v: number of edges from v to the root
- **Height** of a tree T: maximum depth
- Ordered tree: rooted tree with a fixed ordering for children of each vertex



Binary Tree

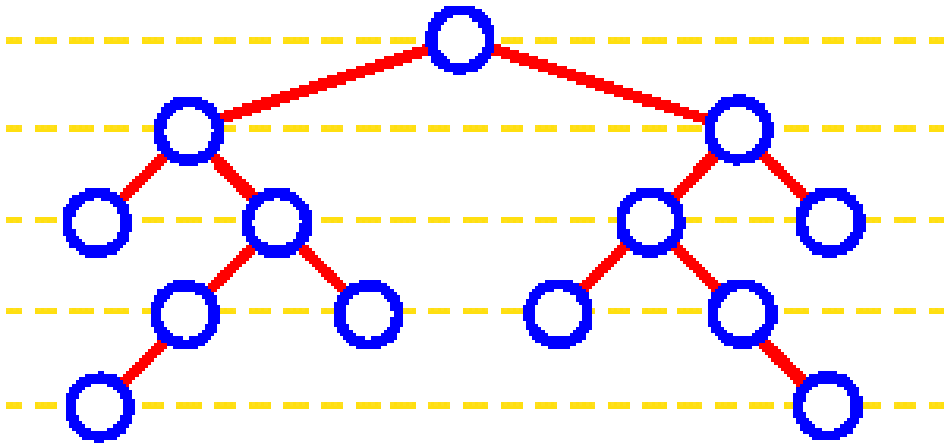
- **Binary** tree: rooted tree with every node has at most two children
  - Children: Left child and Right child
  - Subtrees: Left subtree and Right subtree

- **Binary Tree Traversal**
  - **Inorder** Traversal: Left subtree-Root-Right subtree
  - **Preorder** Traversal: Root-Left subtree-Right subtree
  - **Postorder** Traversal: Left subtree-Right-Root subtree



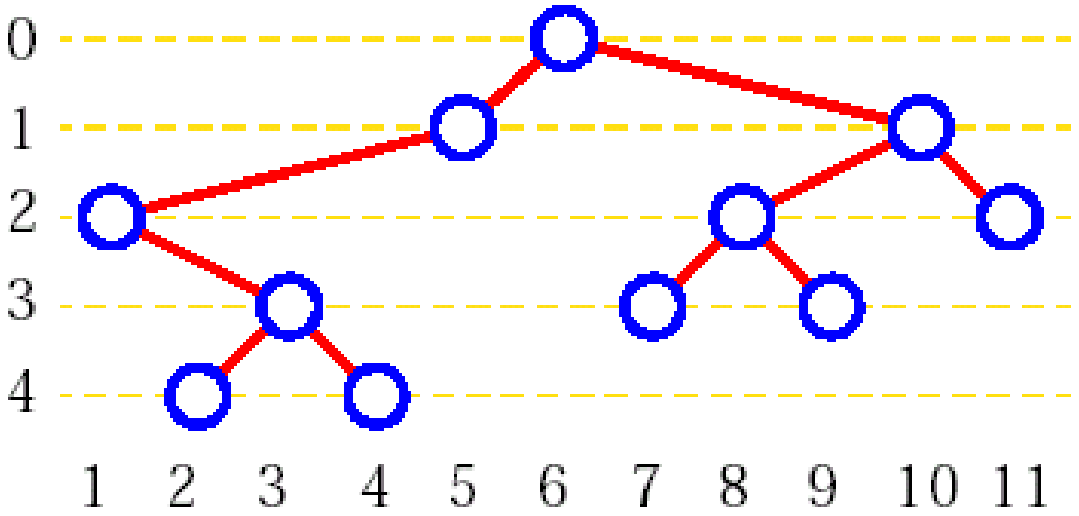
2. Layered Drawing

- rooted (binary) tree T
- assign layer according to the depth
  -> y-coordinates: *y(v) = depth of v*
- how to compute x-coordinates?



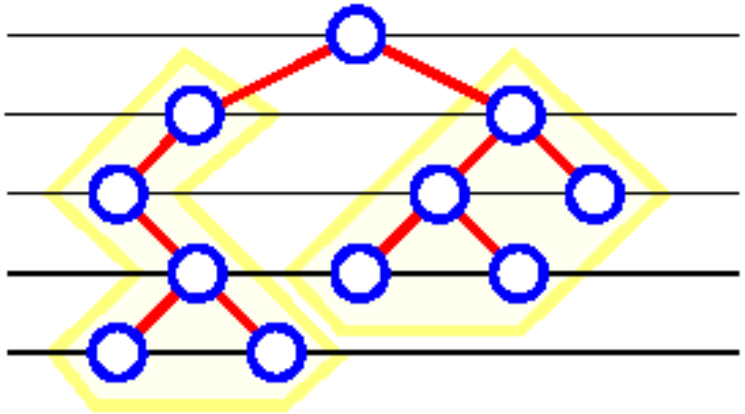
Simple Method

- Inorder Tree Traversal algorithm
  - layered grid drawing
  - two drawback:
    - too wide: *width n- 1*
    - parent vertex is not centered with respect to the children



[Reingold-Tilford 81] Tidier Drawing Algorithm

- **Divide**: recursively apply the algorithm to draw the left and right subtrees of T.
- **Conquer**
  - move the drawings of subtrees until their horizontal distance equals 2.
  - place the root r vertically one level above and horizontally half way between its children.
  - If there is only one child, place the root at horizontal distance 1 from the child.

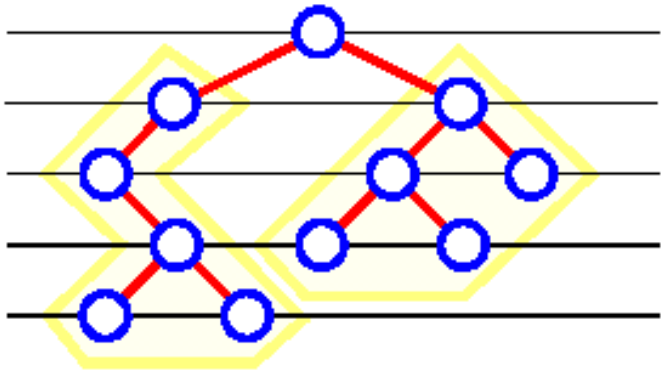


Tidier Drawing Algorithm

- Two traversals:
  - ***Step 1. Postorder traversal***

    For each vertex v, recursively computes the horizontal displacement of the left & right children of v with respect to v.
  - ***Step 2. Preorder traversal***

    Computes x-coordinates of the vertices by accumulating the displacements on the path from each vertex to the root.

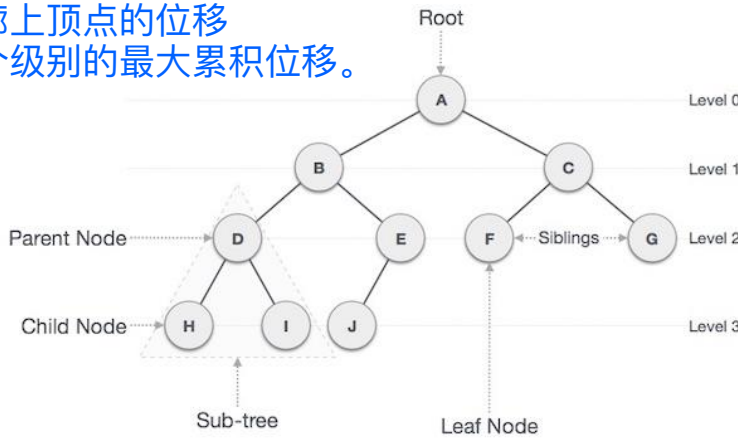


Left/Right Contours for Postorder Traversal

计算v的轮廓：
- 扫描左子树的右轮廓和右子树的左轮廓（按链接列表）
- 累积轮廓上顶点的位移
- 跟踪每个级别的最大累积位移。

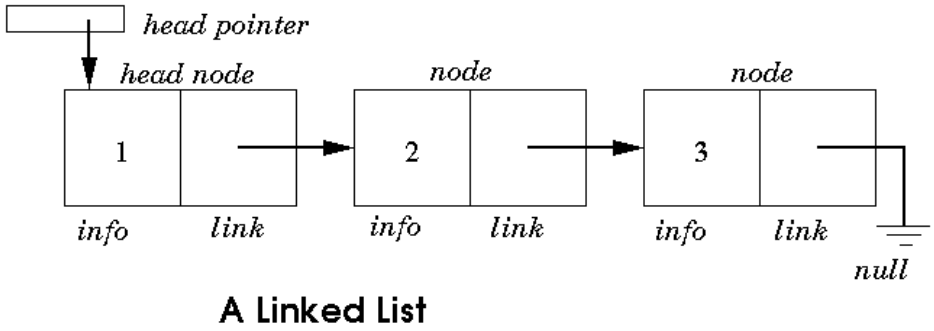
Left (Right) Contour:

- sequence of vertices v_i such that v_i is the Leftmost (Rightmost) vertex of T at level i.
- store in linked list for each v.

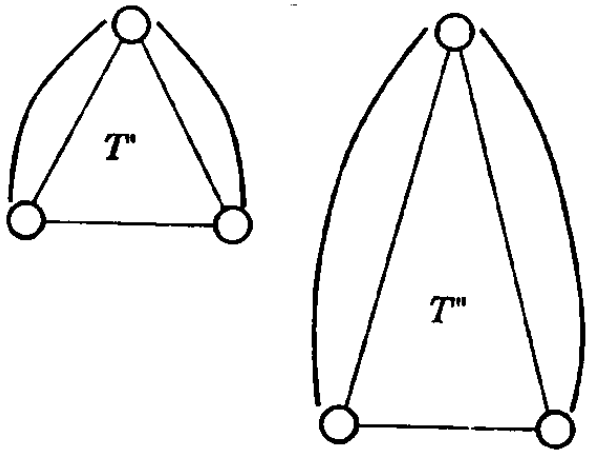


Compute contours of v:

- scan the Right Contour of the Left Subtree, and the Left Contour of the Right Subtree (follow the linked list).
- accumulate displacements of vertices on the contours
- keep track of the maximum cumulative displacement at each level.



A Linked List



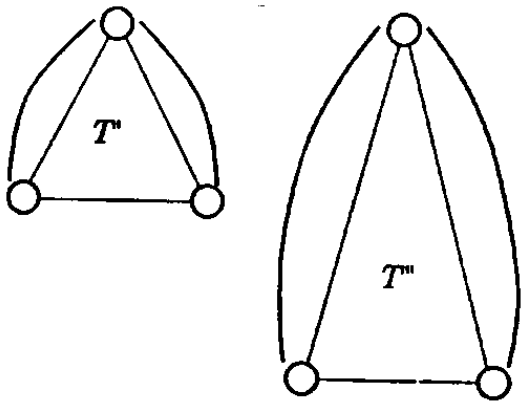
- *T: subtree rooted at v*
- *T' (T"): left (right) subtree of T*
- *L(T)* (*R(T)*): left (right) contours of *T*

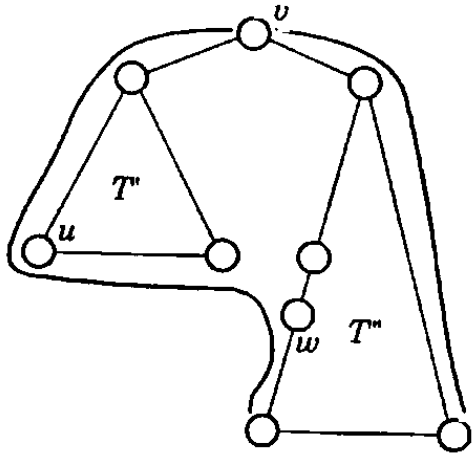
How to Construct L(T) (R(T)) ?

- *Case 1: height(T') = height(T")*
  - L(T) = v + L(T')
  - R(T) = v + R(T")

- *Case 2: height(T') < height(T")*
  - R(T) = v + R(T")
  - L(T) = v + L(T') + {part of L(T") starting from w}
    - h': height of T'
    - u: vertex of L(T') with depth h' (bottom-most)
    - w: vertex on L(T") with depth = h'+1

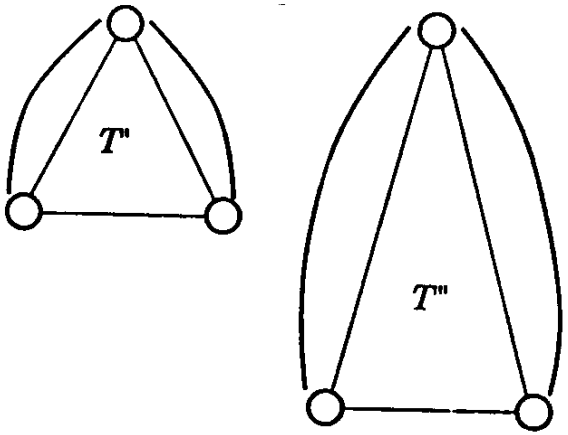
- *Case 3: height(T') > height(T")* : similar to case2

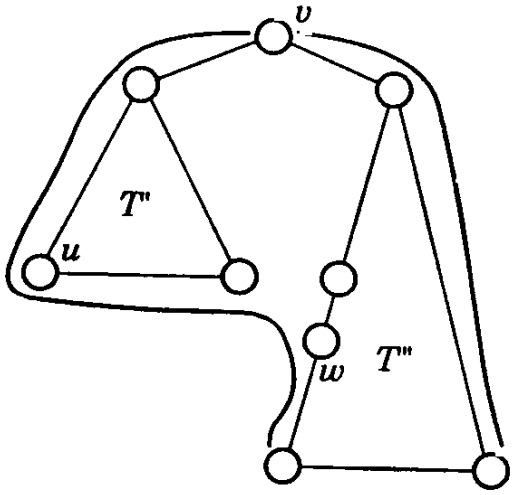




Implementing Postorder Traversal in Linear Time

- It is necessary to travel down the contours of two subtrees T’ and T” only as far as the height of the subtree of *lesser height*.
- the time spent processing vertex v is proportional to the *minimum heights of T’ and T”.*
- The sum over all vertices v of the minimum height of the subtrees of v is no more than the number of vertices of the tree.

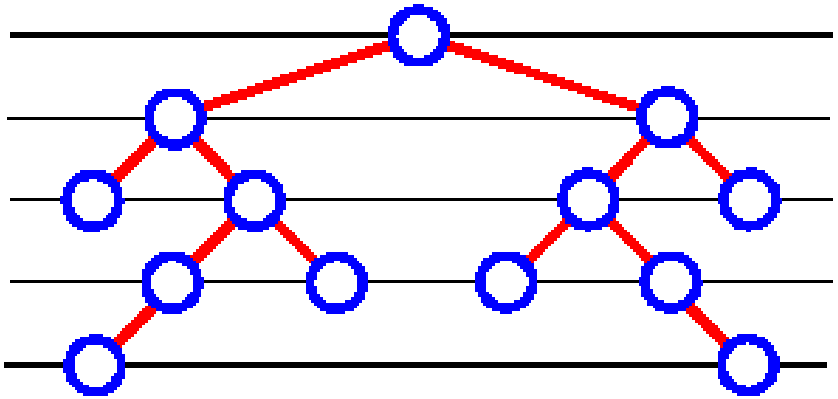




[Theorem]

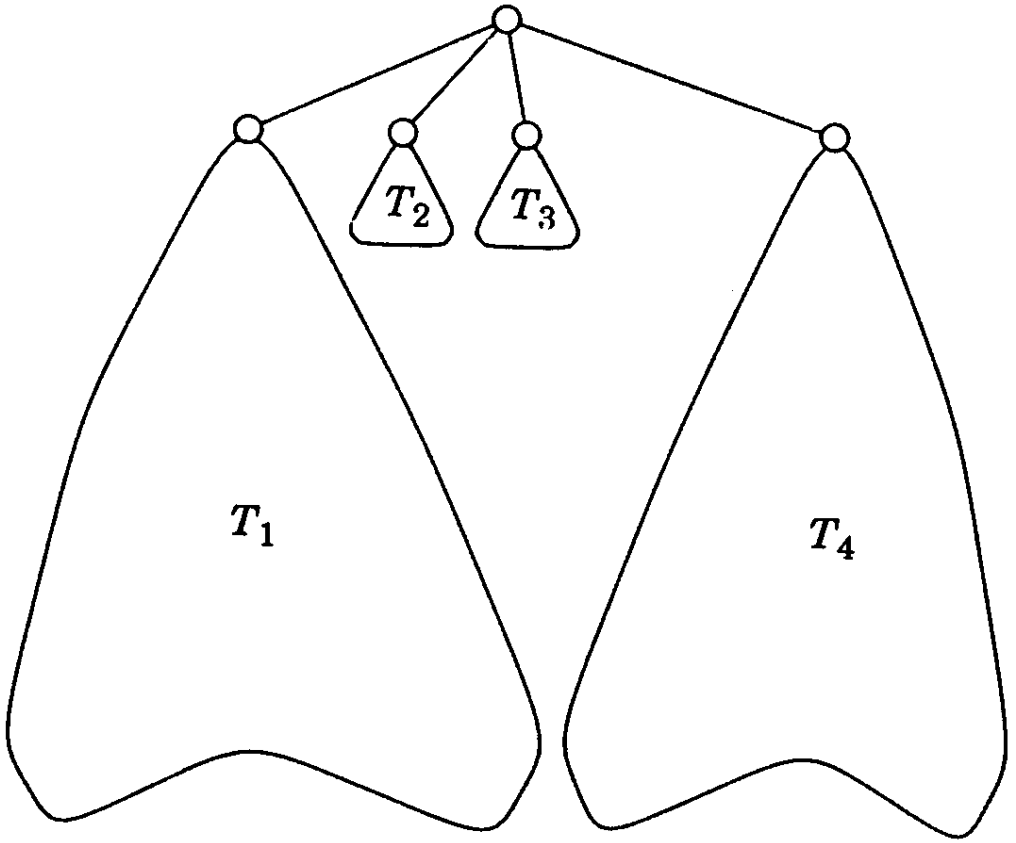
Tidier Drawing Algorithm constructs a drawing of a binary tree T in linear time such that the drawing is

- layered, planar, straight-line and strictly downward
- $O(n^2)$ area
- two vertices are at horizontal & vertical distance at least 1
- parent vertex is centered with respect to its children
- isomorphic subtrees have congruent drawing up to a translation
- axially isomorphic subtrees have congruent drawings, up to a translation & a reflection in y-axis



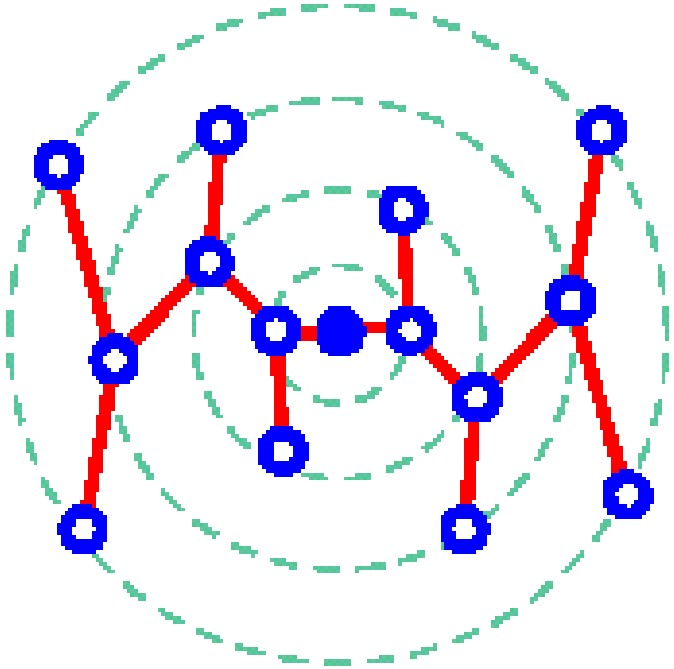
Generalization to Rooted Trees

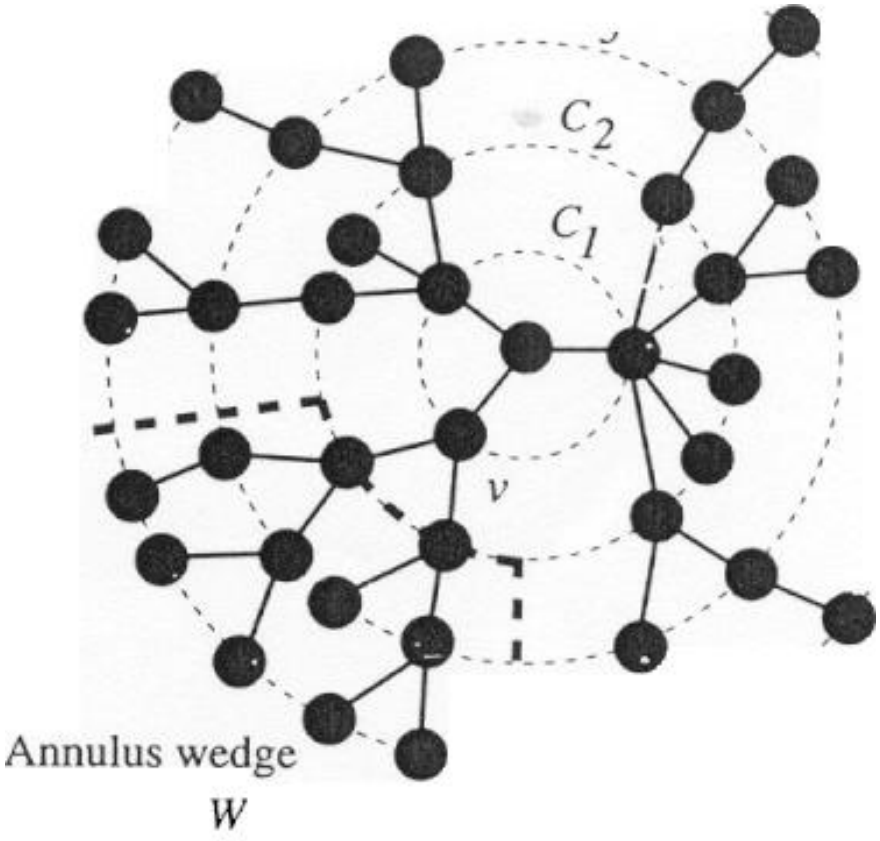
- root is placed at the average x-coordinates of its children
- small imbalance problem: T2 and T3 much closer to T1 than T4.
- Modify conquer step or postprocessing
  - [Walker 90]: layered drawing for trees



3. Radial Drawing

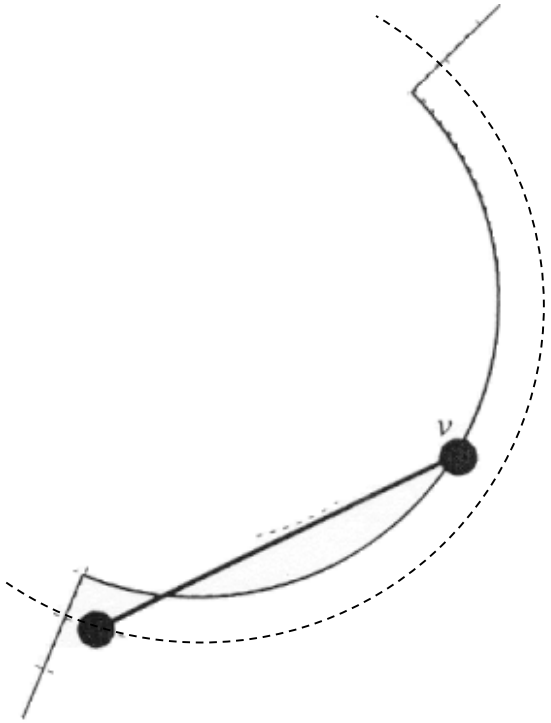
- Layers are represented as concentric circles.
- Draw each subtree in annulus wedge W.
- Angle of wedge: proportional to # of leaves of each subtree

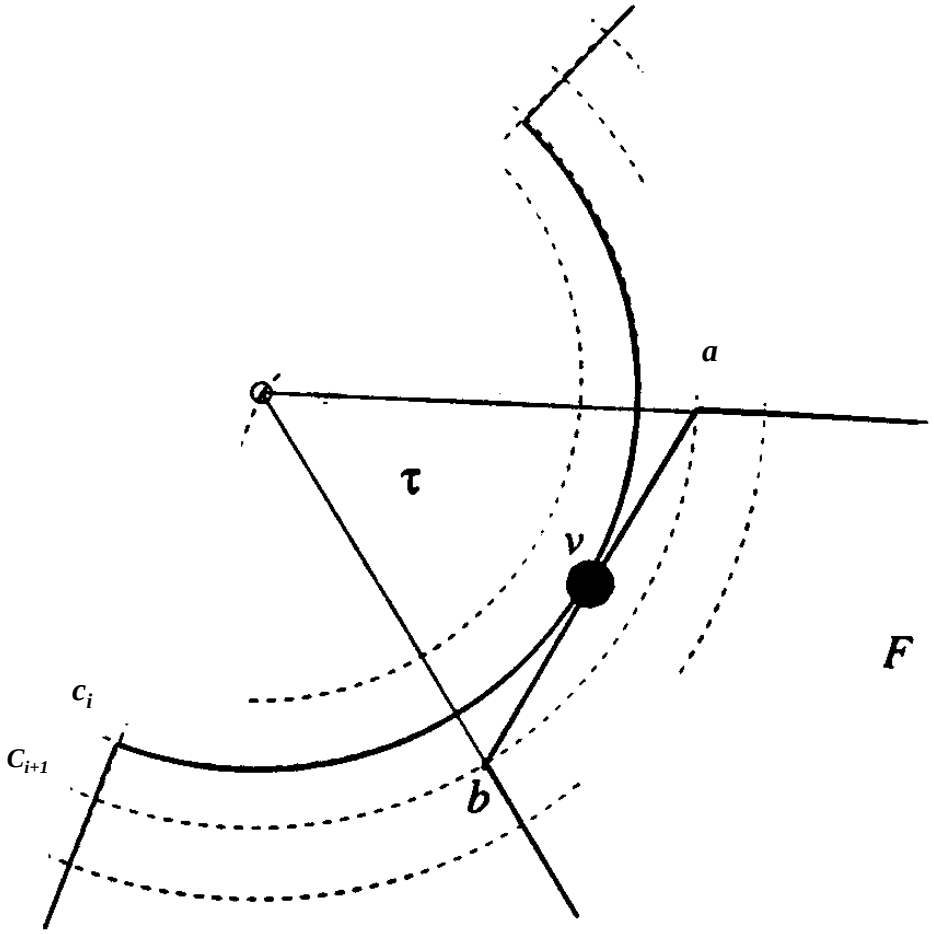




Planar Drawing

- to guarantee planarity, ***define convex subset F*** of the wedge.
- Draw a subtree inside F





Radial Drawing

- Running time: linear
- used for free trees
  - select the center as a root
- can be used to display symmetry
  - Symmetry detection: [Manning, Atallah 88]
- Variations: [Eades 92], [Bernard 81], [Esposito 88]
  - choice of root
  - radii of the circles
  - how to determine the size of the wedge

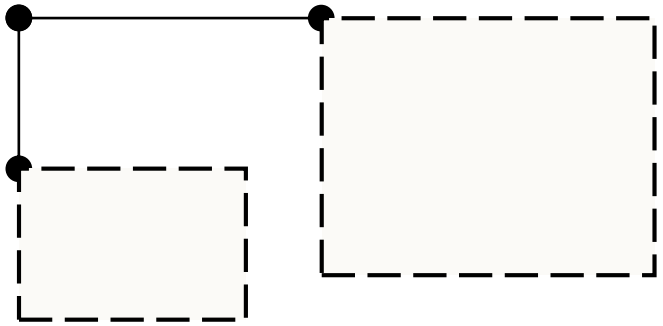
4. hv-Drawing

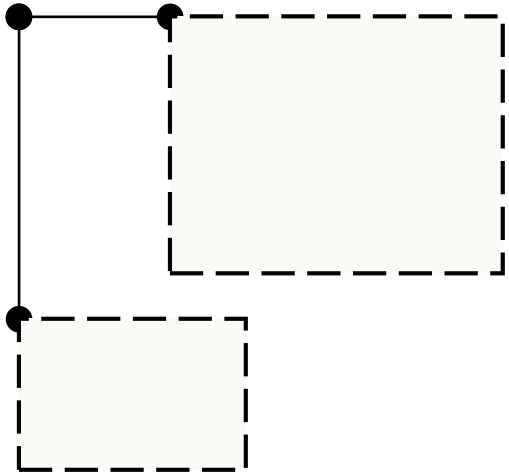
- hv-drawing of a binary tree T:

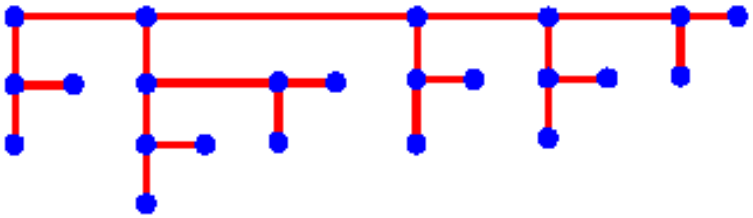
straight-line grid drawing such that for each u, a child of u is either

  - horizontally aligned to the right of u, or
  - vertically aligned below u
  - bounding rectangles of the subtrees of u do not intersect

- planar, straight-line, orthogonal, and downward







Divide & Conquer Algorithm

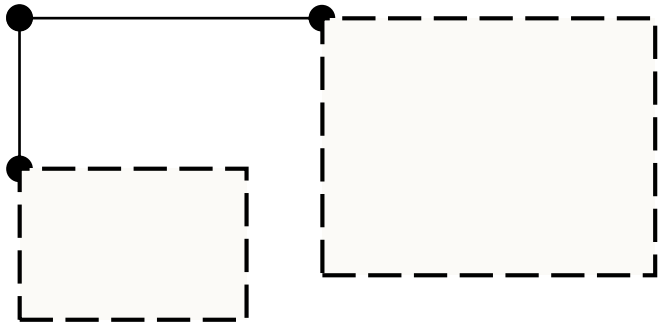
- **Divide:**

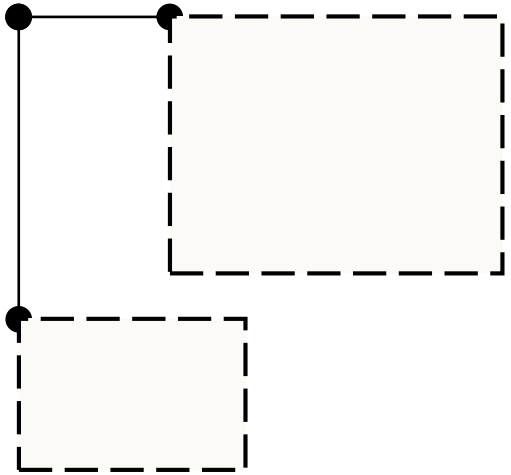
recursively construct hv-drawings for the left & right subtrees

- **Conquer:**

perform either *horizontal* combination or a *vertical* combination

- The height & width are each at most n-1

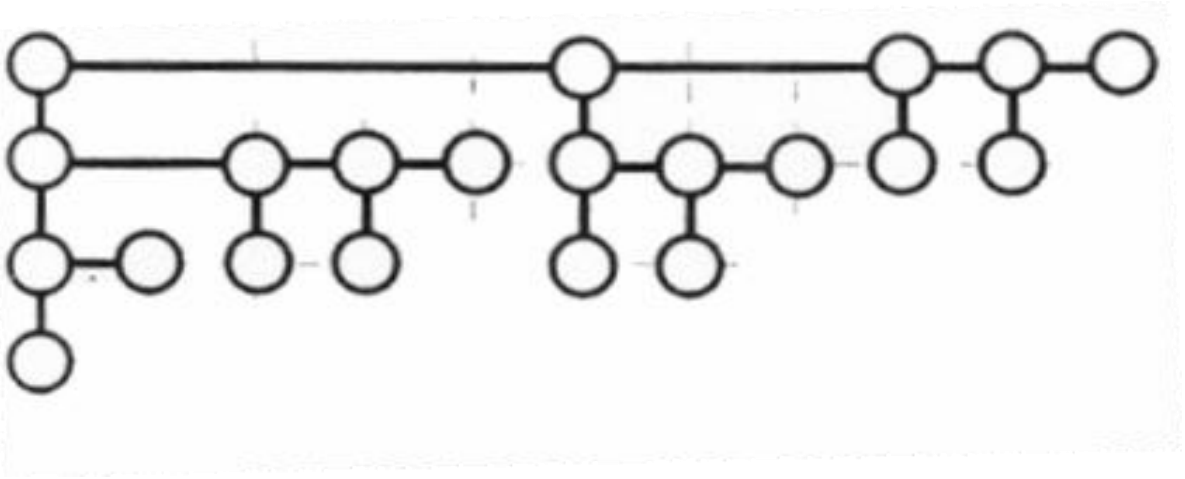




Algorithm Right-Heavy-HV-Tree-Draw

1. Recursively construct drawing of the left & right subtrees.

2. Using only ***horizontal*** combination, place the subtree with the ***largest*** number of vertices to the ***right*** of the other one.



Algorithm Right-Heavy-HV-Tree-Draw

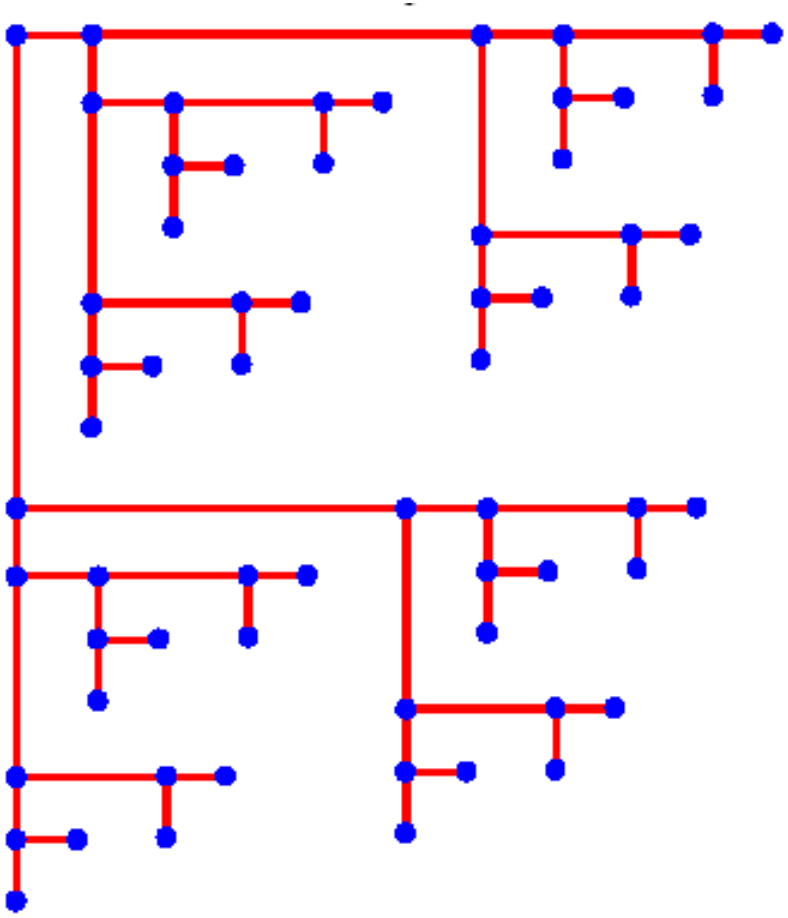
[Theorem]

Algorithm **Right-Heavy-HV-Tree-Draw** construct a drawing of binary tree T with n vertices such that the drawing is

- hv-drawing (downward, planar, grid, straight-line and orthogonal)
- area **O(nlogn)**
- width is at most **n-1**
- height is at most **logn**
- simply and axially isomorphic subtrees have congruent drawings, up to a translation

Algorithm Right-Heavy-HV-Tree-Draw

- Good area bound, but bad aspect ratio
- **Better aspect ratio**: use both horizontal/vertical combinations for odd/even depth.
- **Complete binary tree**: O(n) area and constant aspect ratio.



General Binary Tree

[Eades, Lin, Lin 92] [Eades, Lin, Lin 93]

It is possible to construct an **hv-drawing** of a general binary tree that is optimal with respect to "area" or "perimeter" in $O(n^2)$ time.

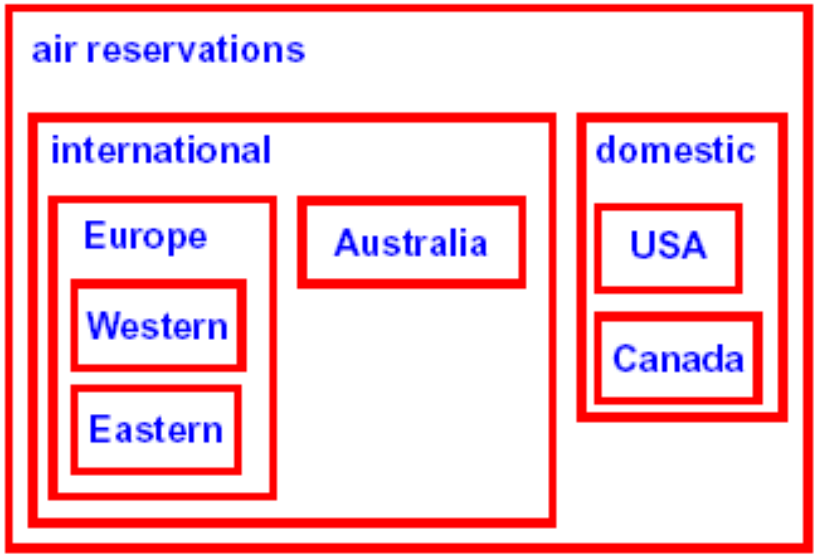
•use dynamic programming algorithm.

5. Inclusion Tree Drawing

[Eades, Lin, Lin 93] ***Inclusion Drawing*** of Rooted Trees

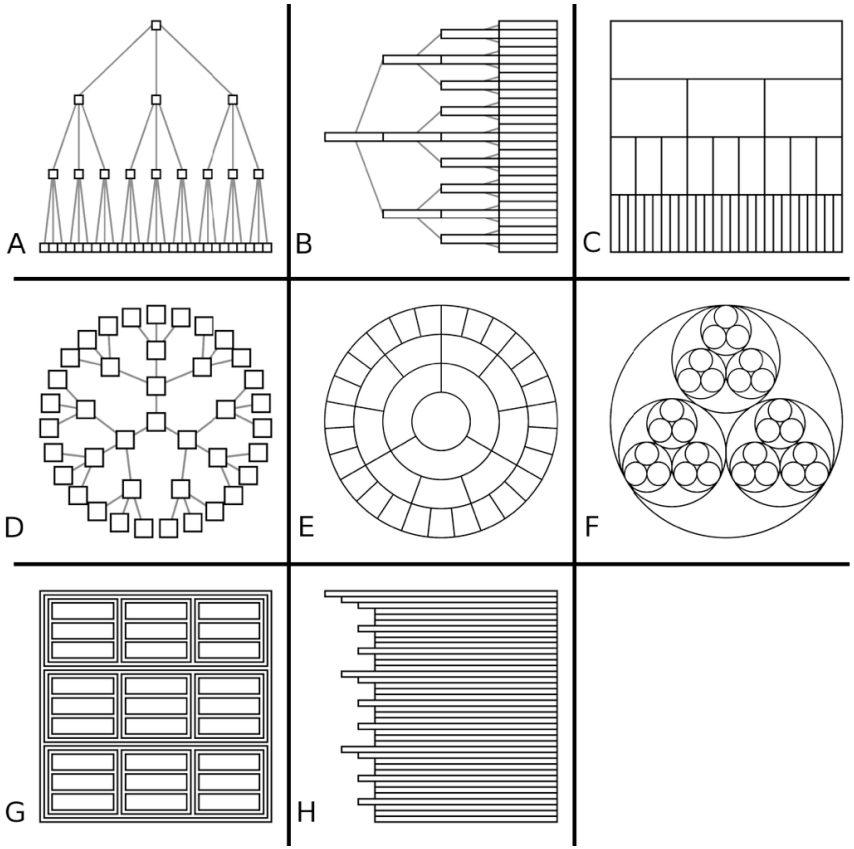
Display the parent-child relationship by the inclusion between isothetic rectangles.

- Minimization of area (perimeter, width, height)
  - NP-hard for general trees
  - Polynomial time algorithm for balanced trees



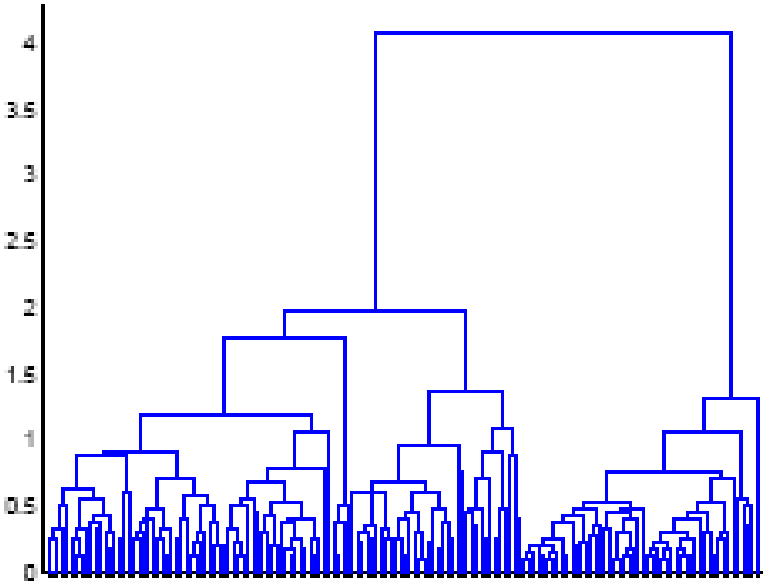
- used for *compounds graphs* (union of a graph and a tree)
- allow better fit the drawing in a prescribed region

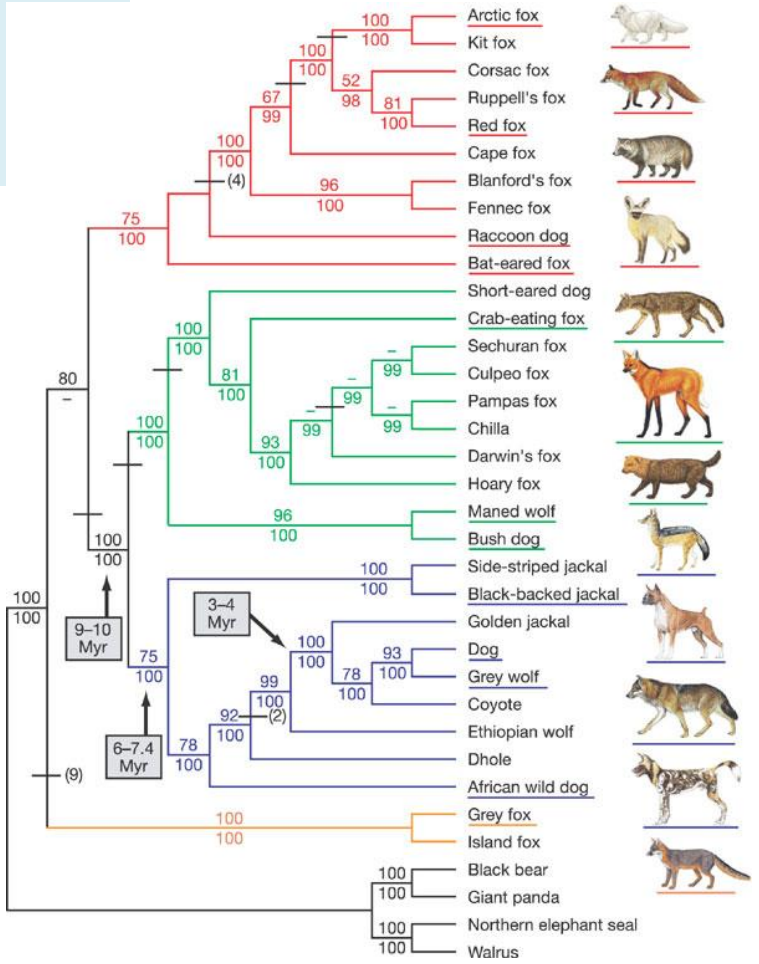
(4C) Tree visualization methods



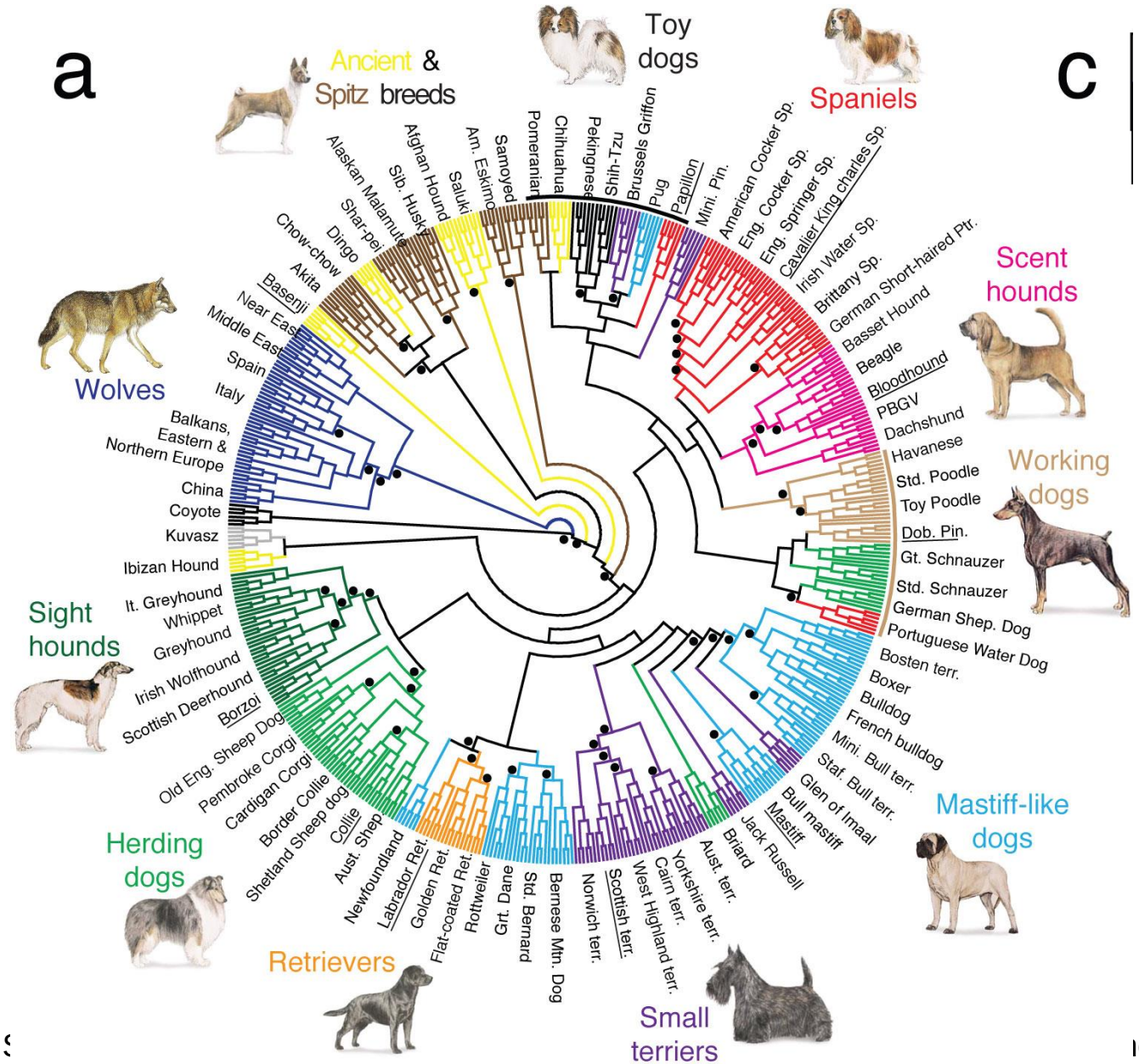
Dendrogram

- layered drawing with bended orthogonal edges.
- all the leaves are on the same layer.
- good for drawing large trees in small area.
- used in bioinformatics to represent
  - hierarchical clustering
  - phylogenetic trees





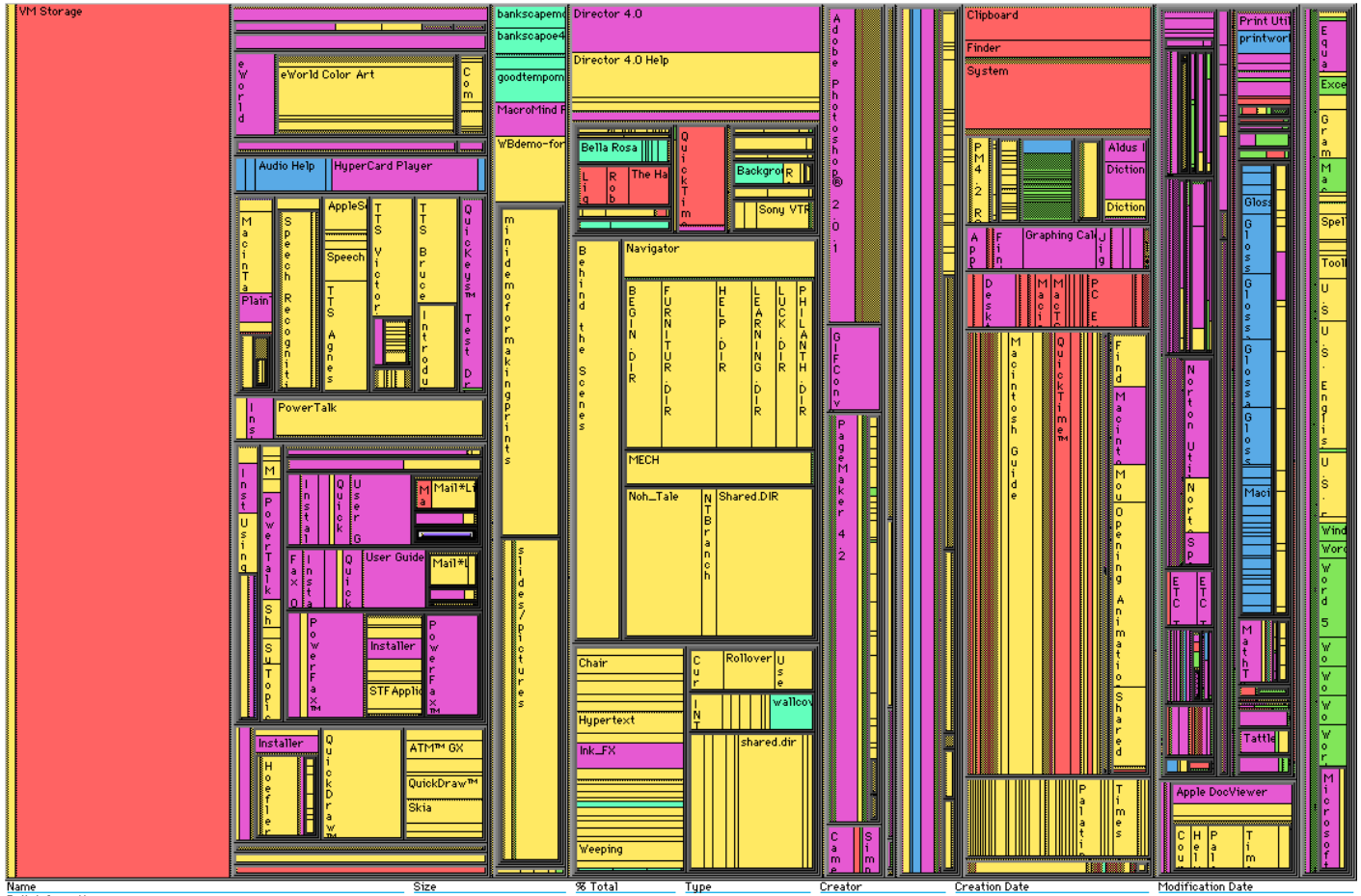
Radial Dendrogram



Treemap

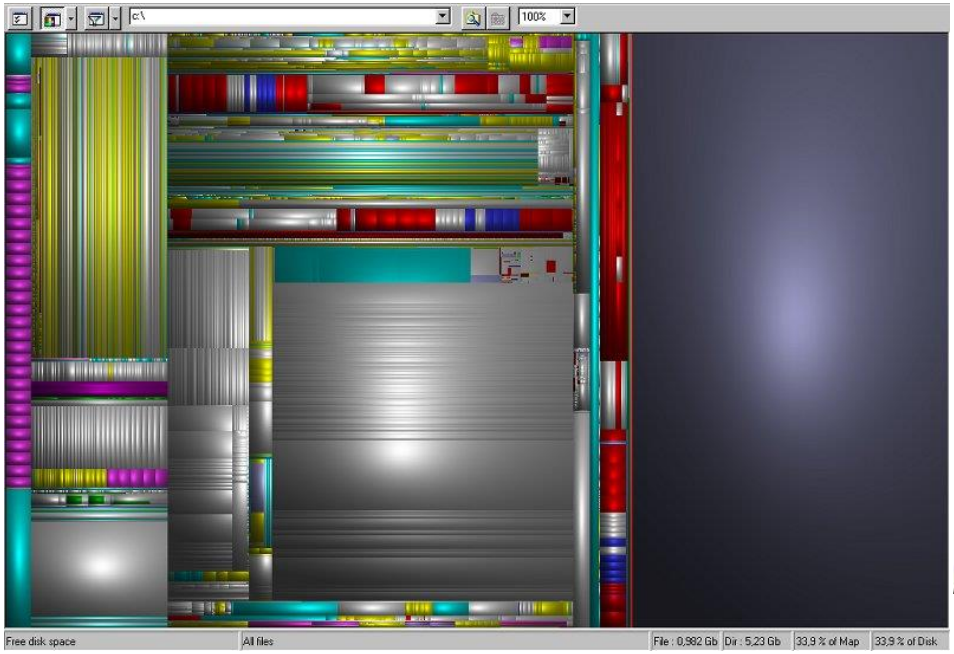
[Shneiderman 92] use containment to show the hierarchy.

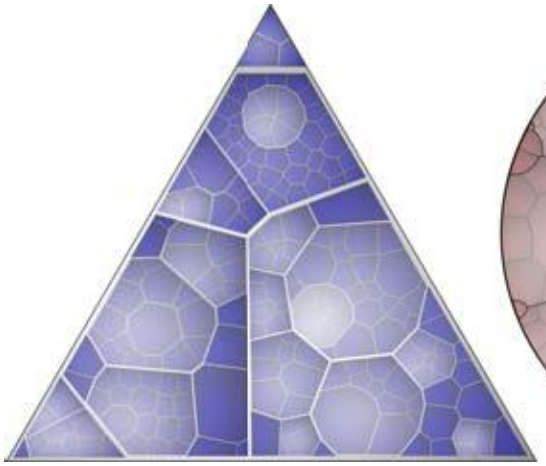
- partition the space recursively according to the size of subtrees.
- space-efficient compare to node-link diagram.
- difficult to follow parent-child relationship.

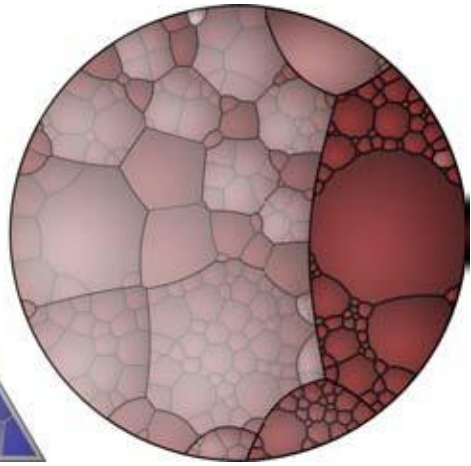


Variations of Treemap

- Cushion treemap [Wijk 00]
  - uses shading to help identify the levels in a treemap.
- Voronoi treemap
  - uses voronoi diagram as partition.



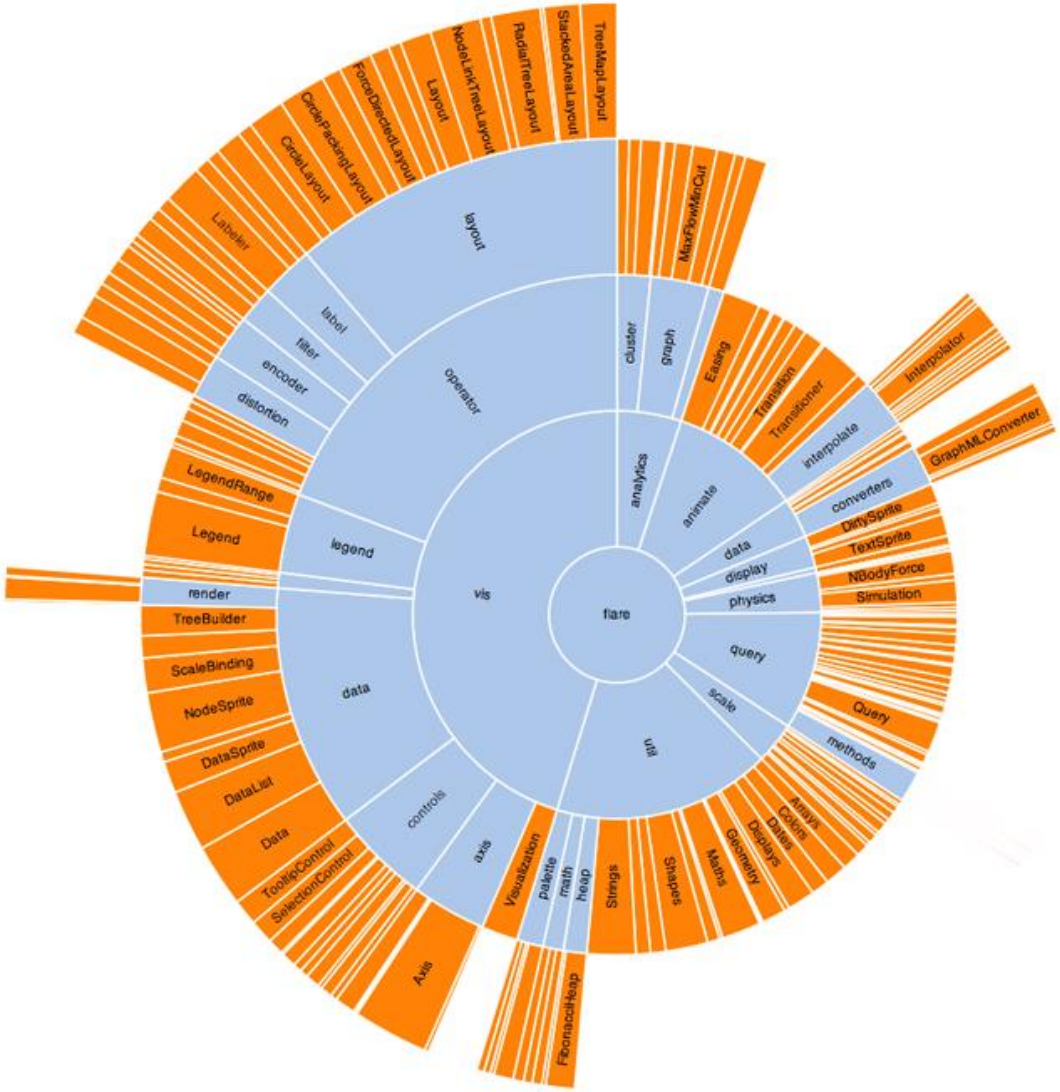




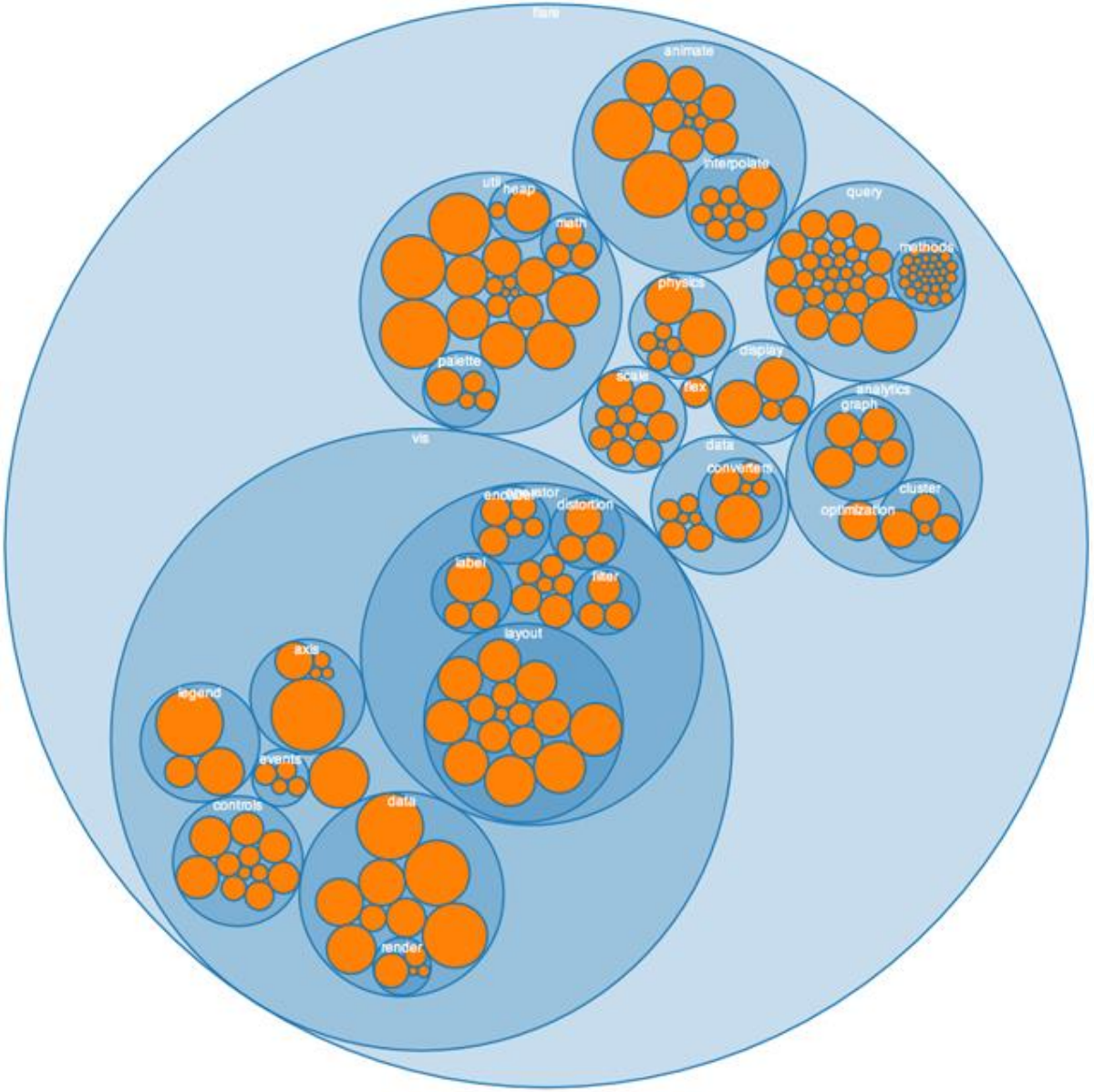
Sunburst Diagrams

[Stasko 00] space-filling visualization method

– radial version of tree map



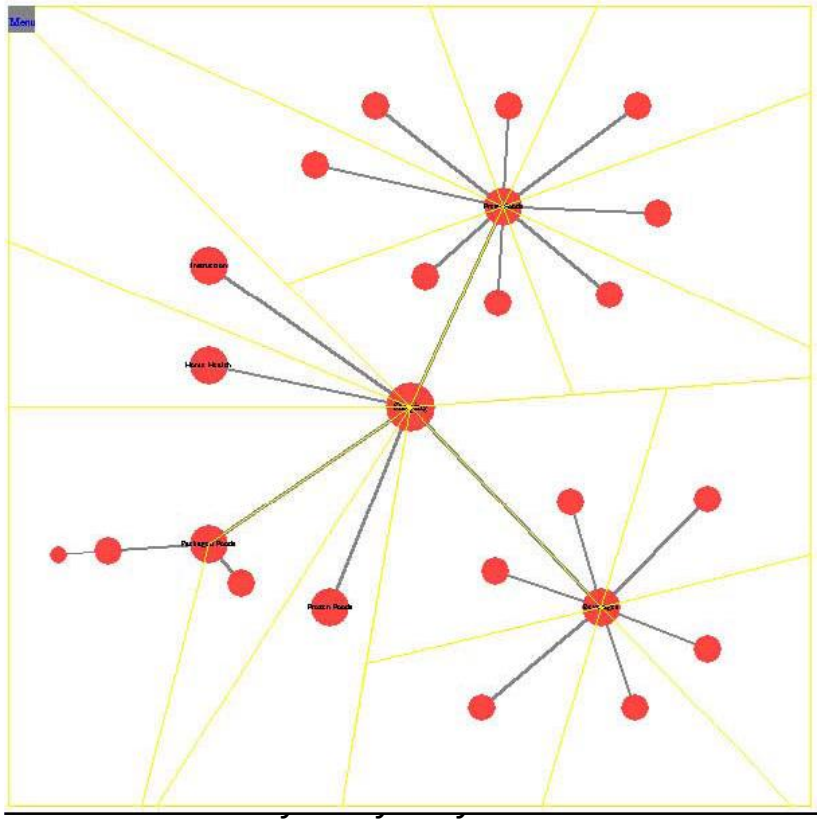
Space-filling by Circle Packing

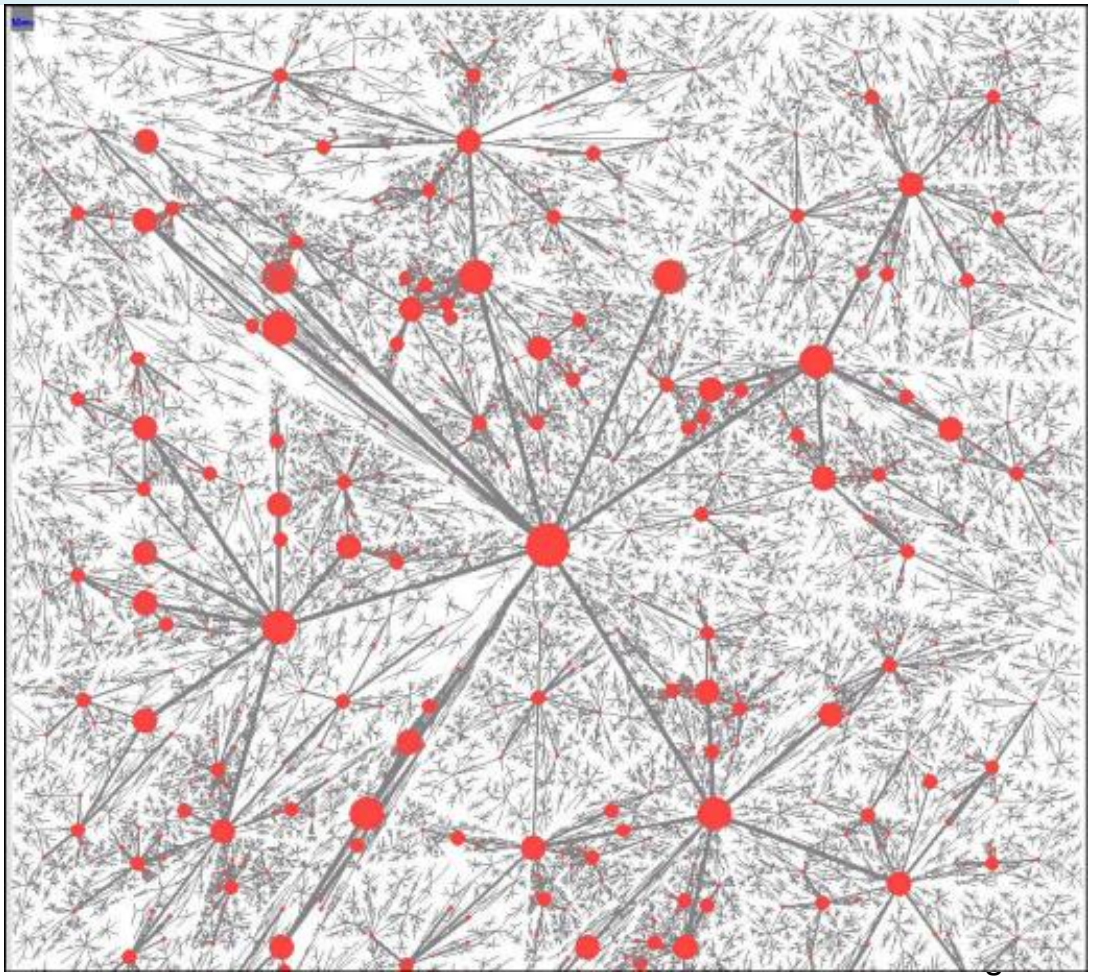


Space-filling Tree Layout

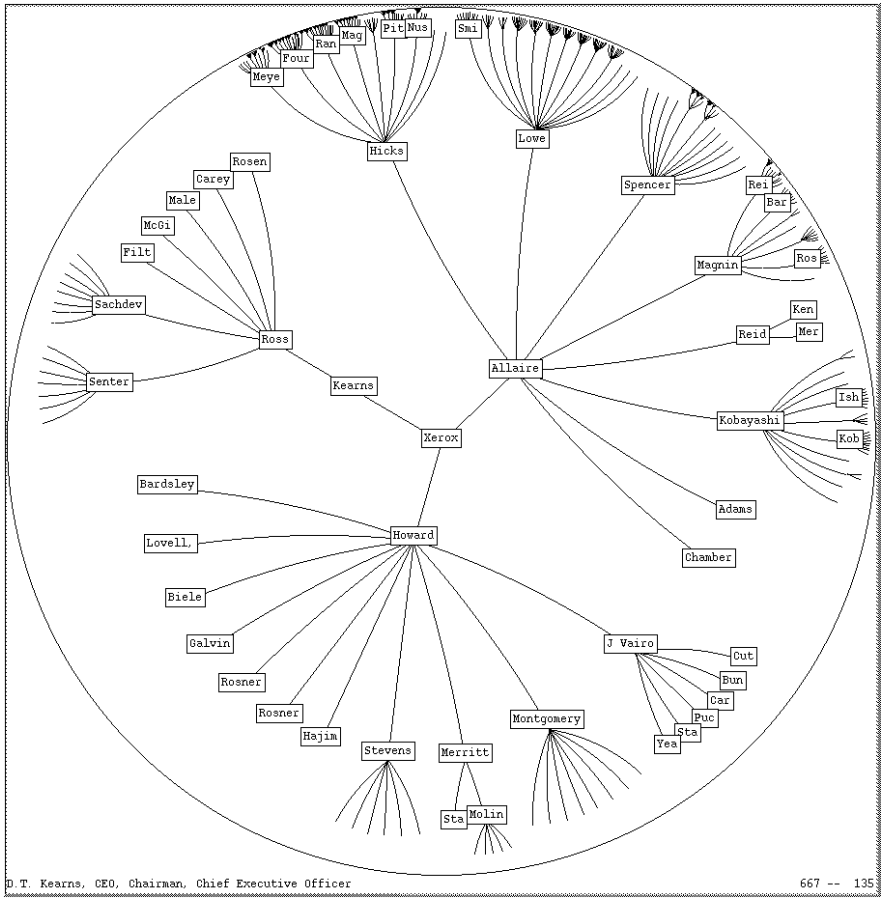
[Huang Nguyen 00]

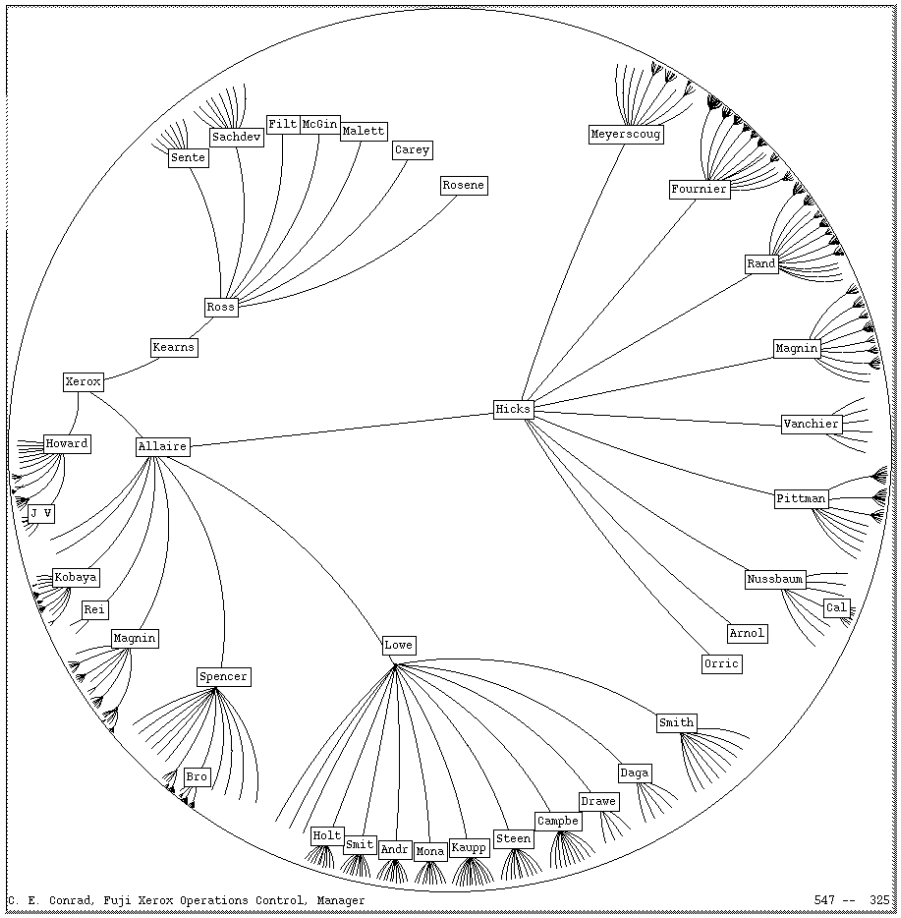
- Layout a tree according to the recursive partition of the screen space
  - area allocated to a subtree is proportional to its size
- example: 55000 nodes (use all the screen space)





Hyperbolic Tree Browser

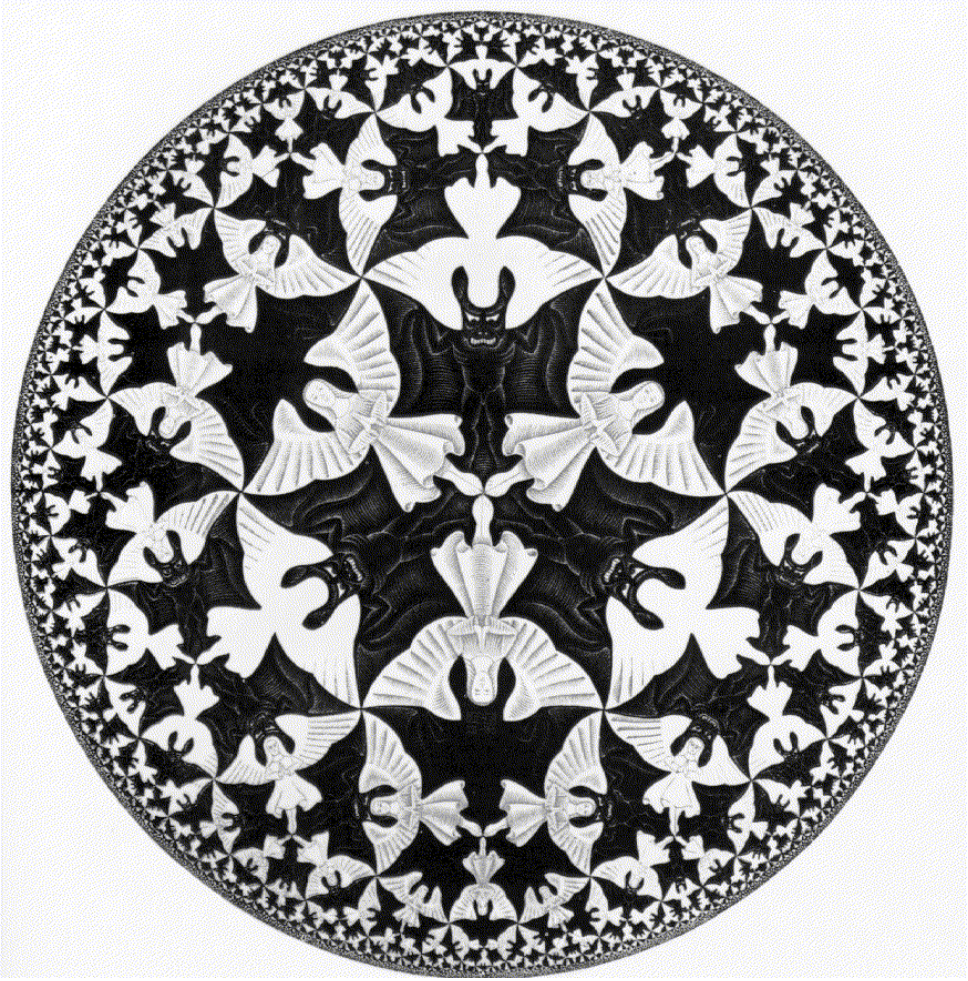


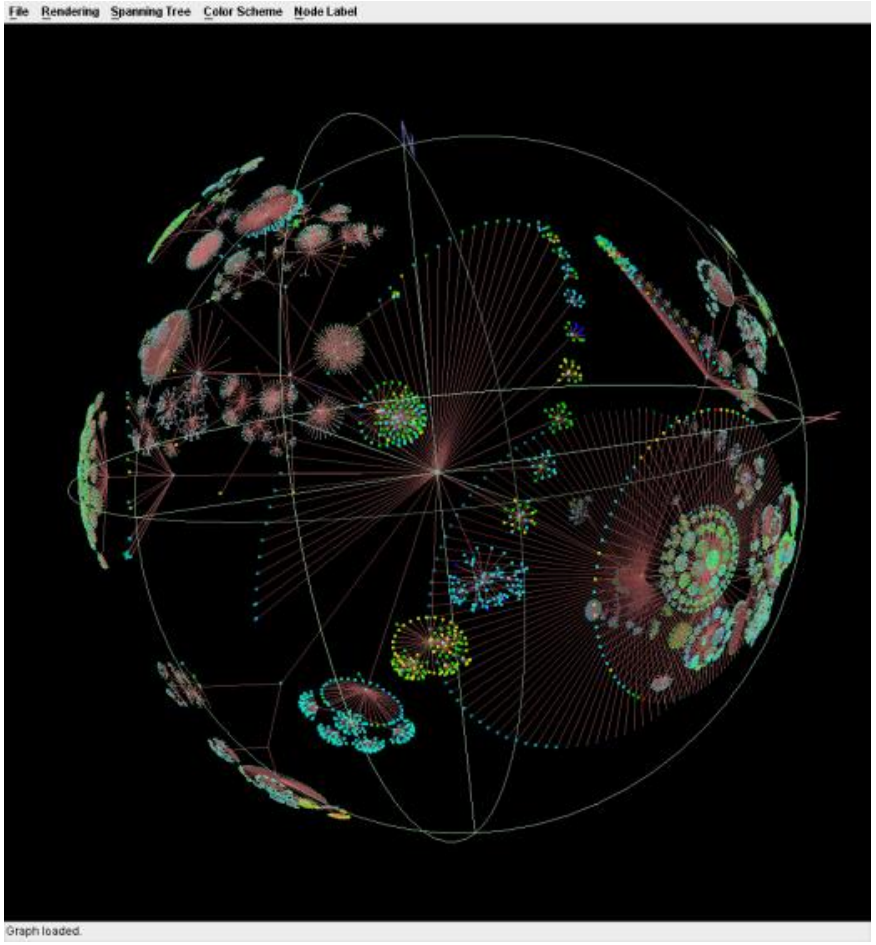


[Lamping, Rao, Pirolli 95]

- A Focus+Context Technique Based on Hyperbolic Geometry
- Distortion effect of fisheye lens
- Interaction method

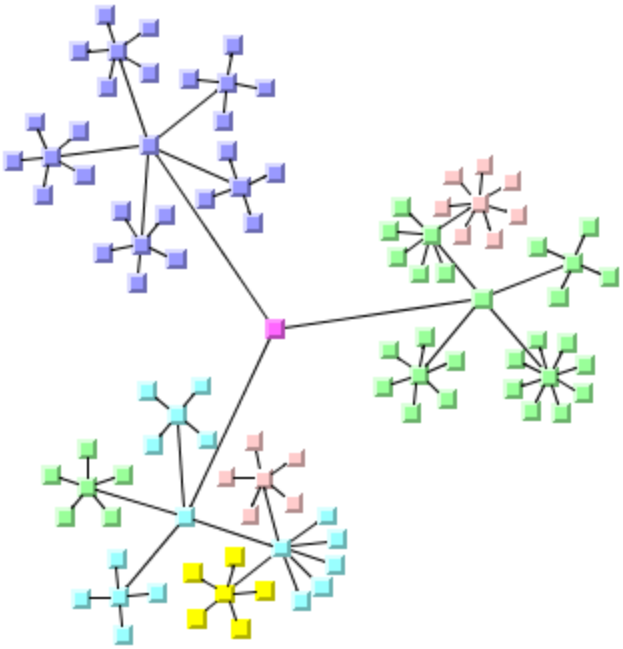
- M.C. Escher, Circle Limit IV (Heaven and Hell)
- **3D hyperbolic tree** [Munzner]
  - projecting a graph on hyperbolic sphere
  - produces a distortion effect

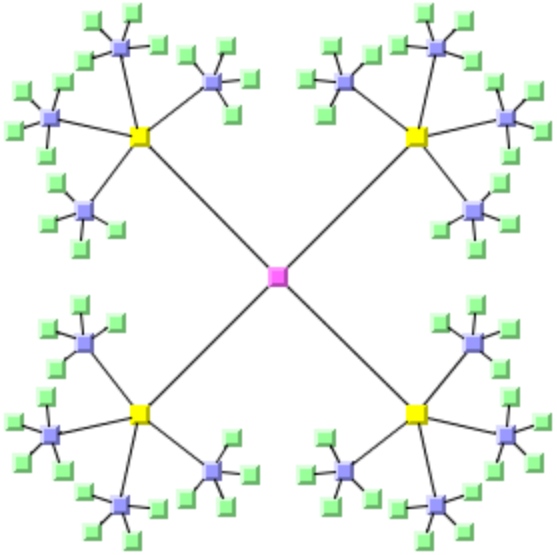


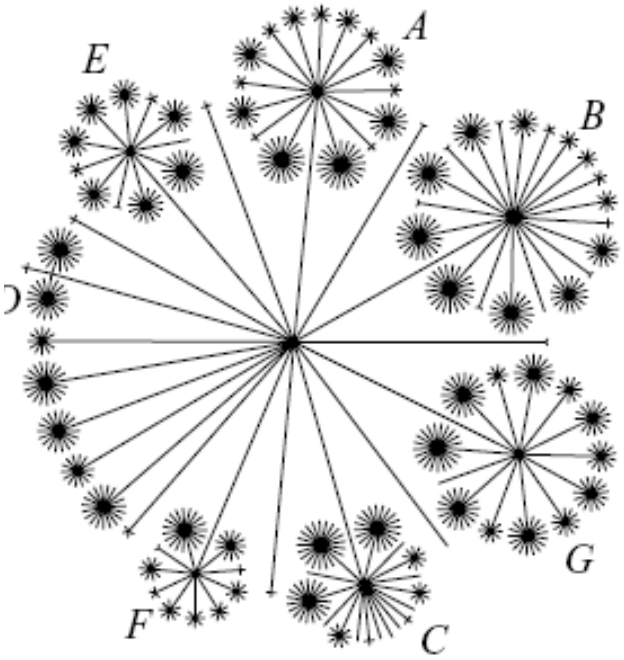


Balloon Tree Drawing

- A variation of radial layout.
- Children are drawn in a circle centered at their parents.
- [Yen 05] Deciding a good ordering: NP-hard problem



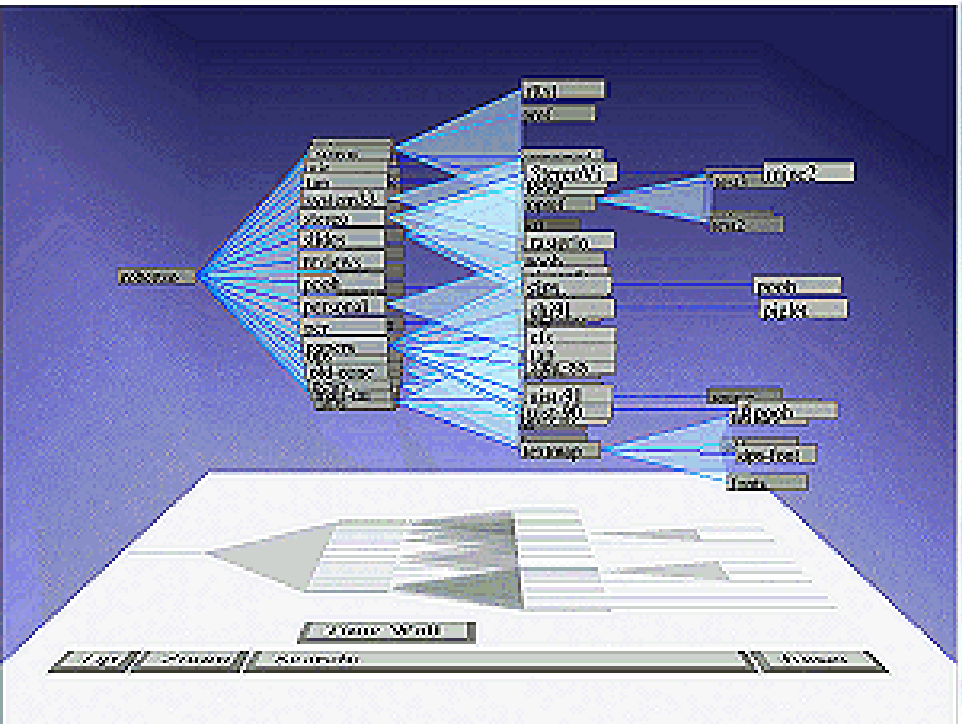


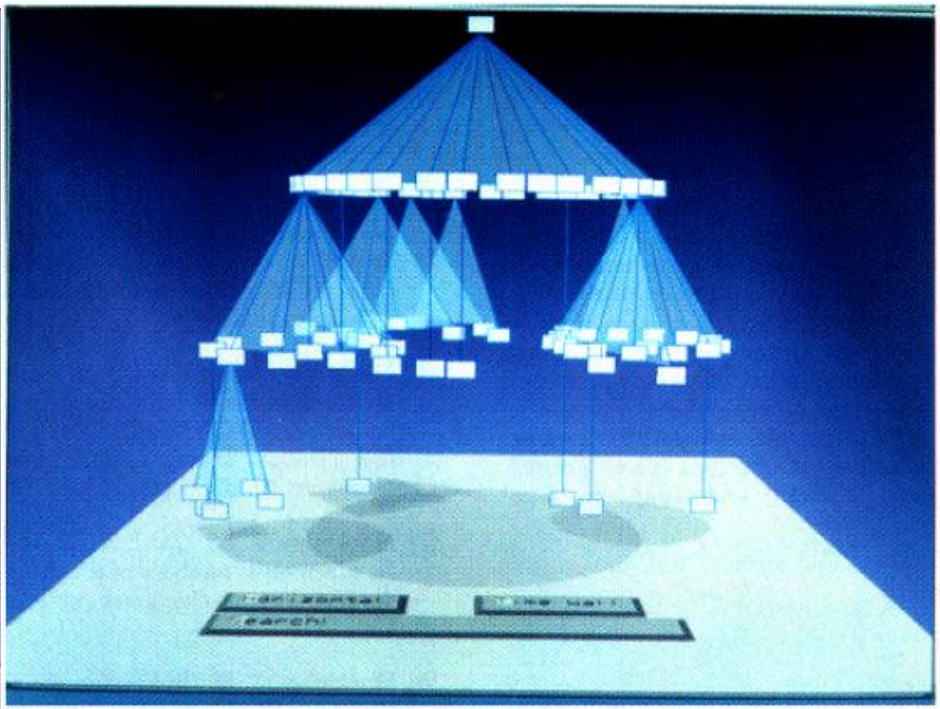


Cone Tree

[Roberston et al. 91]

- 3D extension of the 2D layered tree drawing method.
- The extension to 3D does not necessarily means more information can be displayed
  - occlusion problem
  - interaction is essential

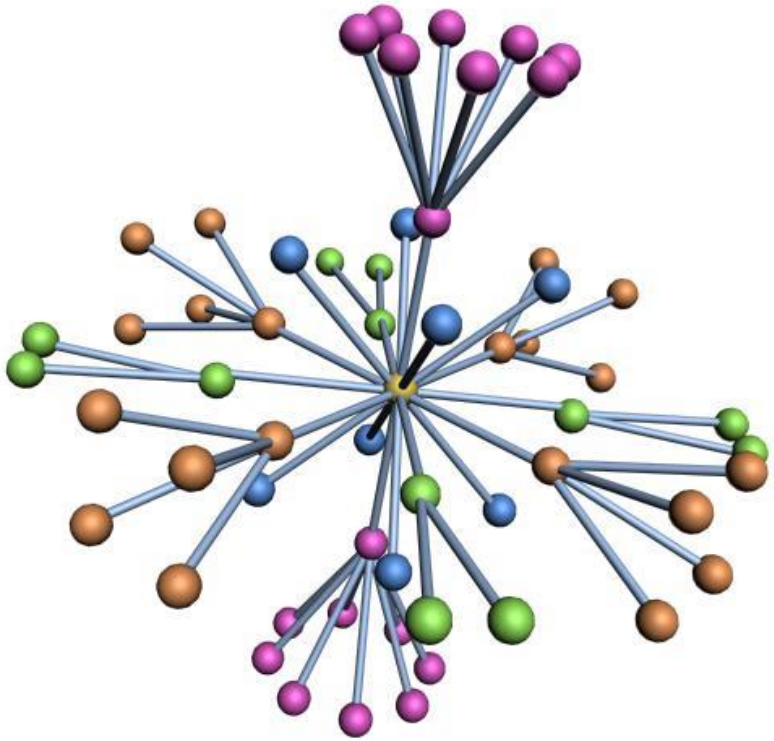


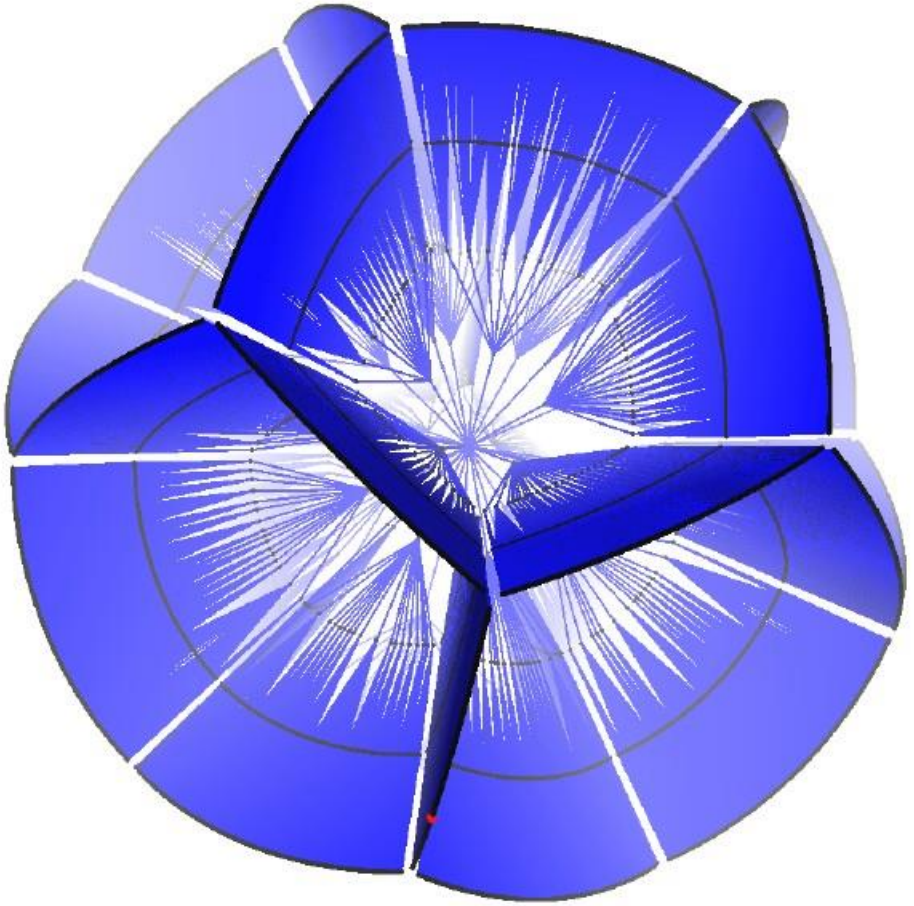


Robertson Plate 1

Polyplane

- [Hong Murtagh 04] 2.5D tree visualisation
  - Place subtrees on 2D planes
  - Divide & Conquer algorithm
  - Arrange these planes in 3D to reduce occlusion using Platonic solids





Phyllo Tree

[Carpendale 06]

- use of nature's phyllotactic patterns
- optimal packing effect

Collapsible Cylindrical Tree

- Telescope metaphor: a set of nested cylinders
  - A cylinder is constructed for the children of a node with a smaller radius.
  - It can be pulled out to the right of the parent cylinder or collapsed.

- Only one path of the hierarchy is visible at once

Botanical Tree

- Resembles botanical trees [van Wijk]
  - The root is the tree stem.
  - Non-leave nodes are branches .
  - Leave nodes are “bulbs” at the end of branches.

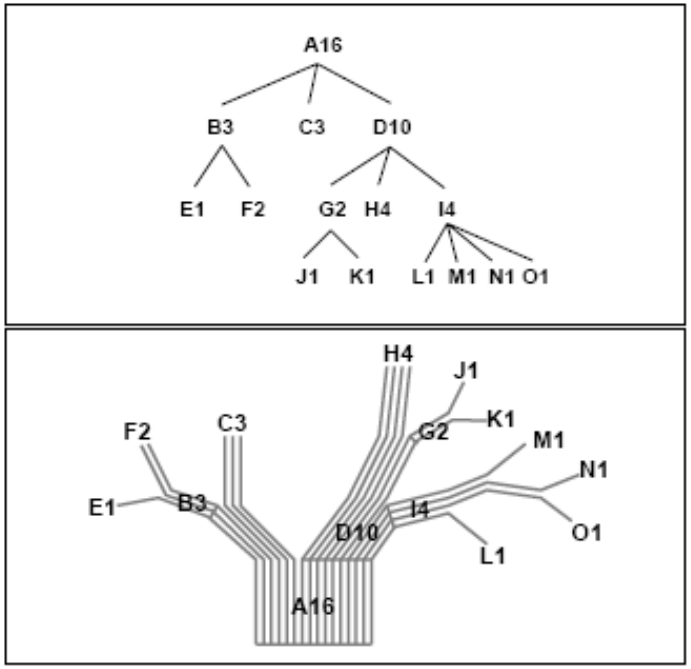
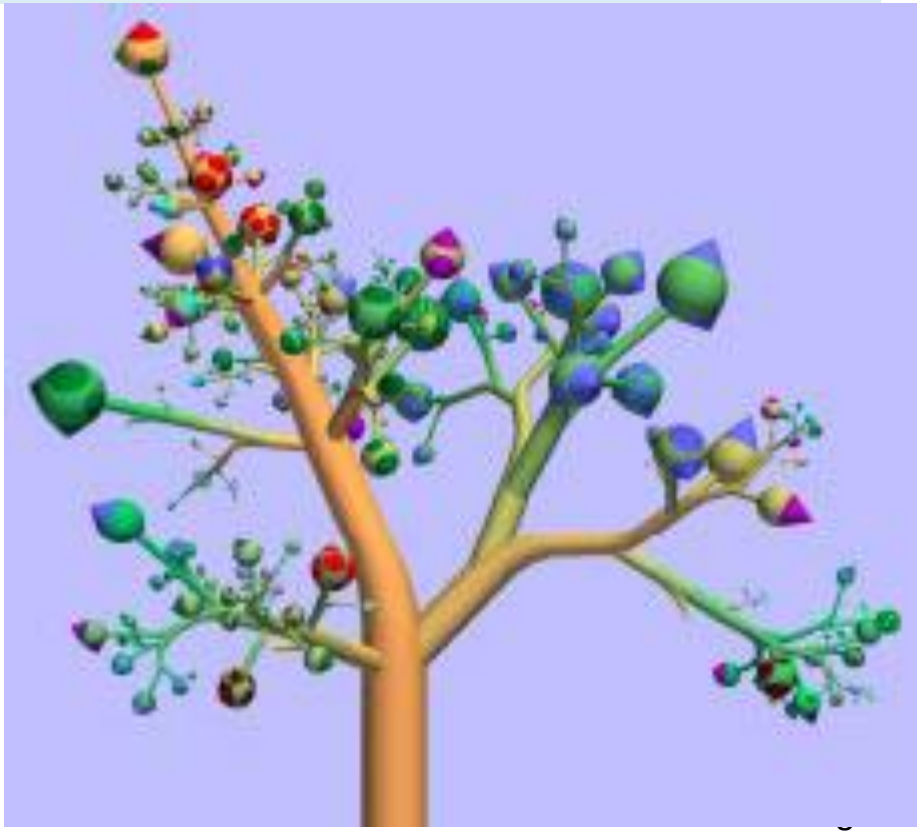


Figure 2. Node and link diagram (t) and corresponding strands model (d).



Summary: Tree Drawing/Visualization

- There are many variations based on
  - edge representation
  - criteria

- Divide and Conquer algorithm
- mostly run in linear time (O(n) time)

- most popular methods
  - tidier tree drawing
  - treemap
  - dendrogram

A Visual *Survey* of *Tree Visualization*

http://vcg.informatik.uni-rostock.de/~hs162/treeposter/poster.html

Homework

- Tutorial/Homework

- Assumed Knowledge

basic terminology on graphs
  - degree
  - path
  - cycle
  - shortest path

- Reference:

"Graph Drawing: Algorithms for the Visualization of Graphs"
by Di Battista, ***Eades***, Tamassia, Tollis
  - Week 2: chapter 3
  - Week 3: chapter 10
  - Week 4: chapter 9