树和图的可视化

信息可视化

曹楠（教授）
https://idvxlab.com
同济大学

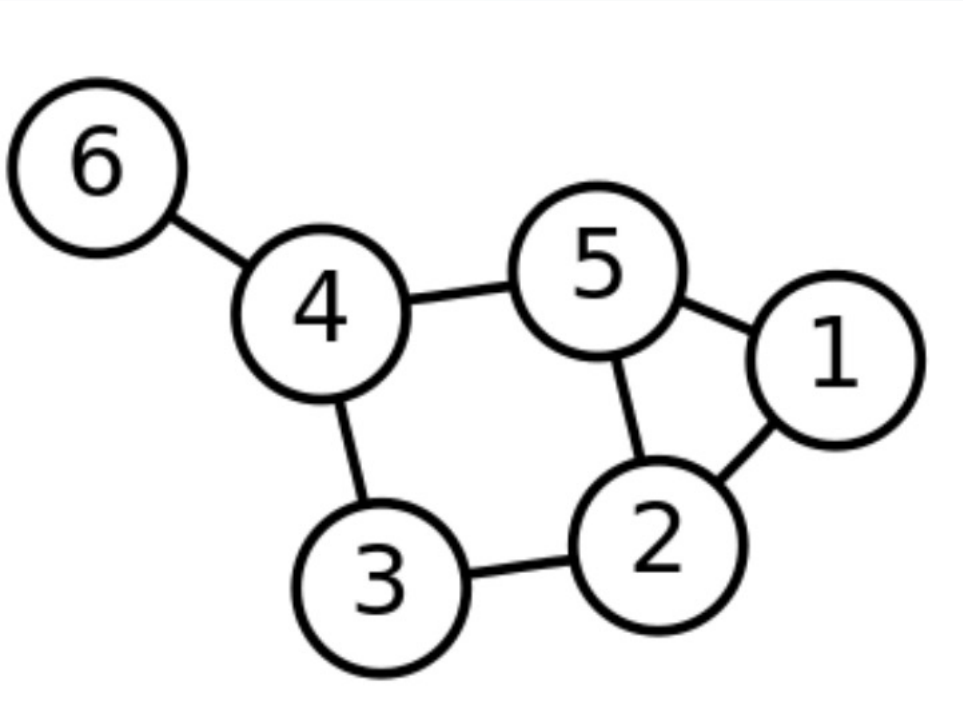
智能大数据可视化实验室

课程大纲

- 多维度数据的可视化
- 树的可视化
  - 点线图（Node-Link Diagram）
  - 邻接图（Adjacency Diagram）
  - 包含图（Enclosure Diagram）
- 图的可视化

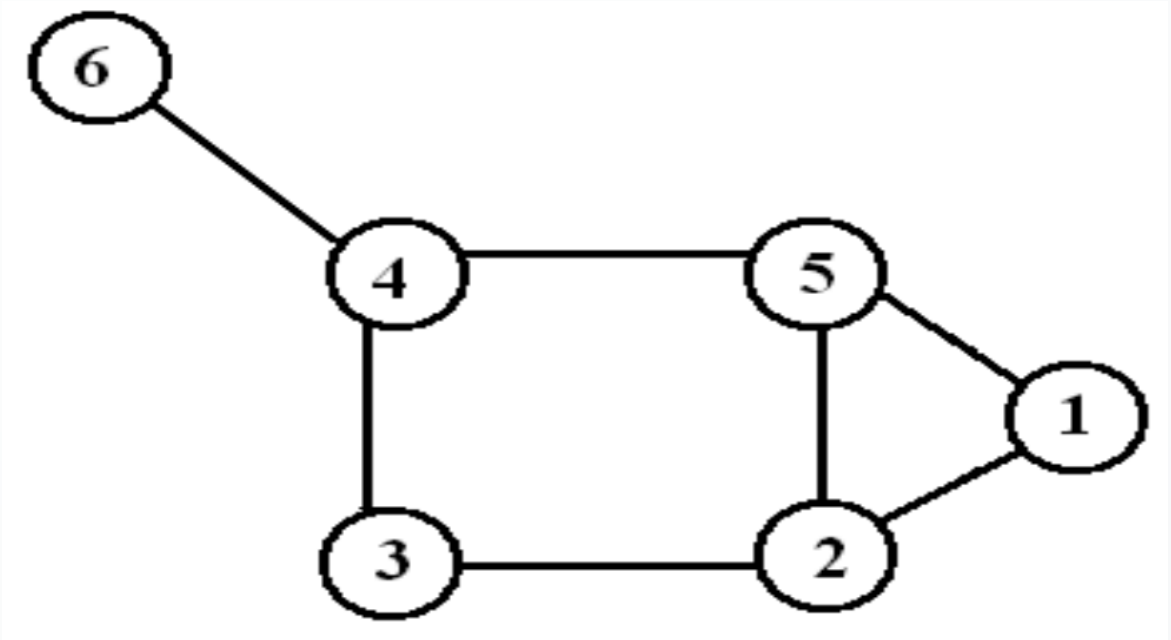
Graph

- A data structure that consists of a set of nodes (vertices) and a set of edges that relate the nodes to each other.
- G <V , E>
  - V : vertices
  - E : edges / links



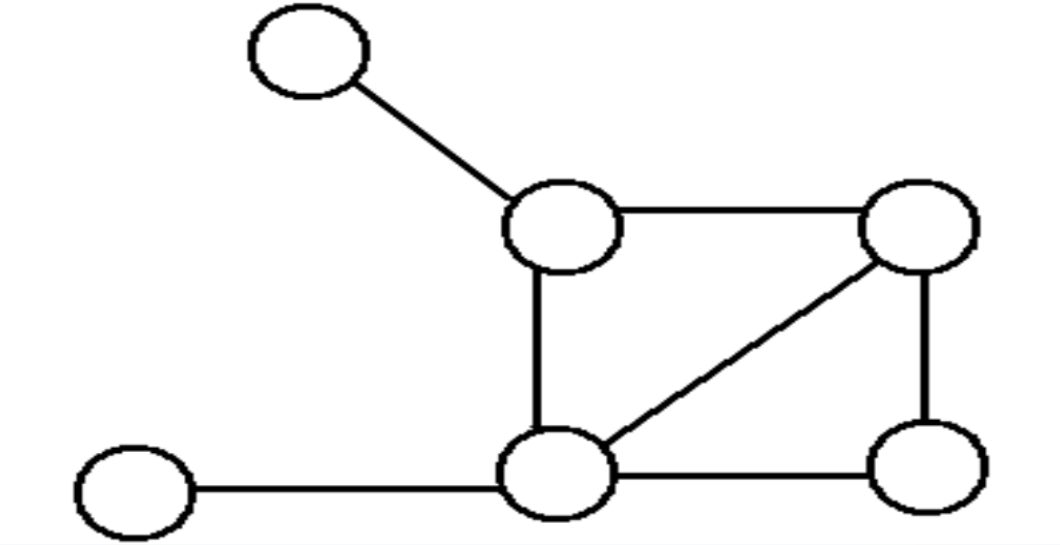
Example

- V:={1,2,3,4,5,6}
  E:={{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}



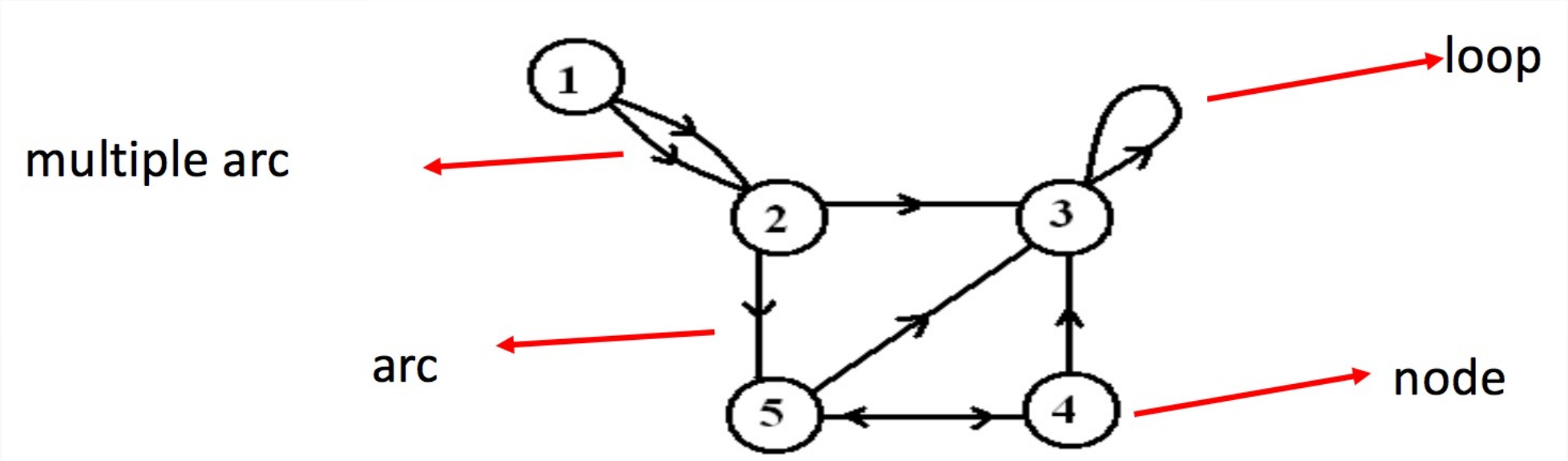
Simple Graph

- Simple graphs are graphs without multiple edges or self-loops.



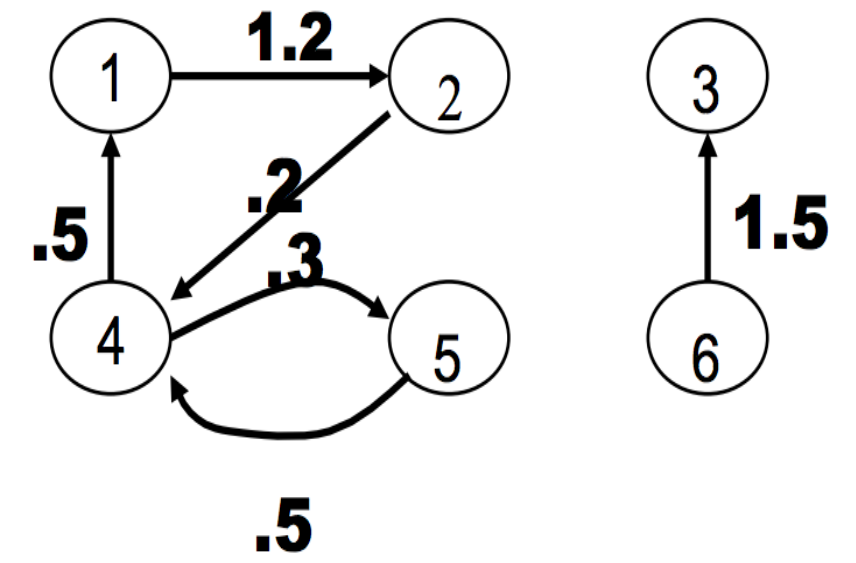
Directed Graph

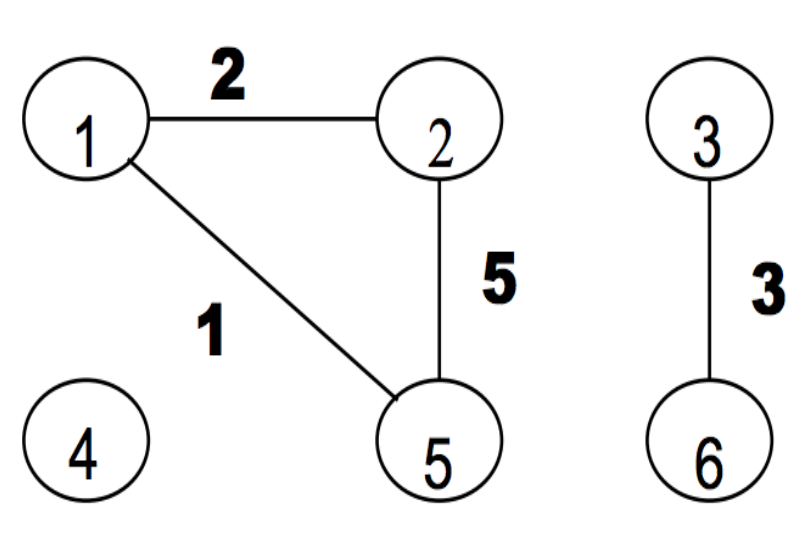
- Edges have directions
  - An edge is an ordered pair of nodes



Weighted Graph

- Weighted graph is a graph for which each edge has an associated **weight**, usually given by a **weight function** w: E->**R**.





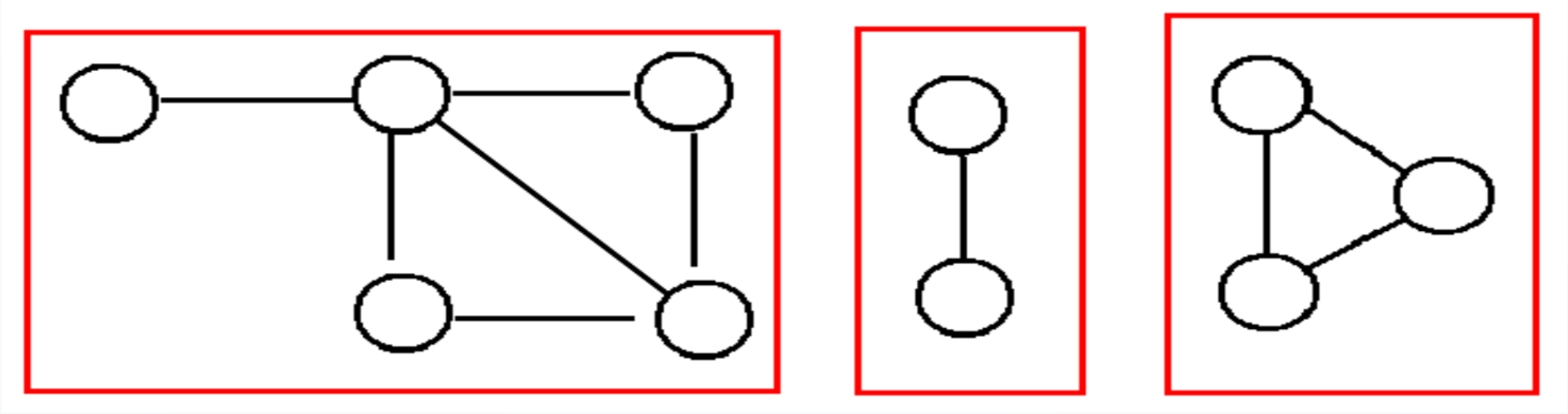
Connectivity

- A graph is connected if
  - you can get from any node to any other by following a sequence of edges OR
  - any two nodes are connected by a path.

- A directed graph is strongly connected if there is a directed path from any node to any other node.

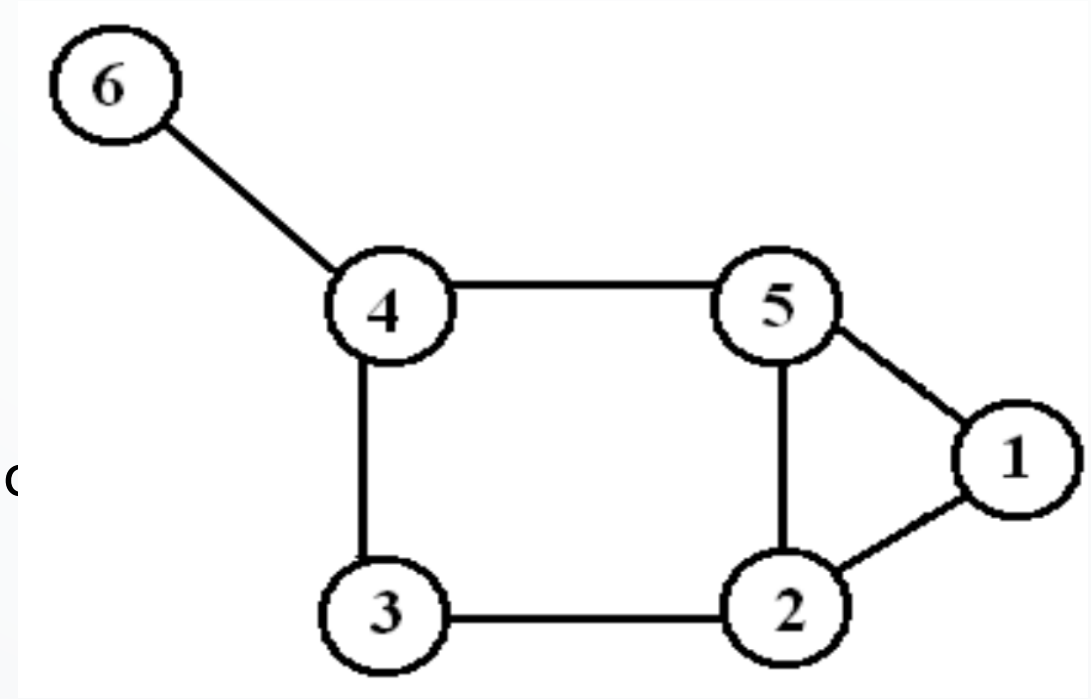
Component

- Every disconnected graph can be split up into a number of connected **components**.



Degree

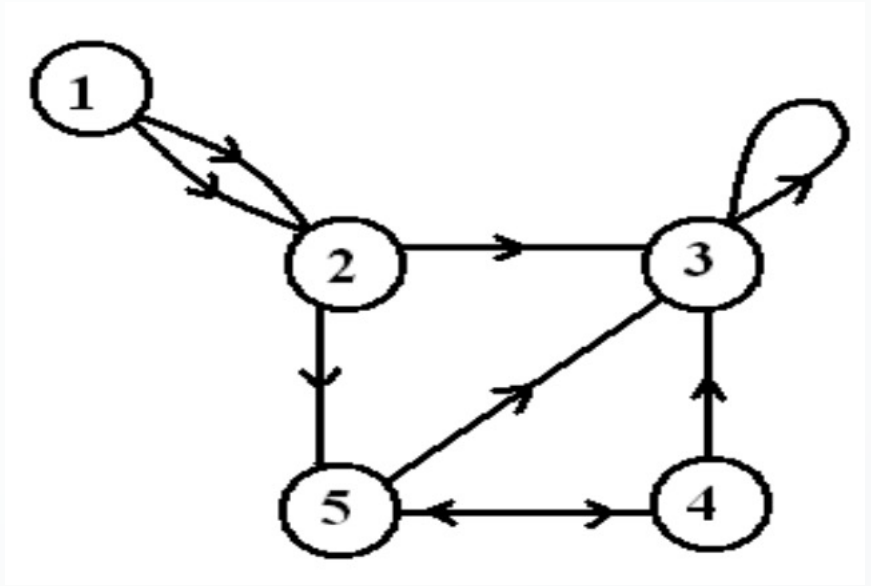
- Number of edges incident on a node



- What is the degree o

Degree (Directed Graph)

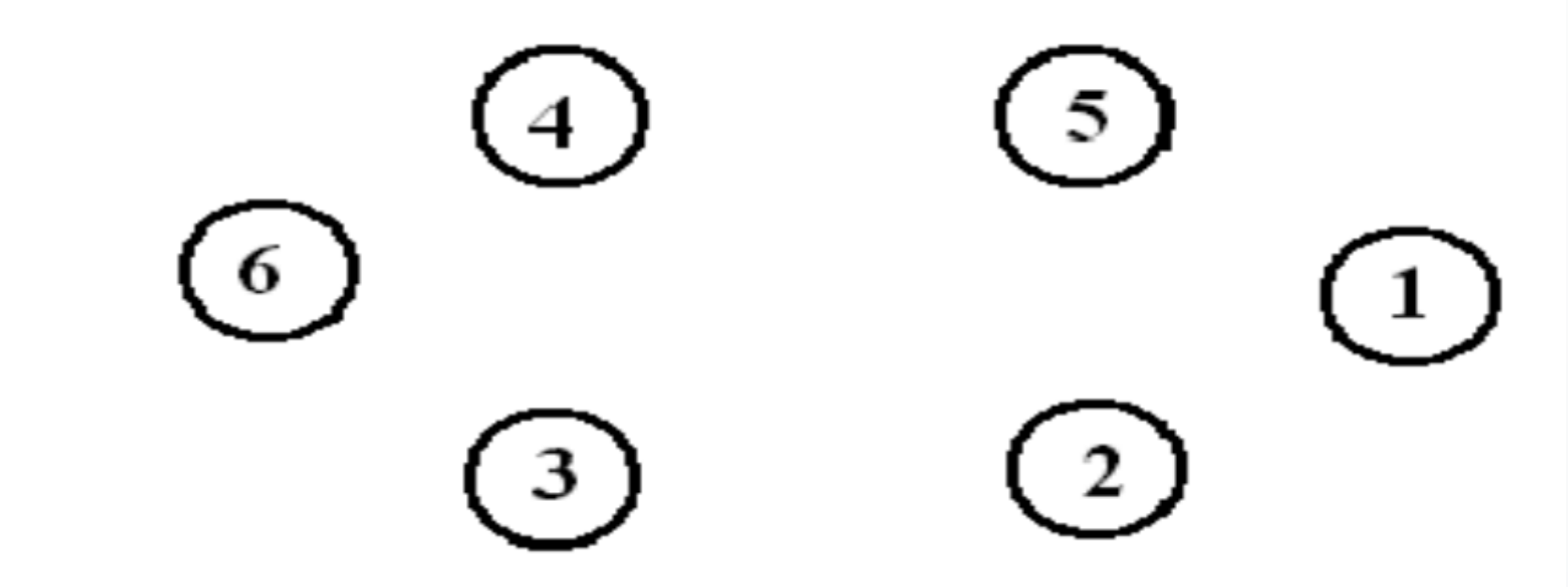
- In-degree: Number of edges entering
- Out-degree: Number of edges leaving
- Degree = indeg + outdeg



Indeg(2)=?
Outdeg(2)=?
Degree(3)=?

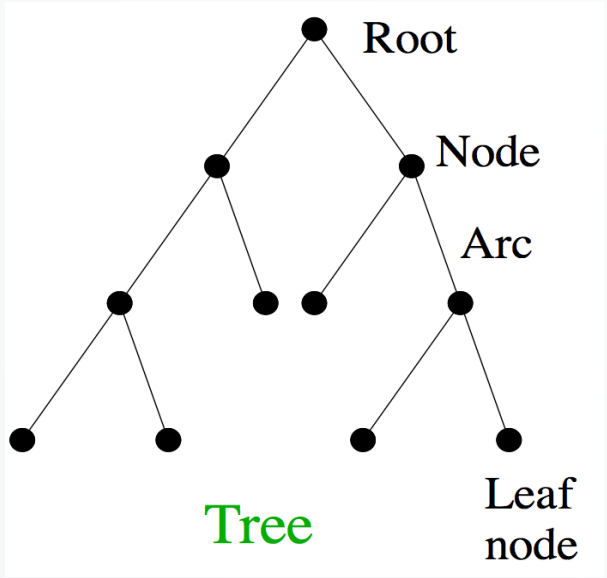
Types of Graphs

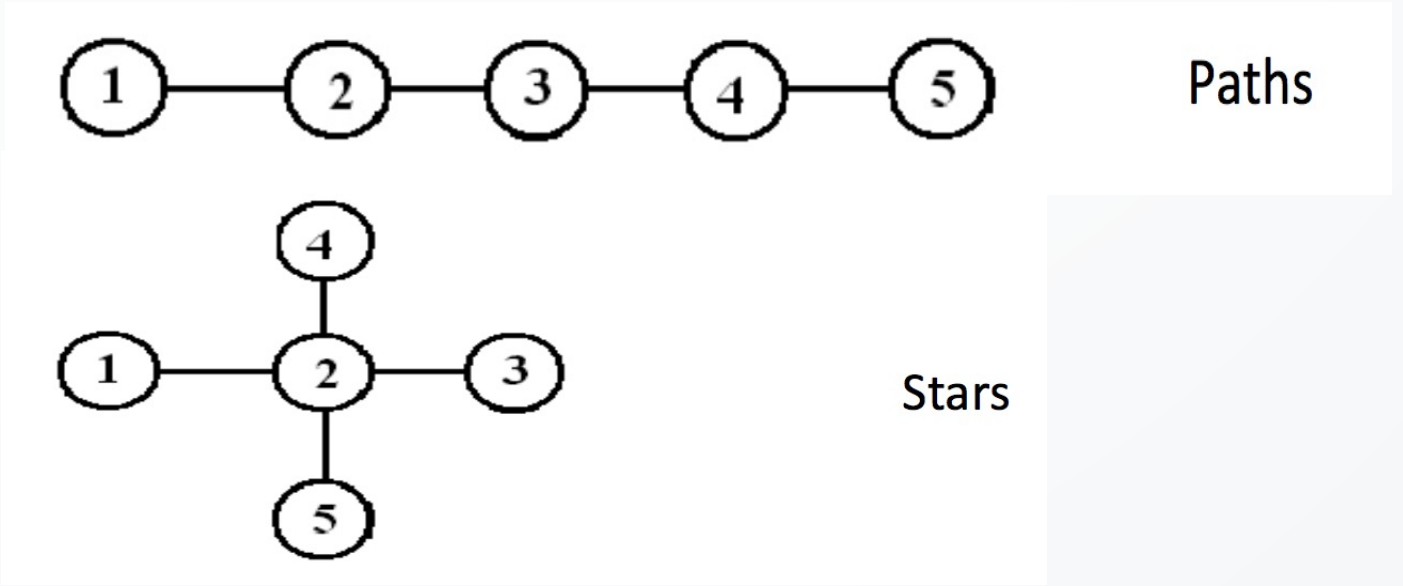
- Empty Graph / Edgeless Graph



Types of Graphs

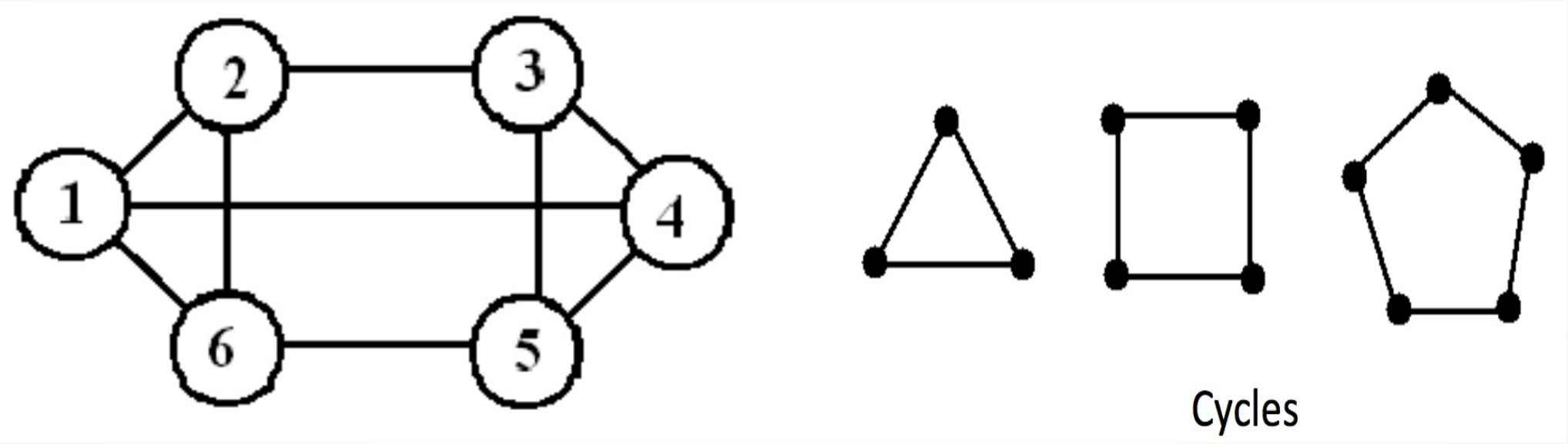
- Trees
  - Connected Acyclic Graph
  - Two nodes have *exactly* one path between them





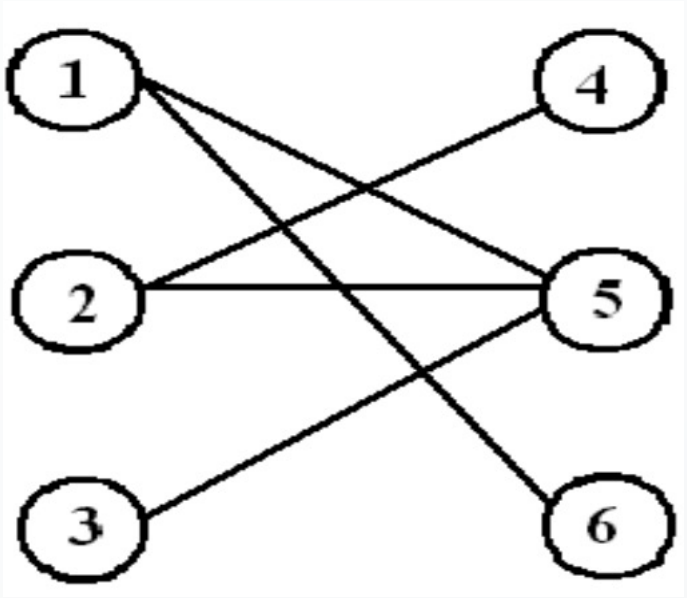
Types of Graphs

- Regular Graph
  - Connected Graph
  - All nodes have the same degree



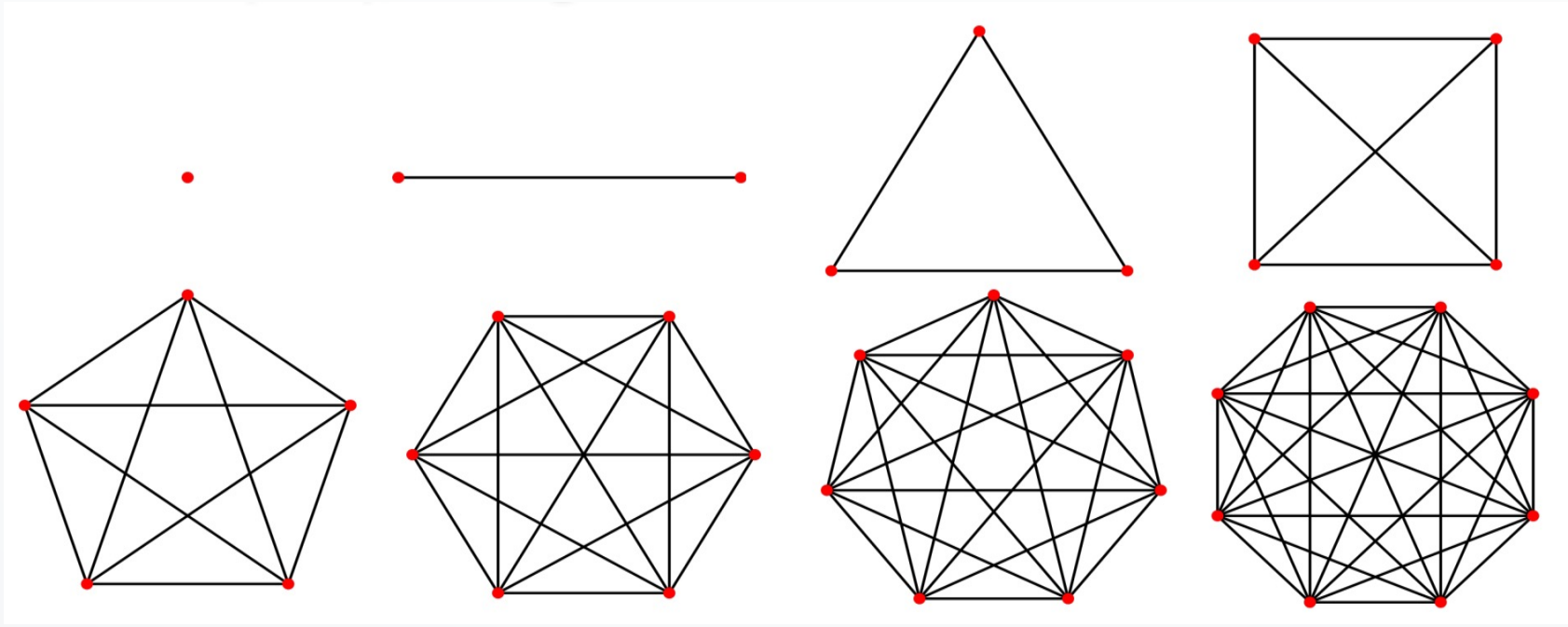
Types of Graphs

- Bipartite Graph
  - V can be partitioned into 2 sets V1 and V2 such that (u,v)∈E implies
    - either u∈V1 and v∈V2
    - – OR v∈V1 and u∈V2.



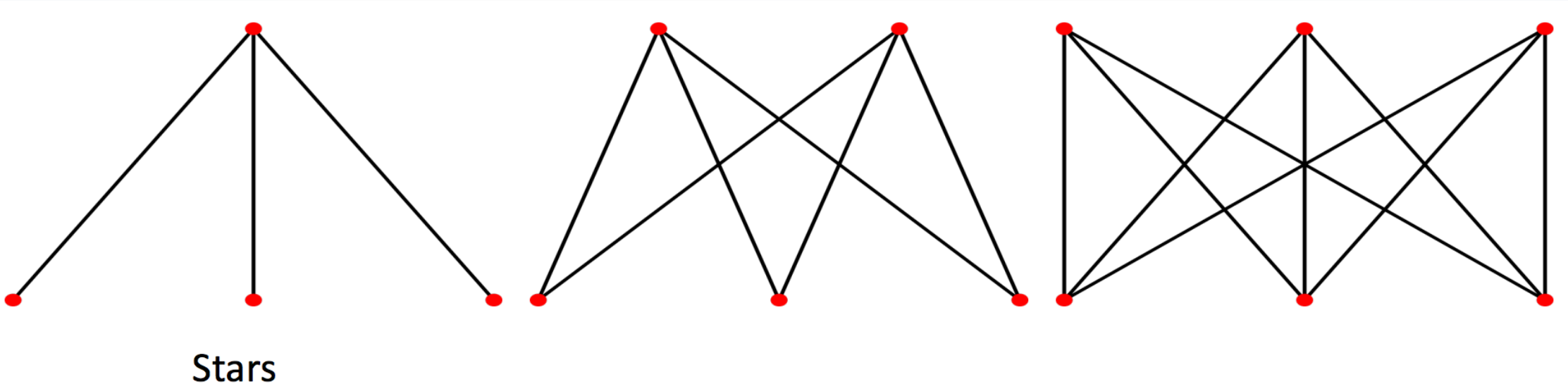
Types of Graphs

- Complete Graph
  - Every pair of vertices are adjacent
  - Has n(n-1)/2 edges



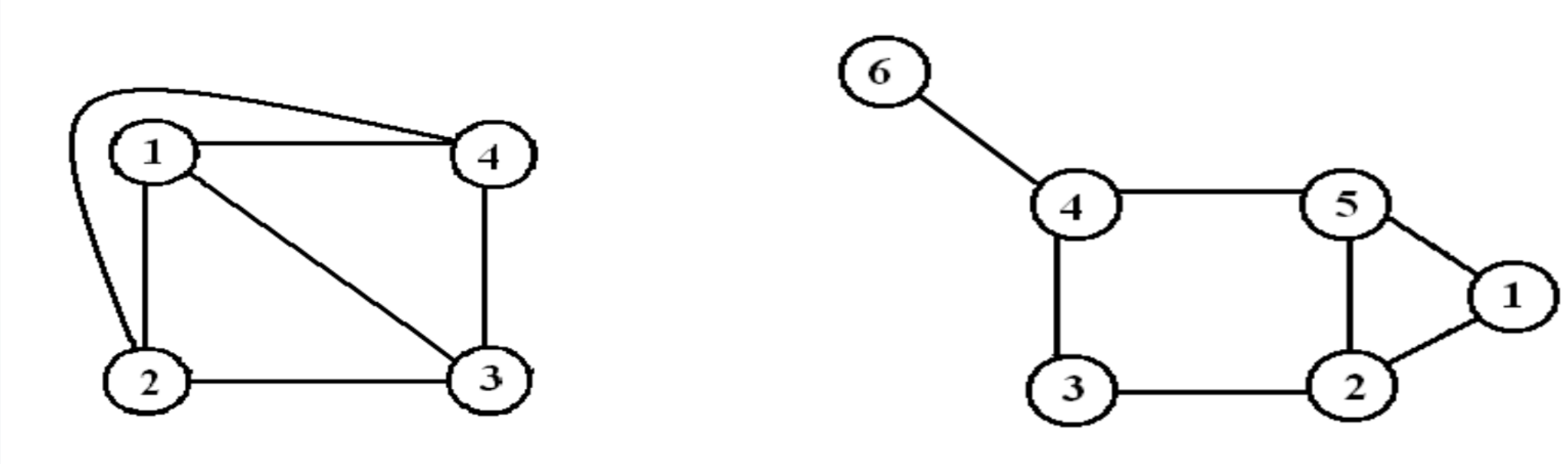
Types of Graphs

- Complete Bipartite Graph
  - Bipartite Variation of Complete Graph
  - Every node of one set is connected to every other node on the other set



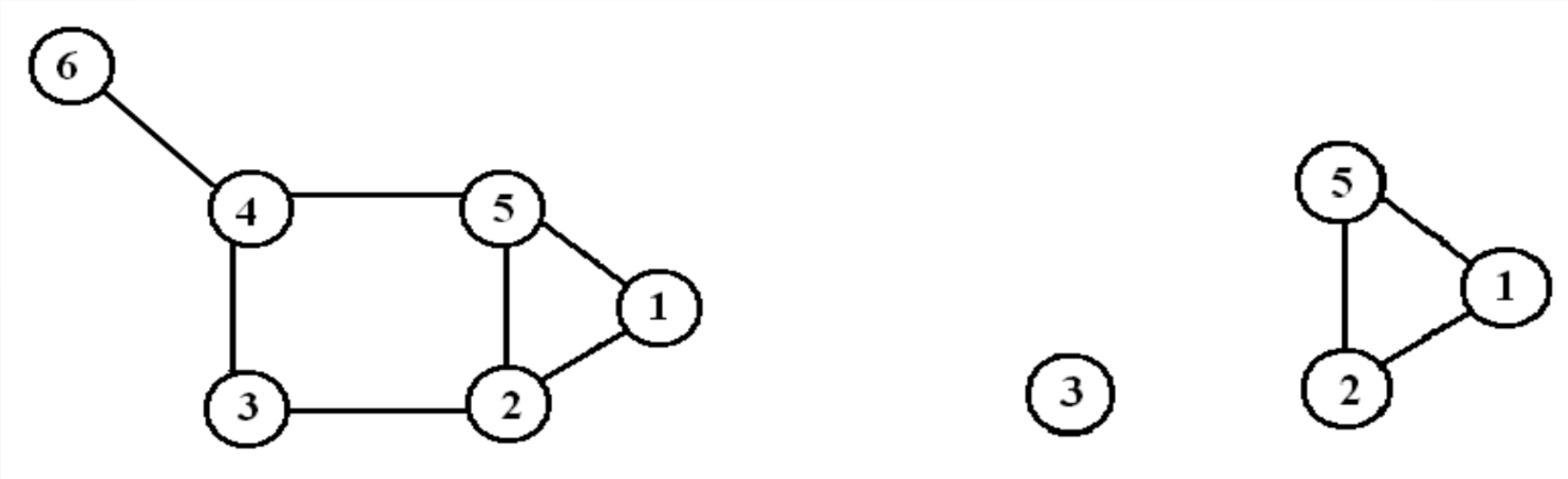
Type of Graphs

- Planar Graph
  - Can be drawn on a plane such that no two edges intersect



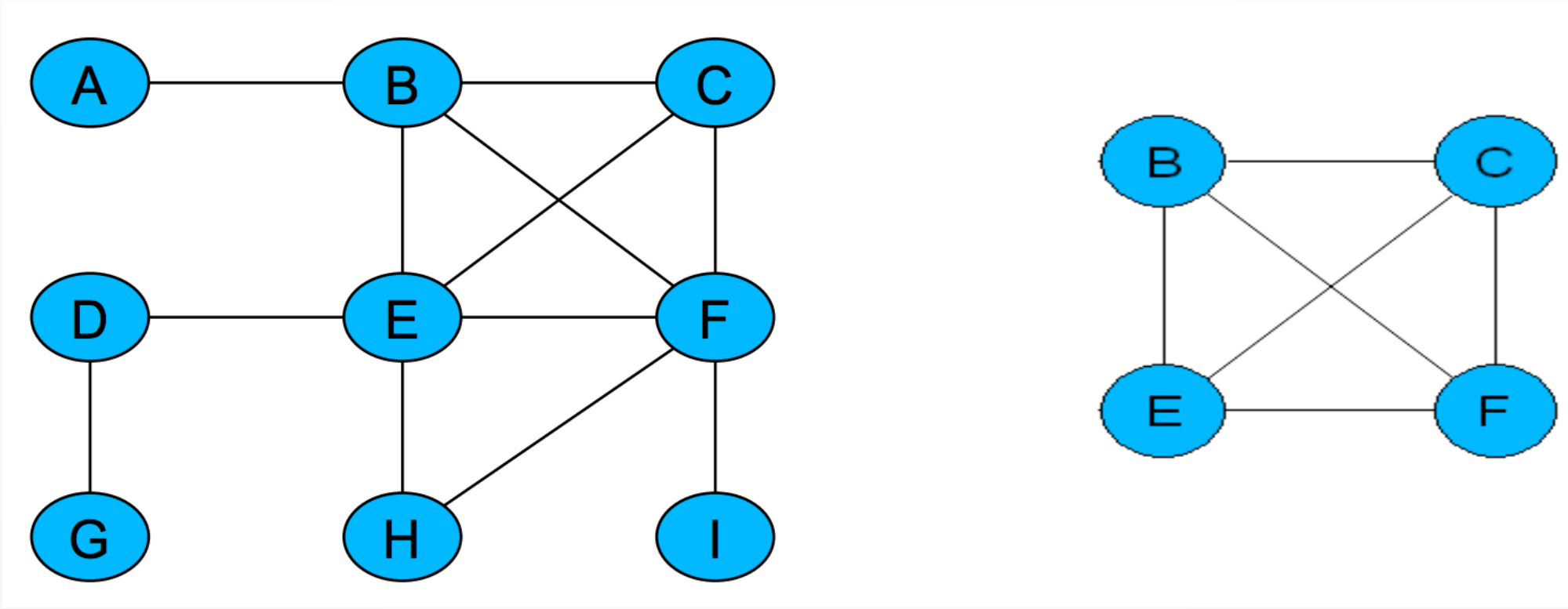
Subgraph

- Vertex and edge sets are subsets of those of G
  - a supergraph of a graph G is a graph that contains G as a subgraph.



Clique

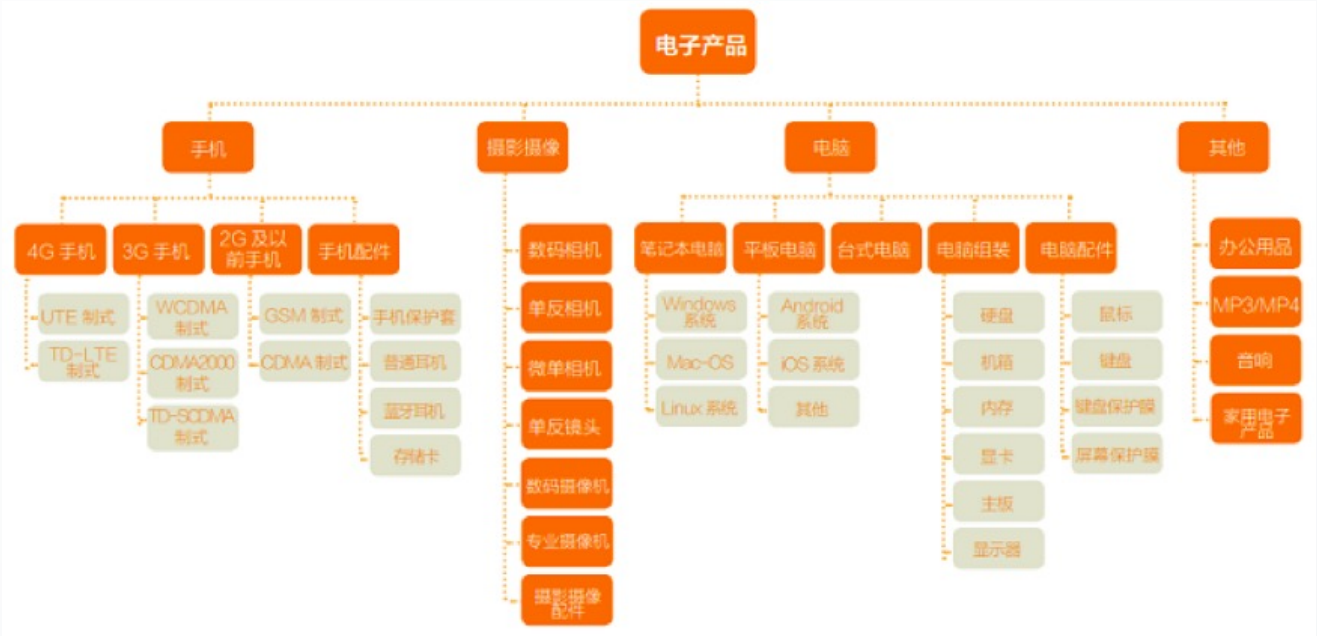
- A **clique** is a maximum complete connected subgraph.



树的可视化

- ”树“是一种基本的数据结构，很多数据中存在层次结构，例如
  - 公司的组织结构图，电脑中的文件目录管理结构，商品的层次化分类等

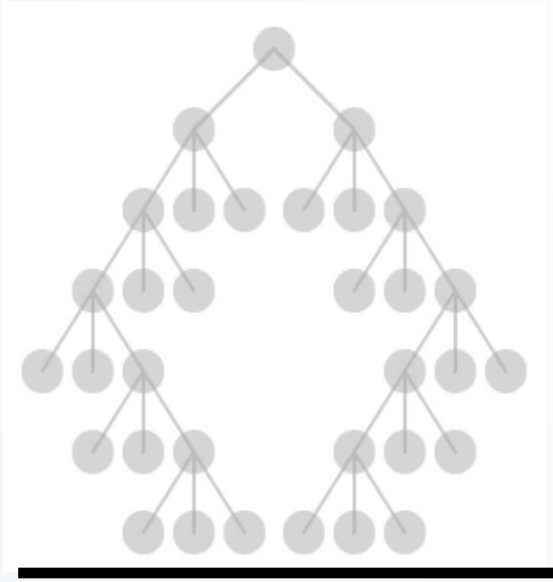
- ”树“的可视化要求必须能够清晰展示层次结构，有三种基本的可视化方式
  - 点线图（Node-Link Diagram）
  - 邻接图（Adjacency Diagram）
  - 包含图（Enclosure Diagram）



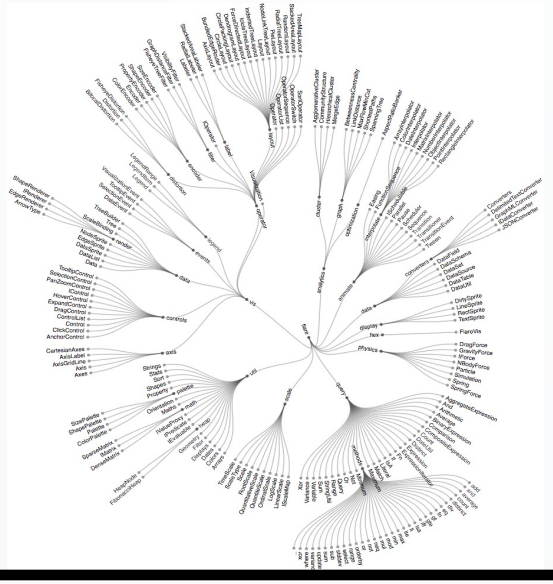
树的可视化

- 点线图（ Node-Link Diagram ）
  - 一个 整洁的 树的布局（Tidy Tree）需要做到： （1）构图紧凑； （2）相同层次的节点被放置在相同的可视化层级之上； （3）父亲节点在子节点的中心

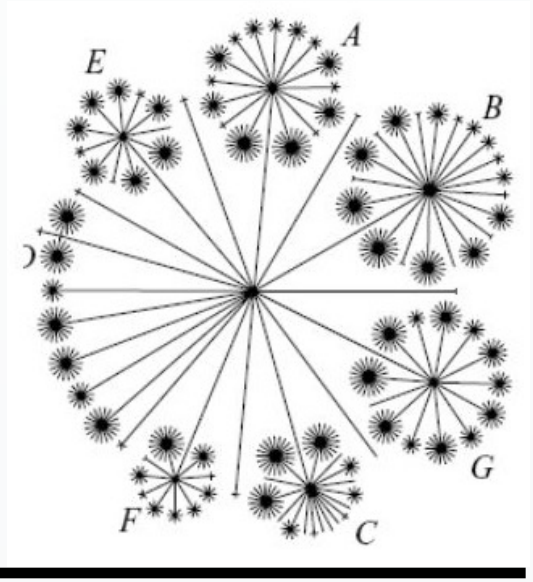
Hierarchy Layout



Radial Layout

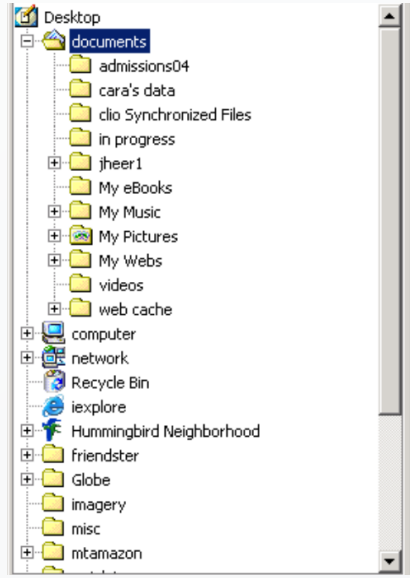


Circle Layout



Tidy Tree

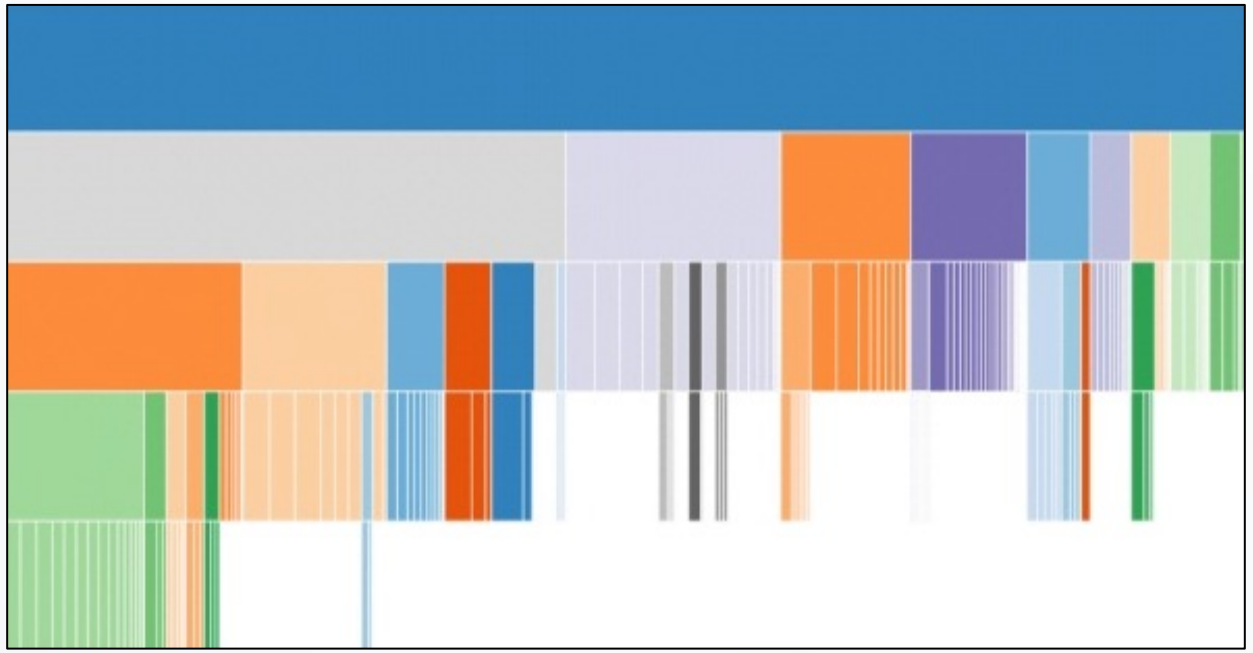
Intended Layout



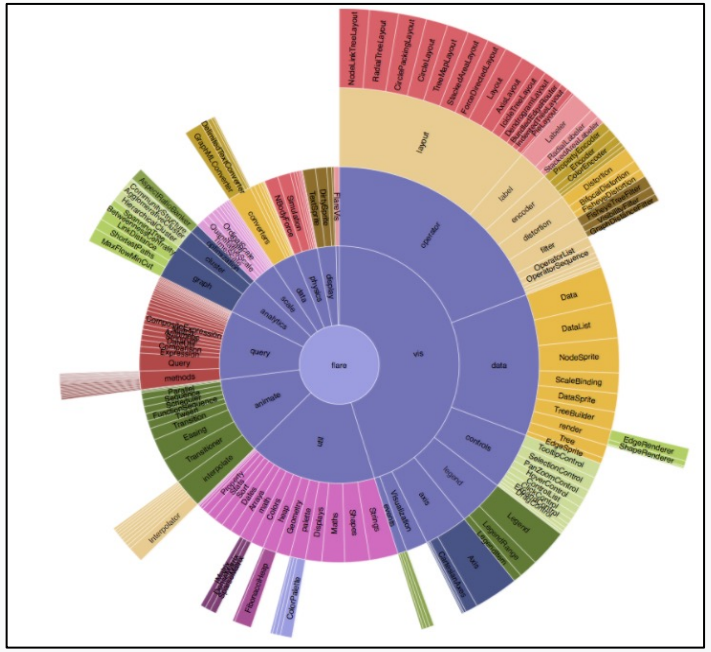
非 Tidy Tree

树的可视化

- 邻接图（ Adjacency Diagram ）
  - 通过可视化节点之间的相邻位置关系以展现树的拓扑结构
  - 节点的大小，宽度，可以用来展现另外一个数据维度



Icicle tree

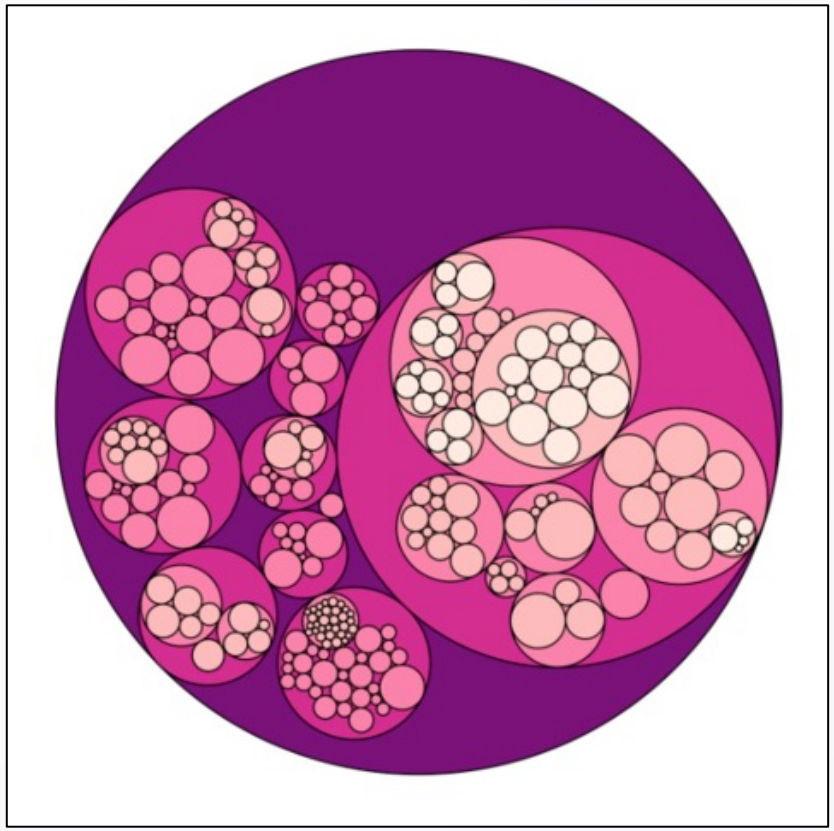


Sunburst Tree

树的可视化

- 包含图（ Enclosure Diagram）
  - 通过包含关系来展现树中的层次结构

Treemap (树图)



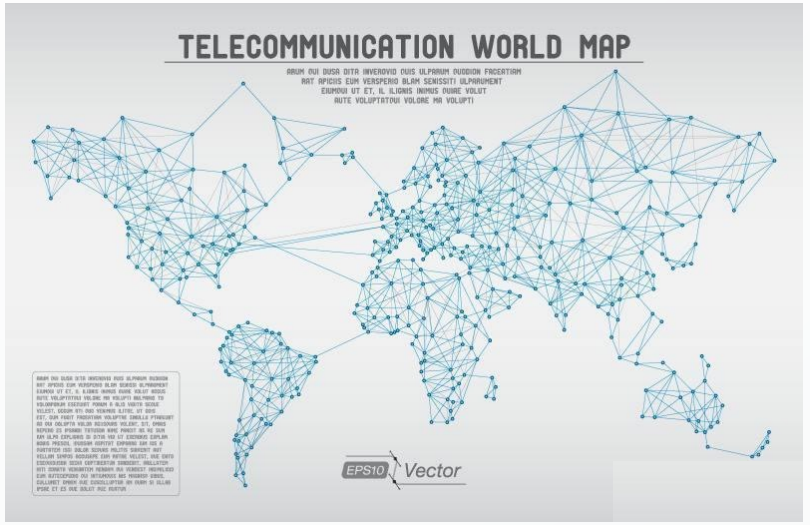
Circle Packing

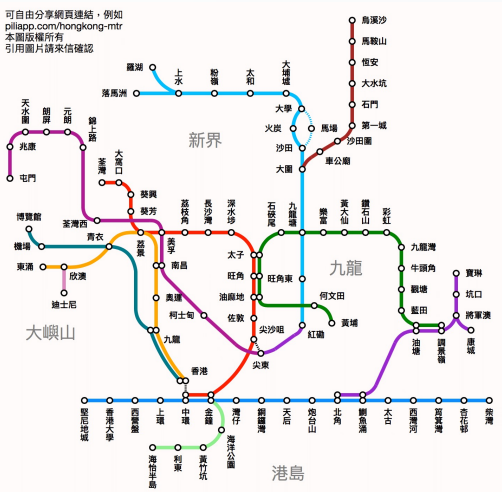
课程大纲

- 多维度数据的可视化
- 树的可视化
- 图的可视化
  - 简介及分类
  - 力导向图及布局
  - 邻接矩阵
  - 混合可视化方法

图的可视化

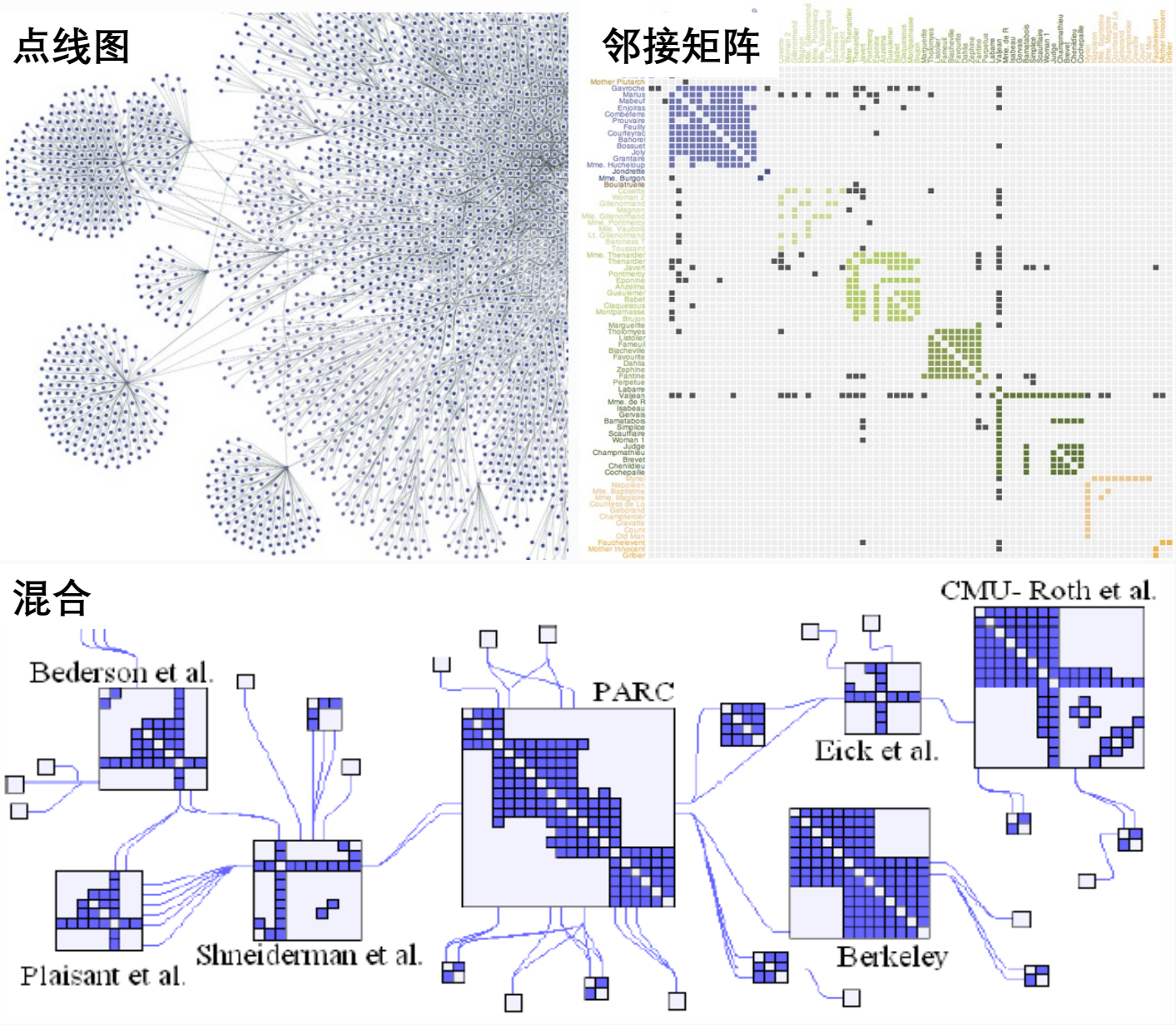
- 图是最普遍的数据结构，几乎无处不在，通信网络，互联网，地铁网
- 对于图结构的可视化可以追溯到100多年前，而现代有关图可视化的研究开始于上世纪70年代
- 今天围绕着图结构，在不同的应用领域，诞生了各种各样丰富多彩的可视化设计





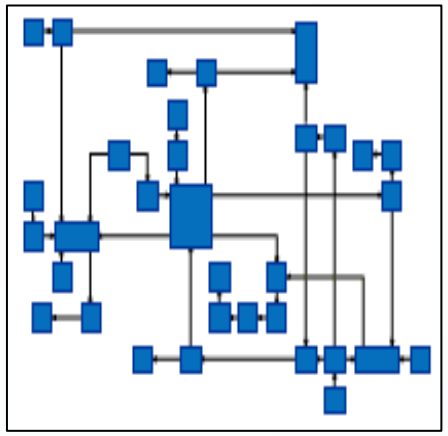
图的可视化

- 图可视化的基本形式
  - 点线图（Node-Link Diagram）
  - 邻接矩阵（Adjacency Matrix）
  - 混合可视化方法（NodeTrix）

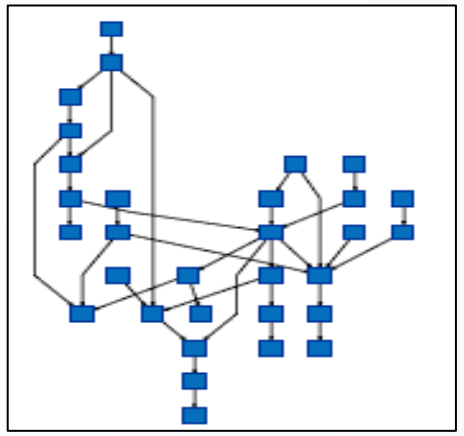


图的可视化

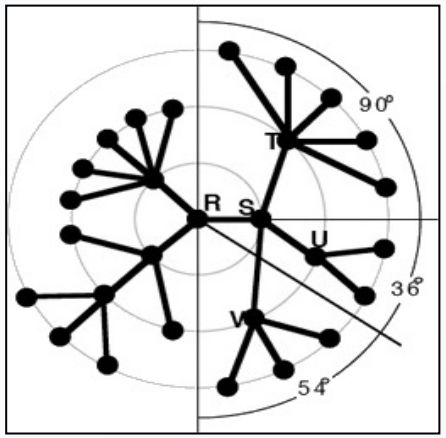
- 点线图（Node-Link Diagram）
  - 最直观的“图”表达方式
  - 布局问题是点线图可视化的关键
  - 布局问题：计算点在屏幕上的位置，以及边的绘制方式，从而使得尽可能的减少点的覆盖与线的交叉
  - 点线图的布局可以大致分为四类：
    直角折线布局（Orthogonal）、放射布局（Radial）、层次布局（Hierarchy）、力导向布局（Force-Directed）
  - 接下来我们将重点讲解 **力导向布局** 的计算方法



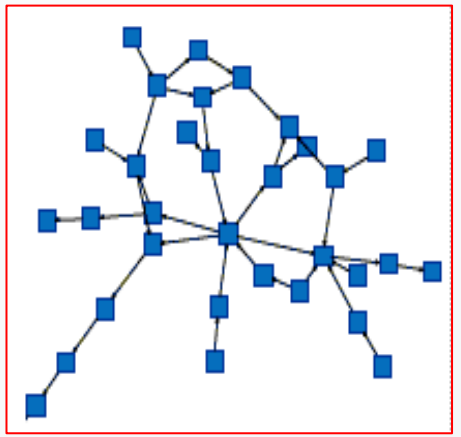
直角折线布局



层次布局图



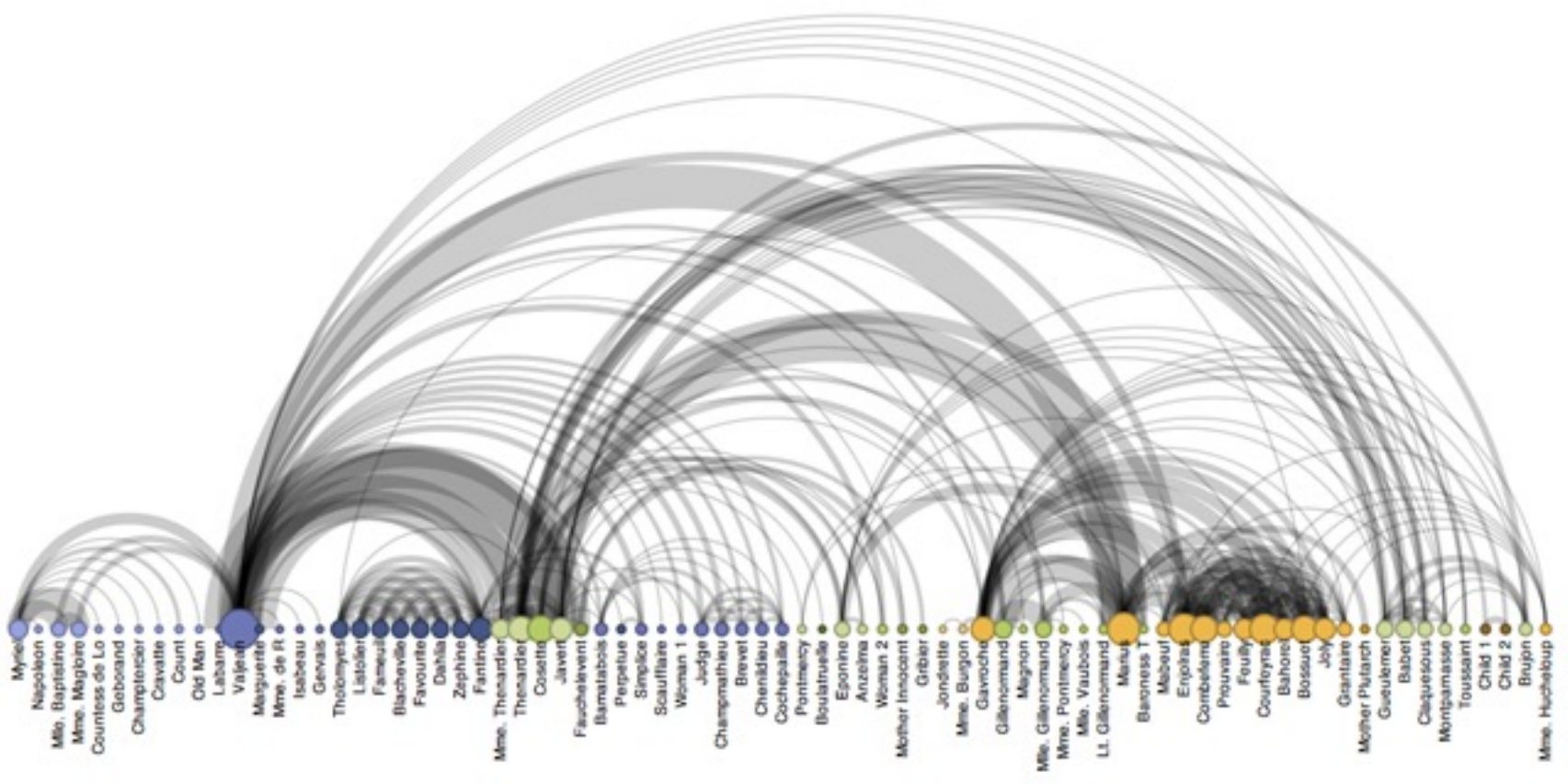
放射布局



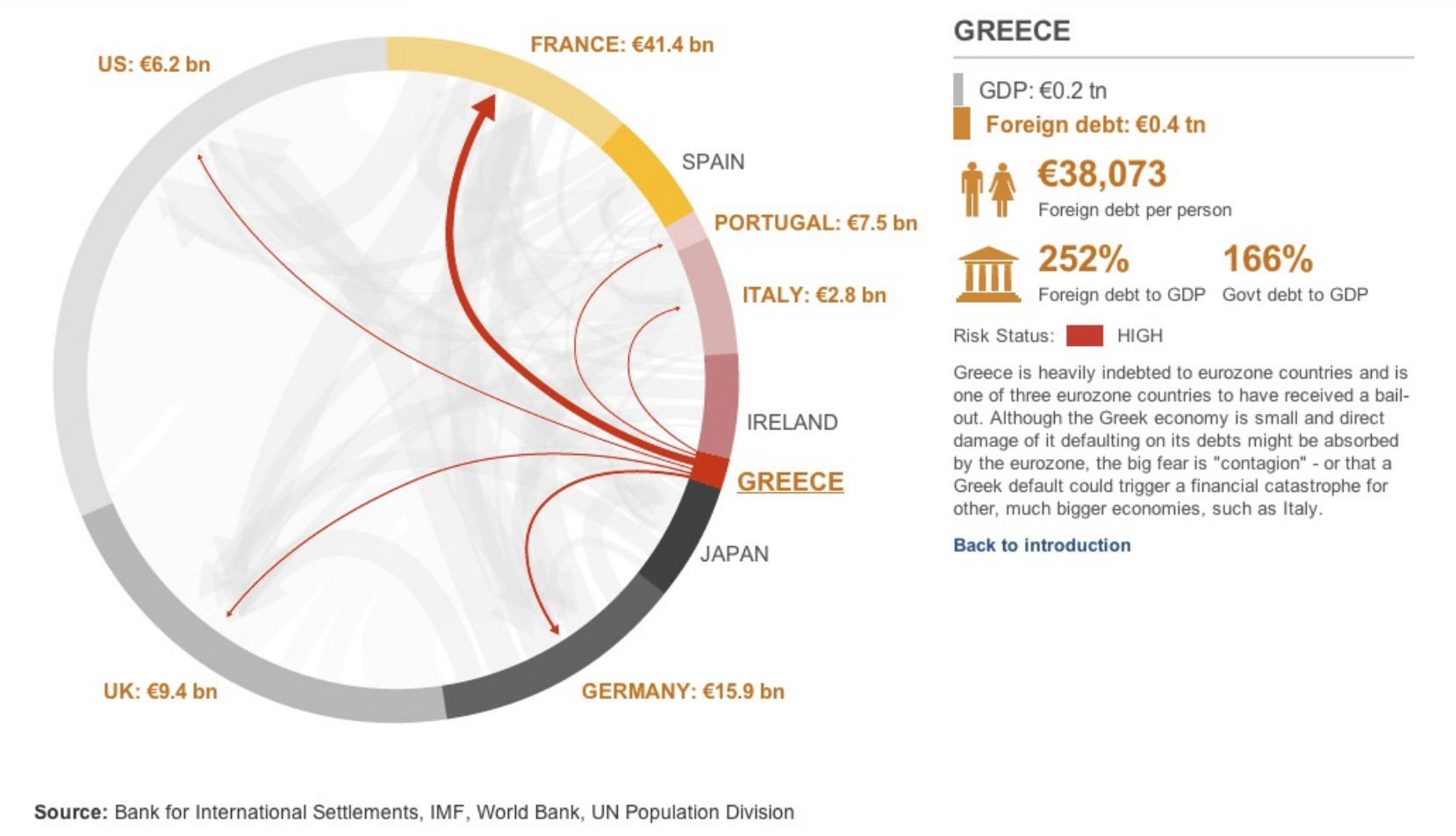
力导向布局

Various Node-Link Diagram Designs

Arc Diagram


Myriel
Napoleon
Mlle. Baptistine
Mme. Magloire
Countess de Lo
Geborand
Champtercier
Cravatte
Count
Old Man
Labarre
Valjean
Marguerite
Mme. de R
Isabeau
Gervais
Tholomyes
Listolier
Fameuil
Blacheville
Favourite
Dahlia
Zephine
Fantine
Mme. Thenardier
Thenardier
Cosette
Javert
Fauchelevent
Bamatabois
Perpetue
Simplice
Scaufflaire
Woman 1
Judge
Champmathieu
Brevet
Chenildieu
Cochepaille
Pontmercy
Boulatruelle
Eponine
Anzelma
Woman 2
Mother Innocent
Gribier
Jondrette
Mme. Burgon
Gavroche
Gillenormand
Magnon
Mlle. Gillenormand
Mme. Pontmercy
Mlle. Vaubois
Lt. Gillenormand
Marius
Baroness T
Mabeuf
Enjolras
Combeferre
Prouvaire
Feuilly
Courfeyrac
Bahorel
Bossuet
Joly
Grantaire
Mother Plutarch
Gueulemer
Babet
Claquesous
Montparnasse
Toussaint
Child 1
Child 2
Brujon
Mme. Hucheloup

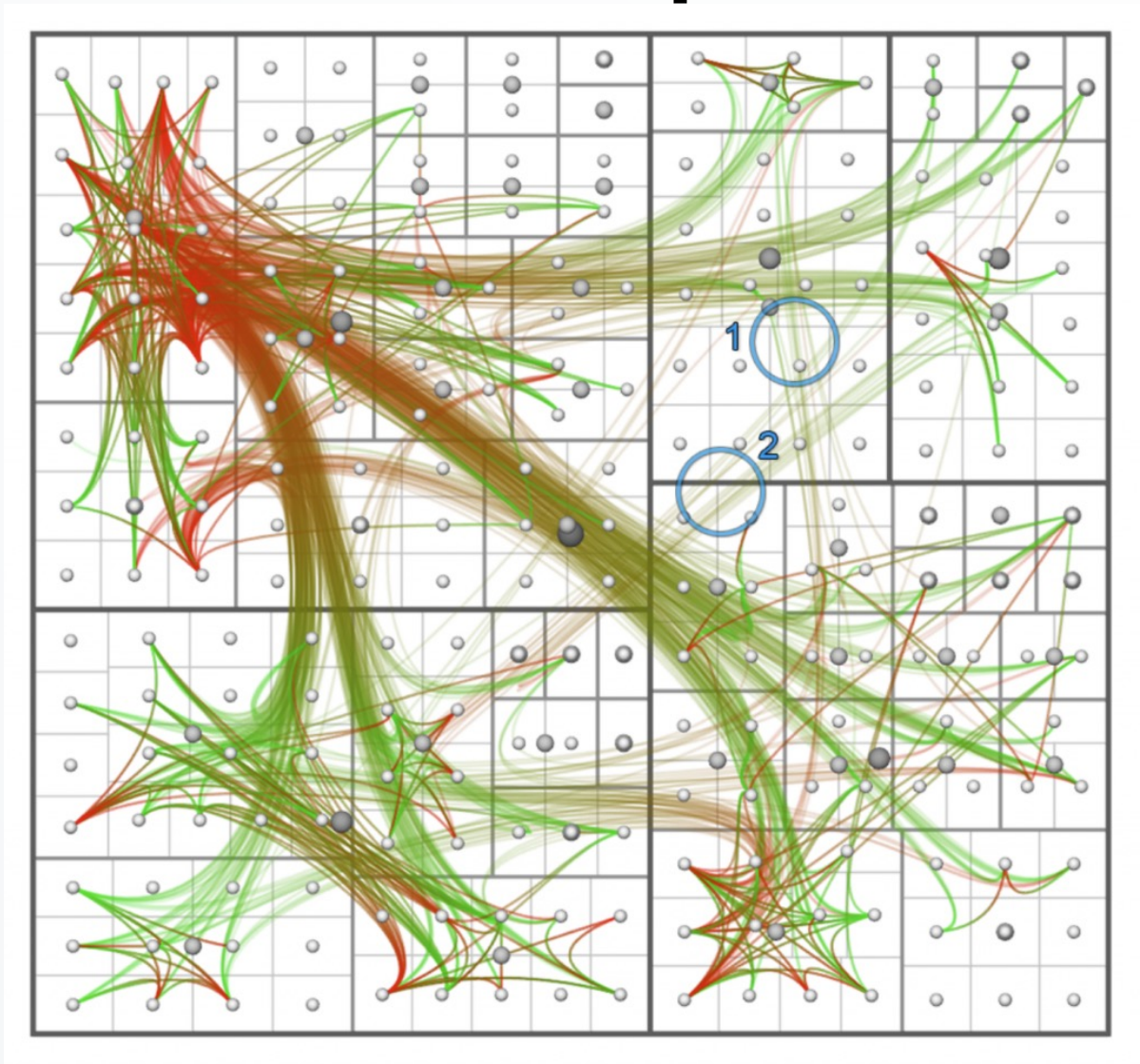
Arc Diagram

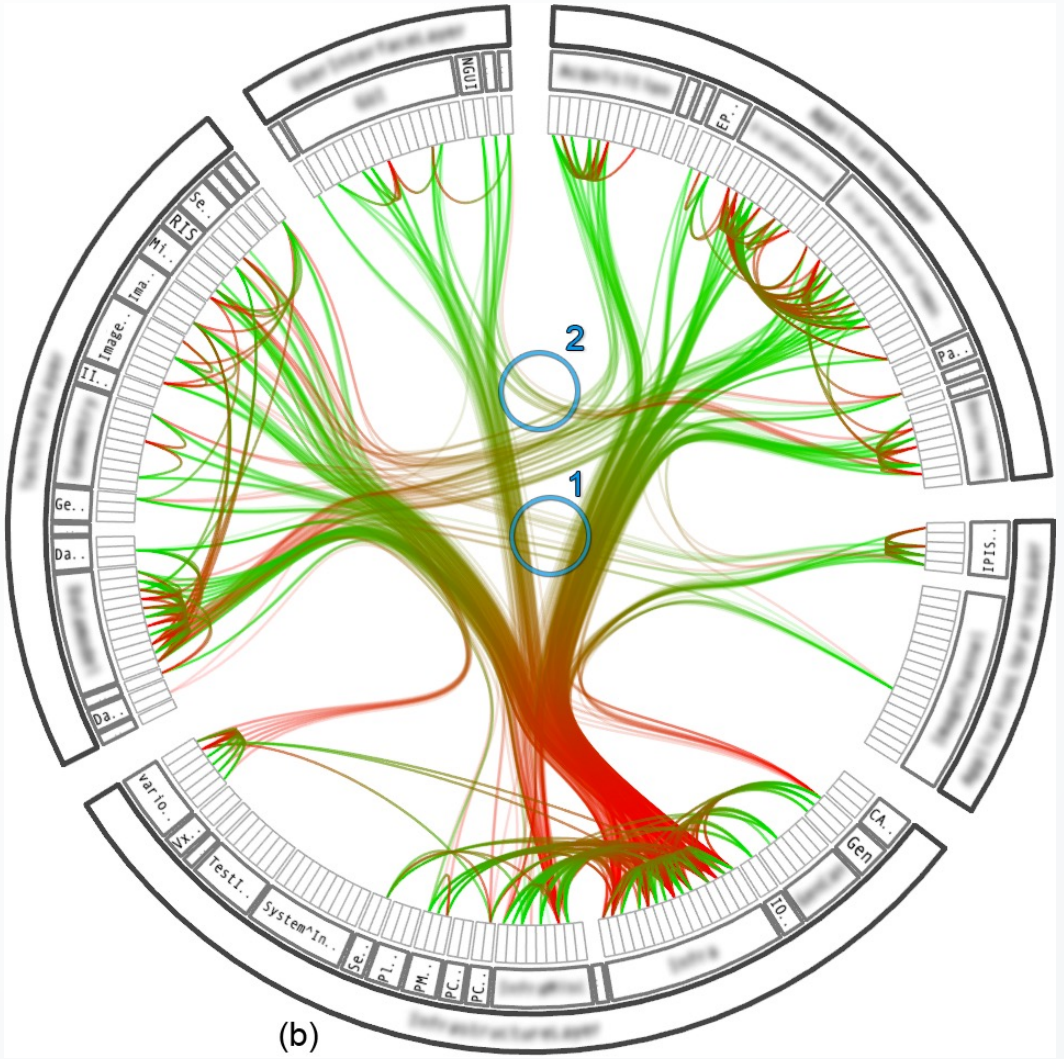


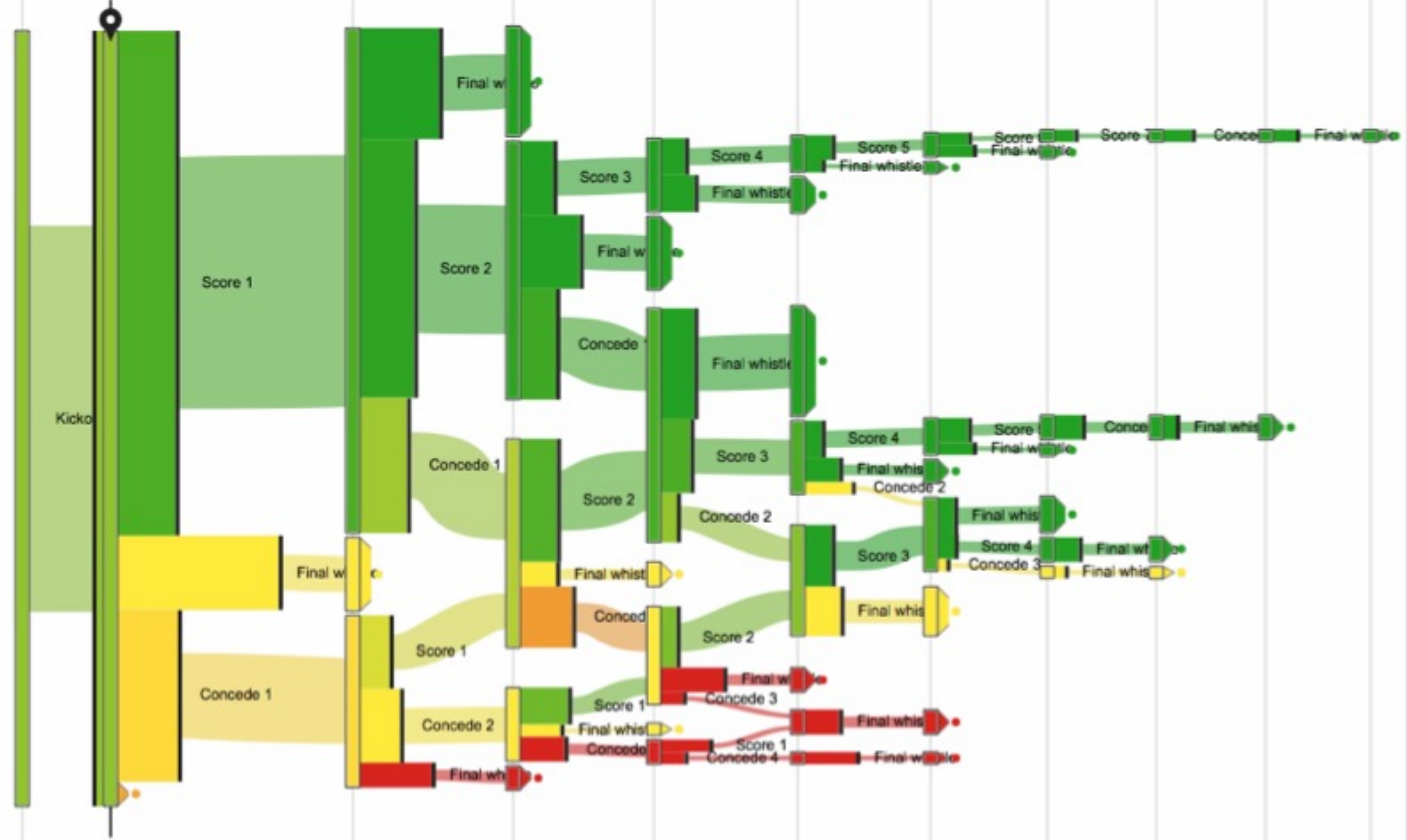
Eurozone debt web: Who owes what to whom?

http://www.bbc.co.uk/news/business-15748696

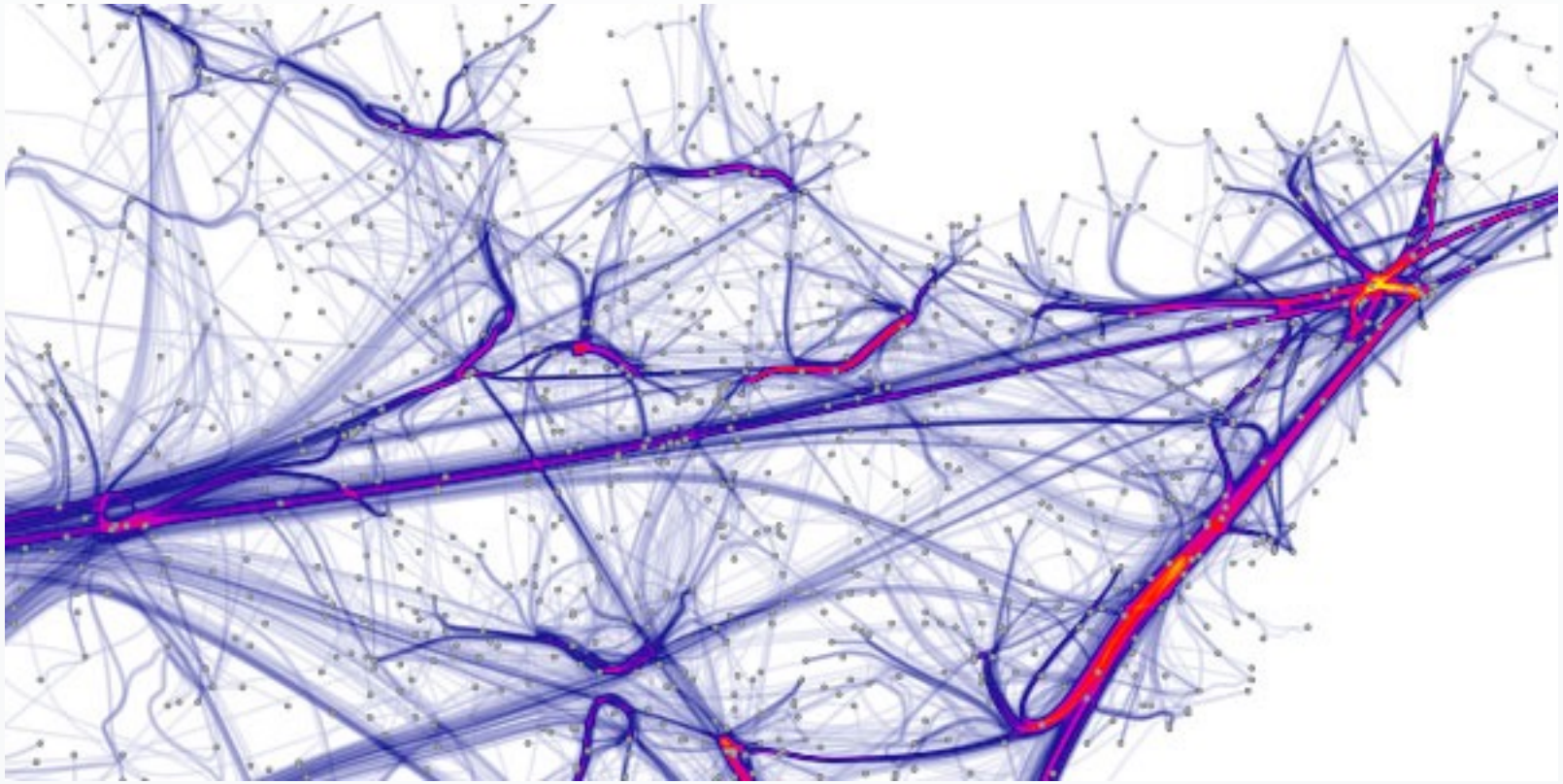
Hierarchical Graph





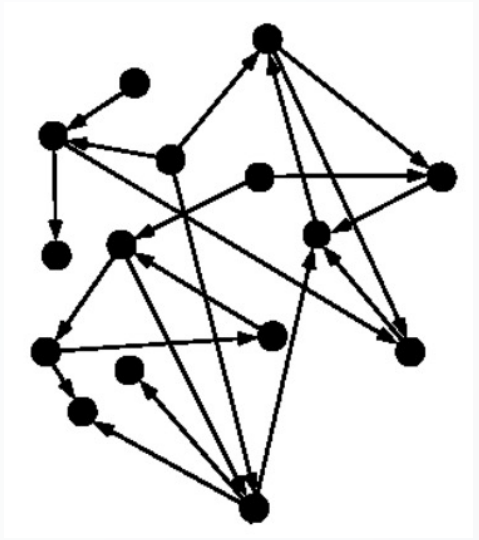

Kicko
Score 1
Concede 1
Final w
Score 2
Concede 1
Score 1
Concede 2
Final wh
Final w
Score 3
Final w
Concede
Score 2
Final whist
Conced
Score 1
Final whis
Concede
Score 4
Final whistle
Final whistle
Score 3
Concede 2
Score 2
Final w
Concede 3
Score 1
Concede 4
Score 5
Final whistle
Score 4
Final whis
Concede 2
Score 3
Final whis
Final whis
Final whis
Final w
Score 4
Concede 3
Final whis
Score
Conce
Final whis

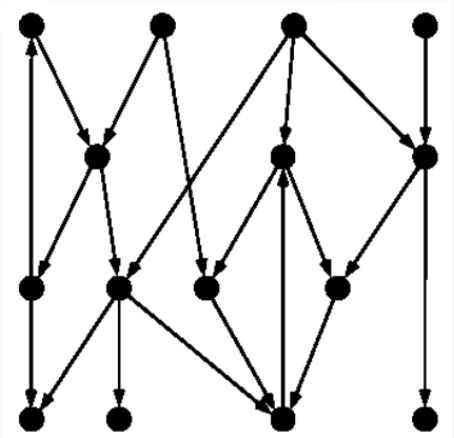
Graph Edge Bundling



Sugiyama Layout

- Drawing Conventions and Aesthetics
  - Edge pointing upward should be avoided
  - Nodes should be evenly distributed
  - Long edges should be avoided
  - Minimize edge crossing
  - Edges should be as straight vertical as possible

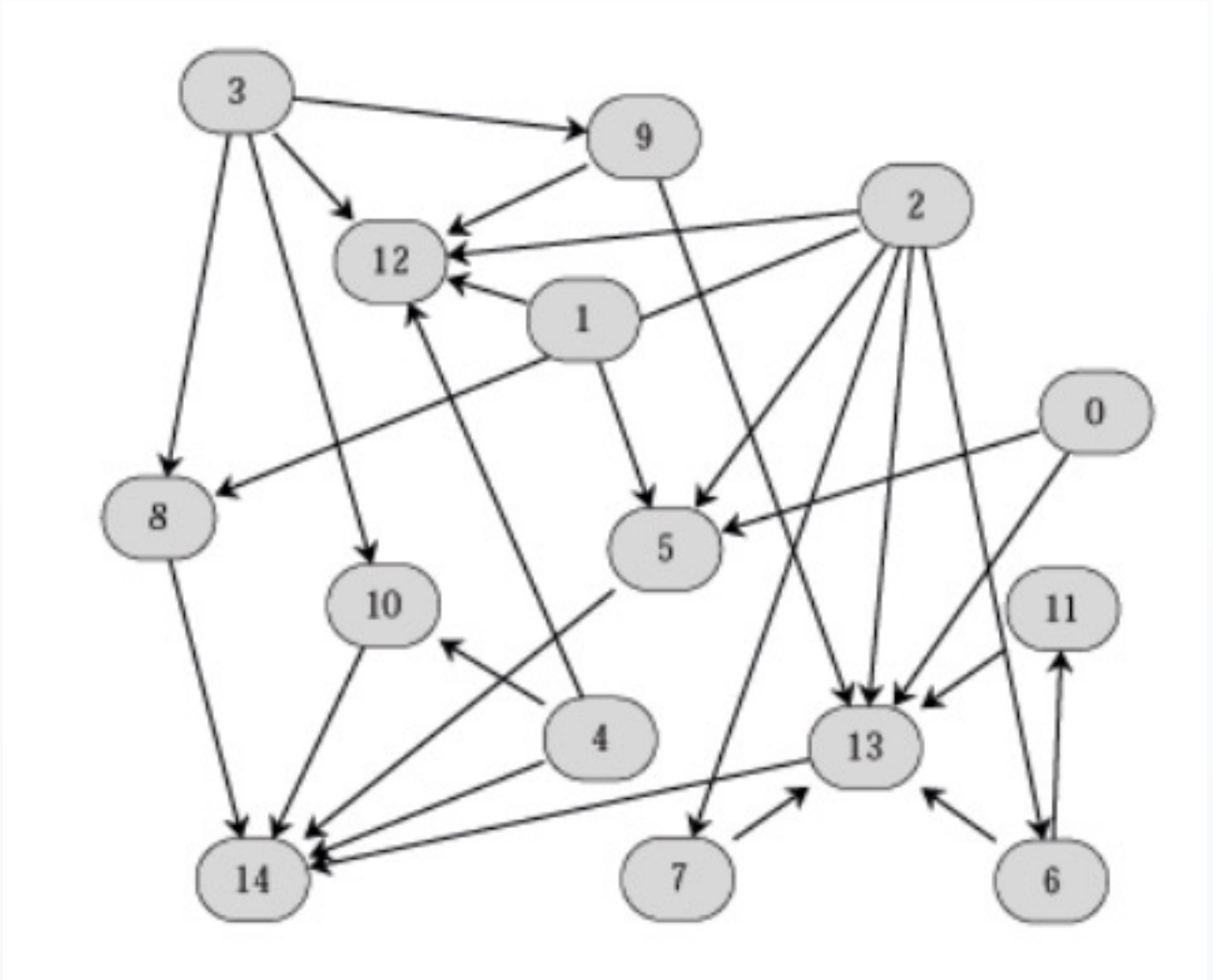




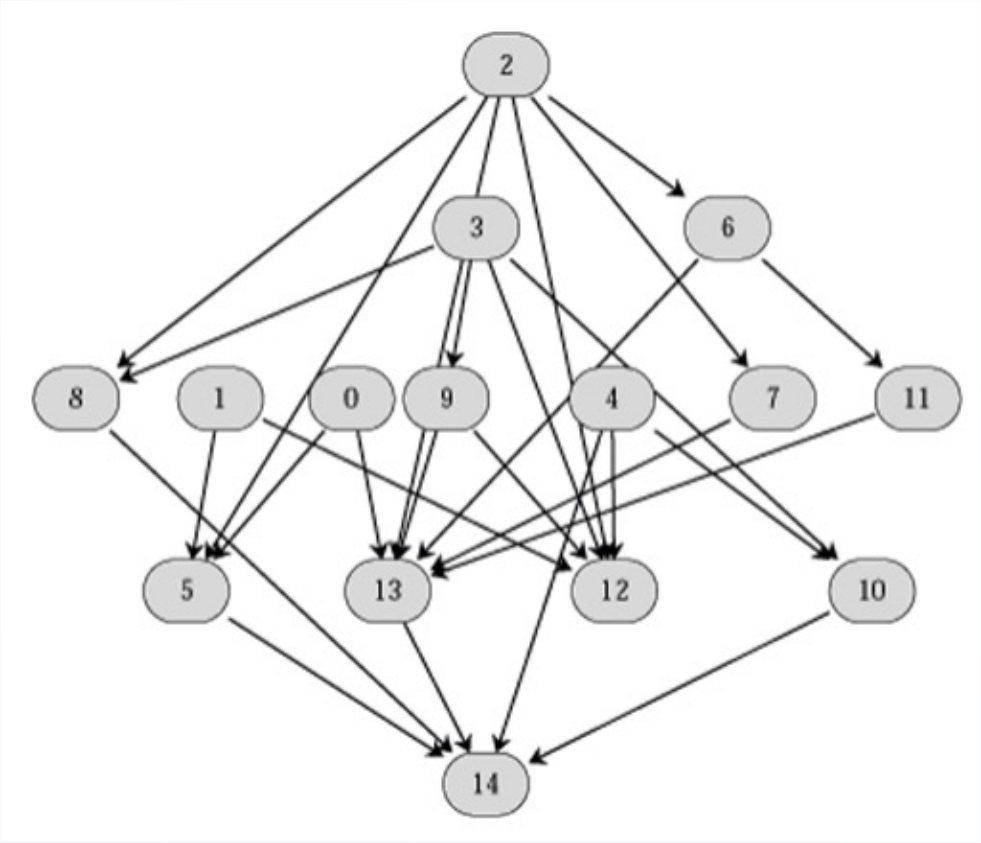
Sugiyama Layout

- Layered networks are often used to represent dependency relations.
- Sugiyama et al. developed a simple method for drawing layered networks in 1979.
- Sugiyama’ s aims included:
  - few edge crossings
  - edges as straight as possible
  - nodes spread evenly over the page

Sugiyama Layout

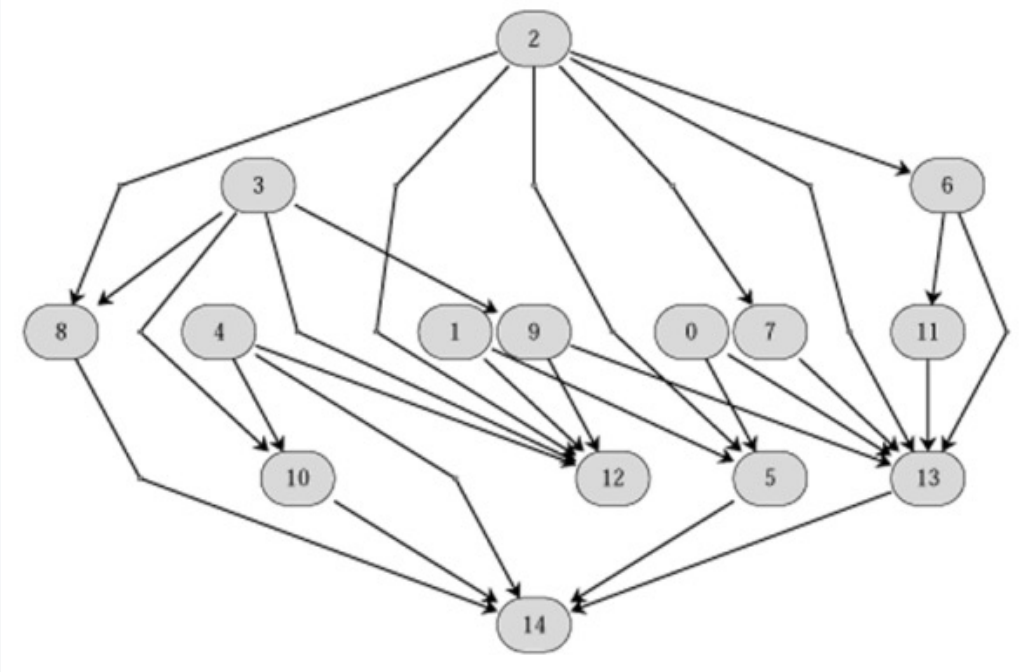


Sugiyama Layout



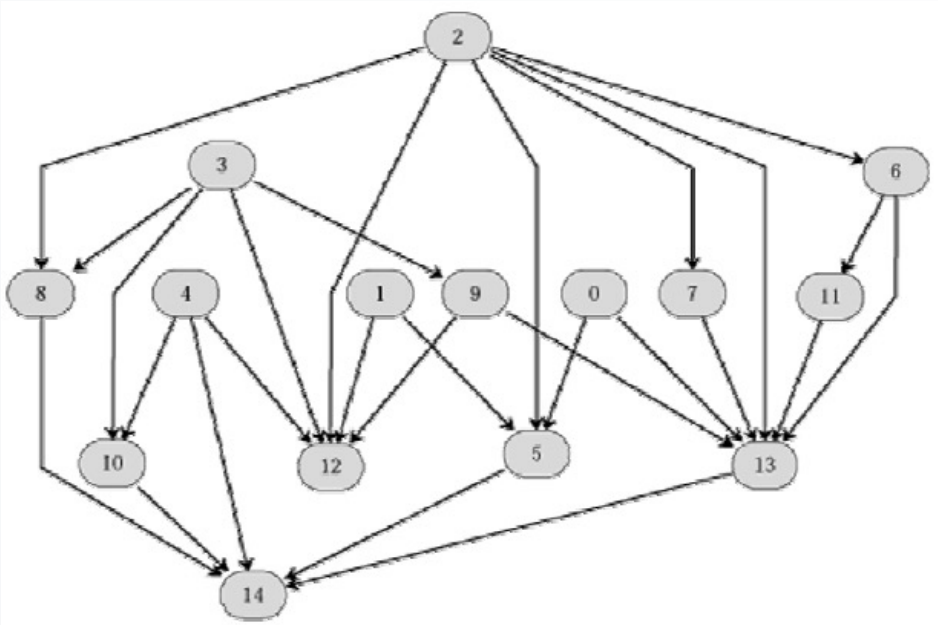
- Step 1: Circle Removal
- Step 2: Layering

Sugiyama Layout



- Step 1: Circle Removal
- Step 2: Layering
- Step 3: Node Ordering

Sugiyama Layout



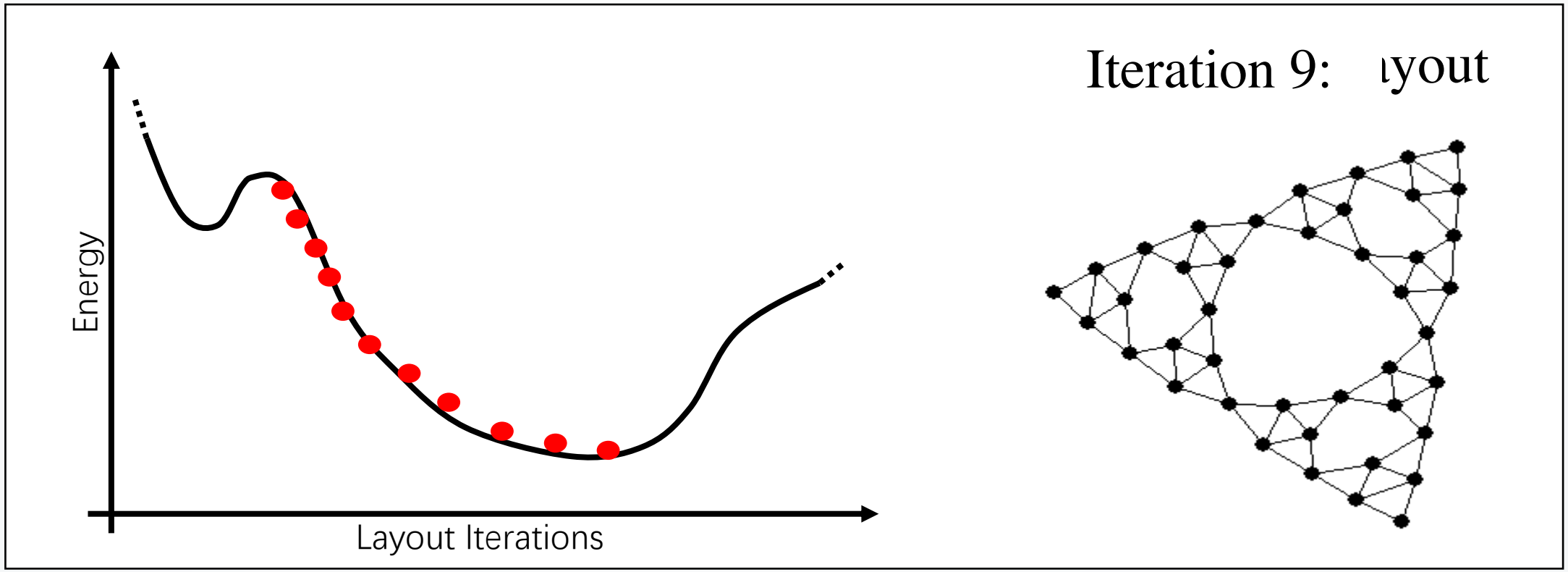
- Step 1: Circle Removal
- Step 2: Layering
- Step 3: Node Ordering
- Step 4: Coordiante

Sugiyama Layout

- High readability
- Difficult to implement (http://www.graphviz.org)
- Not so good for graph without intrinsic order

力导向布局

- 点线图的布局 - 力导向布局
  - 力导向布局是一个迭代求解能量方程的过程
  - 布局的目的是让整个图中因为点与点之间相互位置而构成的“势能”降到最低
  - 力导向布局的能量方程有多种实现方式



力导向布局

- 点线图的布局 - 力导向布局

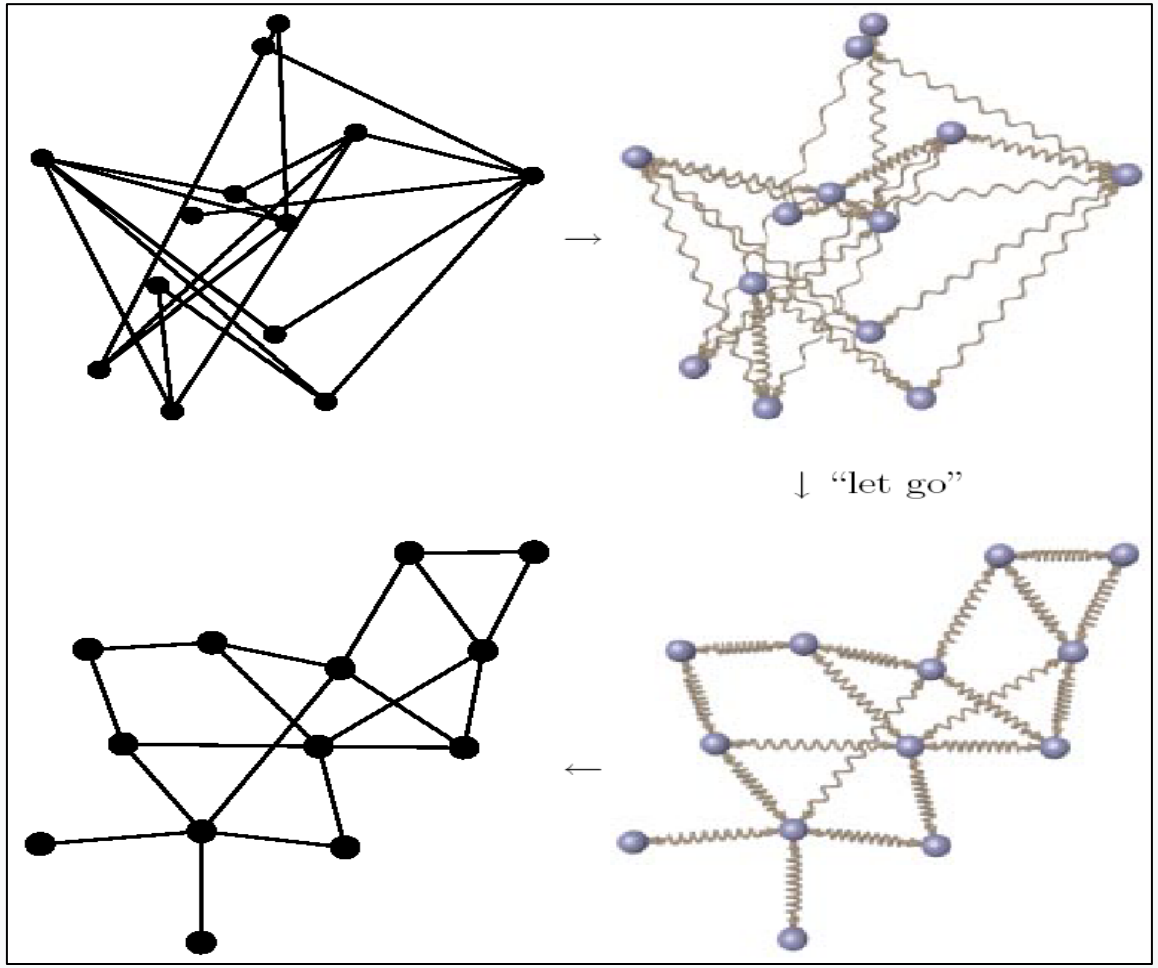
  - 方法I：物理模型
    点看做电荷，而边看做弹簧，无需人为干预，达到力平衡

  - 相连的点之间拥有相互引力，任意两点之间存在相互斥力

    引力 $f_a(d) = d^2/k$　　斥力 $f_r(d) = -k^2/d$

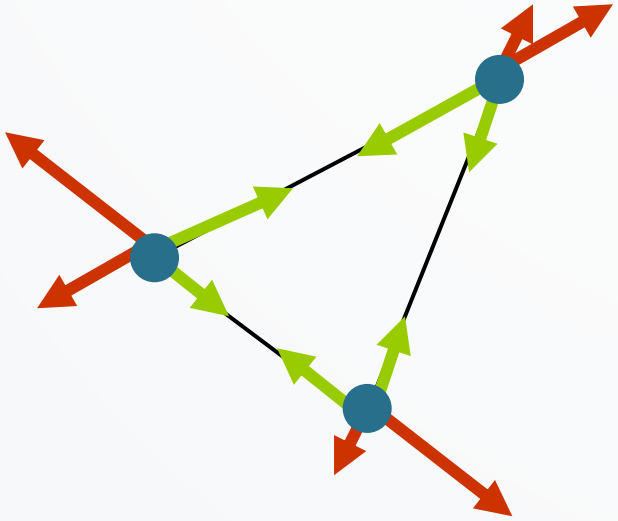
  - 当每个点上的力达到平衡时，势能最小，图的布局得以完成

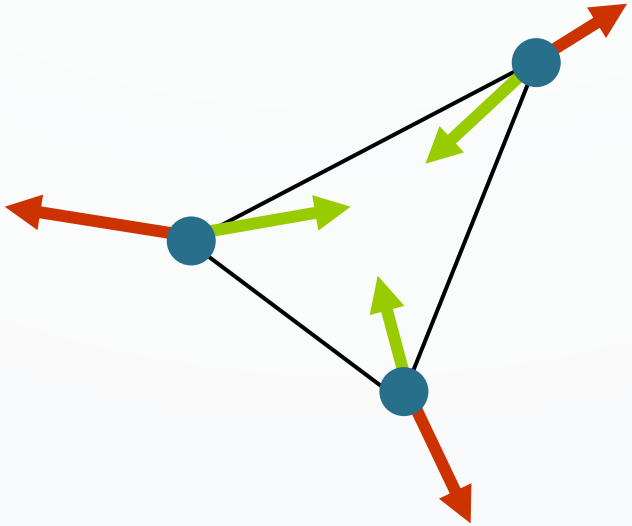
$$F(v) = \sum_{\{u,v\} \in E} f_{uv} + \sum_{u \in V} g_{uv}$$

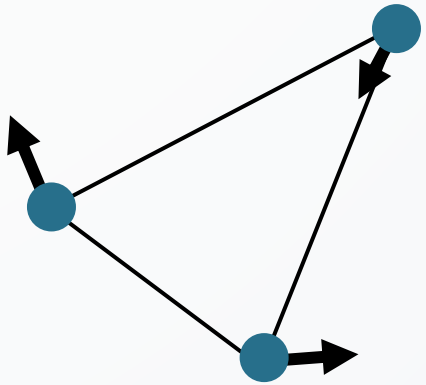


Force-Directed Graph

Forces







近似算法

```
f_a(d) = \d -> d^2 / k # attractive force
f_r(d) = \d -> k^2 / d # repulsive force
for i in [0..iterations)
  u.displacement = zero vector
  # regard repulsive forces
  for (u, v) in E
    d = u.pos - v.pos // vector
    u.displacement += d / |d| * f_r(|d|)
  # regard attractive forces
  for (u, v) in E
    d = u.pos - v.pos
    u.displacement += d / |d| * f_a(|d|)
    v.displacement -= d / |d| * f_a(|d|)
  # tune positions
  for u in V
    u.pos += u.displacement
```

Force Simulator (Physical Model)

Force on each vertices is

$$f(i,x,\mathrm{k},C)=\sum_{i\neq j}-\frac{Ck^2}{\left\|x_i-x_j\right\|^2}(x_i-x_j)+\sum_{i\leftrightarrow j}\frac{||x_i-x_j||}{k}x_i-x_j$$

- $F=ma=m\frac{dv}{dt}=m\frac{d^2x}{dt^2}$
- $\frac{d^2x}{dt^2}=\frac{d(\frac{dx}{dt})}{dt}$ $\frac{dx}{dt}=\frac{F}{m}t+c_1$
- $x=\int\left(\frac{F}{m}t+c_1\right)dt$ $x=\frac{1}{2}\frac{F}{m}t^2+c_1t+c_2$

力导向布局

- 方法2：KK-Layout

两点之间的屏幕距离　　图中的理论距离

$$Cost=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{2}k_{ij}(\|\boldsymbol{y}_i-\boldsymbol{y}_j\|-l_{ij})^2$$

$$l_{ij}=L\cdot d_{ij}$$

- l_{ij}：desired on-screen length of edge (i,j)
- d_{ij}：length of shortest path (n. of steps) from i to j
- L：desired on-screen length of an edge

$$k_{ij}=K/d_{ij}^2$$

- K：constant
- k_{ij}：strength of spring (i,j)

当两点间距离越大的时候，改系数较小，表示精确保留两点之间的相对距离，并不是那么重要；相反，当两点间距离较小时，该系数较大，表示精确保留两点之间的相对距离比较关键

力导向布局

- 点线图的布局 - 力导向布局

  - 方法3：Node-Repose LinLog

$$Cost_{NodeLinLog} = \sum_{\{u,v\} \in E} \|y_u - y_v\| - \sum_{u \in V, v \in V} \ln \|y_u - y_v\|$$

当两点相连时，加在边上的引力

图中任意两点之间的斥力

力导向布局

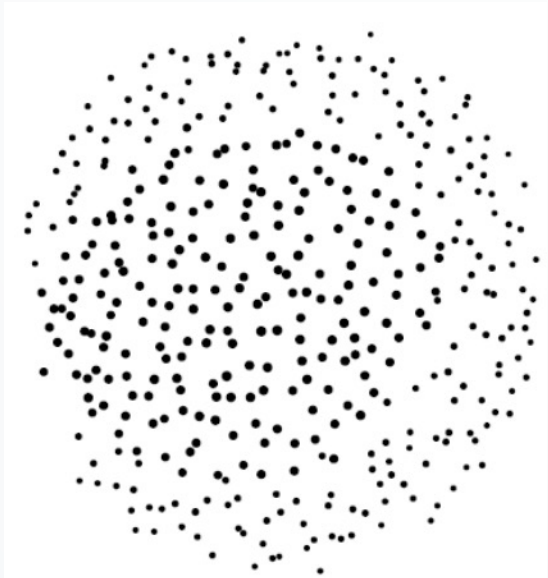
- 点线图的布局 - 力导向布局

  - 方法4：Edge-Repose LinLog

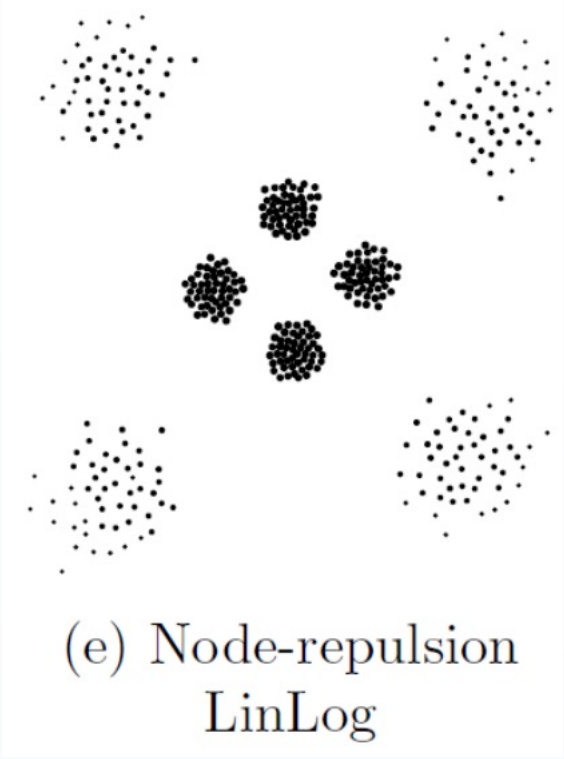
$$Cost_{EdgeLinLog} = \sum_{\{u,v\} \in E} \|y_u - y_v\| - \sum_{u \in V, v \in V} deg(u) deg(v) \ln \|y_u - y_v\|$$

当点的度数比较大的时候，即点连接了较多边的时候，斥力也比较大

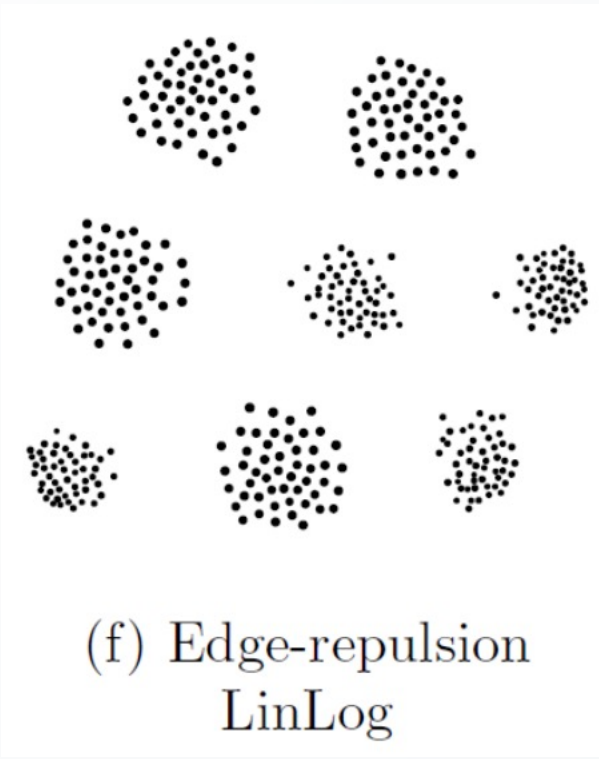
Comparison



(a) Kamada-Kawai



(e) Node-repulsion LinLog



(f) Edge-repulsion LinLog

Hall’s Energy

position of node i

Laplacian Matrix of the graph

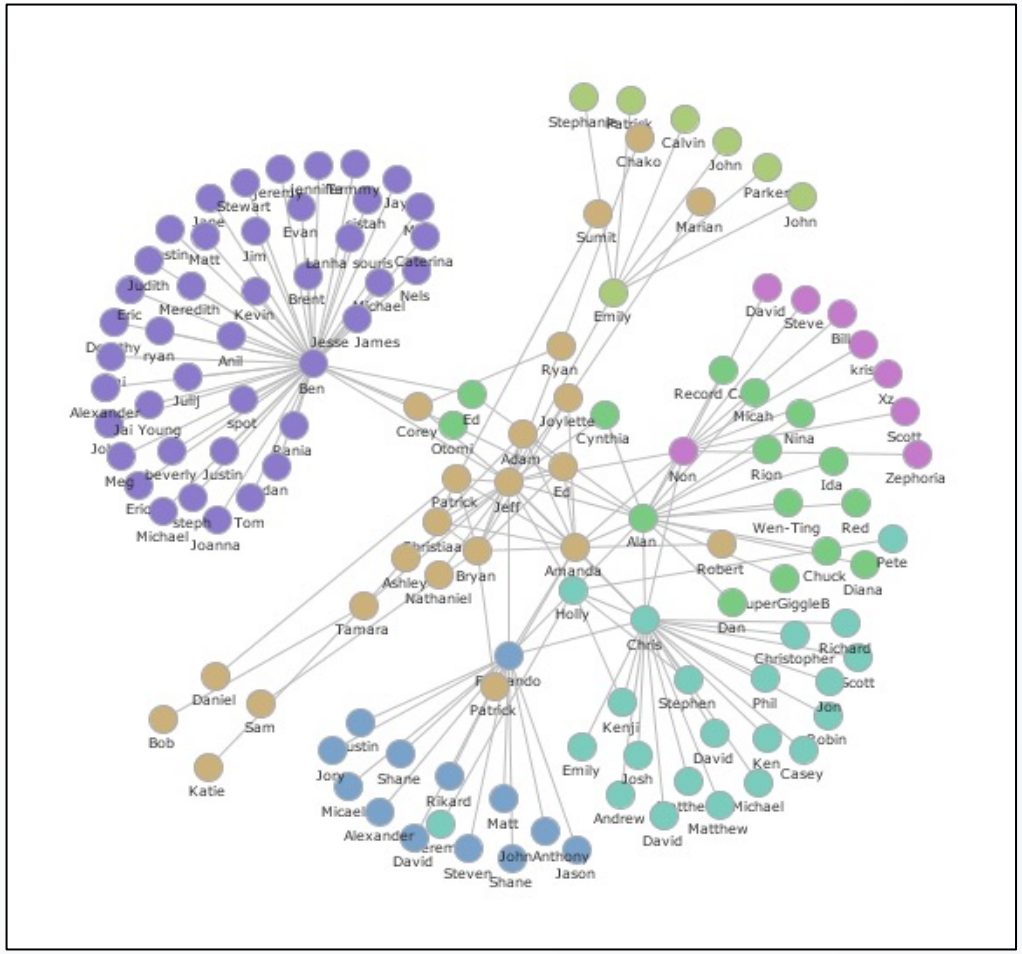
$$\frac{1}{2}\min\sum_{i,j\in E}^{n}\omega_{ij}\,\|X_i - X_j\|^2 = \min\left(X^T L X\right)$$

screen distance between node i and node j

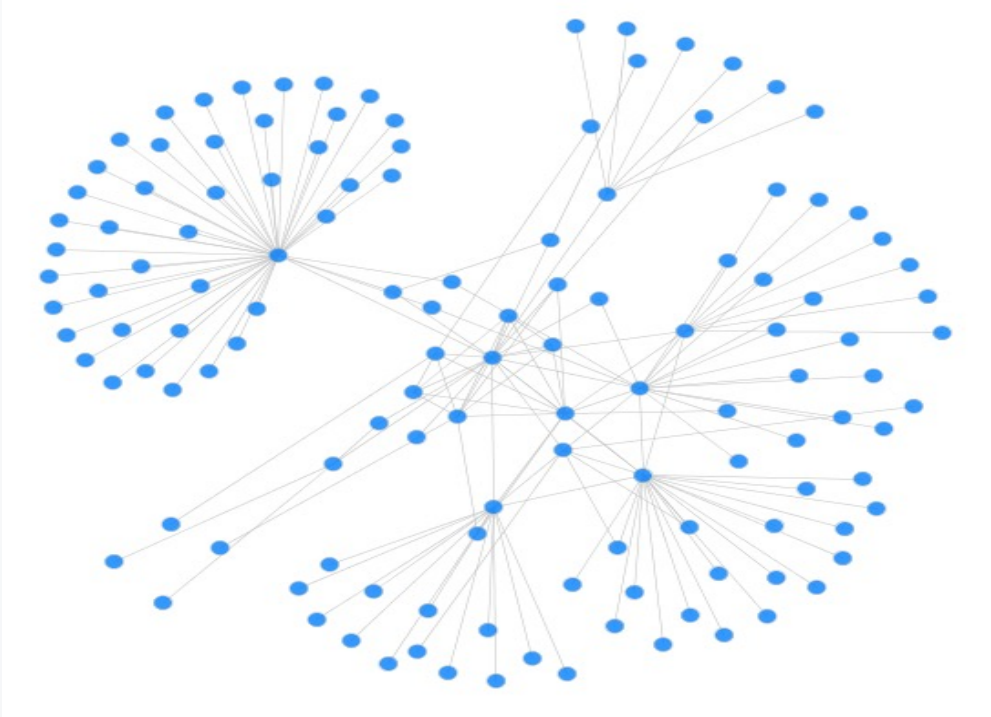
Reference: ACE: A Fast Multiscale Eigenvector Computation for Drawing Huge Graphs

力导向布局

- 点线图的布局 - 力导向布局
  - 能够清晰的显示 图中由点组成的“簇”，符合了格式塔法则中的 相似性原则（Proximity Rule）
  - 布局结果一般具有对称性，满足了美观度的要求
  - 能够最小化边的平均长度，降低了边与其他边相交的可能
  - 没有人为的设计干预，自动根据图的拓扑结构完成布局，抗扰动性较好



高纬度映射布局



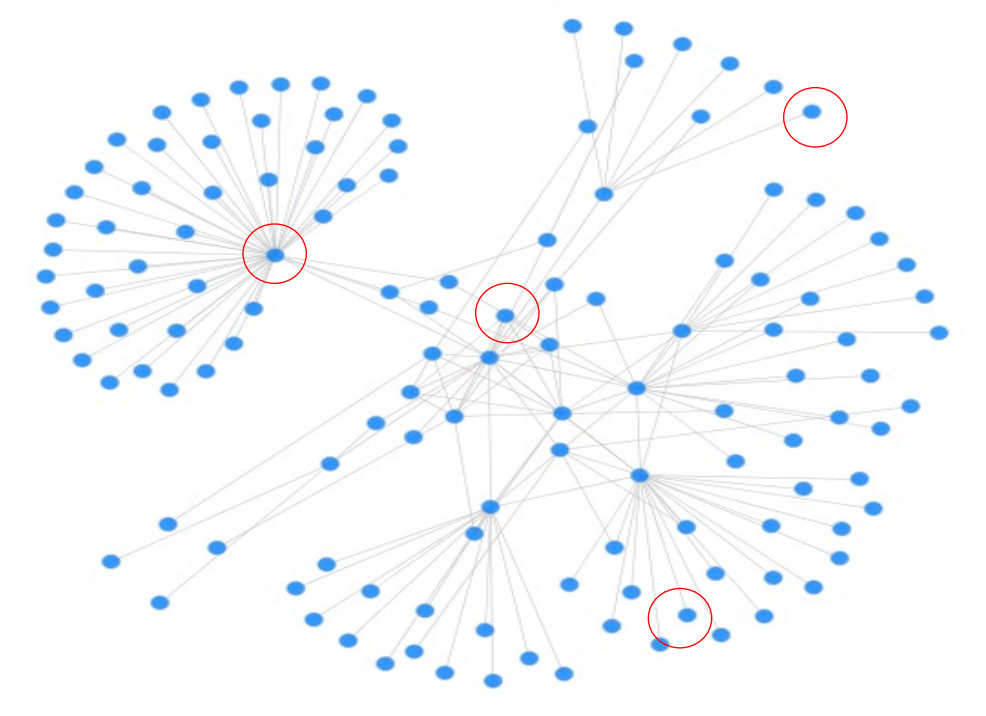
Step 1: Select Pivot

Step 2: Construct High dimensional Space

Step 3: High dimensional Projection (PCA)

Non-Force Directed Layout

高纬度映射布局



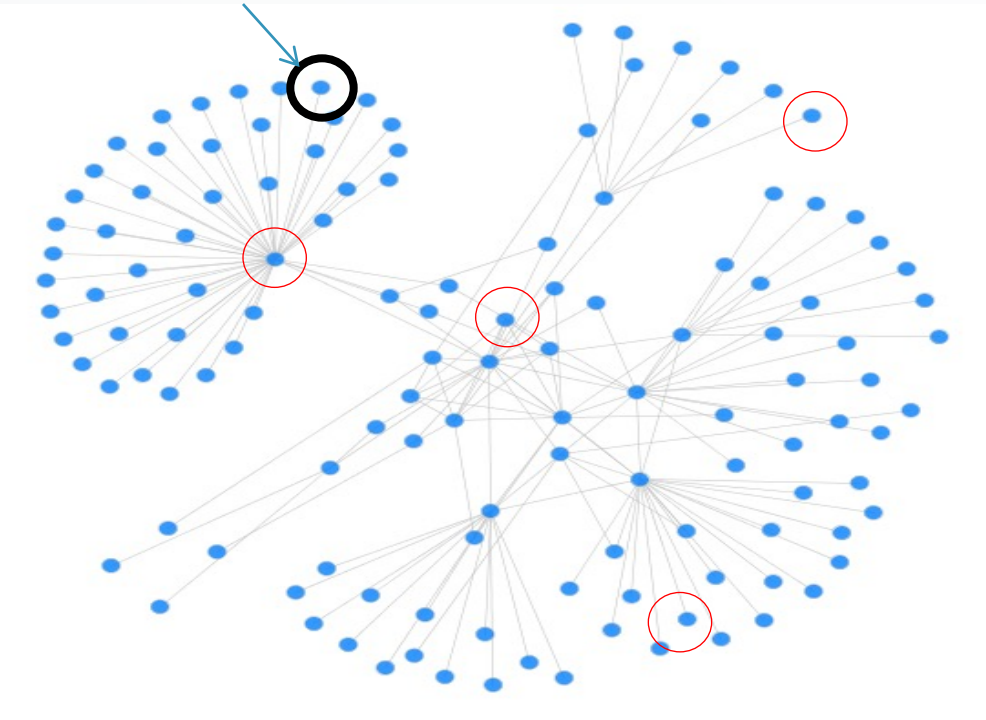
Step 1: Select Pivot

Step 2: Construct High dimensional Space

Step 3: High dimensional Projection (PCA)

高纬度映射布局

A [1, 3, 5, 6]



Step 1: Select Pivot

Step 2: Construct High dimensional Space

Step 3: High dimensional Projection (PCA)

高纬度映射布局

4 - dimensional

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

Step 1: Select Pivot

Step 2: Construct High dimensional Space

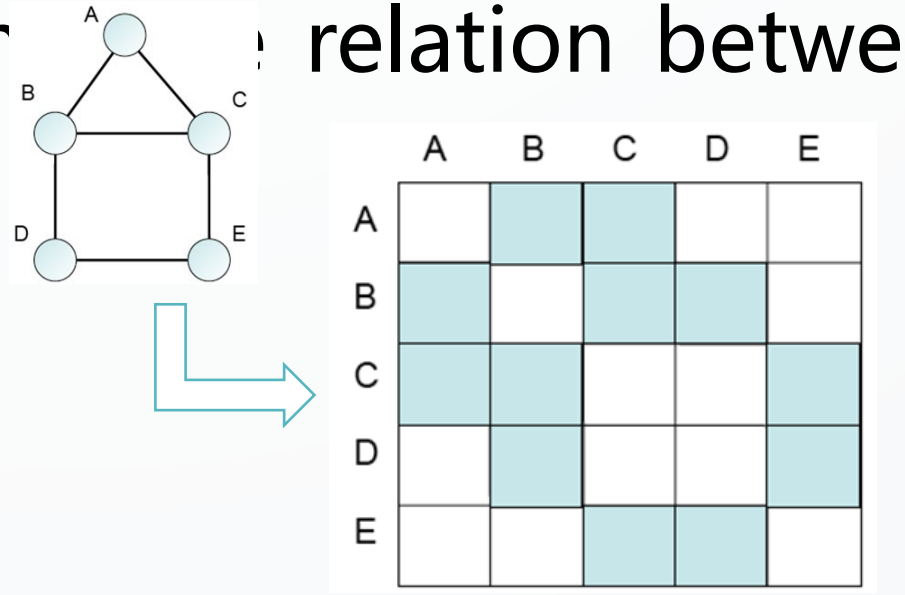
Step 3: High dimensional Projection (PCA)

Outline

- Introduction
- Node-Link Diagram
  - Various Designs
  - Graph Drawing & Layout
- **Matrix Visualization**
- Hybrid

Adjacency Matrix

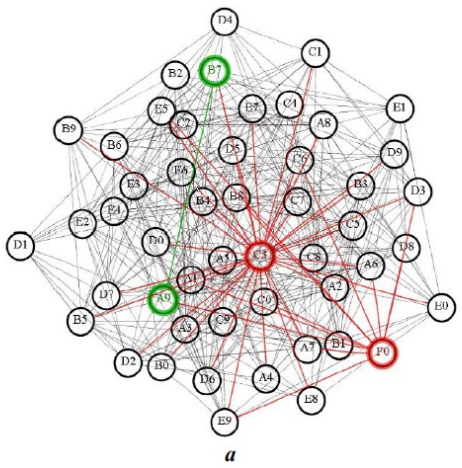
- N×N matrix, representing relations among N objects.
- Position (i, j) represer relation between the ith and the jth object,
  - weight
  - direction
  - Self-reflexivity



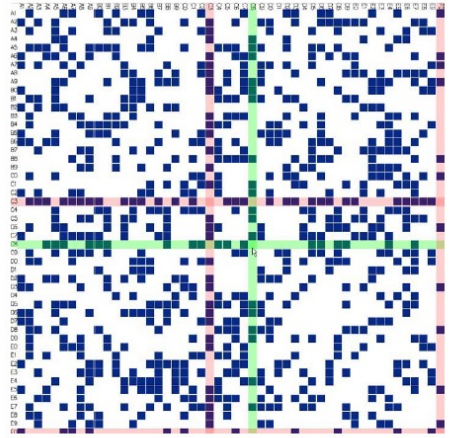
- Related issues
  - ordering
  - Path finding

邻接矩阵

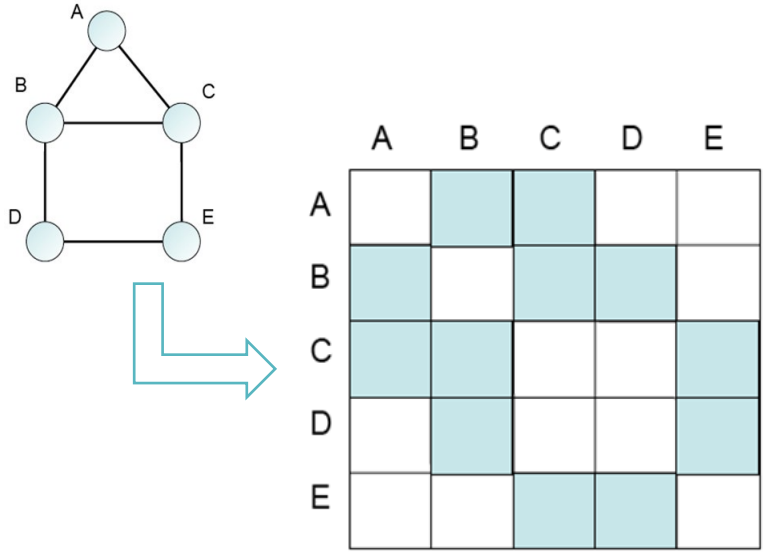
- 点线图 – 邻接矩阵
  - 点线图虽然直观，但是当图比较密集时（Dense），因为线的交叉及点的遮挡，难以清晰的展现图中的信息
  - 使用图的邻接矩阵展现图结构，能够完全避免上述问题
  - 邻接矩阵中的每一行，每一列代表图中的一个点，矩阵中的元素代表图中的边，表示了对应行、列所代表的点之间相互连接
  - 用户无法在邻接矩阵中追踪图中的一条路径



a

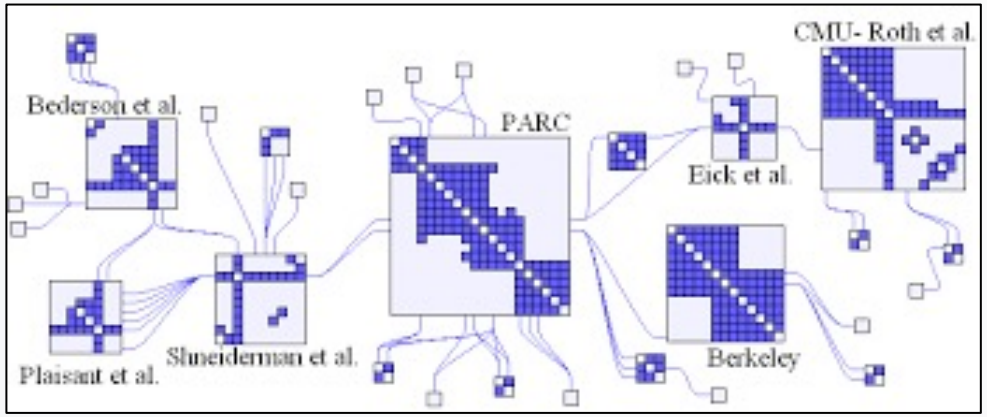


b



混合方法

- 点线图 – 混合方法
  - 结合点线图与连接矩阵的优点
  - 当图中结构较为密集时，用邻接矩阵展现，以避免点的遮挡与线的交叉
  - 当图中结构较为疏松时，用点线图展现，以方便路径追踪

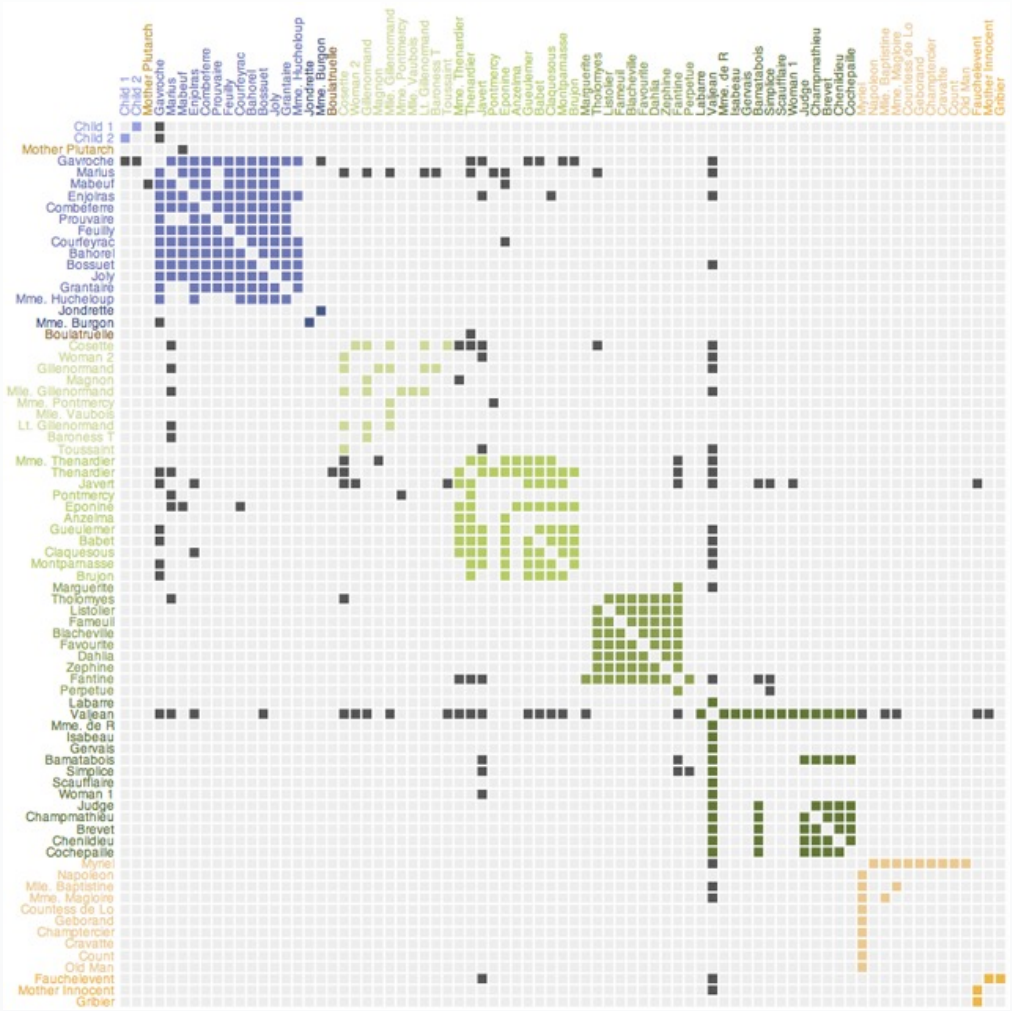


NodeTrix: a Hybrid Visualization of Social Networks,
Nathalie Henry, Jean-Daniel Fekete, and Michael J. McGuffin
IEEE InfoVis 2007

课程总结

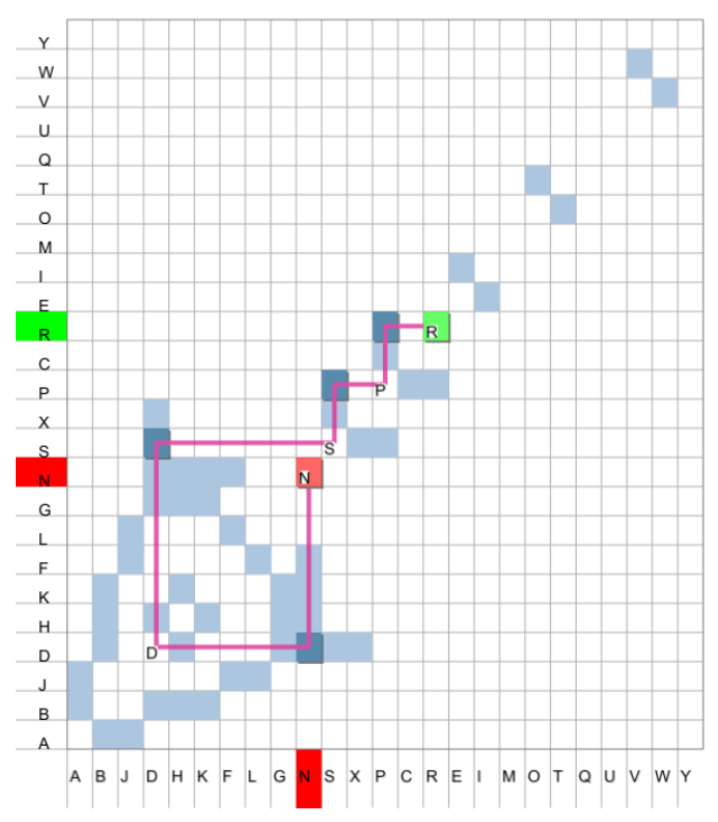
- 本节课
  - 数据基础
  - 多维度数据的可视化
  - 树的可视化
  - 图的可视化

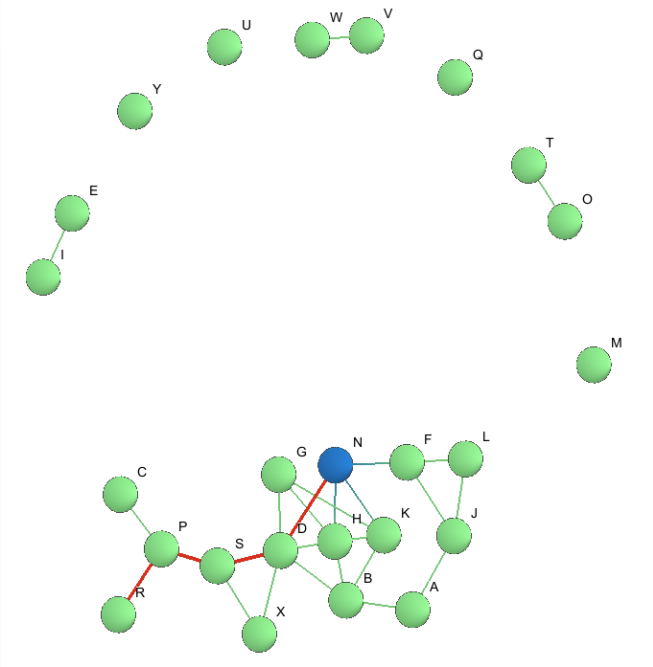
Good Ordering of Adjacency Matrix



http://hci.stanford.edu/jheer/files/zoo

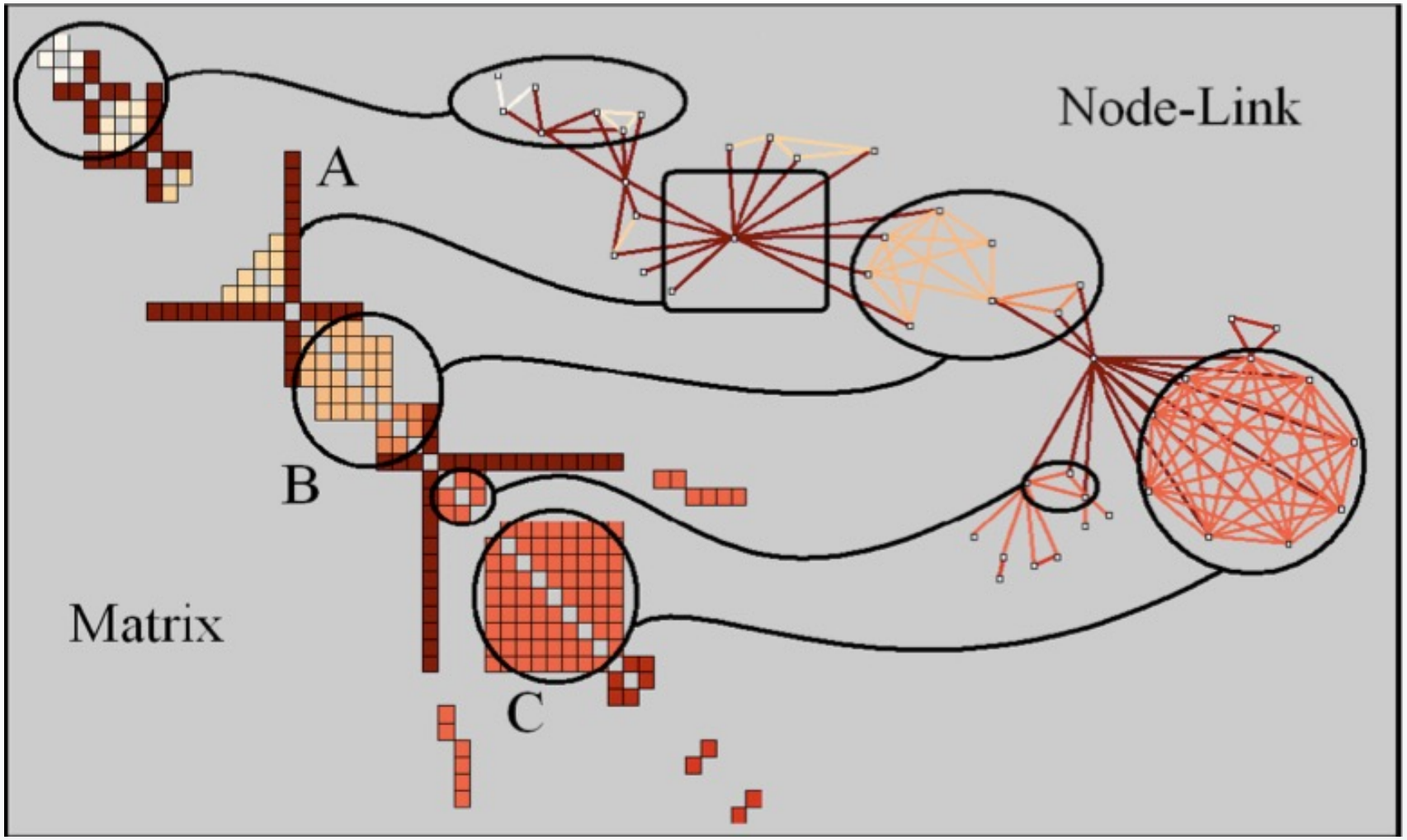
Path Visualization for Adjacency Matrix





Left: Connecting matrix entries and the diagonal.
Right: Path {*N*;*D*;*S*;*P*;*R*} is Highlighted in the Node-link Diagram.

Recognizing the Patterns of Matrix



Adjacency Matrix Summary

- No edge crossing, good for edge cluttered graph
- Good visual scalability
- Good presentation of graph pattern
- Visualization is too abstract to understand
- Difficult to follow a transitive relation path

Outline

- Introduction
- Node-Link Diagram
  - Various Designs
  - Graph Drawing & Layout
- Matrix Visualization
- **Hybrid**


INRIA
FUTURS
UNIVERSITÉ
PARIS-SUD 11
The University of Sydney
UNIVERSITY
OF TORONTO