MTTTS17
Dimensionality Reduction and Visualization

Spring 2020
Jaakko Peltonen

Lecture 12:
Visualization and navigation of graphs - Hairballs and Beyond

Slides originally by Francesco Corona and Manuel J.A. Eugster

Part 1:

Graph-structured data

Structured data and graphs

A simple way to determine the applicability of graph visualizations is to consider the following question

Is there an inherent relation among the data elements to be visualized?

If the answer is "**no**", then the data are "**unstructured**"

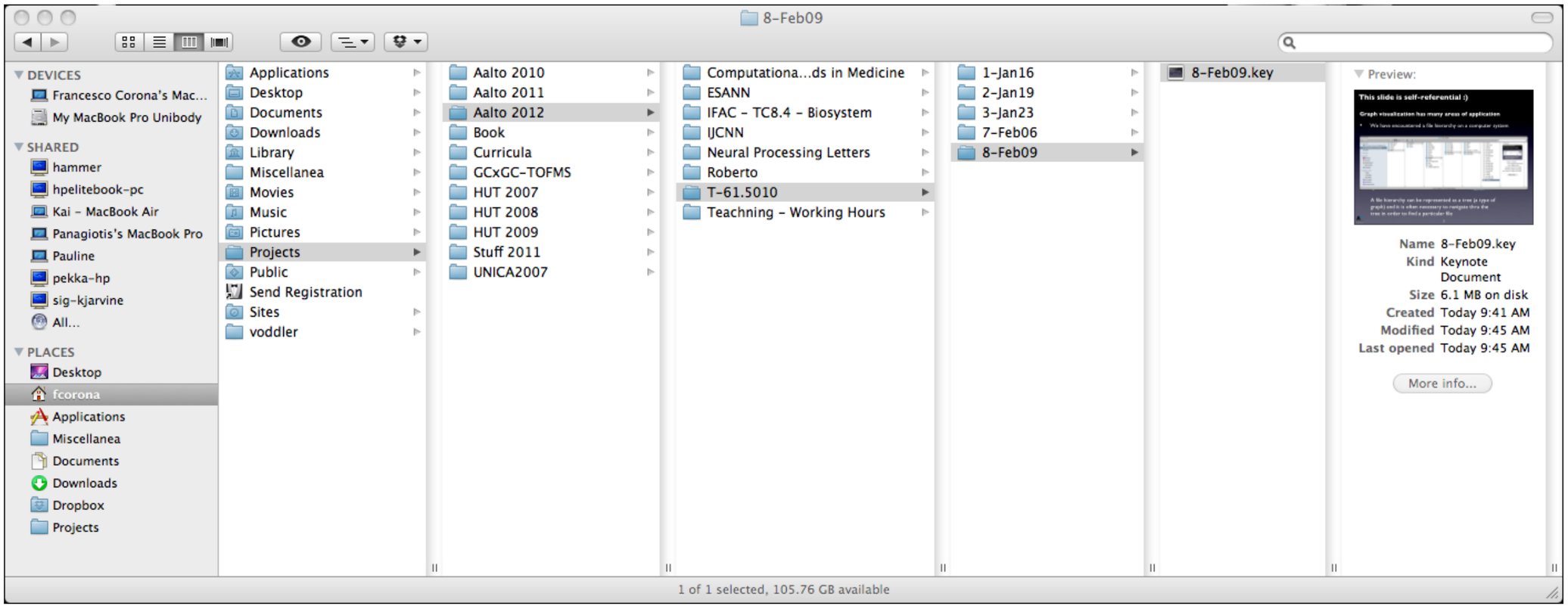
One of the goals of visualization might be to help discover relations among data thru visual means

If the answer is "**yes**", then the data are "**structured**"

The data can be represented by the nodes of a graph with the edges indicating the relations

Graph visualization has many areas of application

- We have encountered a file hierarchy on a computer system



A file hierarchy can be represented as a tree (a type of graph) and it is often necessary to navigate thru the tree in order to find a particular file

Structured data and graphs (cont.)

- Other familiar types of graphs are organizational charts, website maps as well as history browsing, ...
- In biology and chemistry, graphs are applied to molecular maps, evolutionary trees, biomedical pathways, ...
- In dynamic systems analysis, graphs are used to represent Petri nets and automata ...

For structured data, graphs are the fundamental structural representation of the data

Structured data and graphs (cont.)

The size of the graph to view is a key issue in graph visualization, large graphs pose problems

- If the number of elements is large, it can compromise performance and reach the viewing limits of the platform

There is an interface problem

- While the size of the structures we work with grows, the "windows" through which we look at them remains small

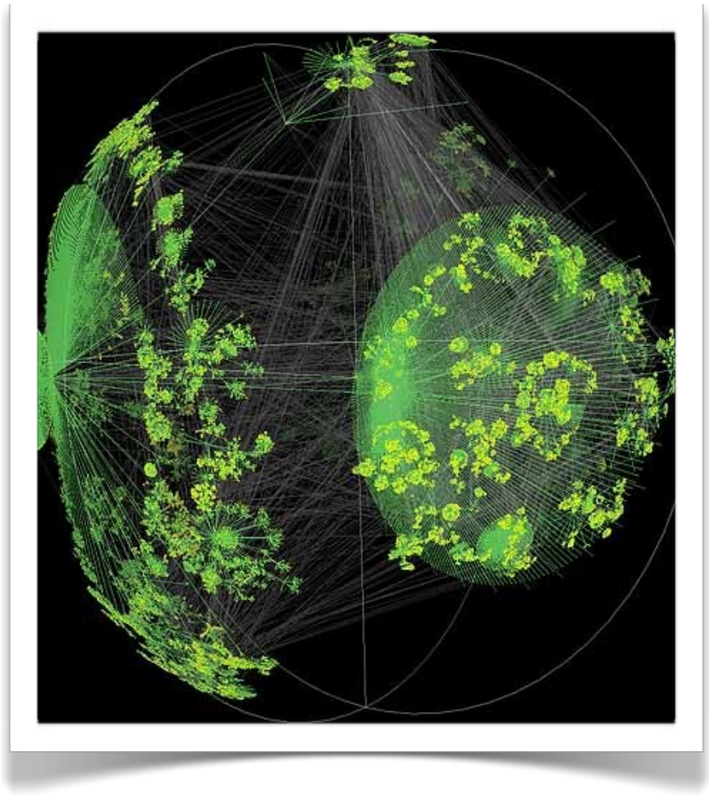
Structured data and graphs (cont.)

Even when it is possible to layout and display all the elements, the issue of *viewability* or *usability* arises

- It will become impossible to discern between nodes and edges

It is not uncommon that usability becomes an issue even before the problem of *discernability* is reached

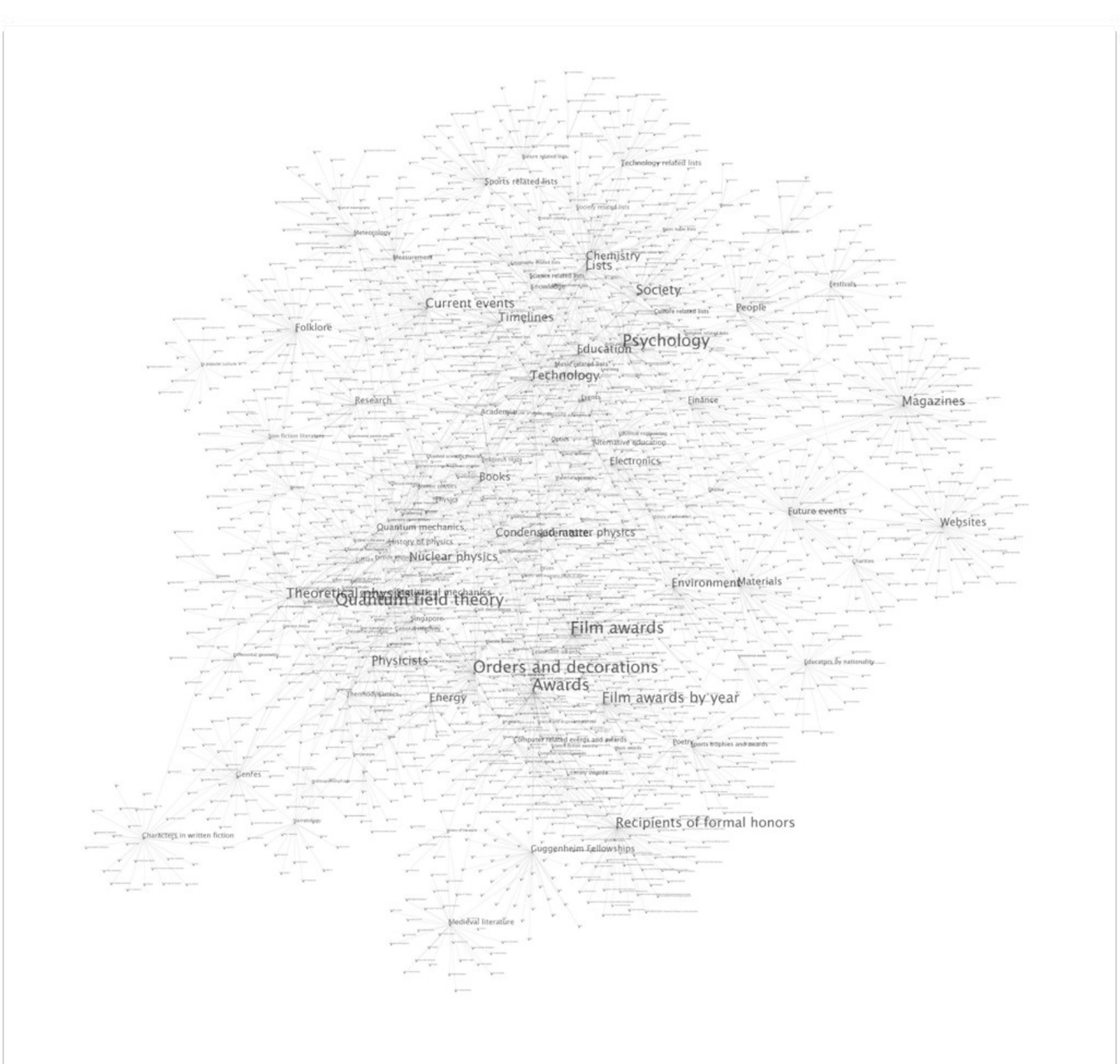
In general, displaying an entire large graph may give an indication of the overall structure, but it makes it difficult to comprehend it



Structured data and graphs (cont.)

Chris Harrison (2006): WikiViz (v3)

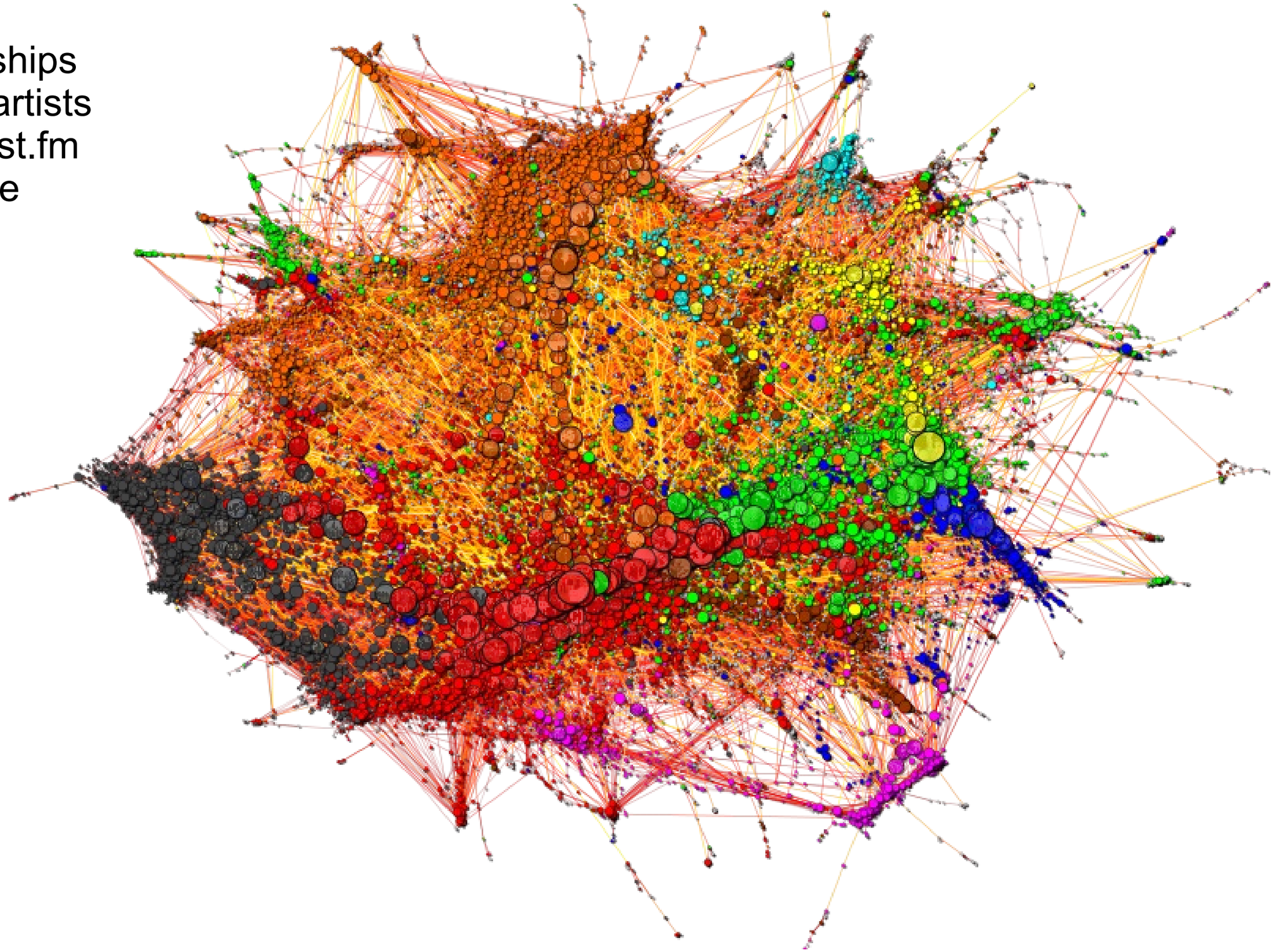
Visualization of wikipedia articles up to three levels deep in the hierarchy, centered on physics



http://www.chrisharrison.net/index.php/Visualizations/WikiViz

Structured data and graphs (cont.)

relationships among artists in the last.fm database



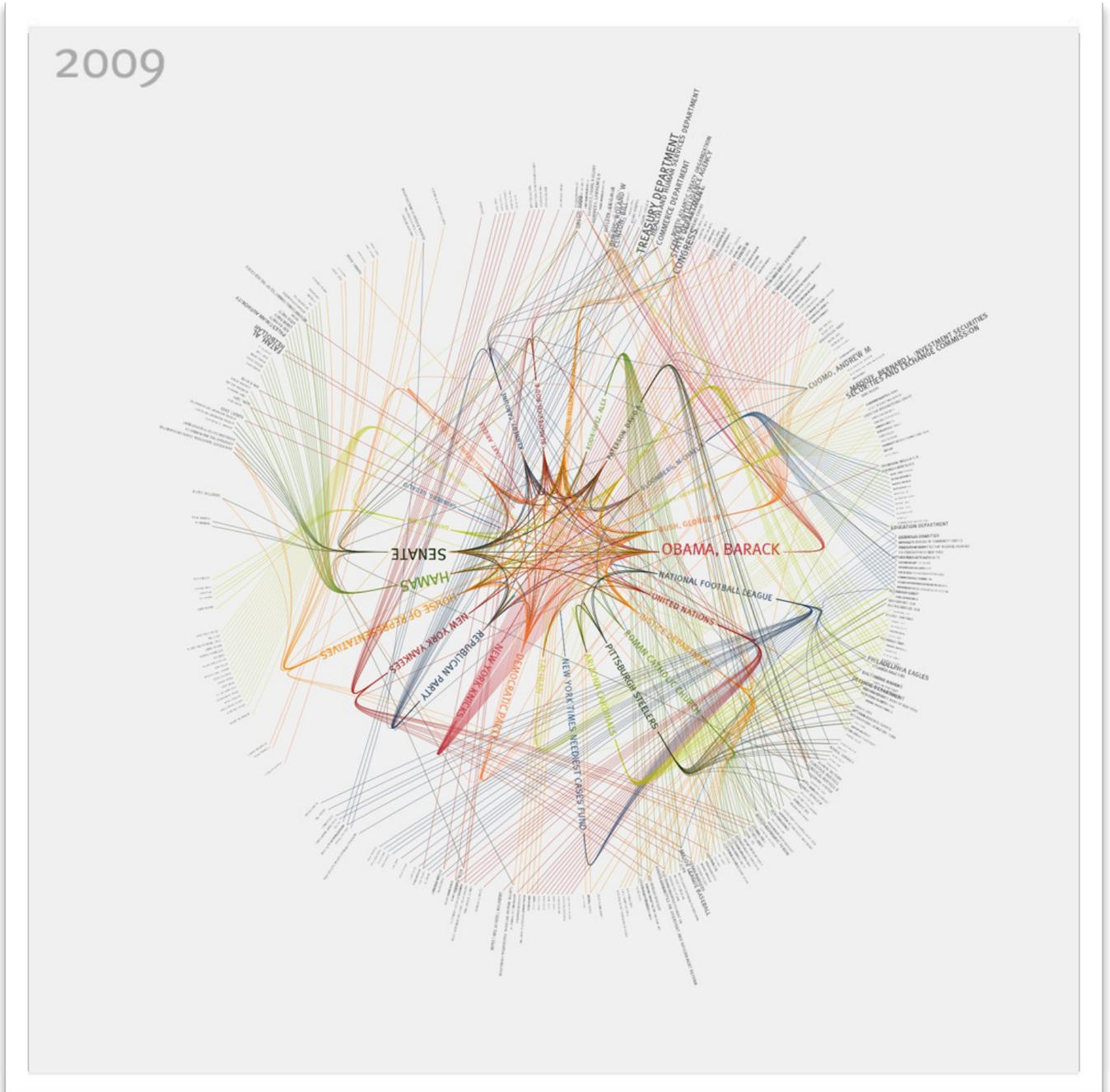
http://drewconway.com/zia/2013/3/29/the-value-of-edges-in-complex-network-visualization

Structured data and graphs (cont.)

Jer Thorp (2009): NYTimes: 365/360

New York Times stories from 2009

Probably nodes = entities (people/organizations) mentioned, edges = articles mentioning both entities



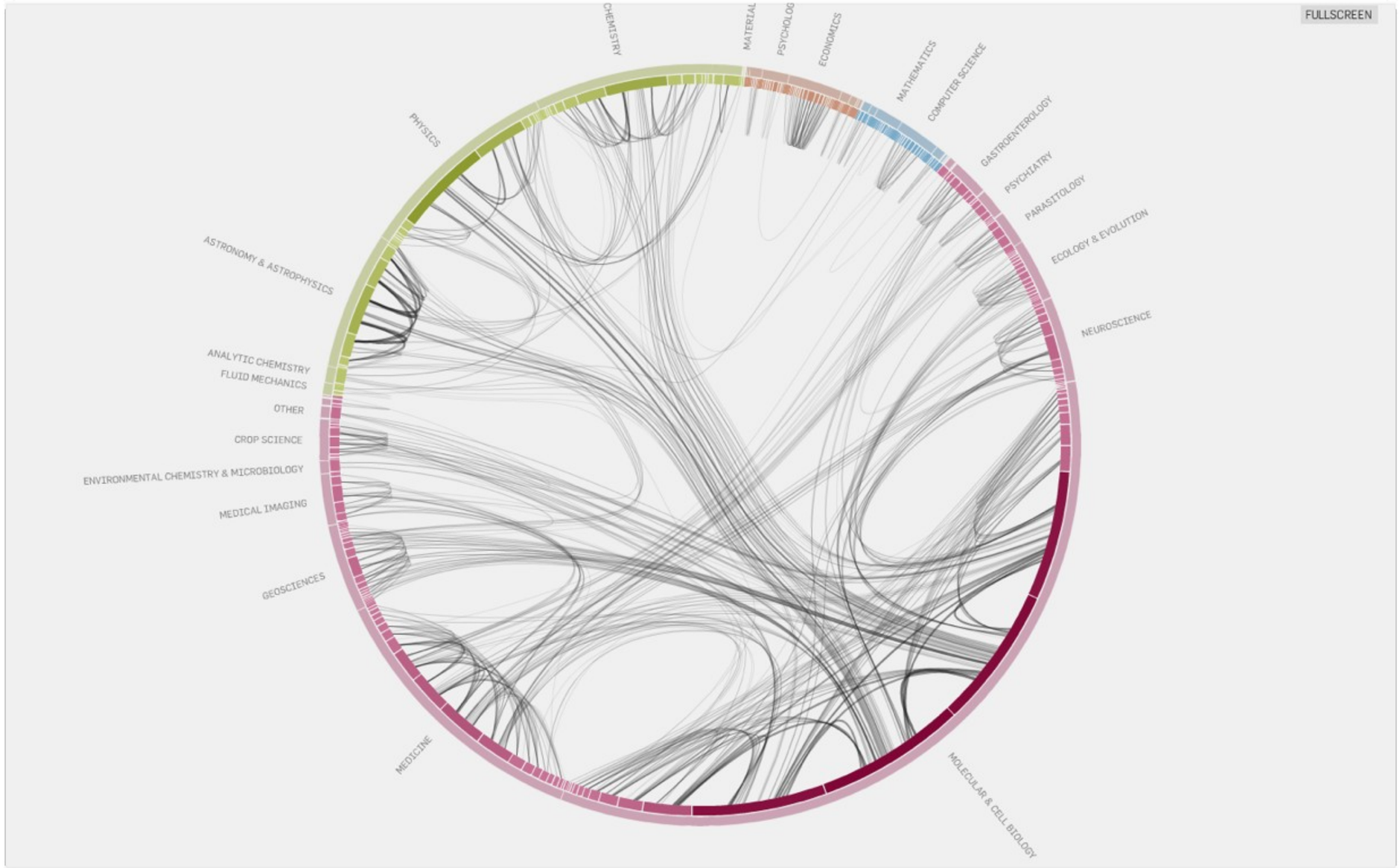
http://blog.blprnt.com/blog/blprnt/7-days-of-source-day-2-nytimes-36536

Structured data and graphs (cont.)

Moritz Stefaner (2009): Citation patterns

Nodes on the circumference = journals arranged by field,

edges = Amount of citations between different journals.



http://weloveinfographics.info/post/11903980187/citation-patterns

Structured data and graphs (cont.)

Comprehension and analysis of data in graph structures is easiest when the size of the displayed graph is small

The interface design problem amounts to deciding what parts of a large structure to show and how

- Graph layout issues and limitations with regard to scaleability
- Approaches to navigation of large graphs

Today: An impression of graph drawing and layout algorithms from the point of view of information visualization.

Next week: a few important layout algorithms in detail

Part 2:
Graphs and their properties

Graph: definition

A graph G = (V, E) is a set of vertices (nodes) V together with a set of edges (lines) E; where E consists of two-element subsets of V.

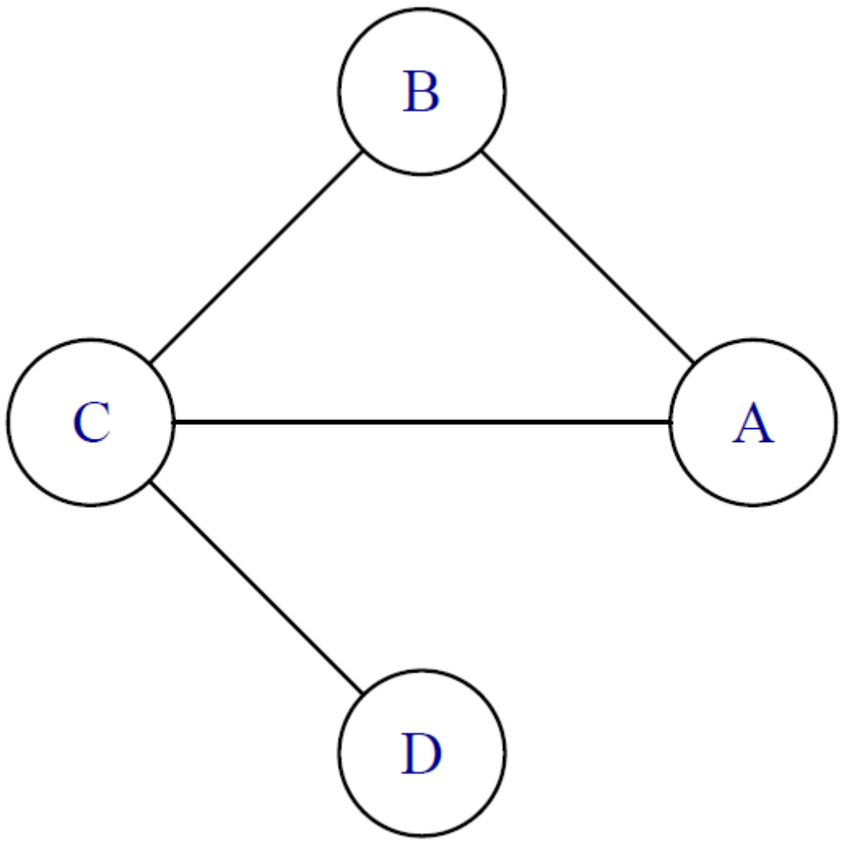
For example:

- V = {A, B, C, D}
- E = {{A, B}, {A, C}, {B, C}, {C, D}}

(*) see, e.g., Easley and Kleinberg (2010)

Graph: definition - example

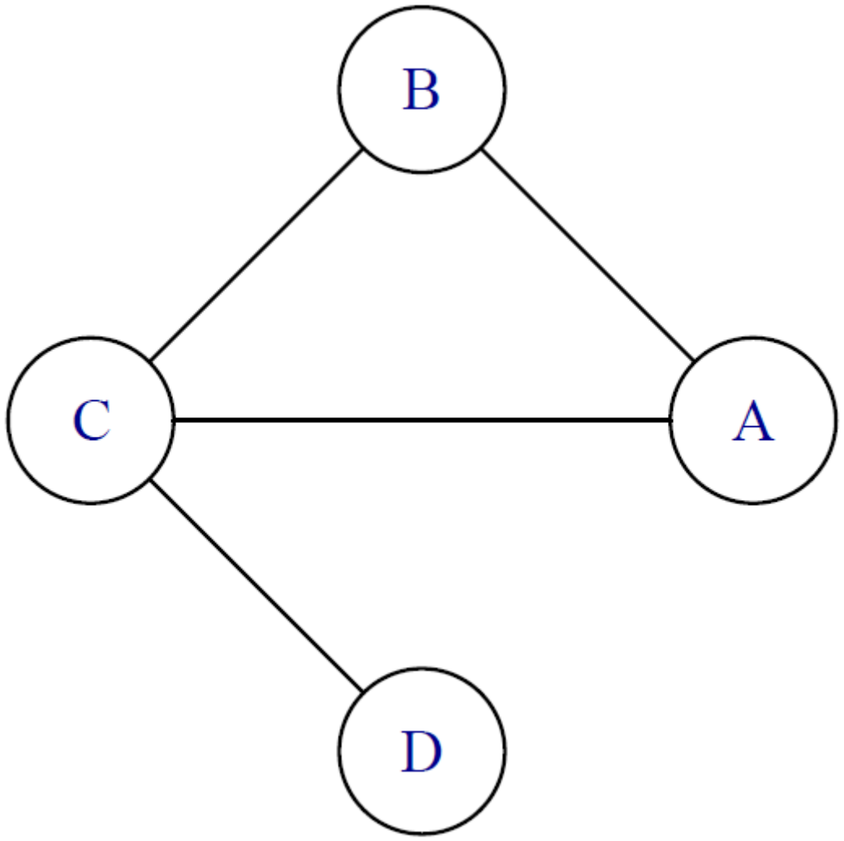
V = {A, B, C, D} E = {{A, B}, {A, C}, {B, C}, {C, D}}



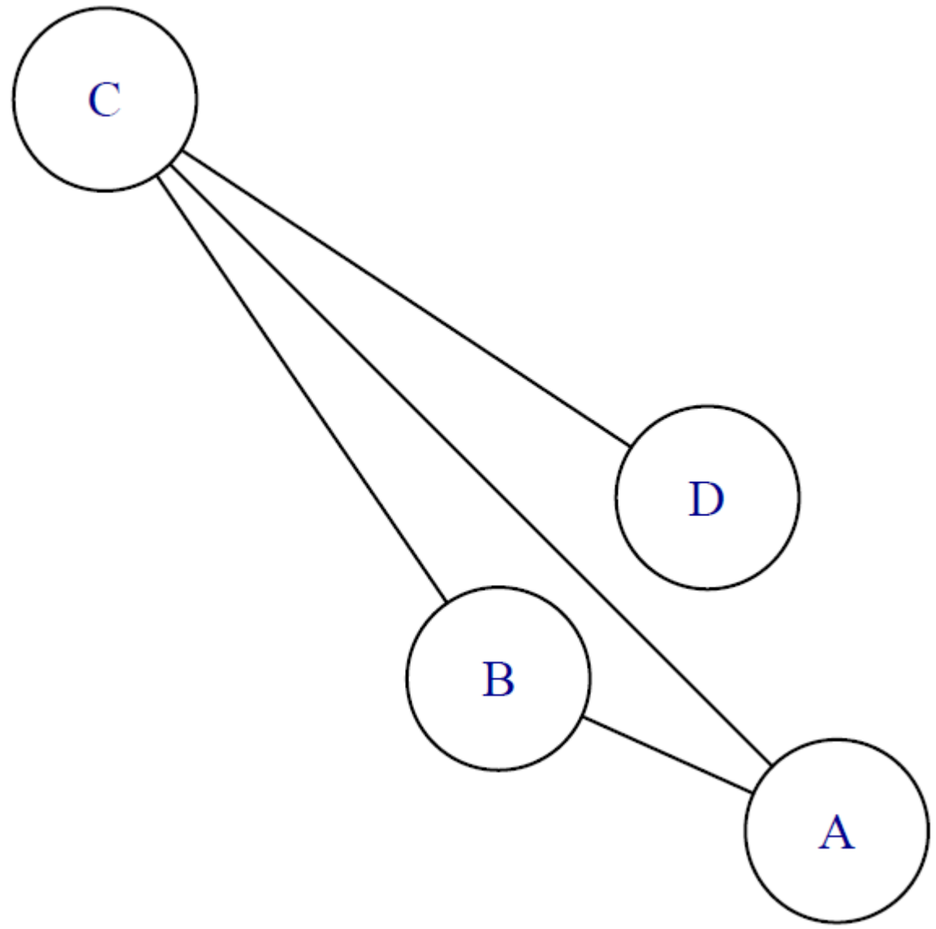
One way to draw the graph

Graph: definition - example

V = {A, B, C, D} E = {{A, B}, {A, C}, {B, C}, {C, D}}



One way to draw the graph



Another way to draw the same graph

Graph: definition, continued

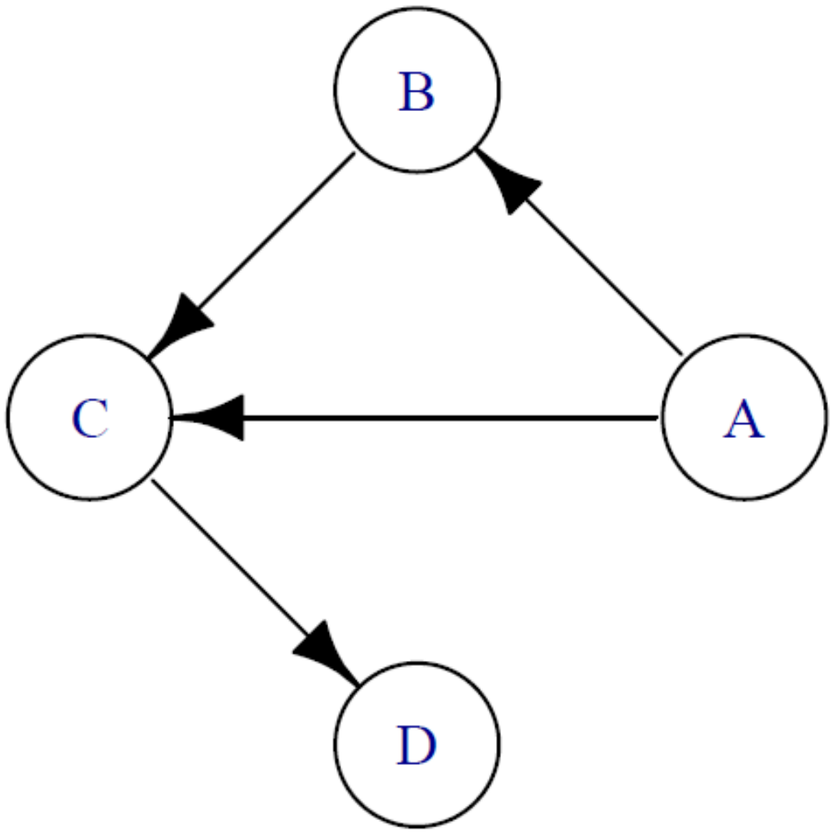
Undirected or directed edges:

In an undirected graph the edges have no orientation, i.e., whenever $\{A, B\} \in E$ it is implied that $\{B, A\} \in E$.

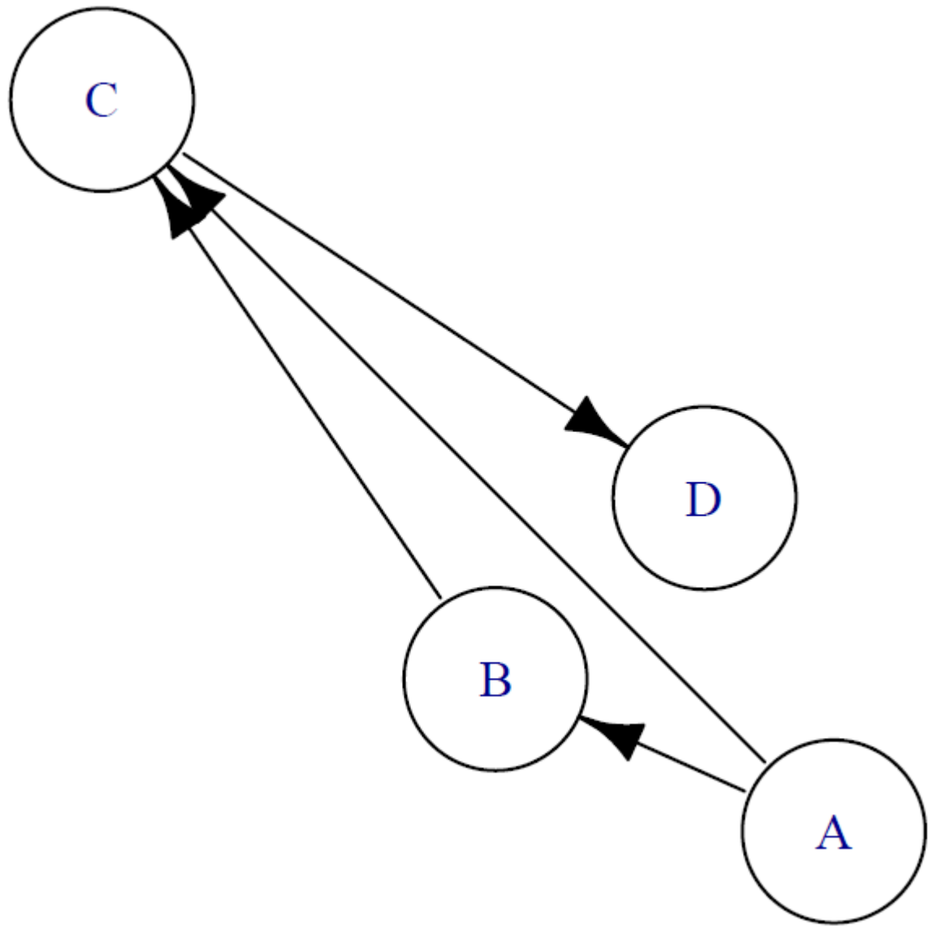
In an directed graph the edges have an orientation, i.e., the edges are ordered pairs where $(A, B) \in E$ means that there is a directed arc from the node A to the node B.

Graph: definition - example of directed edges

V = {A, B, C, D} E = {(A, B), (A, C), (B, C), (C, D)}



One way to draw the graph



Another way to draw the same graph

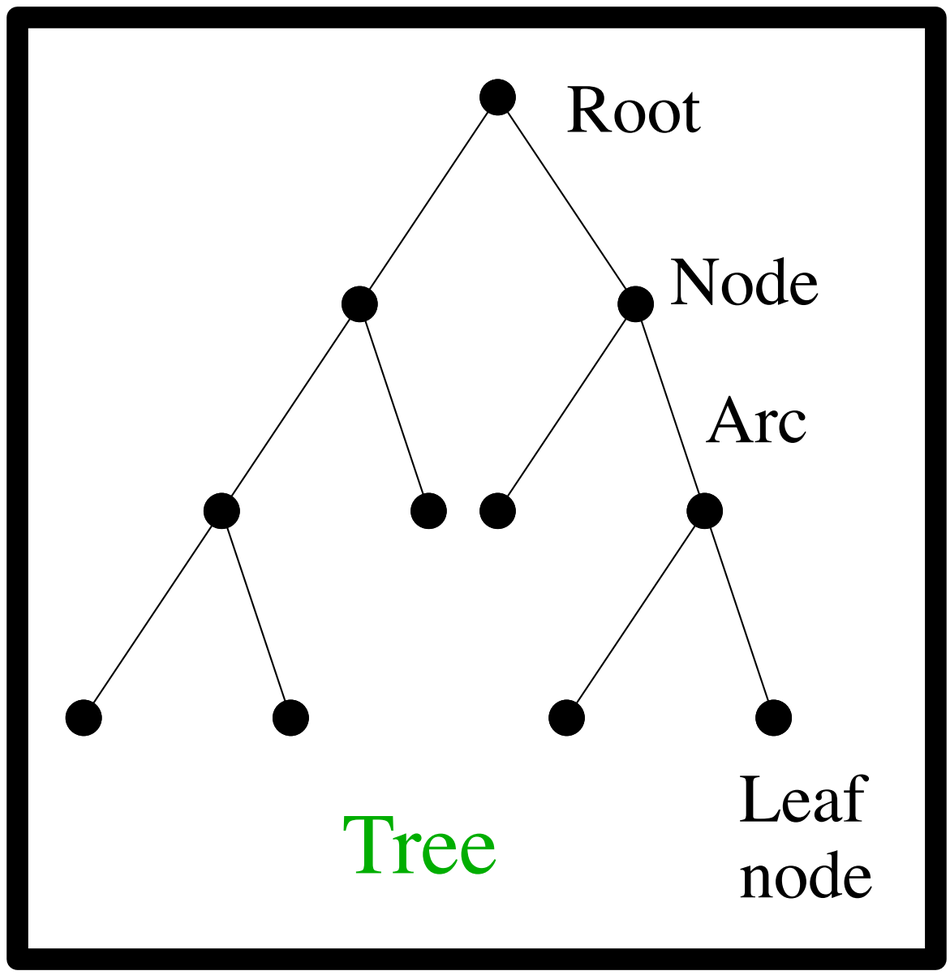
Simple Graph Properties

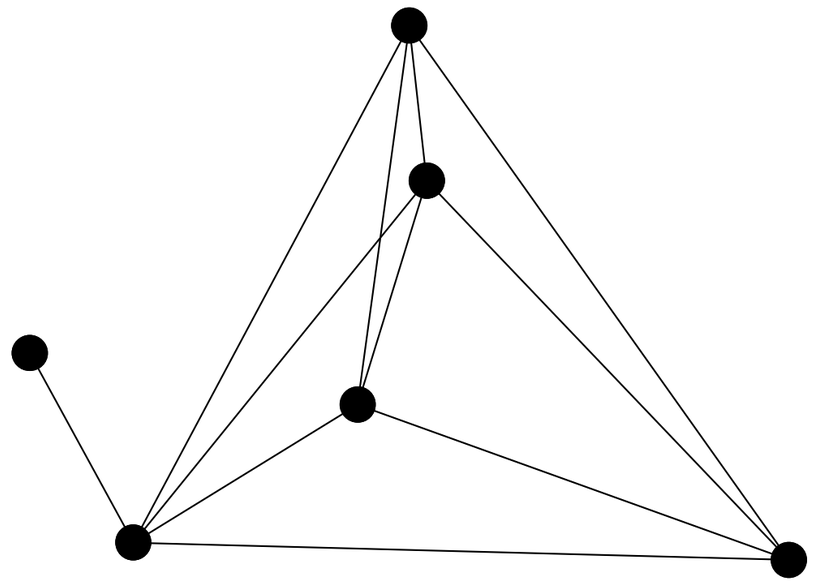
order: the number of vertices |V|
size: the number of edges |E|
path: a sequence of edges which connect a sequence of vertices
cycle: a path starting and ending at the same vertex
shortest path: a path between two vertices with minimal (weighted) number of edges
distance: the number of edges in a shortest path between two vertices

See, e.g., Wikipedia for a list of graph properties.

Small-World experiment/Six degrees of separation, Bacon number are examples of applied graph theory.

Types of graphs: trees vs. networks





Graph labeling/attributes

G = (V, E) is the abstract representation of the graph, all that matters are the pairwise relations, i.e., which vertices are connected by edges.

In terms of application (social network analysis, information visualization, etc), the abstract representation of the graph has to be connected to *data*!

Attributes

Attributes for vertices and edges allow to connect the graph components to data (loose definition):

$$v^{A}_{\text{attr}} : V \to V_A$$

$$e^{B}_{\text{attr}} : E \to E_B$$

A, B the specific attributes; V_A, and V_B the corresponding set of possible attribute values.

For example: names of people for the vertices, money flow between two people for edges (i.e., the direction of an edge can also be seen as an attribute).

US airport network (2010 December)

Passenger flights between airports in the United States. The dataset was compiled based on flights in 2010 December and is available in the **igraphdata** R-package.

743 vertices:

	name	City	State	Latitude	Longitude
1	BGR	Bangor	ME	44.80	-68.84
2	BOS	Boston	MA	42.35	-71.00
3	ANC	Anchorage	AK	61.17	-150.01
4	JFK	New York	NY	40.63	-73.78
5	LAS	Las Vegas	NV	36.07	-115.18
6	MIA	Miami	FL	25.78	-80.30

US airport network (2010 December)

Passenger flights between airports in the United States. The dataset was compiled based on flights in 2010 December and is available in the **igraphdata** R-package.

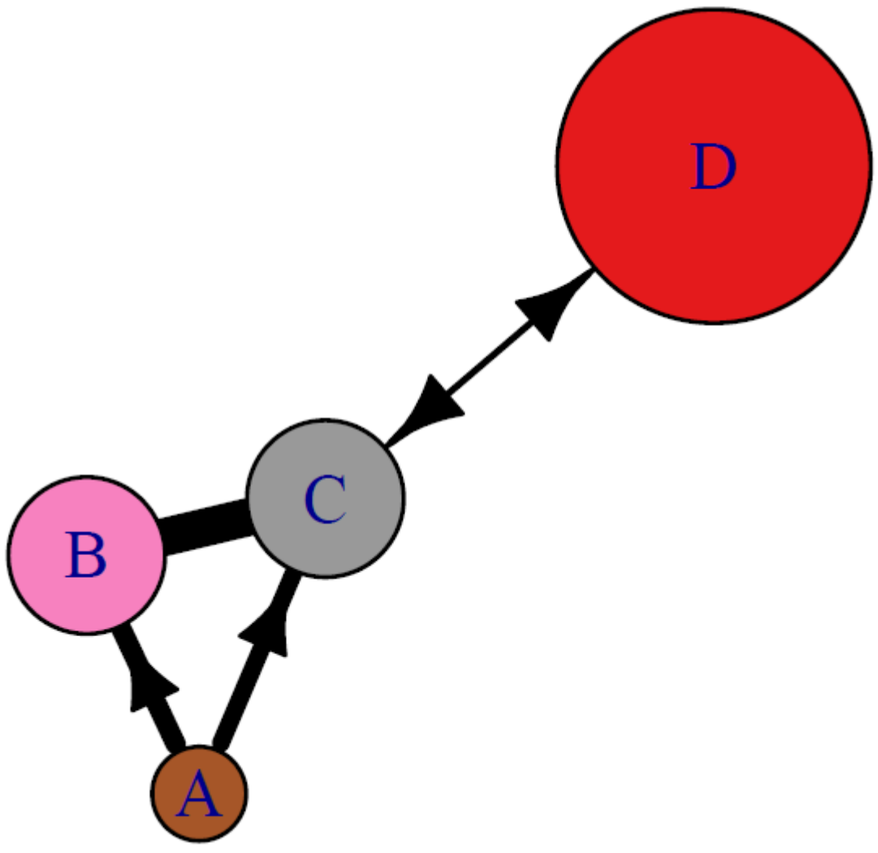
8131 directed edges:

	from	to	Departures	Seats	Passengers	Distance	Flights
1	1G4	VGT	52	988	777	79	1
2	A23	HOM	13	78	15	19	2
3	A23	PGM	1	6	1	30	1
4	A27	FAI	4	36	10	91	2
5	A29	ADQ	4	16	8	39	2
6	ABE	ATL	29	1785	1481	692	2

Part 3:
Graph drawing

Graph Drawing

Map the graph attributes to the aesthetics of geometric objects, which represent the graph.



Node-link diagram:
"When you think of a graph, you likely already think of a node-link diagram - unless you're a mathematician."
Kosara (2012)

Remember: "Everything shown by a visualization should exist in the data!" (cf. Scheidegger, 2012)

“Simple” Graph Representations

1. Combinatorial description

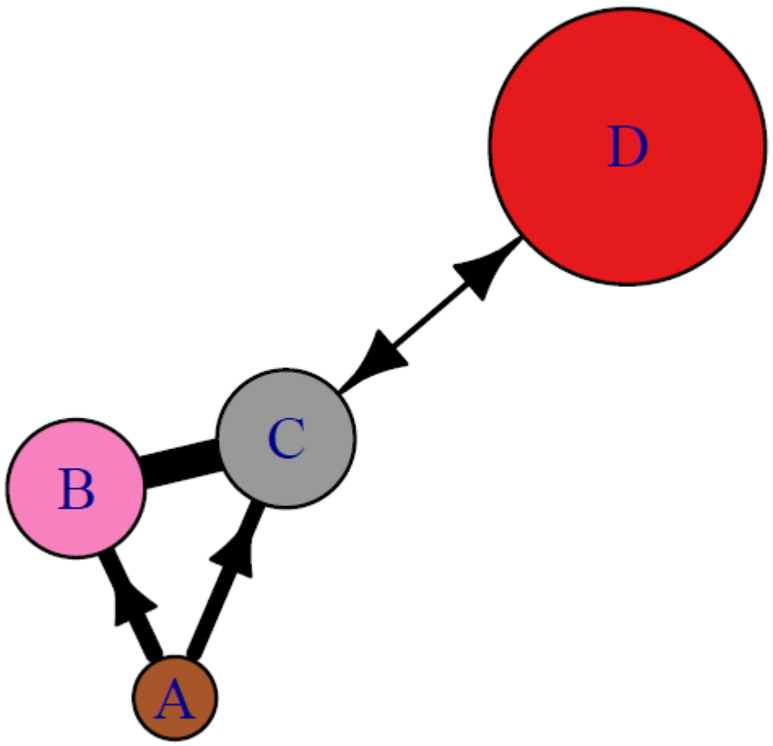
2. Edge list

3. Adjacency matrix

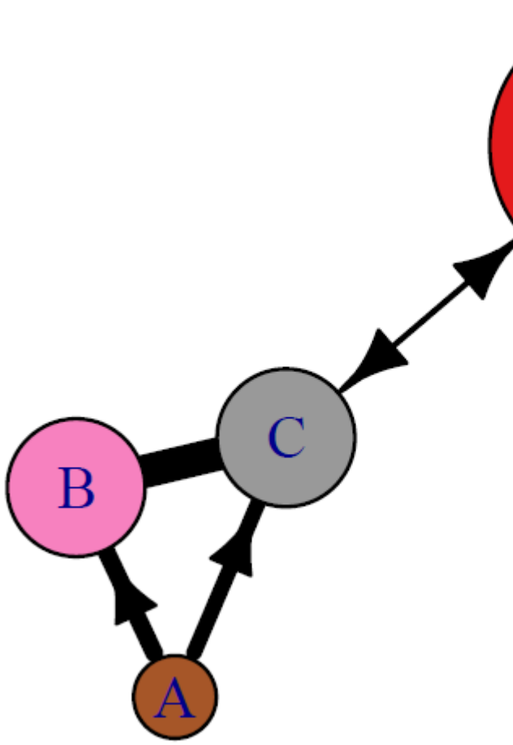
4. Incidence matrix

These representations only require knowledge of the nodes and edges, and an order to show them. In contrast, visual drawing of graphs like node-link diagrams require an on-screen location for every node, and a line or curve for every edge.

Adjacency Matrix



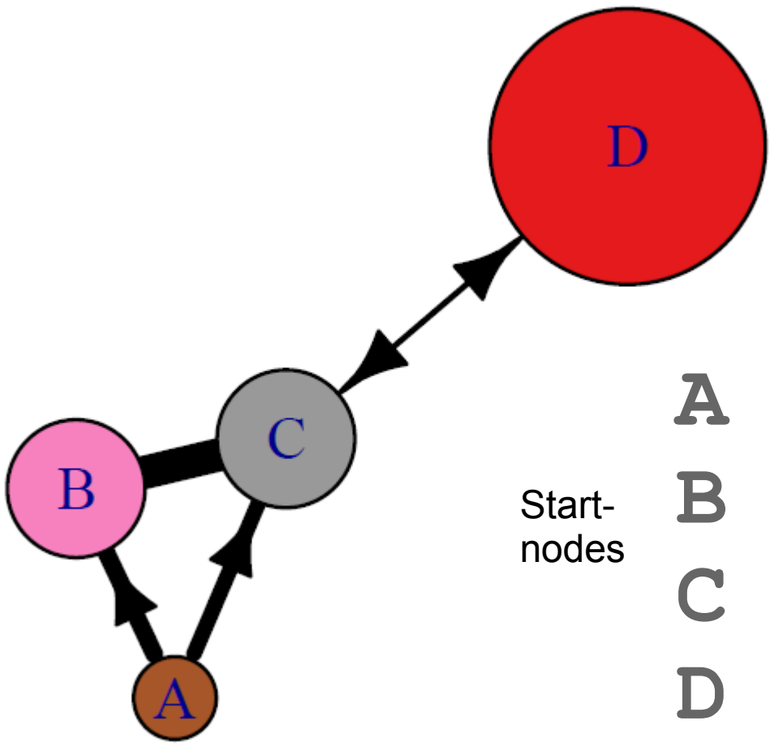
Adjacency Matrix



End-nodes

Start-nodes	A	B	C	D
A	0	1	1	0
B	0	0	1	0
C	0	1	0	1
D	0	0	1	0

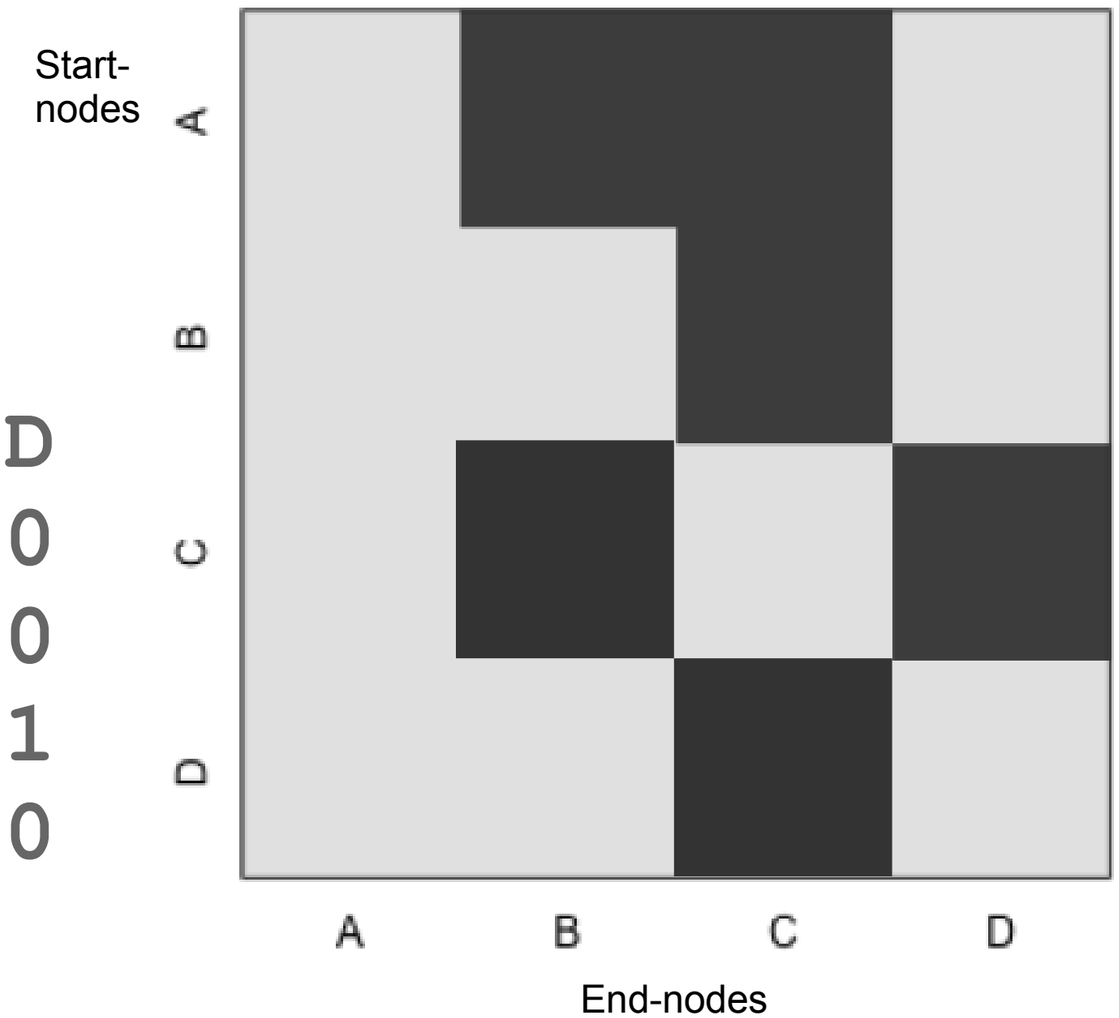
Adjacency Matrix



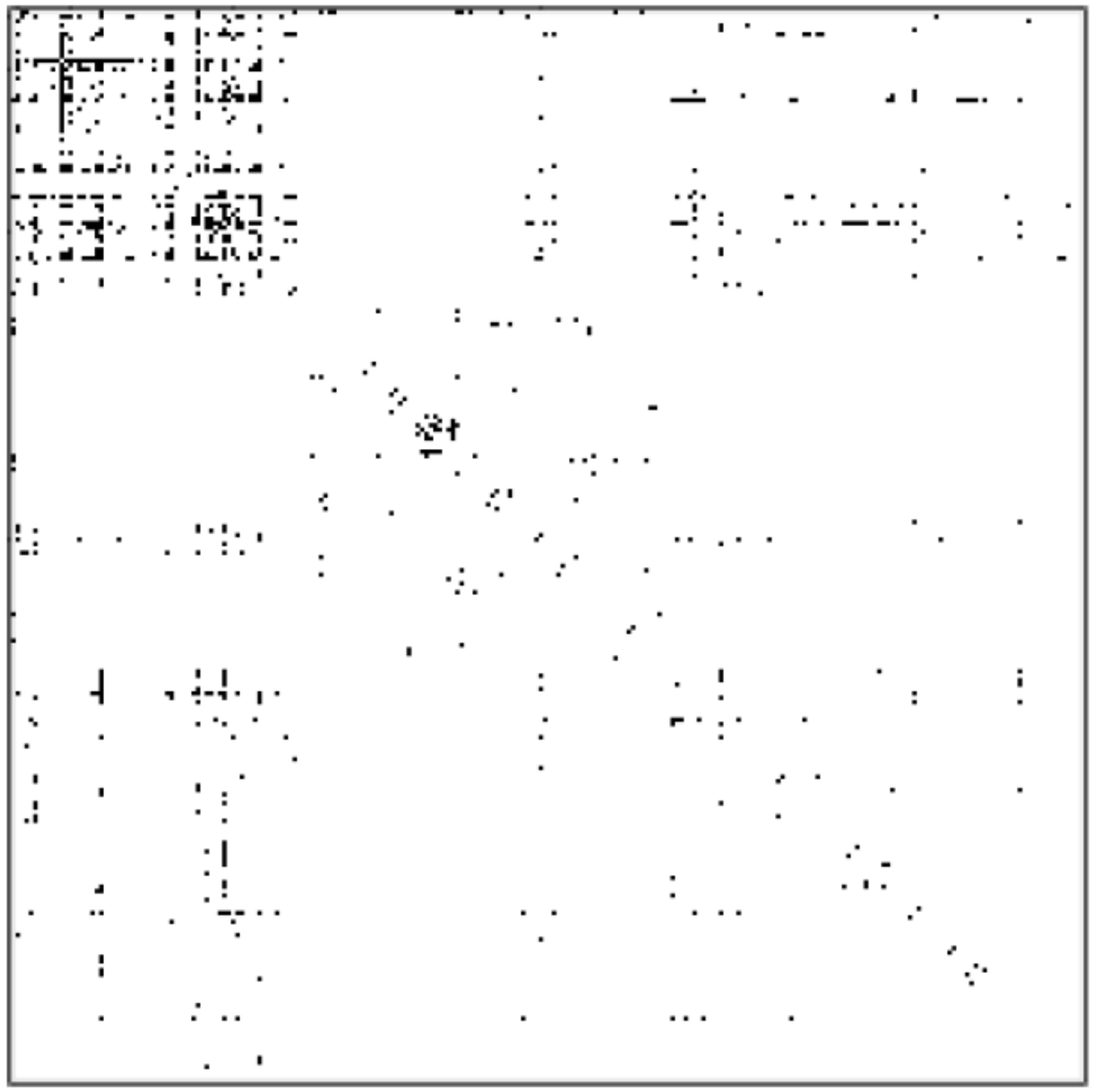
Start-nodes

End-nodes

	A	B	C	D
A	0	1	1	0
B	0	0	1	0
C	0	1	0	1
D	0	0	1	0



US Airport Network



Visualization of the adjacency matrix, i.e., which airports are connected via one or more flights (ignoring the direction).

Visualization by Graph layout

The basic graph drawing problem can be put simply:

- Given a set of nodes (data) with edges (relations) calculate the position of the nodes and the curve to be drawn for each edge
- In other words: For a given combinatorial description of a graph G, derive a node-link diagram representation in the plane.

It's not a new problem: It has always existed for the simple reason that a graph is often defined by its own drawing

Usually, the vertices are represented by symbols such as circles or boxes, and each edge is represented by a simple open curve between the symbols associated with the vertices.

Visualization by Graph layout

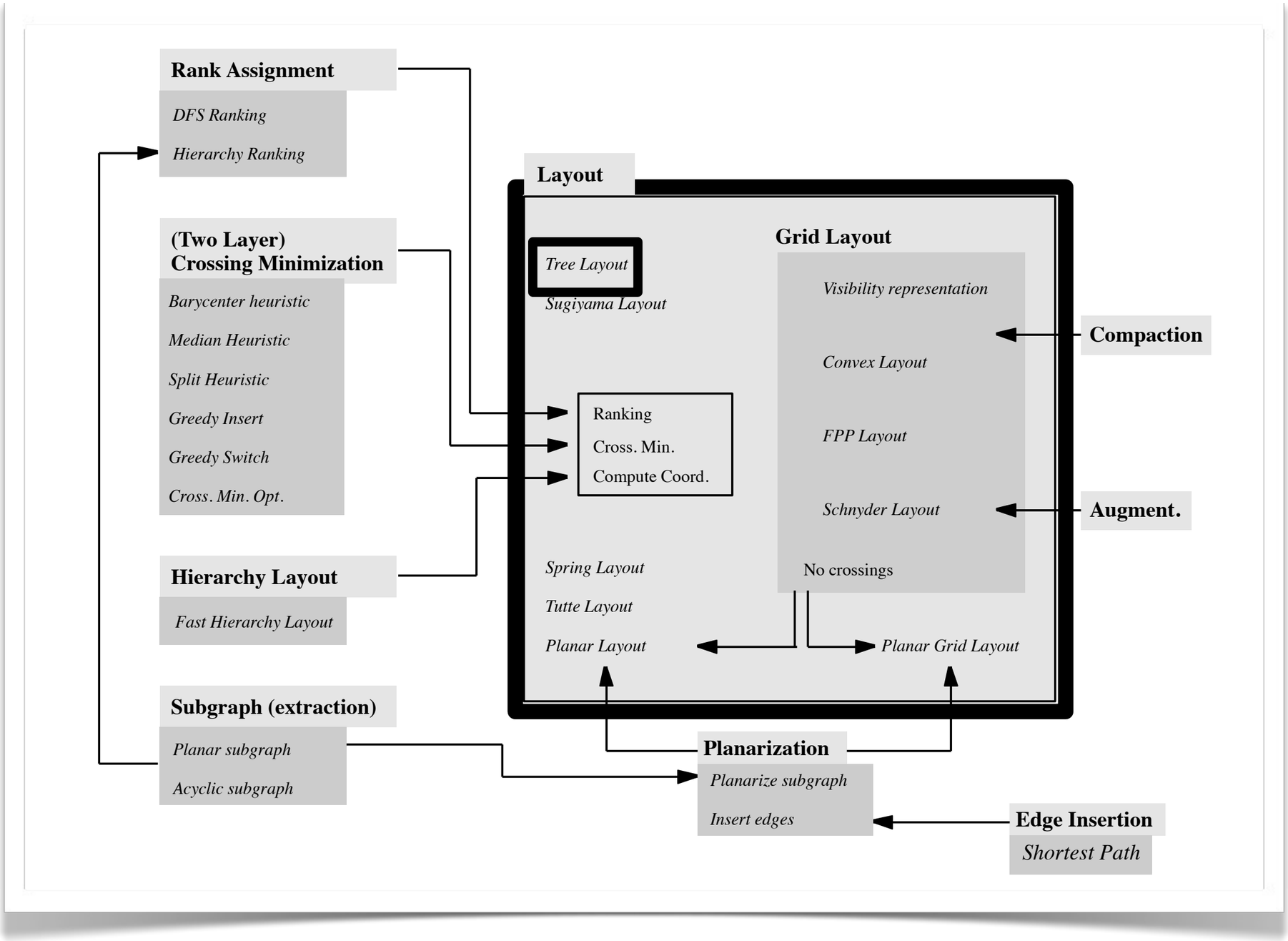
"Isn't that a simple problem ...?!?"

There is a plethora of layout algorithms, each using different techniques and optimizing different criteria

Battista et al. (1994) lists more than 300 publications; the Graph Drawing E-print Archive more than 800 publications (Oct 2013).

• G. di Battista, P. Eades, R. Tamassia and I.G. Tollis, *Graph drawing: Algorithms for the visualization of graphs*, *Prentice Hall*, 1999

Graph layout (cont.)



Mutzel *et al.*. A library of algorithms for graph drawing, 1998

Graph drawing: Optimization algorithms

Usually, a graph has infinitely many different drawings. However, the usefulness of a drawing of a graph depends on its readability.

Readability issues are expressed by means of aesthetics, which can be formulated as optimization goals for the drawing algorithms.

A fundamental and classical aesthetic is the minimization of **crossings between edges**.

(*) cf. Battista et al. (1994)

Graph drawing with fixed node locations

Example: US airport network

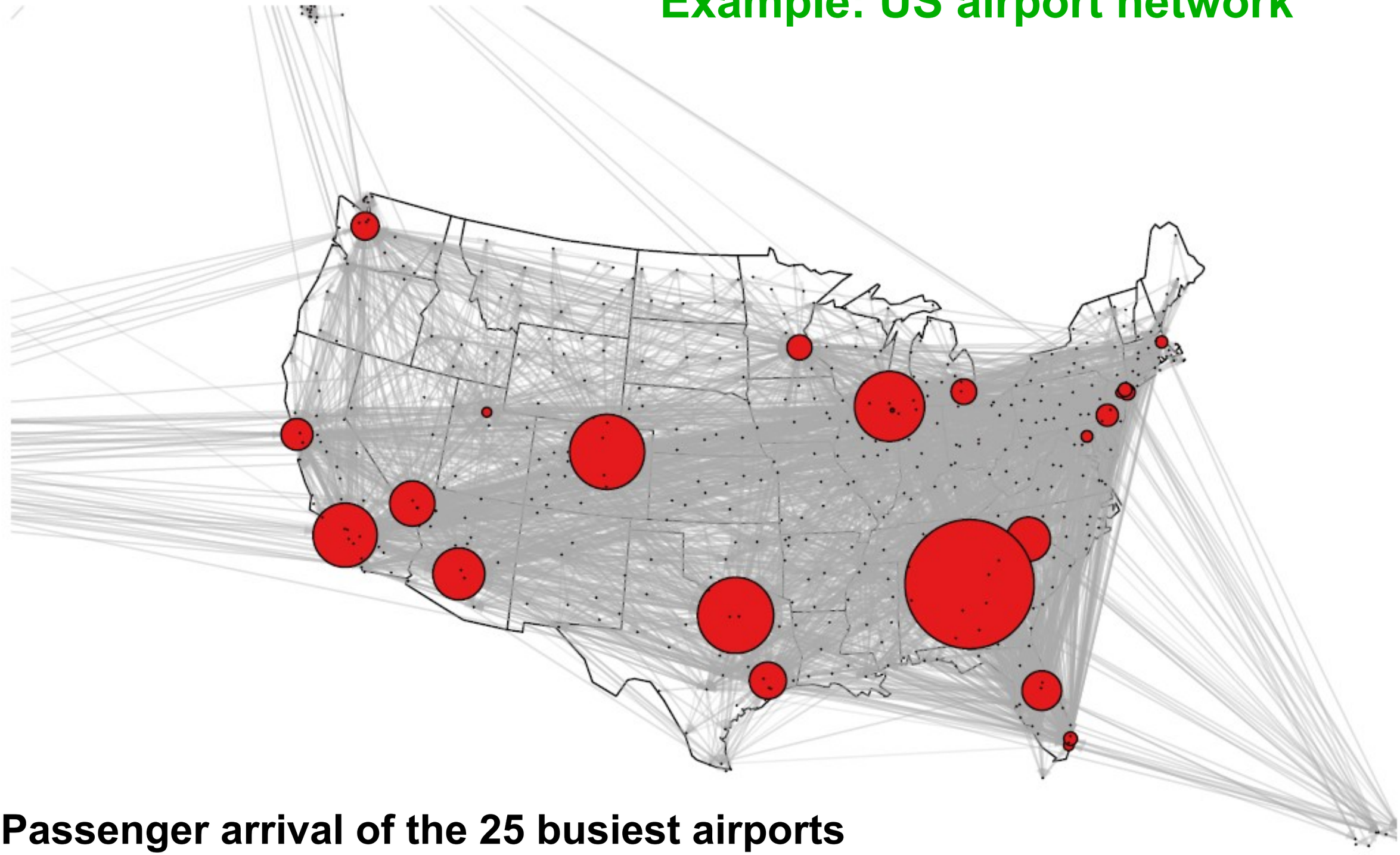
Graph drawing with fixed node locations

Example: US airport network

Airways.

Graph drawing with fixed node locations

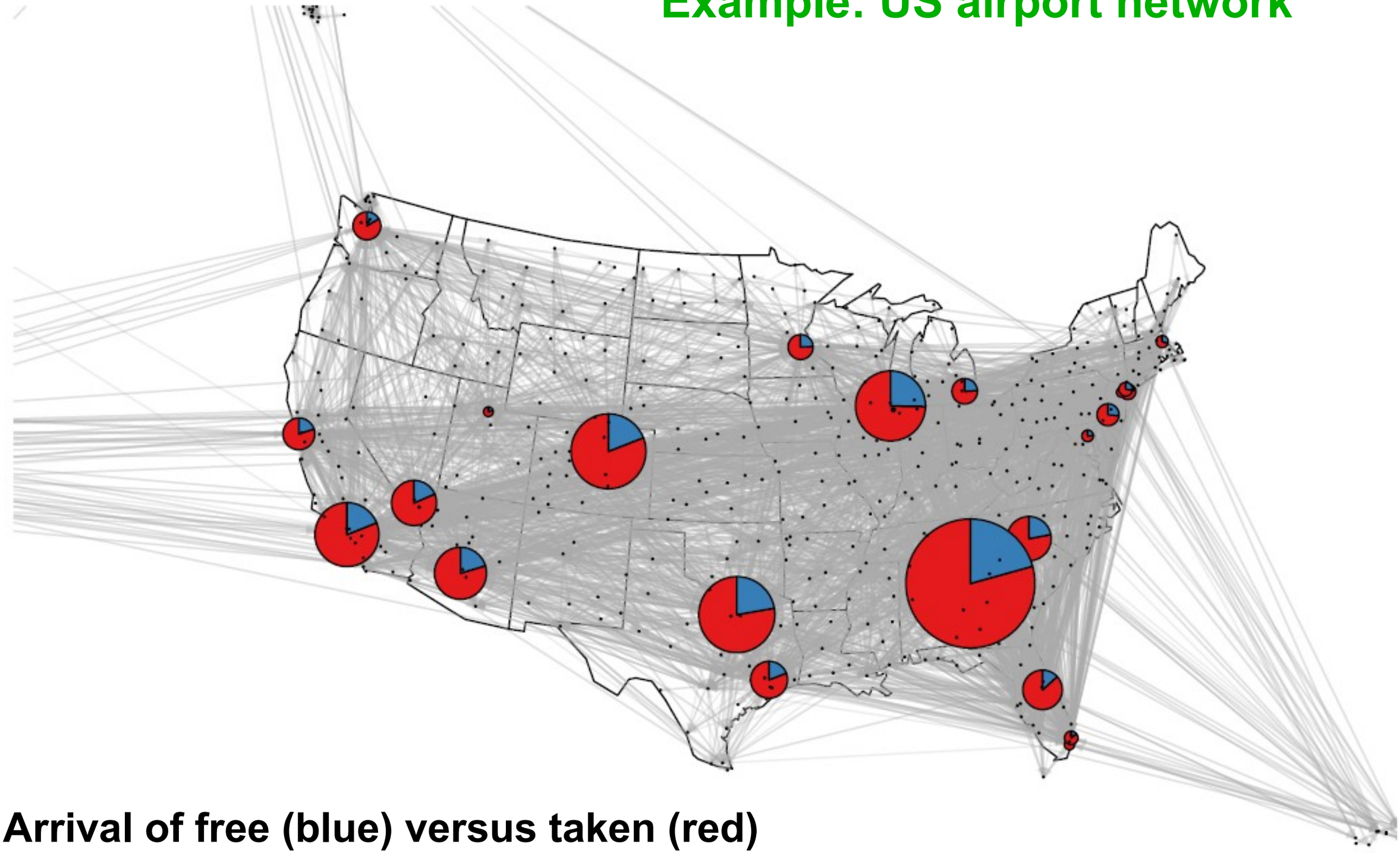
Example: US airport network



Passenger arrival of the 25 busiest airports (difference linearly scaled).

Graph drawing with fixed node locations

Example: US airport network



Arrival of free (blue) versus taken (red) seats of the 25 busiest airports.

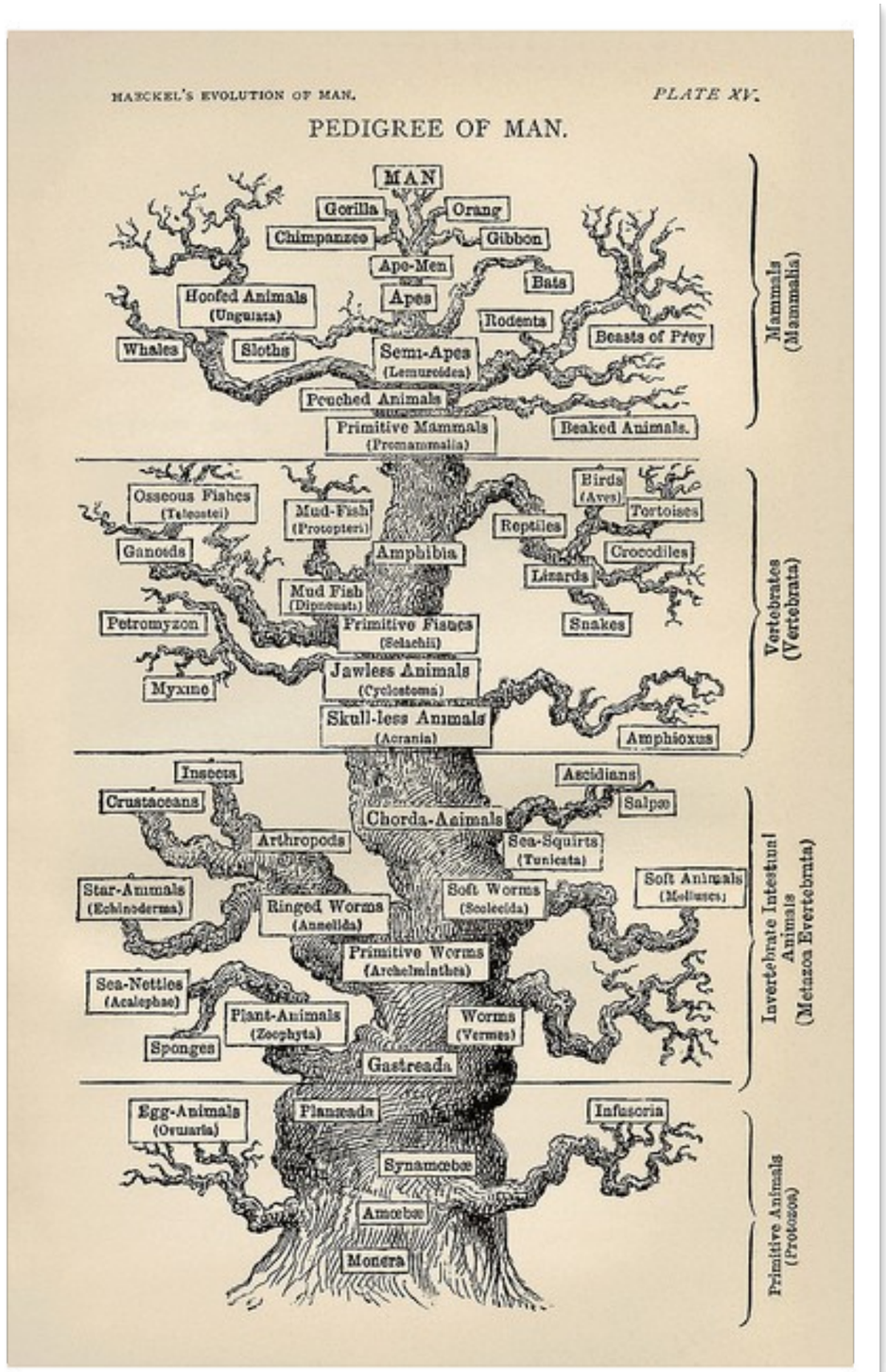
Part 4:
Layout for tree-structured graphs

Graph layout (cont.)

Trees have received most of the attention

A classical **tree layout** will position children nodes *below* their common ancestors

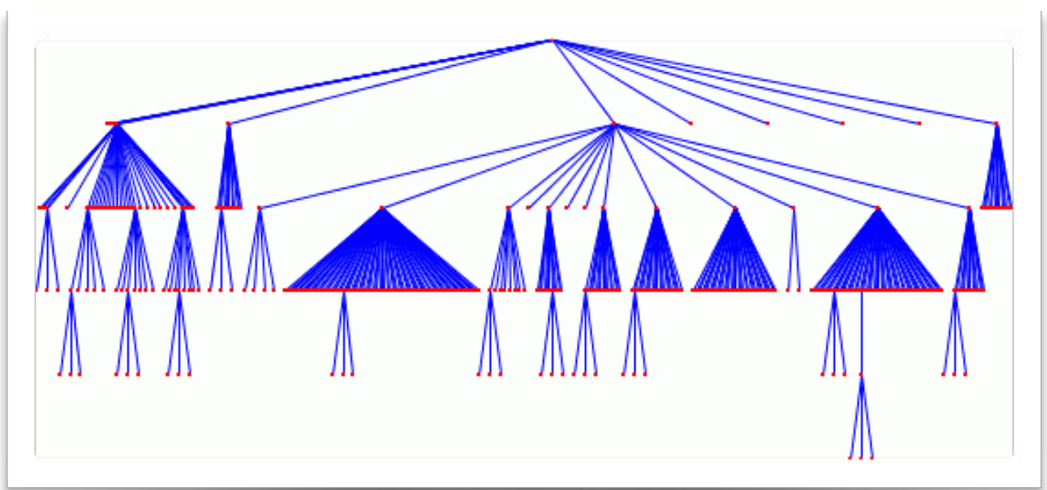
Ernst Haekel (1879):
Pedigree of Man
(here children are *above* the ancestors)



Graph layout (cont.)

This layout satisfies a graph property referred to as *planarity*

- It is possible to draw a tree on a plane without edge crossings

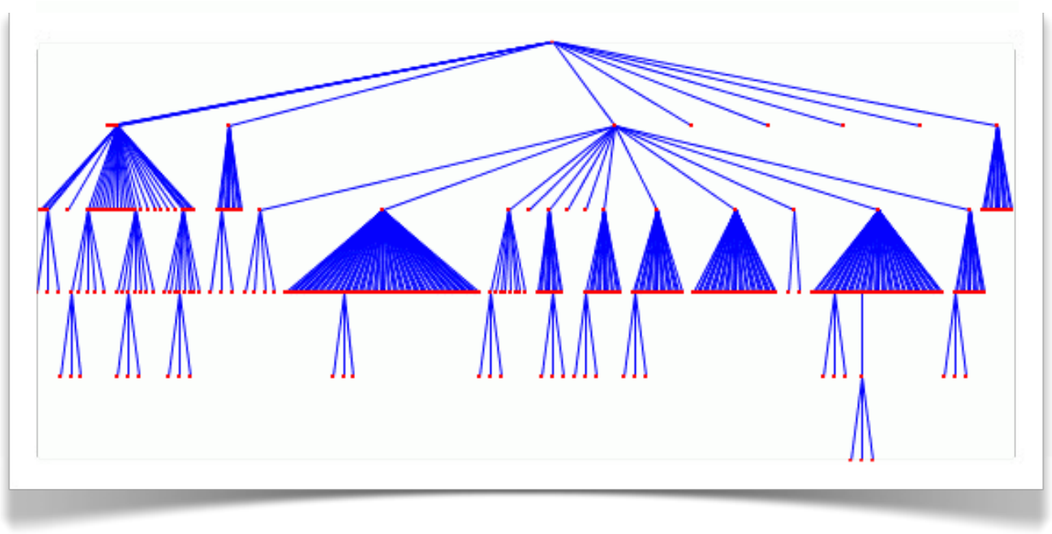


It fulfills many *aesthetic rules* imposed on the final layout

- Nodes with equal depth can be placed on the same horizontal line, distance between sibling nodes can be often fixed, ...
- The drawing clearly reflects the intrinsic hierarchy of the data

Graph layout (cont.)

The Reingold and Tilford (1990, 1997) algorithm for trees is a *tidy* example for achieving these aesthetic goals
(details: http://emr.cs.iit.edu/~reingold/tidier-drawings.pdf)



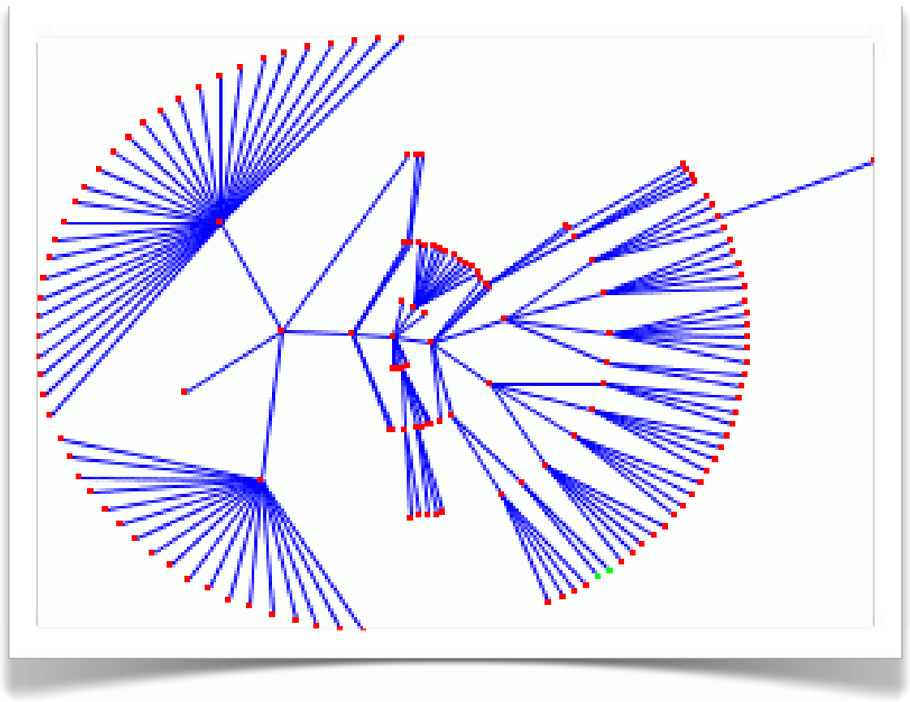
Isomorphic subtrees are laid down analogously

Distance between nodes is a parameter of the algorithm

Only a few hundred nodes and the layout is already very dense

Difficult interaction with the graph (too little space)

Graph layout (cont.)



Radial positioning places graph nodes on concentric circles, according to their depth in the tree

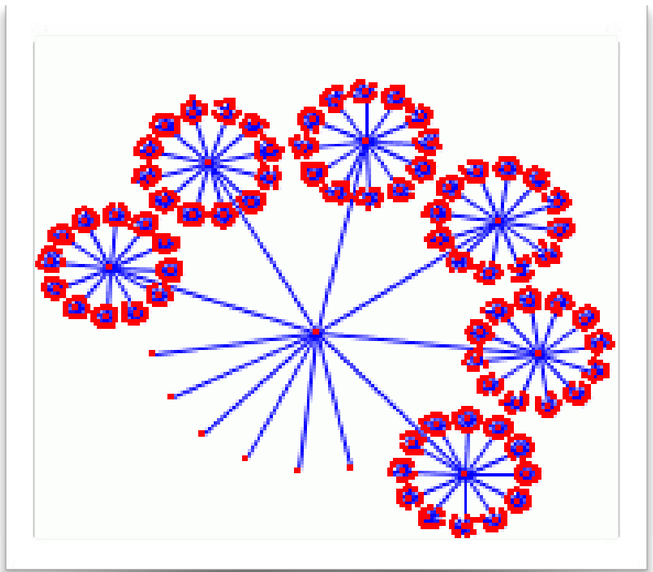
A **subtree** is can be laid out over a sector of the circle so that two adjacent sectors do not overlap

The Eades algorithm (1992) performs this layout

The Reingold-Tilford algorithm generates a more classical drawing

- In the radial positioning it is less clear where the root of the tree is and one might explore the graph in a less hierarchical way

Graph layout (cont.)

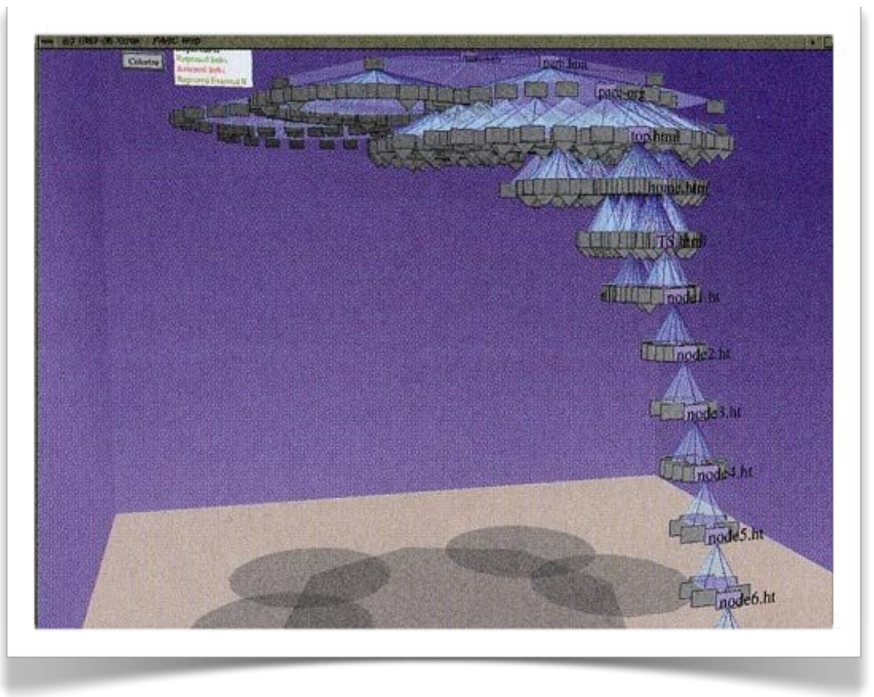


Hierarchy can be recovered using the **balloon view layout**

- Sibling subtrees are drawn in circles centered at their parent nodes
- At the cost of node details

It's a variant of the radial layout, it can be constructed by projecting a **conetree** (a 3D graph) onto a plane

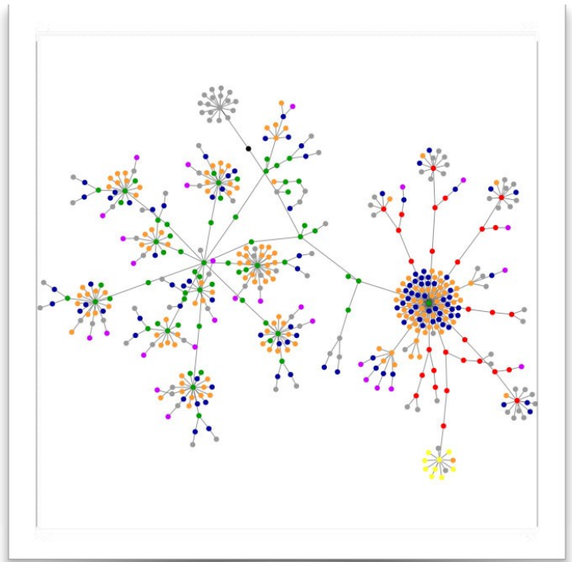
- Parent at the apex of a cone, and its children spaced evenly along its base



Graph layout (cont.)

Tree layout algorithms are often characterized by low complexity and simple implementation

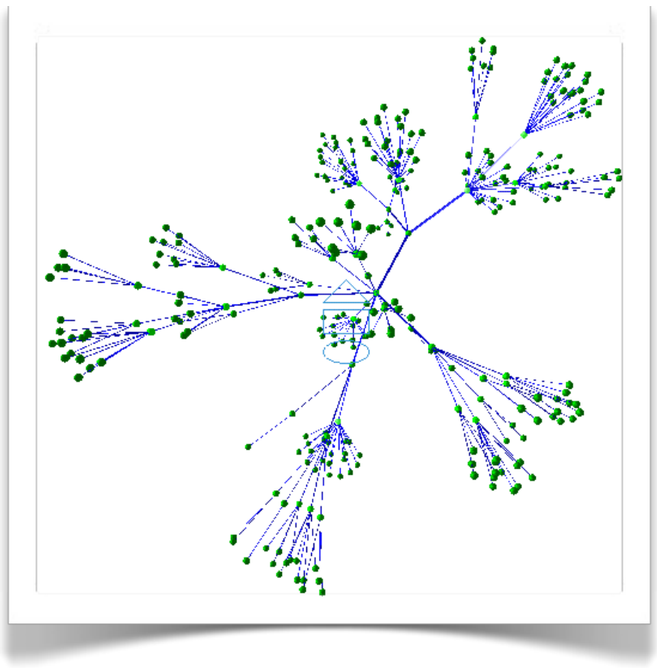
- They work well and they are applicable only for relatively small graphs



One very popular technique is to display larger graphs in three dimensions rather than two

- The idea is that the extra dimension gives literally more space and eases the problem of displaying large structures
- The viewer can navigate to find a view without occlusions

Graph layout (cont.)



The simplest approach is to generalize classical 2D layout algorithms for 3D

- 3D radial tree algorithms

In spite of the apparent simplicity, displaying trees in 3D can introduce new problems

- Additional visual cues (occlusion, transparency, depth, ...)
- Minimizing edge crossing for multiple perspectives may not be as rewarding as hoped

Graph layout (cont.)

In spite of all the technical development and the obviously attractive features, ...

3D graph visualization techniques have still significant difficulties

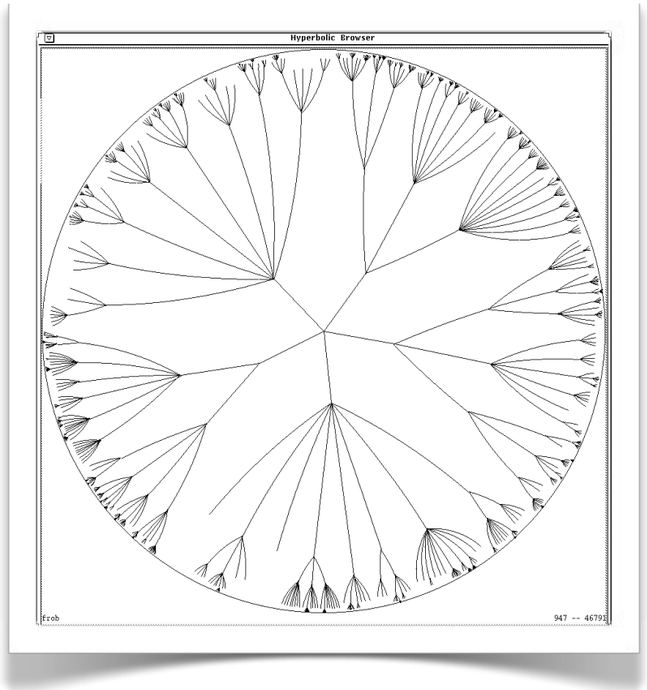
- The main reason lies with the inherent cognitive issues of 3D navigation/interaction in 2D screens
- Perceptual and navigational conflicts

Graph layout (cont.)

The **hyperbolic layout** of trees is one of the recent forms of layout (Lamping *et al.* and Munzner in the mid-late 90's)

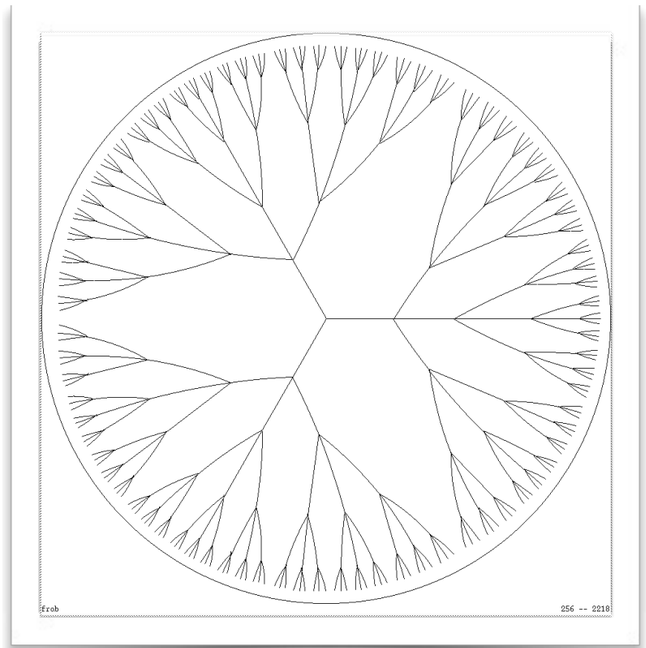
Developed with visualization and interaction in mind

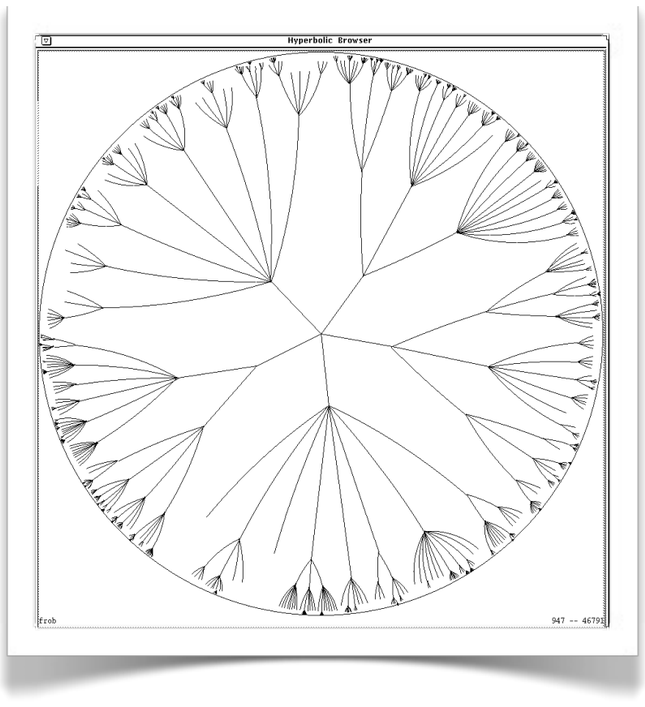
- It provides a *distorted* view of a tree layout
- It resembles the effect of using a fish-eye lens
- It can be implemented in either 2D or 3D



Graph layout (cont.)

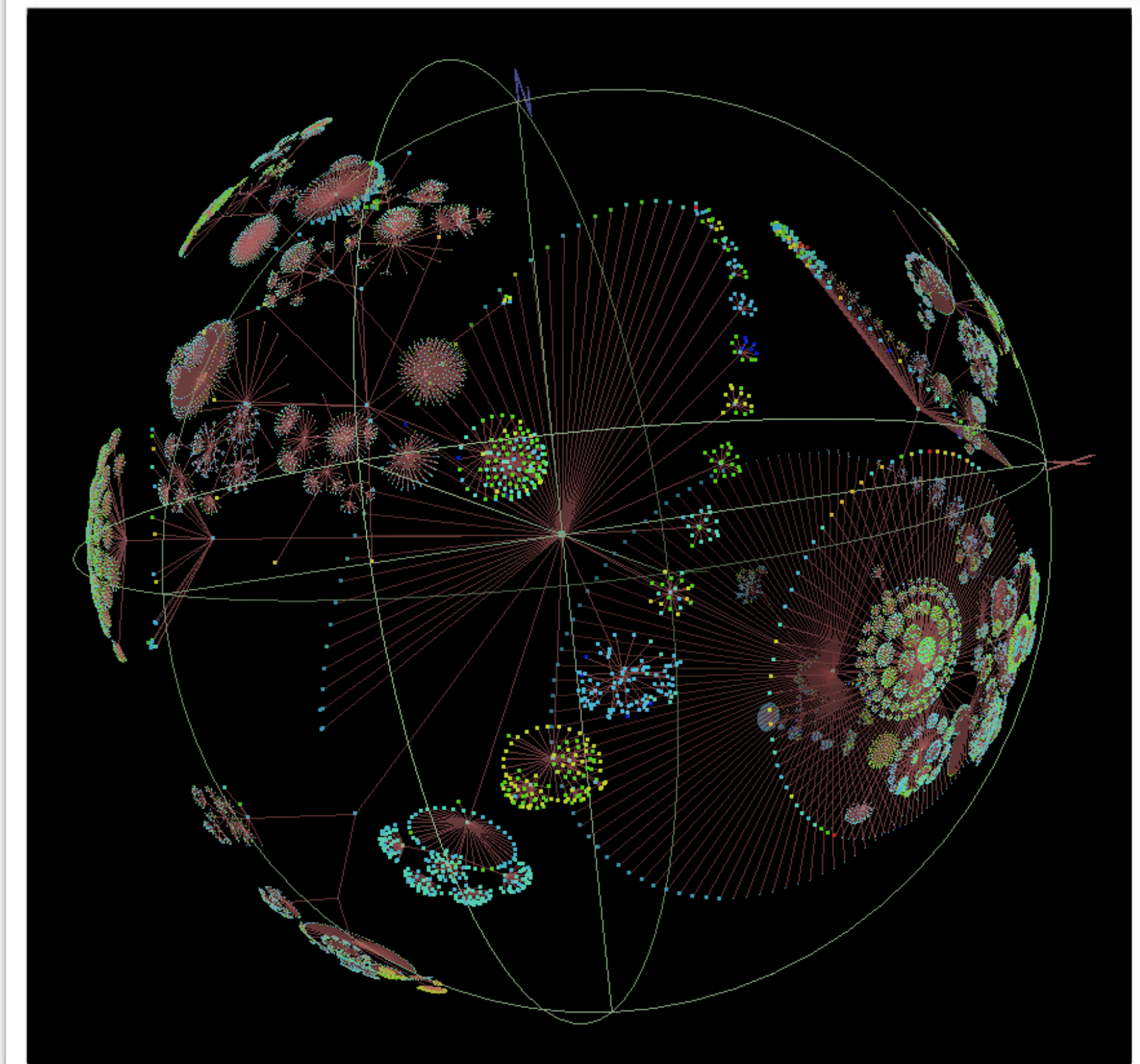
Hyperbolic views are radically different, due to their different geometrical background





This distorted view makes it possible to interact with potentially large trees making it suitable for some demanding real-world applications

Graph layout (cont.)



Part 4:
Seriation

Addendum: Seriation

Seriation is to finding a suitable *linear order* for a set of objects given data and a loss or merit function (in order to reveal structural information).

Can be done without creating a layout

(*) cf. Hahsler et al. (2008)

Seriation - Combinatorial optimization problem:

Given is a symmetric (dissimilarity/binary) matrix D = (dij) where d_{ij} for $1 \leq i, j \leq n$ represents the dissimilarity between objects i and j , and d_{ii} = 0 for all i .

The permutation function Ψ is a function which reorders the objects in D by simultaneously permuting rows and columns.

The seriation problem is to find a permutation function Ψ^* which optimizes the value of a given loss function *L* or merit function *M*:

$$\Psi^* = \text{argmin}_\Psi L(\Psi(\mathbf{D})) \quad \text{or} \quad \Psi^* = \text{argmax}_\Psi M(\Psi(\mathbf{D}))$$

Hard problem: the number of possible permutations for *n* objects is *n*!.

Seriation - Combinatorial optimization problem:

Partial enumeration methods: Search for the exact solution in a clever way; independent of the loss/merit function.

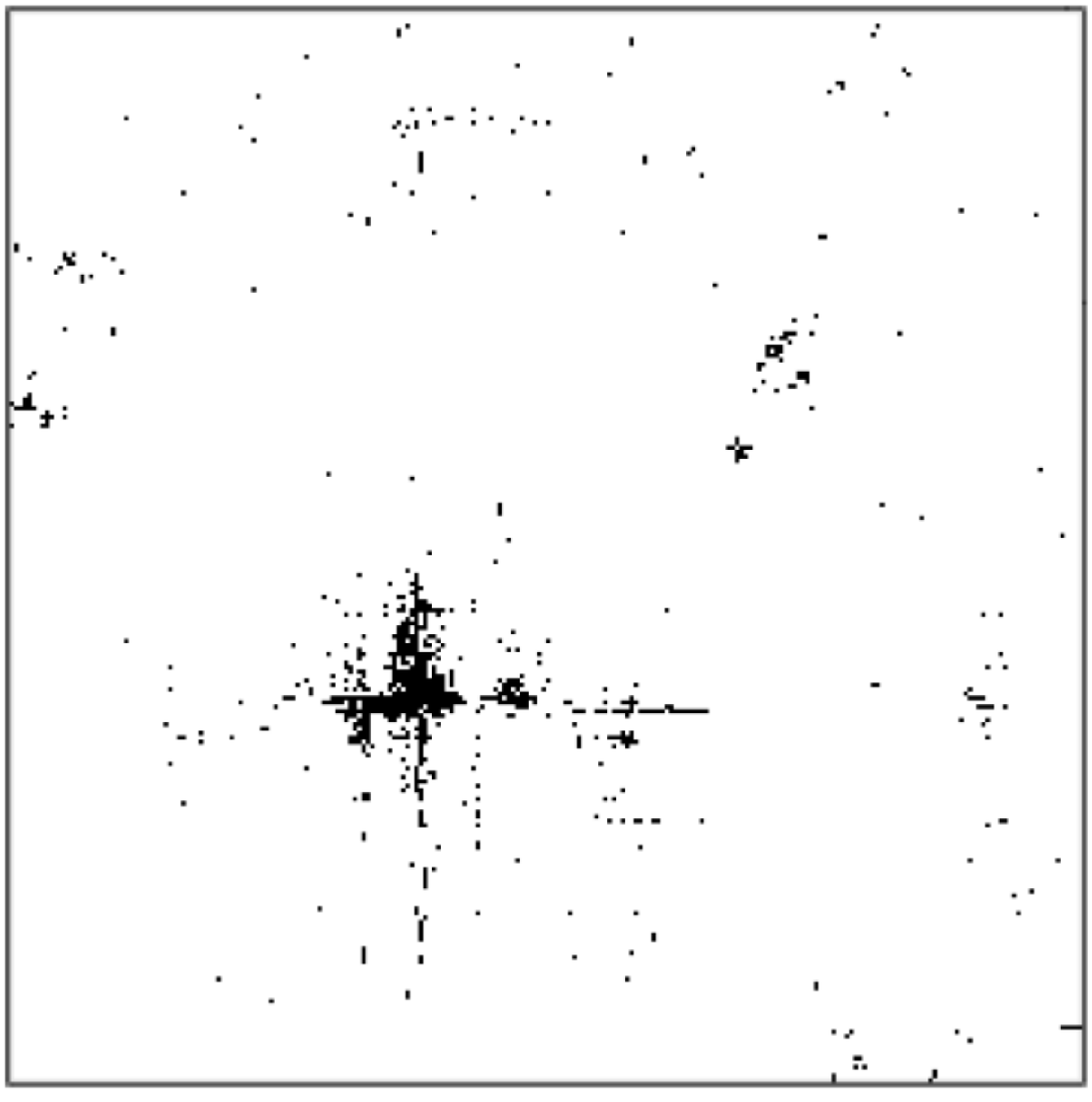
Traveling salesperson problem solver: Search for a *Hamilton path* (i.e., a path where each vertex is visited once); wide array of heuristics are available.

Bond energy algorithm: Rearrange columns and rows of a matrix such that each entry is as closely numerically related to its four neighbors as possible; simple heuristic.

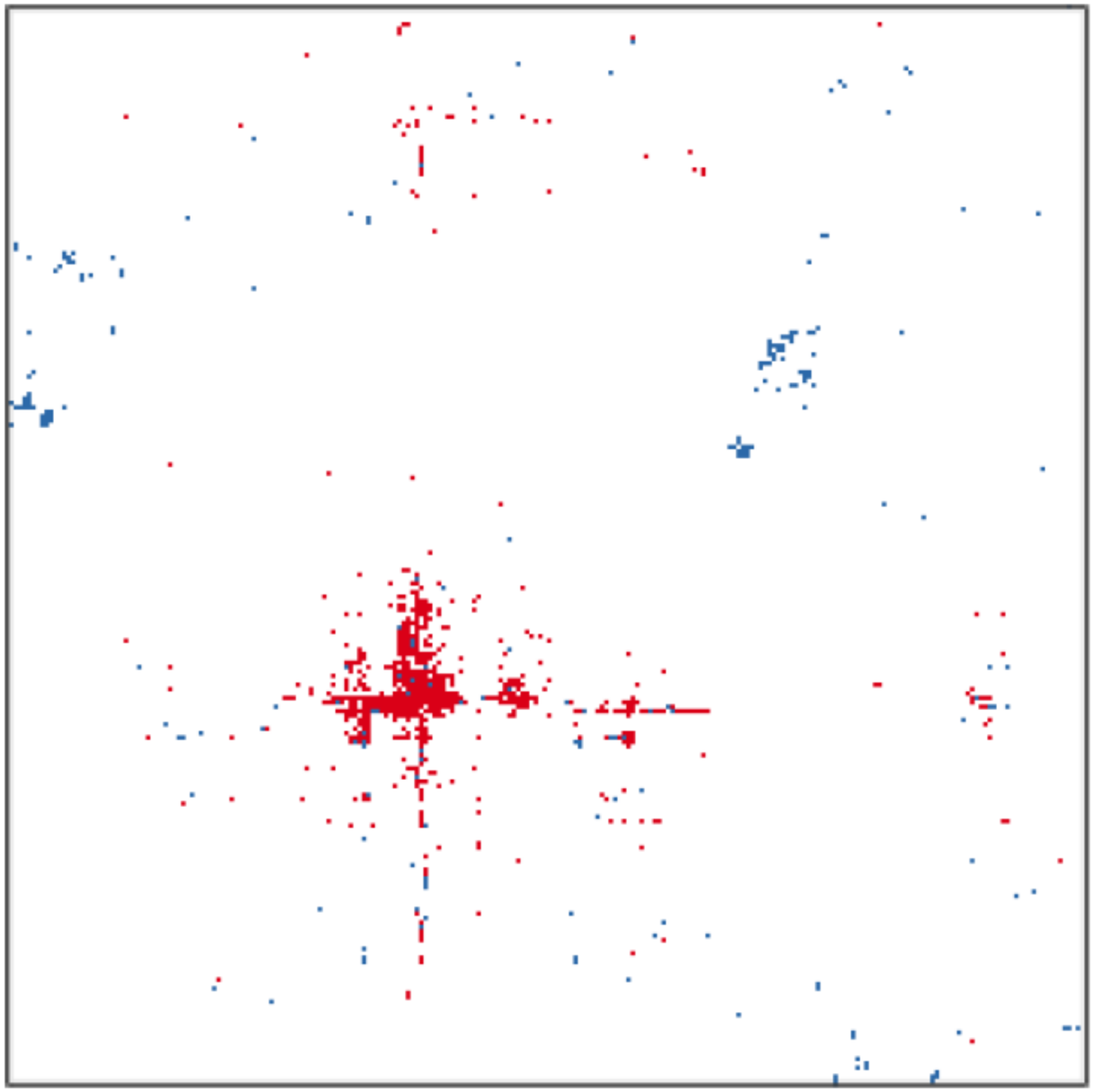
Hierarchical clustering: Use the order of the leaf nodes in a hierarchical clustering; simple heuristic.

(*) for details see Hahsler et al. (2008)

Seriation example - US Airport Network

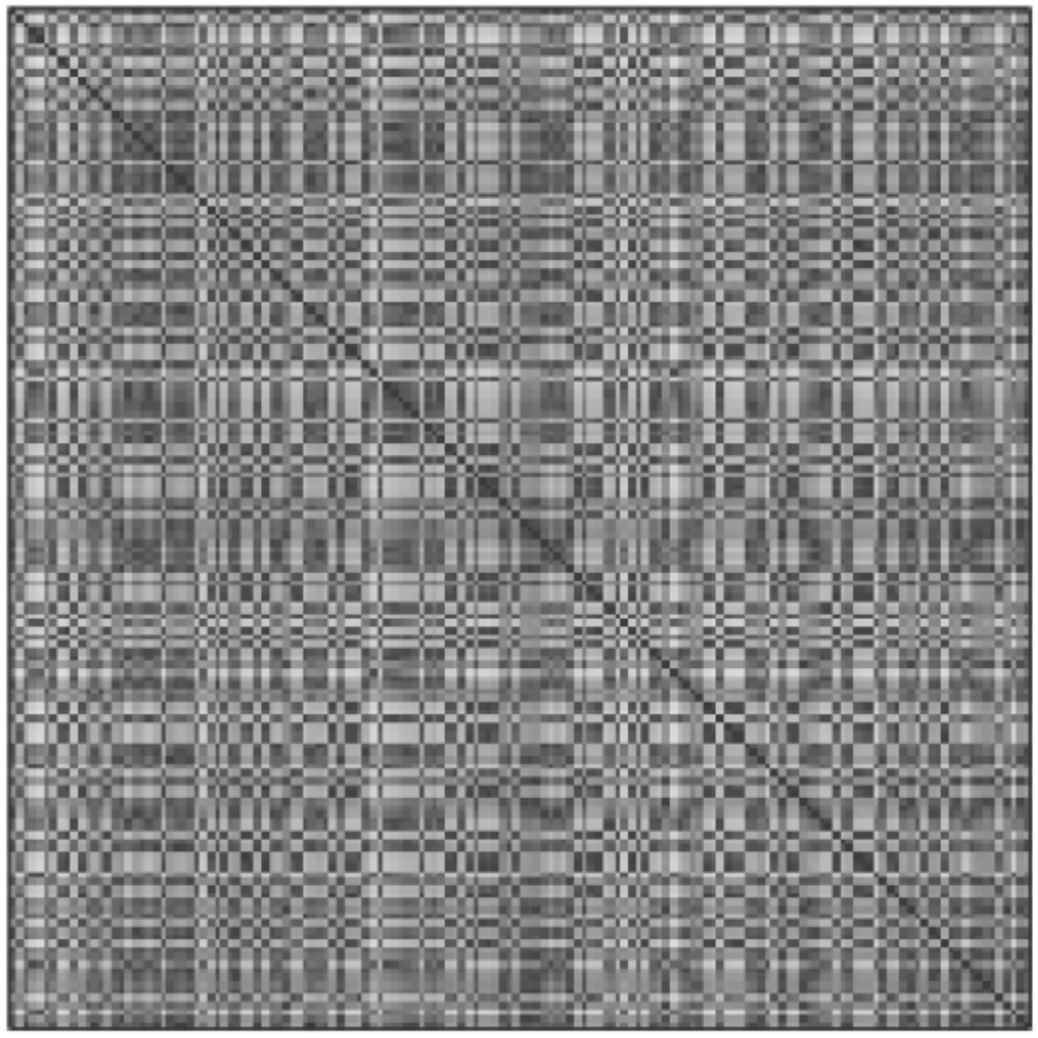


Seriation example - US Airport Network

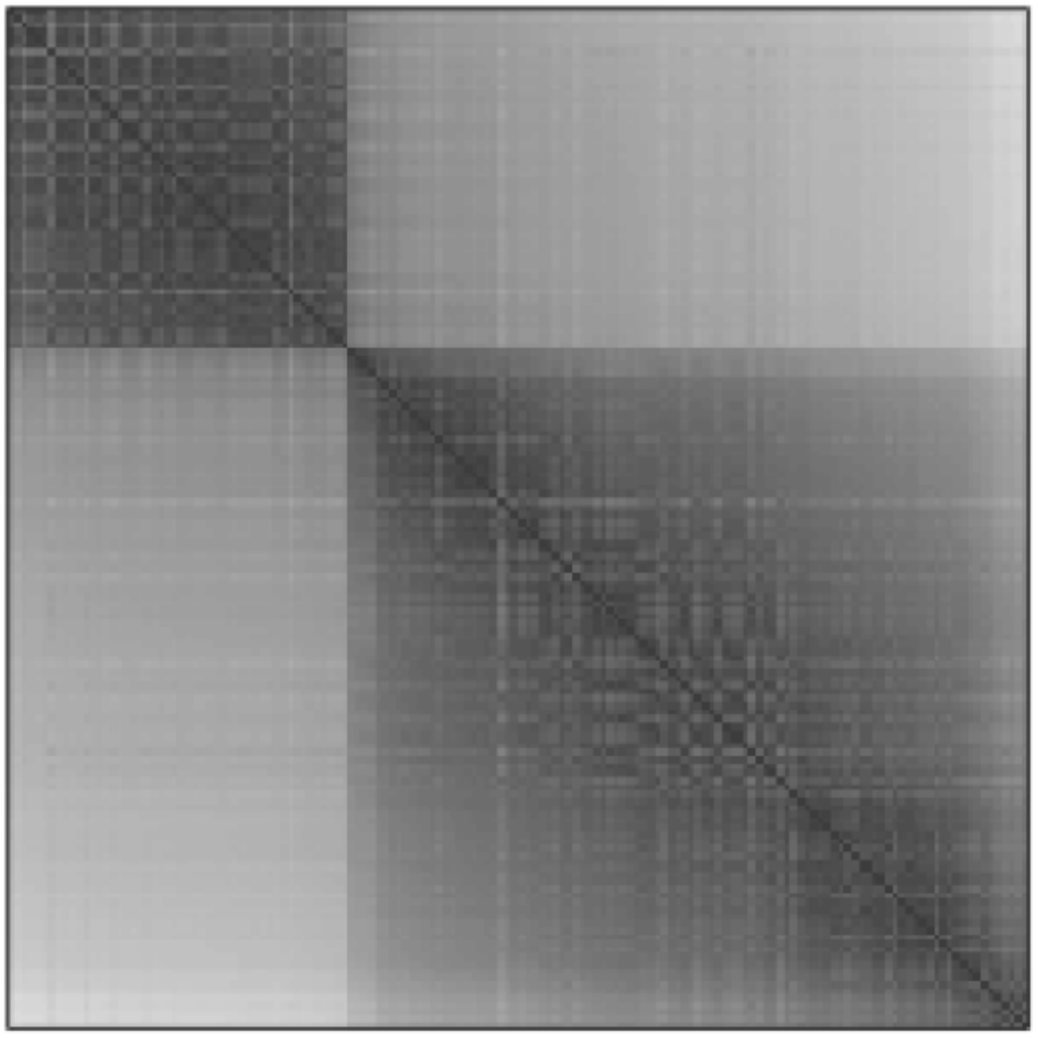


Color indicates whether flight is within the same state (blue) or between two different states (red).

Note: Seriation can be used for vectorial data too...



Iris data, unordered distance matrix



Iris data, ordered distance matrix

References

I. Herman, G. Melançon and S. Marshall, “Graph visualization and navigation in information visualization: A survey”, *IEEE Transactions on Visualization and Computer Graphics*, 6(1), 24-43, 2000

Manuel Lima, *Visual complexity: Mapping patterns of information*. Princeton Architectural Press (2011)

Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for drawing graphs: An annotated bibliography. Computational Geometry: Theory and Applications, 4(5):235{282, 1994. http://dx.doi.org/10.1016/0925-7721(94)00014-X.

David Easley and Jon Kleinberg. Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press, 2010. http://www.cs.cornell.edu/home/kleinber/networks-book/.

References, continued

Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: An introduction to the r package seriation. Journal of Statistical Software, 25(3):1{34, 2008. URL http://www.jstatsoft.org/v25/i03.

Robert Kosara. Graphs beyond the hairball, 2012. URL http://eagereyes.org/techniques/graphs-hairball. Blog entry.

Carlos E. Scheidegger. So you want to look at a graph, part 1, 2012. URL http://cscheid.net/blog/so_you_want_to_look_at_a_graph__part_1. Blog entry.

Roberto Tamassia, editor. Handbook of Graph Drawing and Visualization. CRC Press, 2013. http://cs.brown.edu/~rt/gdhandbook/.

Part 5:
Layout for general-structured graphs

Graph drawing: Optimization algorithms

Popular graph layout strategies:

- force-based
- circular
- tree-based

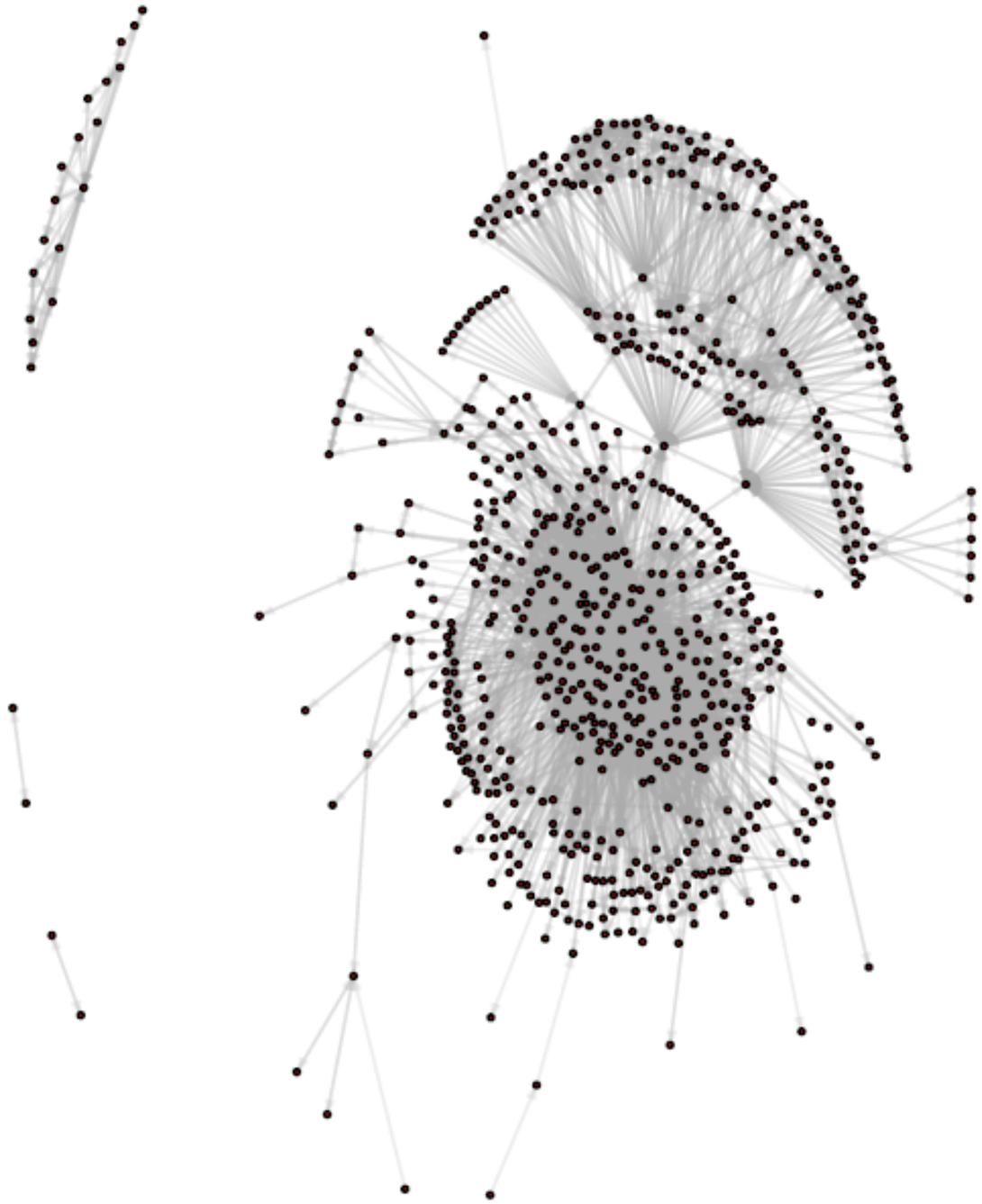
Force-Directed layout algorithms

Force-directed methods define an objective function which maps each graph layout into a number in $\mathbb{R}+$ representing the energy of the layout.

This function is defined in such a way that low energies correspond to layouts in which adjacent nodes are near some pre-specified distance from each other, and in which non-adjacent nodes are well-spaced. A layout for a graph is then calculated by finding a minimum of this objective function.

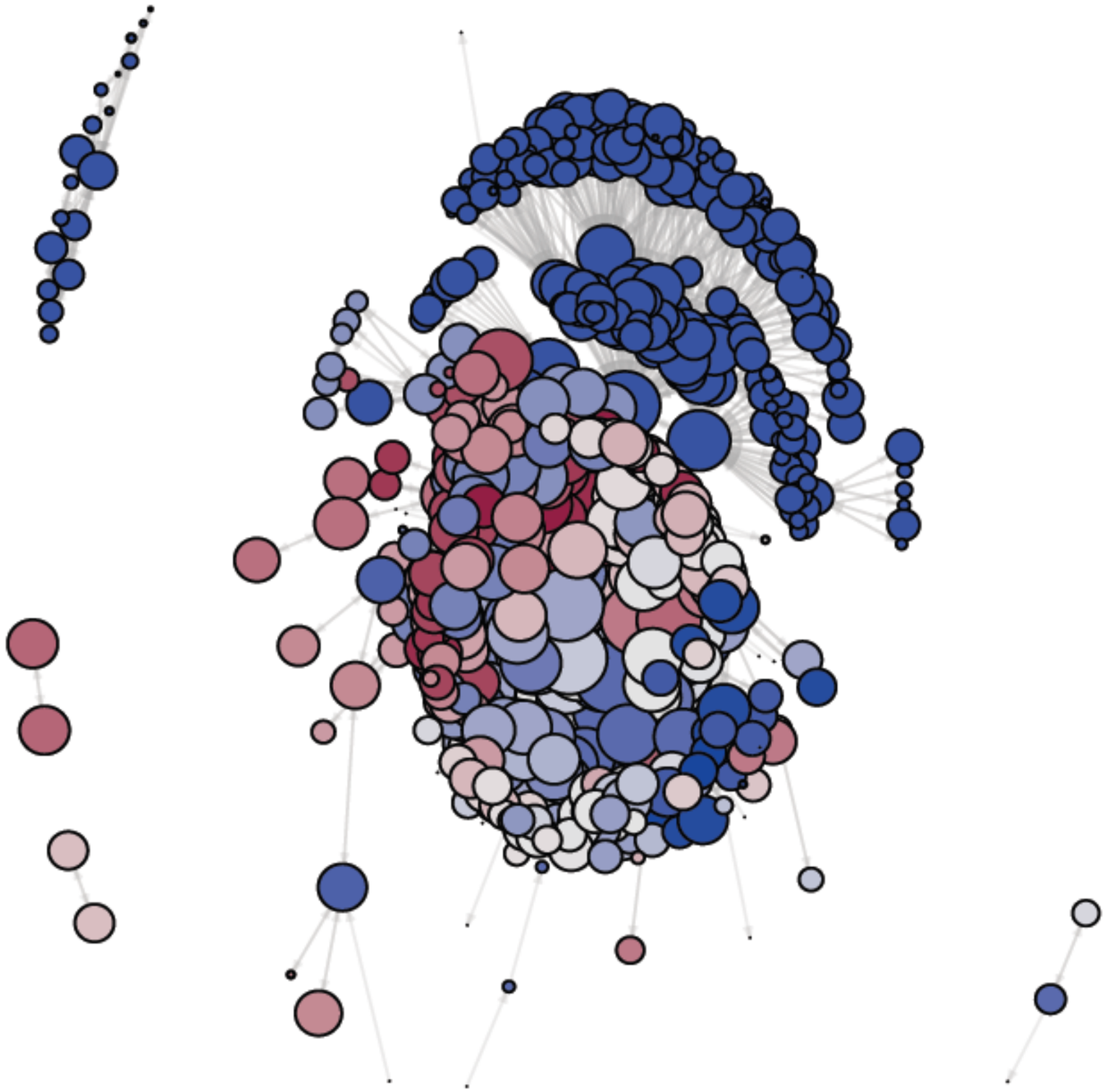
(*) Tamassia (2013, Chapter 12)

US airport network



Kamada-Kawai layout with weighted edges based on the distance.

US airport network



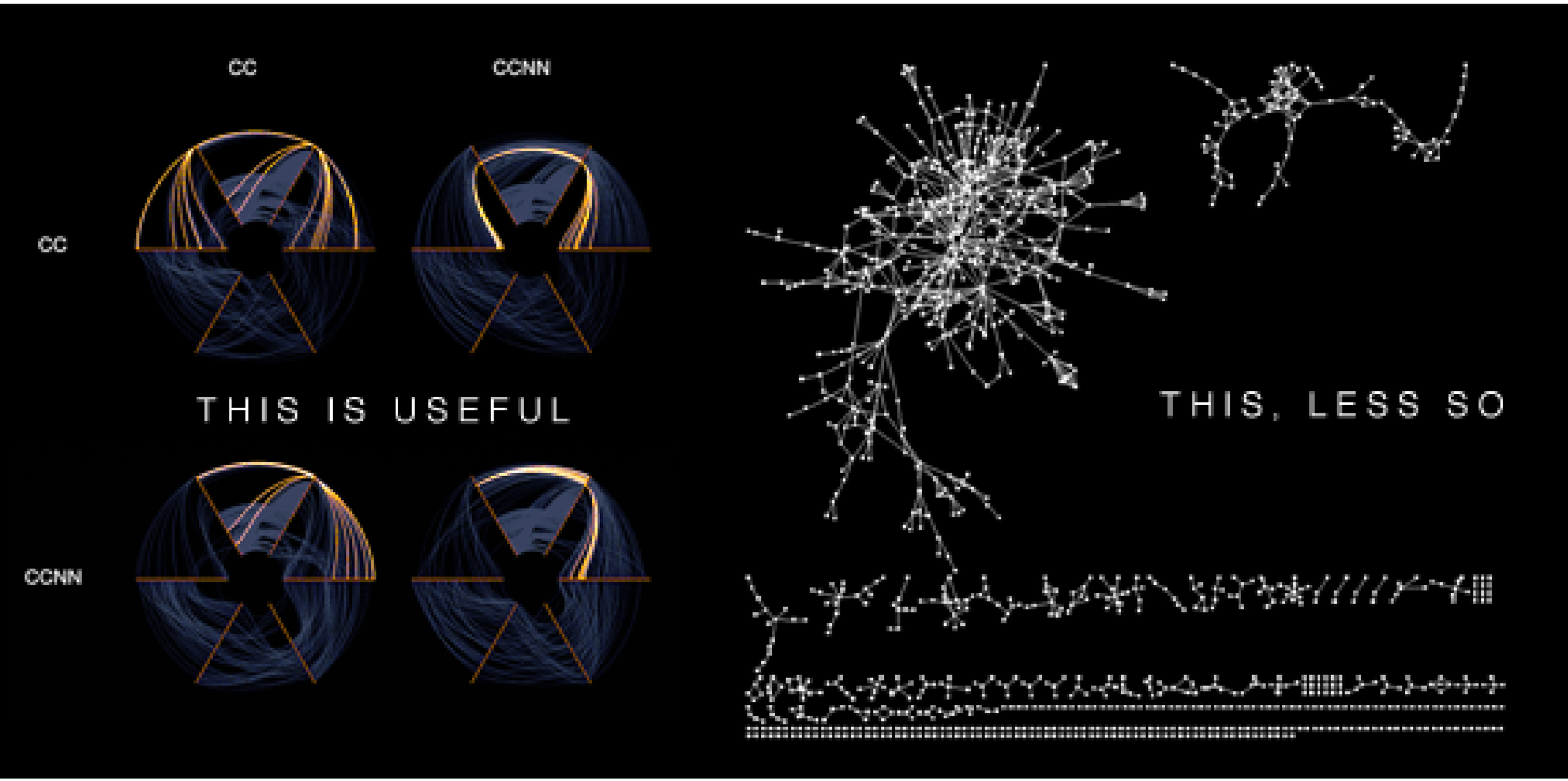
Kamada-Kawai layout with weighted edges based on the distance; vertex sizes based on total passengers, vertex color based on state.

How to cure the Hairball?

Possible parts to play with:

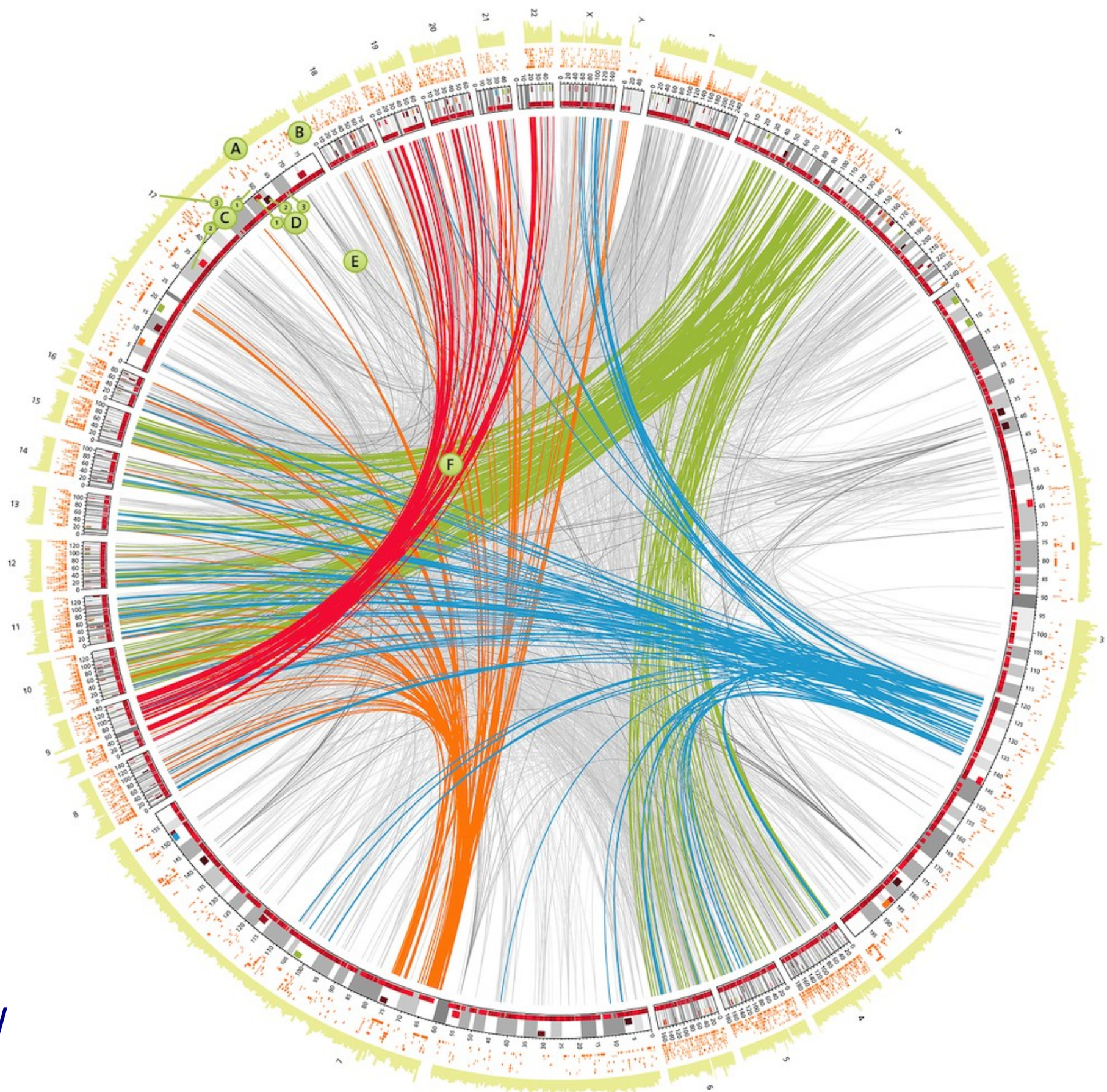
- Visual attributes of edges and vertices
- **Layout algorithm** **<---- we will concentrate on this**
- Edge bundling
- **Interaction** **<---- we will also talk about this**

Hive plots



http://www.hiveplot.net/

Circos



http://circos.ca//

Part 6:
Interactive visualization of graphs

Navigation and interaction

Navigation and interaction are essential in infoviz

- Layout algorithms alone cannot overcome the problems raised by the large sizes of the graphs occurring in many applications

Zoom and pan (cont.)

Zoom and pan are conventional tools in visualization

- Indispensable when large graph structures are to be explored

Zooming is well-suited for graphs, because the graphics used to display them are usually fairly simple

- Mostly lines, sometimes curves, and other simple geometric forms

Zoom and pan (cont.)

Zooming can take on two main forms

- **Geometric:** simply provides a blowup of the graph content
- **Semantic:** information content changes and more details

are shown when approaching a particular area of the graph

http://ucjeps.berkeley.edu/map2.html

Zoom and pan is thus conceptually simple, but it might create problems when used in interactive environments

Zoom and pan (cont.)

Road map of Europe, the user has zoomed into the area around London but wants to change to a view of Berlin

- Doing this without changing the zoom factor (at least temporarily) might be slow (zoom out, pan to Berlin and zoom in again)
- The user wants smooth changes (perform zoom and pan in parallel)

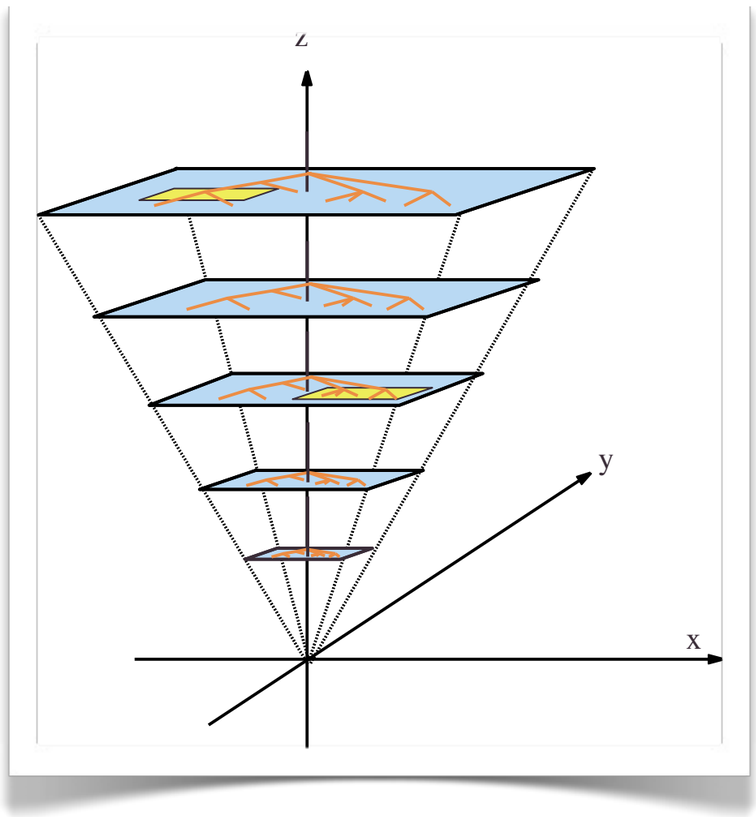
When zooming in, the view expands exponentially fast and the target point moves always faster than the pan can keep up with

The net result is a target that is approached non-monotonically

Zoom and pan (cont.)

Furnas and Bederson proposed an elegant solution to alleviate the effects of the zoom and pan problem

Space-scale diagrams



Create copies of the original 2D picture, one at each possible magnification and stack them up

- Combinations of (continuous) zoom and pan actions are described as paths
- An optimal path in this abstract space can be found e.g., thru minimum path length

Focus and context

Another well-known problem with zooming is that if one zooms on a focus, all contextual information is lost

- The loss of context can become a considerable usability obstacle

A set of approaches allow to focus on some detail without losing the context

Focus+Context techniques

They do not replace zoom and pan but rather complement them

Focus and context (cont.)

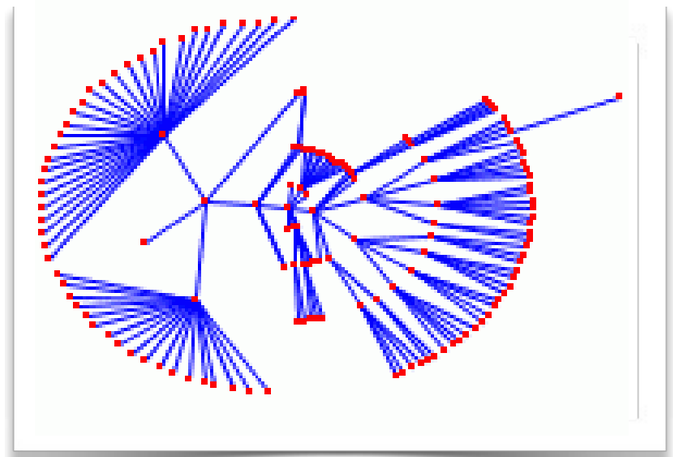
Graphical fish-eye distortion is a popular technique for F+C

- Enlarging the area of interest and showing other portions of the image with successively less detail

Conceptually, the graph is mapped onto the plane and a "focus" point is defined by the user

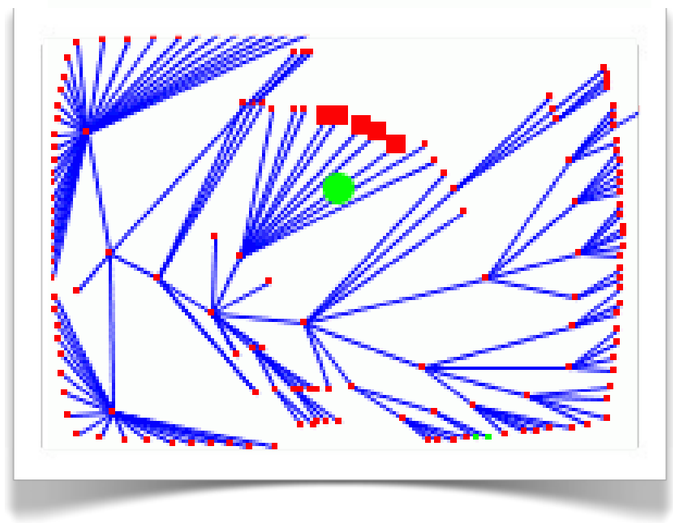
- The distance from the focus to each node of the tree is then distorted by a function and the distorted points and connecting edges are displayed

Focus and context (cont.)

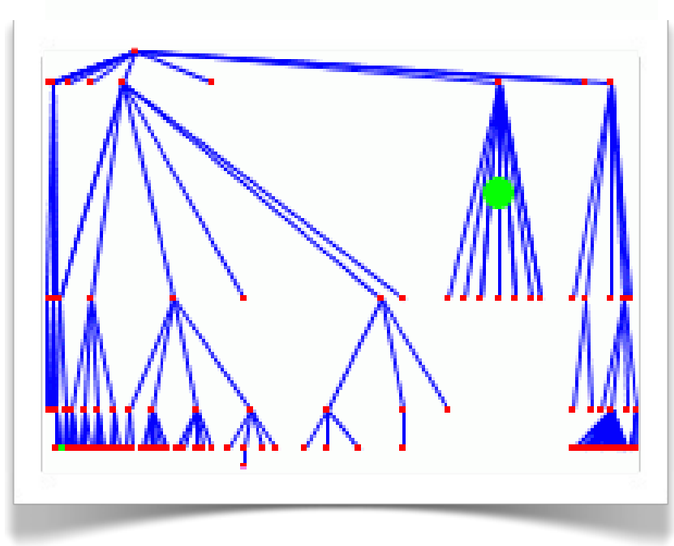


The created distortion depends on the form of the distorting function

Two basic variations:



- Polar distortion: It applies to the nodes radially in all directions, starting from the focus point



- Cartesian distortion: It is applied on each direction (x and y) independently before establishing the final position of the nodes

Focus and context (cont.)

This simple but powerful technique is an important form of navigation but has at least one major pitfall

- The essence of the fish-eye is to distort the position of each node
- If the distortion is faithfully applied, straight edges connecting the nodes will also be distorted, the result is a general curve

Standard graphic systems don't seem to offer the necessary facilities to transform lines into curves

- Mostly because of the prohibitively large number of calculations
- ... so, we get straight-line edges and distortion only for nodes

The consequence of this solution is edge-crossing

Focus and context (cont.)

Interaction with fish-eye means changing the position of the focus and/or modifying the distortion parameters

The fish-eye technique is independent of the layout algorithm and it is defined as a separate postprocessing step on the graphical layout of the graph

- **(+)** modular organization of the software implementation
- **(-)** may destroy the aesthetics governing the layout algorithm

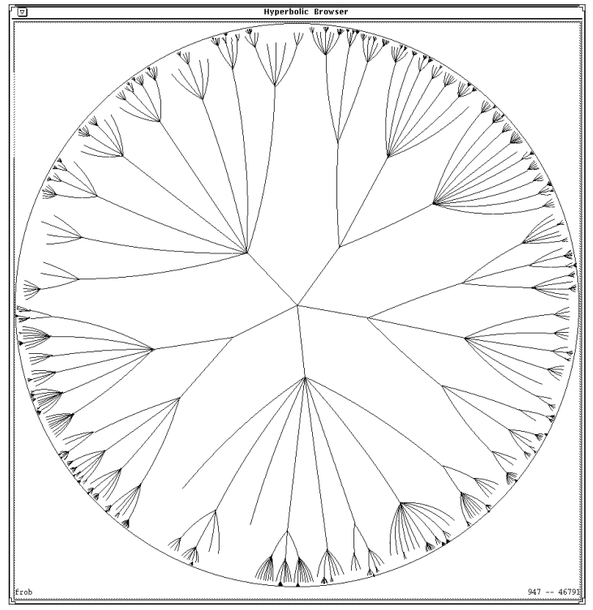
Appropriate distortion possibilities can be built into the layout algorithm itself

- Context+Focus effects that merge with Layout

Focus and context (cont.)

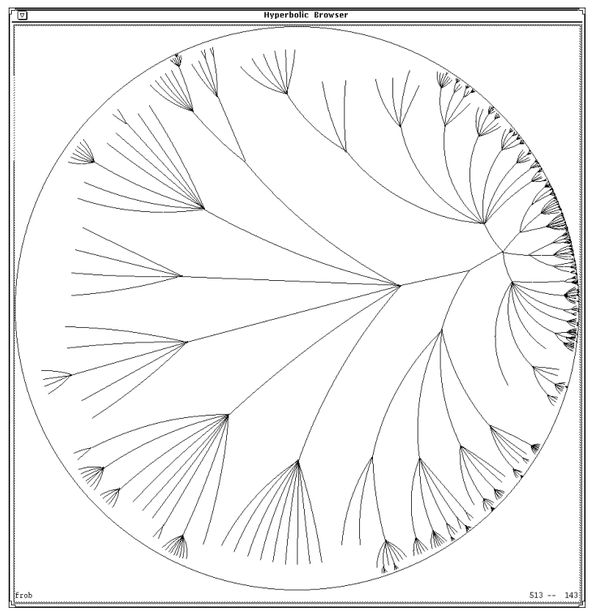
The hyperbolic layout does just that

- Whether in 2D or 3D, it produces a distorted view with a focal point at some fixed location in the graph, in a fish-eye view sense



However, proper interaction with the view means changing the position of the center point within the graph

- The root has been shifted to the right, putting more focus on the nodes that were toward the left



References

I. Herman, G. Melançon and S. Marshall, “Graph visualization and navigation in information visualization: A survey”, *IEEE Transactions on Visualization and Computer Graphics*, 6(1), 24-43, 2000

Manuel Lima, *Visual complexity: Mapping patterns of information*. Princeton Architectural Press (2011)

Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for drawing graphs: An annotated bibliography. Computational Geometry: Theory and Applications, 4(5):235{282, 1994. http://dx.doi.org/10.1016/0925-7721(94)00014-X.

David Easley and Jon Kleinberg. Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press, 2010. http://www.cs.cornell.edu/home/kleinber/networks-book/.

References, continued

Michael Hahsler, Kurt Hornik, and Christian Buchta. Getting things in order: An introduction to the r package seriation. Journal of Statistical Software, 25(3):1{34, 2008. URL http://www.jstatsoft.org/v25/i03.

Robert Kosara. Graphs beyond the hairball, 2012. URL http://eagereyes.org/techniques/graphs-hairball. Blog entry.

Carlos E. Scheidegger. So you want to look at a graph, part 1, 2012. URL http://cscheid.net/blog/so_you_want_to_look_at_a_graph__part_1. Blog entry.

Roberto Tamassia, editor. Handbook of Graph Drawing and Visualization. CRC Press, 2013. http://cs.brown.edu/~rt/gdhandbook/.