Network clustering: from models to methods

Sonia Cafieri

ENAC – Ecole Nationale de l'Aviation Civile
Toulouse, France

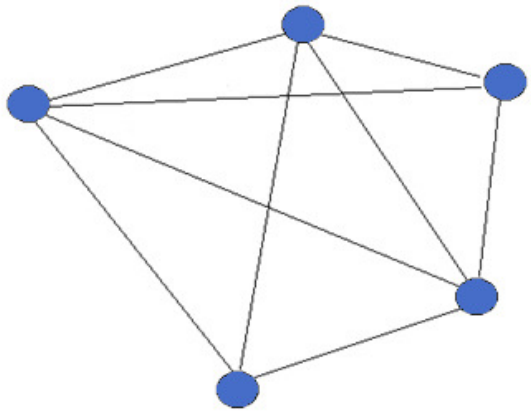
Summer School on Operational Research and Applications
May 2013

Outline

Networks

Network = (in its simplest form),
a collection of points joined together in pairs by lines.

- In many domains, many objects of interest or *complex systems* are composed of parts or components linked together in some way.
- $\Rightarrow$ Many complex systems can be thought of as networks



Networks

Networks often used to represent complex systems

- social networks
- telecommunication networks
- transportation networks
- ...

Why are we interested in networks?

Many complex systems are composed of parts or components linked together in some way

Different ways to study systems:

- study the nature of the components
  Ex.: study how a computer works
- study the nature of the connections between components
  Ex.: study the communication protocols on the Internet
- study the *pattern* of connections between components
  $\Rightarrow$ this can be represented as a network

Why are we interested in networks?

The pattern of connections (structure of the network)
can have a big effect on the behaviour of the system

Examples

- computer network: the pattern of connections affects the routes that data take over the network (efficiency of data transmission)
- social network: the connections affect the way people learn, form opinions, gather informations
- ...

$\Rightarrow$ Study systems using a simplified representation reducing them to abstract structures capturing basically the pattern of connections $\rightarrow$ networks

$\Rightarrow$ Use mathematical, computational and statistical tools for analyzing networks

Example: Social Networks

Example: Social Networks

Social Network: components are people and connections represent some form of social interaction between them. *Actors* and *ties*.

Many possible definitions of a connection:

- friendship
- professional relationships
- romantic relationships
- communication patterns
- exchange of goods or money
- etc...

Example: Social Networks

Social Network: components are people and connections represent some form of social interaction between them. *Actors* and *ties*.

Many possible definitions of a connection:
- friendship
- professional relationships
- romantic relationships
- communication patterns
- exchange of goods or money
- etc...

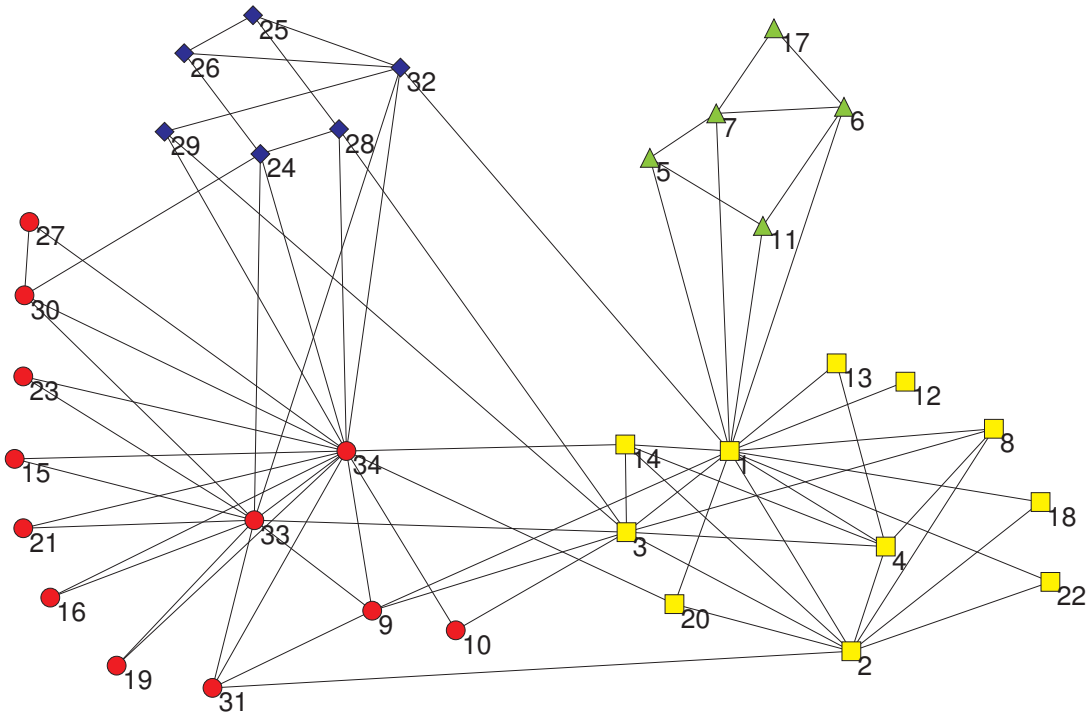
Empirical methods to get data:
- direct questioning, interviews
- direct observation (watching interactions between individuals)
- use of archival records
- other: email exchange etc...

Foundation of the field: Jacob Moreno (psychiatrist), 1930s

Social network I

Zachary's karate club network

it describes friendship relations between 34 members of a karate club observed over two years by Zachary. In that period the club split into two groups after a dispute between the club owner and the karate instructor.

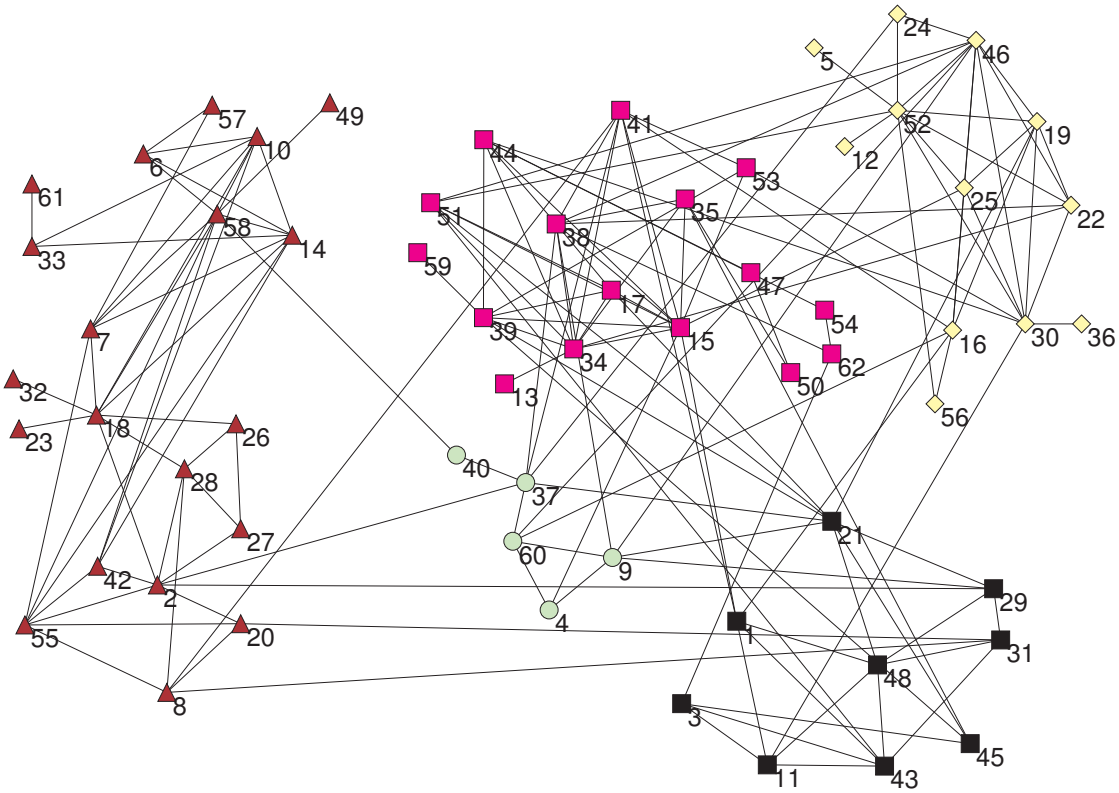


Social network II

Dolphins network:

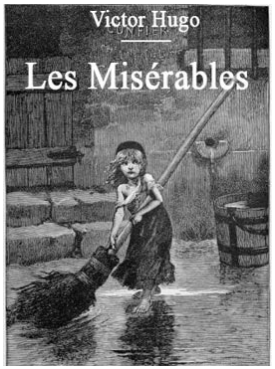
bottlenose dolphins studied by Lusseau in Doubtful Sound, New Zealand.

Network with 62 components corresponding to the dolphins and 159 links between pairs of dolphins with frequent communications among them.

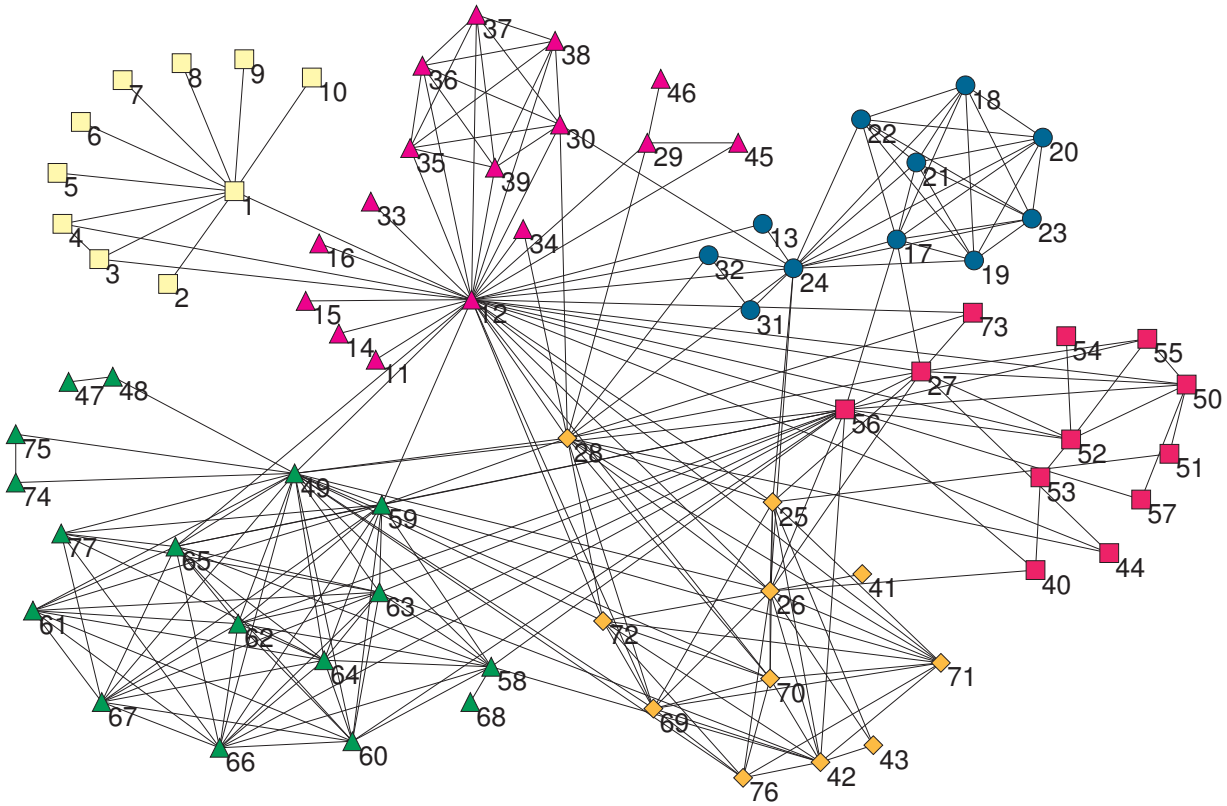


Social network III

Victor Hugo's *Les Misérables* network:



- ★ 77 components associated to characters which interact
- ★ 257 links between pairs of characters appearing jointly in at least one chapter

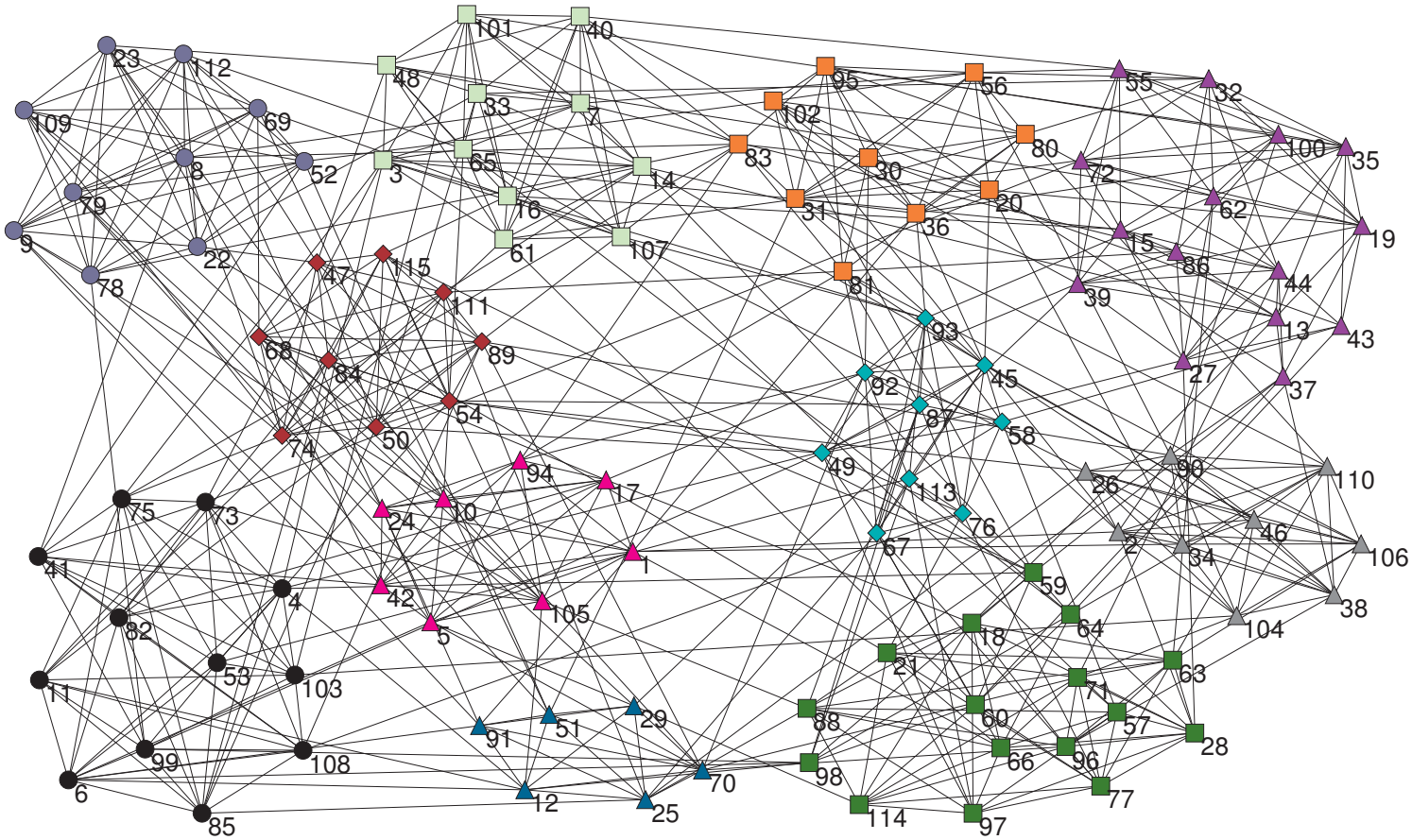


Social network IV

Football game network

it describes the schedule of games between American college football teams in the Fall 2000.

$n = 115$ components, $m = 613$ links



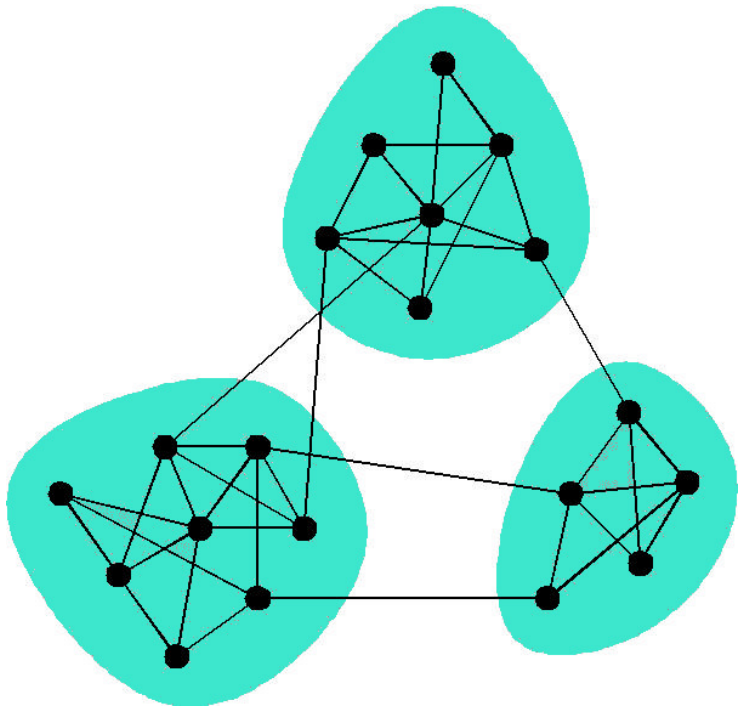
Detection of modules (communities)

Automatic analysis of complex systems represented as networks

$\Downarrow$

identification of communities

community = a subset of entities such that there are more links within the community than links joining it to the outside



Outline

Networks –> Graphs

Network: mathematical representation as a graph

A network, or graph: $G = (V, E)$

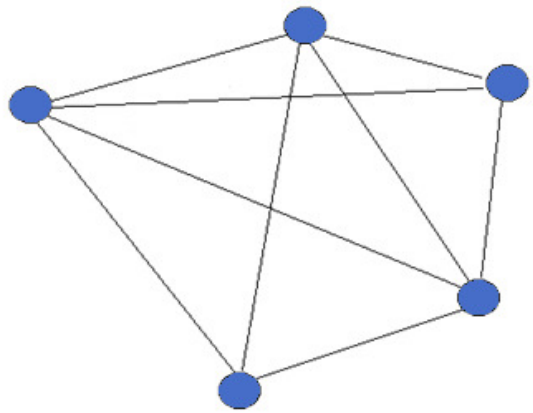
V = *Vertices*, associated with the entities of the system under study

(people, companies, towns, natural species, ...)

represented by points

E = *Edges*, express that a relation defined on all pairs of vertices holds or not for each such pair

represented by lines joining vertices

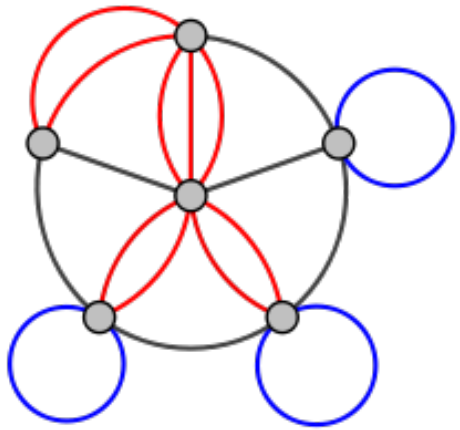


$|V| = n, \quad |E| = m$

Networks –> Graphs

Most of the networks we study have at most a single edge between any pair of vertices

- A **loop** or **self-edge** is an edge for which both end vertices coincide.
- When there is more than one edge between the same pair of vertices, we refer to those edges collectively as a **multiedge**.
- A **multigraph** is a graph such that two vertices can be joined by a multiedge.
- A **simple graph** has neither loops nor multiple edges.

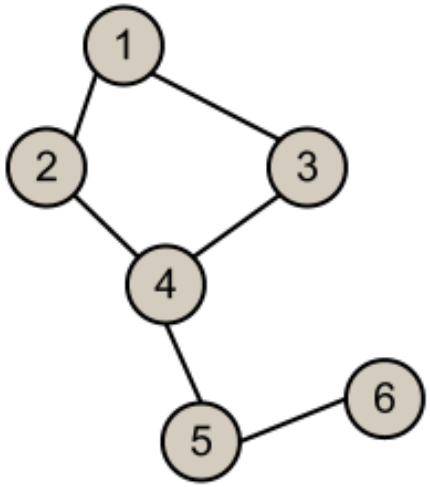


Representation of a network

- **Edge list**
  - label the vertices with integer labels $1, \ldots, n$
  - denote an edge between vertices i and j by (i, j)
  - the network can be specified by giving the value of n and the list of all the edges.

Often used to store the structure of networks on computers.

Example



$n = 6$

(1,2)

(1,3)

(2,4)

(3,4)

(4,5)

(5,6)

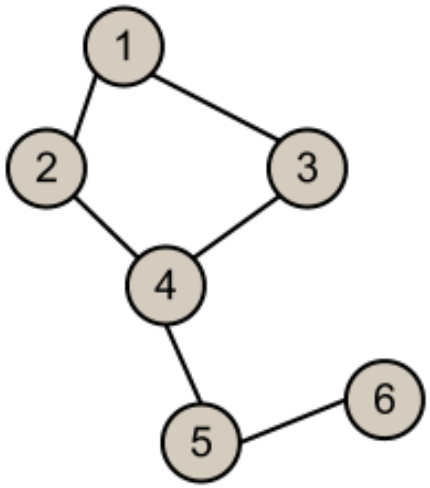
Representation of a network

- **Adjacency matrix**

Matrix A with elments:

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge between vertices } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Example



0	1	1	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
0	1	1	0	1	0
0	0	0	1	0	1
0	0	0	0	1	0

Representation of a network

- **Adjacency matrix**

  Matrix A with elments:

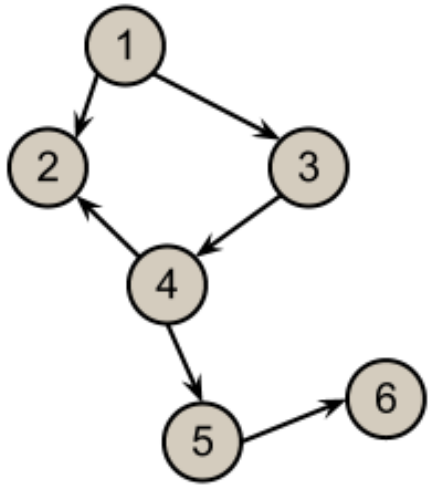
$$A_{ij} = \begin{cases} 1 & \text{if there is an edge between vertices } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

  - symmetric matrix
  - diagonal elements all 0 for graph with no self-edges
  - for a multiedge: the corresponding element A_{ij} is the multiplicity of the edge
  - for a self-edge: it can be represented by the corresponding diagonal element equal to 2

Outline

Types of networks

Directed network or digraph

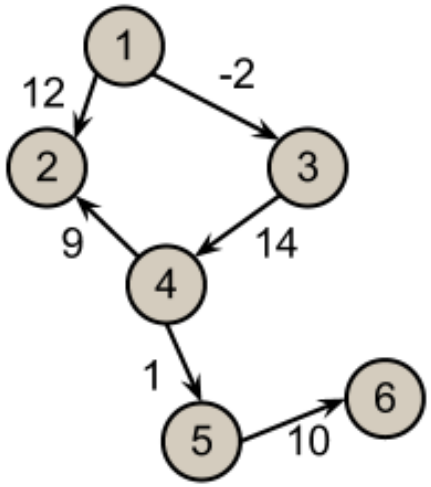


- A network where each edge has a direction
- *directed edges*: point *from* on vertex *to* another, represented by lines with arrows

- Examples: - hyperlinks on the World Wide Web run from one web page to another
  - citation networks point from one paper to another
- Adjacency matrix has elements:

$$A_{ij} = \begin{cases} 1 & \text{if there is an edge from } j \text{ to } i \\ 0 & \text{otherwise} \end{cases}$$

Weighted network

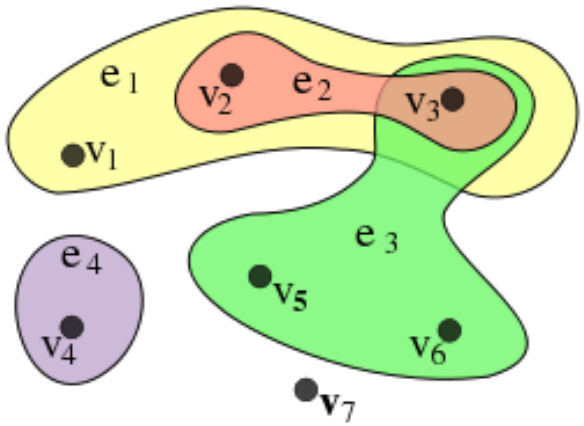


- It can be useful in some situation to represent edges as having a strenght, weight, or value
- Usually, the weight is a real number

- Examples: - the amount of data flowing on a computer network,
  - frequency of contact between people in a social network
- Adjacency matrix has elements with values equal to the weights of the edges

Hypergraph

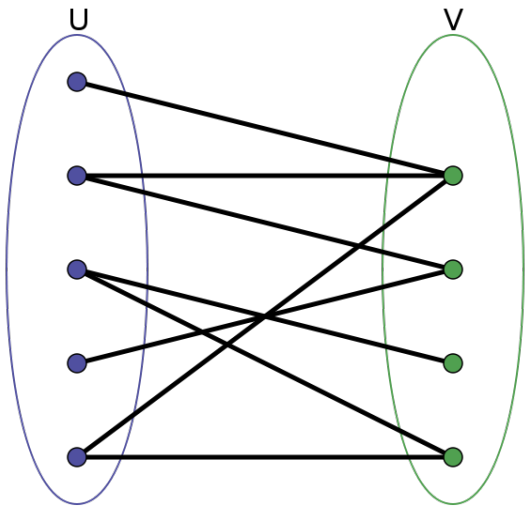
- A generalization of a graph in which an edge can connect more than two vertices at a time. (E is a subset of $\mathcal{P}(V) \setminus \{\emptyset\}$, with $\mathcal{P}(V)$ the power-set of V).
- Example: - representation of family ties in families with more than two people



- Can be viewed as an incidence structure
- Often more conveniently represented by *bipartite graphs*

Bipartite graph (two-mode network)

- Graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V.
- Used to model relations between two different classes of objects.



- Examples:

  - actors and films in which they appear,

  - football players and clubs

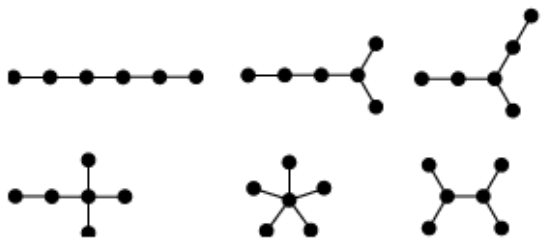
- **Incidence matrix**: $p \times n$ matrix, with n = number of elements, p = number of groups

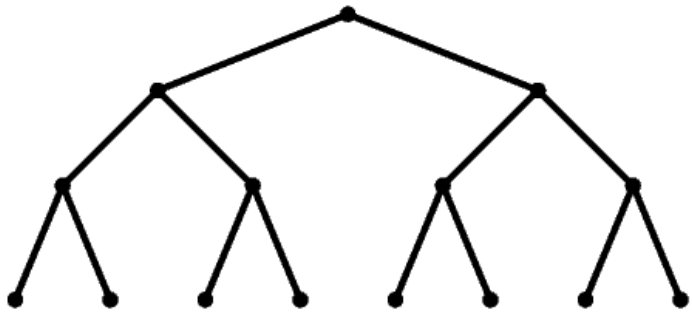
$$B_{ij} = \begin{cases} 1 & \text{if vertex } j \text{ belongs to group } i \\ 0 & \text{otherwise} \end{cases}$$

Types of networks

Tree = connected acyclic graph $T = (G, V)$

- Any two vertices are connected by exactly one simple path (no closed loops)
- T is connected, but is not connected if any single edge is removed from it
- T has n vertices, $n - 1$ edges
- T is labeled if each vertex is given a unique label
- a star graph is a tree with as many leaves as possible
- a path graph is a tree with two terminal vertices (the fewest possible)





Outline

Measures & Properties

A network G can be characterized by several measures

Let us consider an unweighted network.

Measures

- Degree
- Clustering coefficient
- Assortativity
- Centrality

Properties

- Modules
- Small-world networks
- Scale-free networks

Degree

The degree of a vertex is the number of edges connected (incident) to it:

$$k_i = \sum_{j=1}^{n} A_{ij}$$

Degree

The degree of a vertex is the number of edges connected (incident) to it:

$$k_i = \sum_{j=1}^{n} A_{ij}$$

Undirected graphs:

- $2m$ ends of edges $\Rightarrow \quad \sum_{i=1}^{n} k_i = 2m \quad \Rightarrow \quad m = \frac{1}{2} \sum_{ij} A_{ij}$
- regular graph: all vertices have the same degree $\rightarrow$ degree of the graph

  k-regular graph: all vertices have degree k
- degree sequence: non-increasing sequence of its vertex degrees
- isolated vertex: vertex with degree 0

Degree

The degree of a vertex is the number of edges connected (incident) to it:

$$k_i = \sum_{j=1}^{n} A_{ij}$$

- average degree of a graph:

$$c = \frac{1}{n} \sum_{i=1}^{n} k_i = \frac{2m}{n}$$

- density of a graph: fraction of edges that are actually present:

$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)} = \frac{c}{n-1}$$

- if ρ tends to a constant as $n \to \infty$, the network is said dense

- if ρ tends to 0 as $n \to \infty$, the network is said sparse

Clustering coefficient

Clustering coefficient: a measure of the level of transitivity between pairs of vertices

- relation "connected by an edge": a transitive relation would mean
  if u connected to v and v connected to w, then u also connected to w
  " the friend of my friend is also my friend"
- perfect transitivity only possible on cliques
- count all paths of length 2, count how many are closed, divide the 2nd number by the 1st:

$$C = \frac{\text{number of closed paths of lenght 2}}{\text{number of paths of lenght 2}}$$

$C = 1$: perfect transitivity, $C = 0$: no closed triads (Ex: trees)

Clustering coefficient

Clustering coefficient: a measure of the level of transitivity between pairs of vertices

- alternatively:

$$C = \frac{(\text{number of triangles}) \times 3}{\text{number of connected triples of vertices}}$$

Clustering coefficient

Clustering coefficient: a measure of the level of transitivity between pairs of vertices

- alternatively:

$$C = \frac{(\text{number of triangles}) \times 3}{\text{number of connected triples of vertices}}$$

Local clustering coefficient: for a vertex, quantifies how close its neighbors are to being a clique

- go through all distinct pairs of vertices that are neighbors of i, count the number of such pairs that are connected to each other, and divide by the total number of pairs $(1/2)(k_i(k_i - 1))$:

$$C_i = \frac{\text{number of pairs of neighbors of } i \text{ that are connected}}{\text{number of pairs of neighbors of } i}$$

Assortativity and Centrality

A graph is said assortative if vertices with a high degree tend to be connected to other vertices with high degree, and vertices with a low degree to others with low degree.

Assortativity and Centrality

A graph is said assortative if vertices with a high degree tend to be connected to other vertices with high degree, and vertices with a low degree to others with low degree.

Centrality refers to the relative importance of a vertex within the network.

Measures:

- vertex degree

- *betweenness centrality*: fraction of shortest paths that pass through the vertex

- *eigenvector centrality*: value of the vector component, where the vector corresponds to the largest eigenvalue of the adjacency matrix.

A shortest path between two vertices A and B is the path between A and B with the smallest number of edges.

Small-world networks

Stanley Milgram, 1960s

experiments to quantify the typical distance between people in social networks

- Milgram sent a set of packages to recipients randomly chosen in a city of Nebraska, asking them to attempt to get the package to a specified target individual in Boston, MA.
- Each recipient was asked to pass the package to a person who he knew personally who was more likely to know the target.
- Whoever received the package was asked to repeat the process.
- When and if the package eventually reached the contact person in Boston, Milgram could know the entire path taken and count the number of times it had been forwarded from person to person.
- Of the 96 packages, 18 found their way to the target.
- The mean length of the completed paths was just 5.9 steps.
- $\Rightarrow$ **origin of the idea of the "six degree of separation": the popular belief that there are only about 6 steps between any two people in the world.**
- It is now widely accepted that vertex pairs in social networks tend on average to be connected by short paths.

Small-world networks

- A small-world network is a graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of steps.
- ℓ = mean distance between vertices, n = number of vertices

  Small-world network: a network where ℓ grows proportionally to the logarithm of n:

$$\ell \propto \log n$$

Small-world networks

- A small-world network is a graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of steps.
- ℓ = mean distance between vertices, n = number of vertices

  Small-world network: a network where ℓ grows proportionally to the logarithm of n:

$$\ell \propto \log n$$

- In 1998, Duncan J. Watts and Steven Strogatz from Cornell University published the first network model on the small-world phenomenon.
  They showed that networks from both the natural and man-made world exhibit the small-world phenomenon.

Small-world networks

- A small-world network is a graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of steps.
- ℓ = mean distance between vertices, n = number of vertices

  Small-world network: a network where ℓ grows proportionally to the logarithm of n:

$$\ell \propto \log n$$

- In 1998, Duncan J. Watts and Steven Strogatz from Cornell University published the first network model on the small-world phenomenon.
  They showed that networks from both the natural and man-made world exhibit the small-world phenomenon.

- Substantial implications for networked systems:
  - rumor spread over a social network,
  - speed of response from a computer on the Internet,
  - etc..

Degree distribution

How is the frequency of distribution of vertex degree?

p_k = degree distribution of the network = the fraction of vertices that have degree k

p_k can be also thought as the probability that a random chosen vertex has degree k

- A knowledge of the degree distribution does not tell us the complete structure of a network.
- Interestingly, often from the degree distribution we see that the mostly highly connected vertex is connected to only a small percentage of the other vertices (Example: Internet, 12%)
  $\Rightarrow$ *hub*

Scale-free networks

- Scale-free network: network whose degree distribution follows a *power law*, at least asymptotically:

$$p_k \sim Ck^{-\gamma}$$

(the fraction of vertices having k connections goes as $k^{-\gamma}$ for large k)

- C constant, γ typically in the range $2 \leq \gamma \leq 3$

Scale-free networks

- Scale-free network: network whose degree distribution follows a *power law*, at least asymptotically:

$$p_k \sim Ck^{-\gamma}$$

(the fraction of vertices having k connections goes as $k^{-\gamma}$ for large k)

- C constant, γ typically in the range $2 \leq \gamma \leq 3$
- Studied by Barabasi and Albert (1999)
- Examples: the Internet, social networks, etc..

Scale-free networks

- Scale-free network: network whose degree distribution follows a *power law*, at least asymptotically:

$$p_k \sim Ck^{-\gamma}$$

(the fraction of vertices having k connections goes as $k^{-\gamma}$ for large k)

- C constant, γ typically in the range $2 \leq \gamma \leq 3$
- Studied by Barabasi and Albert (1999)
- Examples: the Internet, social networks, etc..
- Characteristics:

  - often presence of hubs

  - scale-free property strongly correlates with the network's robustness to failure

  - the clustering coefficient distribution decreases as the node degree increases.

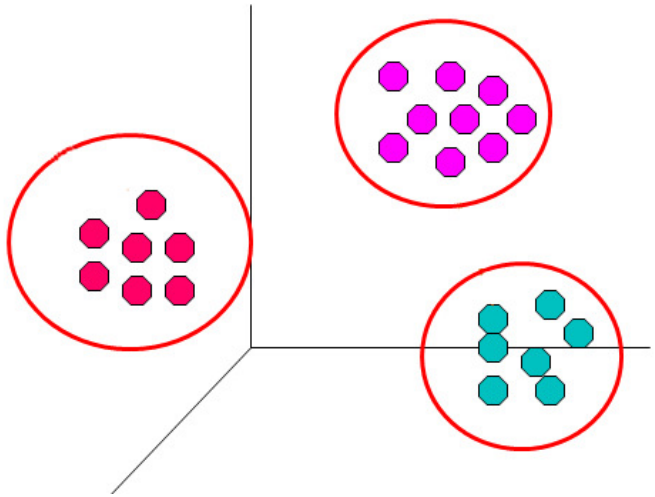
Outline

Clustering

Given a set of entities, find subsets, or **clusters**, which are *homogeneous* and/or *well separated*

Homogeneity: entities in the same cluster must be similar

Separation: entities in different clusters must differ one from the other

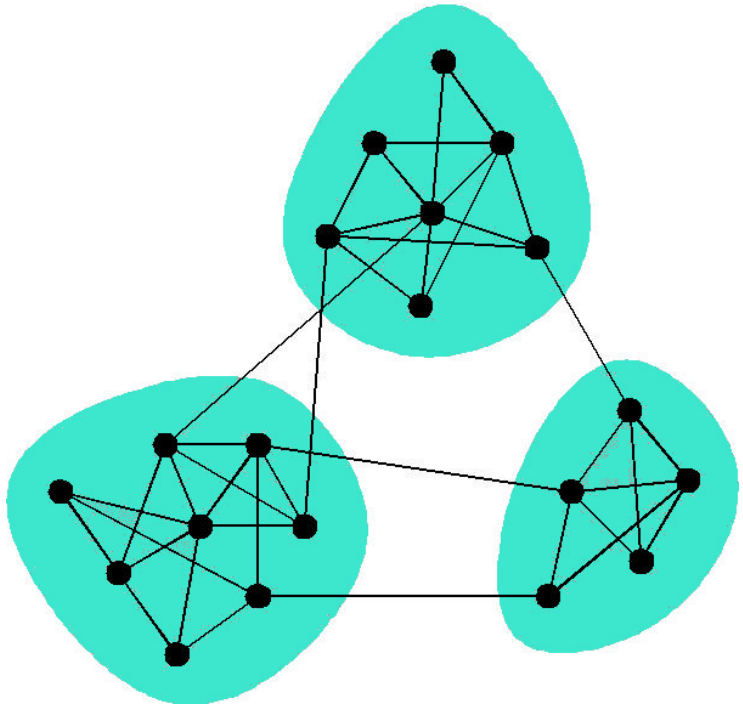


- This problem can be traced back to Aristotle and was already much studied by 18th century naturalists such as Buffon, Cuvier and Linné.
- Applications in the natural sciences, medicine, engineering, economics, marketing and other fields.
- The data usually correspond to *observations* or *measurements* on given entities.

Clustering –> Graph Clustering

Graphical representation of data as graphs

→ finding data patterns → **network (graph) clustering**



- Many vertices inside clusters
- Few edges between clusters

cluster or **community** = a subset of nodes which are more densely linked compared to the rest of the network

Examples

A community...

- in a Social network:
  individuals sharing a common interest or location
- in a Biological network:
  entities with a common function
- in the World Wide Web:
  web-pages having a common topic or language
- etc...

A modular structure characterizes many complex systems, meaning that they contain subgroups of entities sharing some common properties

Clustering on networks

Community detection useful to...

- identify some properties of the system described by the studied network starting from its structural features
- study modules individually based on their properties
- visualize and analyze to a higher level very large and complex networks by compressing its modules in single nodes
- ...

Partitions

A community corresponds to a subgraph $G_S = (S, E_S)$ of a graph $G = (V, E)$:
a graph with vertex set $S \subseteq V$, edge set E_S equal to all edges with both vertices in S.

$\Rightarrow$ finding a partition of V into subgraphs induced by nonempty subsets

$$V_1, V_2, \ldots, V_M$$

such that

$$V_k \cap V_l = \emptyset \quad \forall k \in 1, 2, \ldots M$$

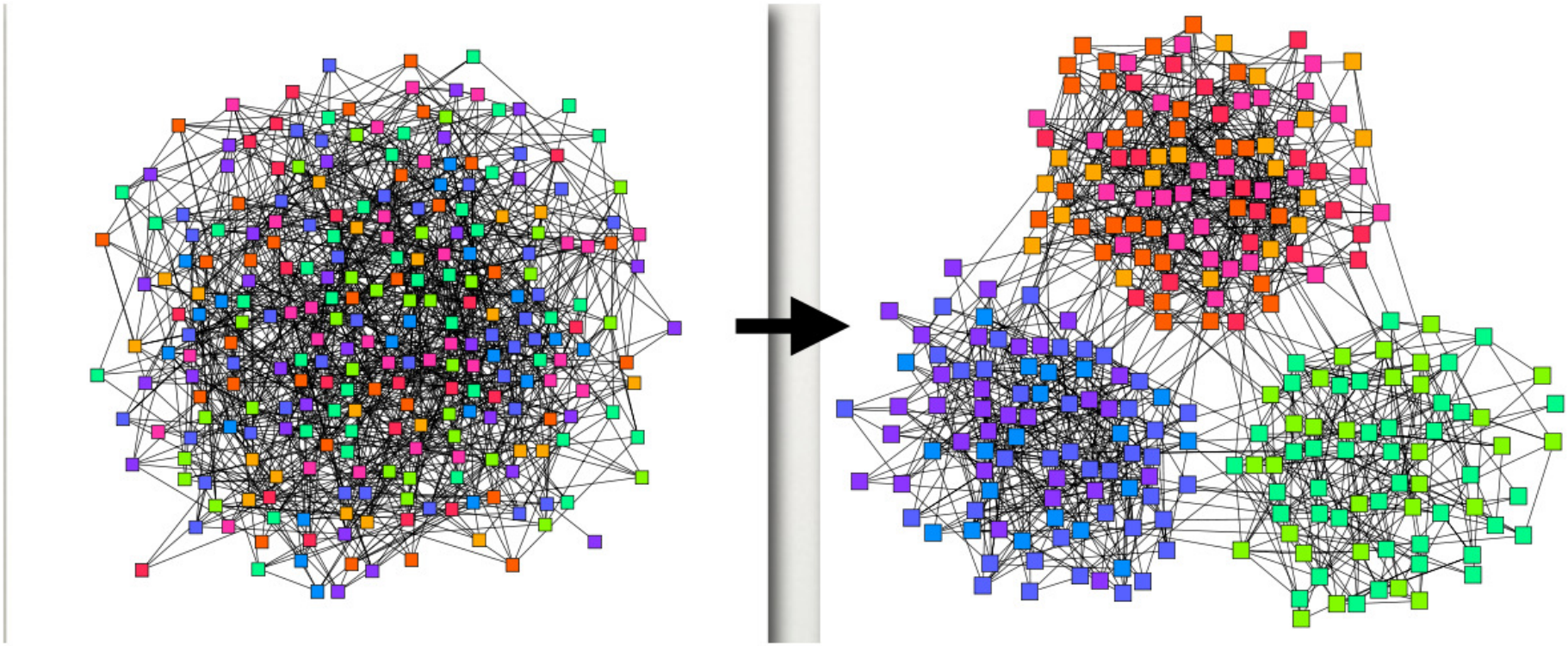
$$V_1 \cup V_2 \cup \ldots \cup V_M = V$$

.

How to find and evaluate a partition?

Clustering: finding communities

How to find and evaluate a partition?



We need

- a clustering criterion / definition of community
- a clustering algorithm

Dividing networks into clusters

* Graph partitioning: dividing the vertices of a graph

such that the number of edges from one subgroup of vertices to another is minimized, usually posing limits on the number of groups as well as on their relative size.

The number and sizes of the groups are fixed.

Typically used in parallel computing.

Dividing networks into clusters

* Graph partitioning: dividing the vertices of a graph
such that the number of edges from one subgroup of vertices to another is minimized, usually posing limits on the number of groups as well as on their relative size.

The number and sizes of the groups are fixed.

Typically used in parallel computing.

* Community detection: dividing the vertices of a graph
such that there are *many edges* within groups and *few edges* between groups, where the size and number of groups are not specified and are determined by the network itself.

Finding the natural fault lines along which a network separates.

The terms community identification, graph clustering and graph partitioning are often used interchangeably in this context.

Graph Clustering

The problem can be formulated using mathematical programming.

Several formulations can be considered, depending on the definition of community as well as the criterion chosen to evaluate the quality of the partition.

The research in this context is in fact generally essentially addressed

- on the one hand, to proposing and evaluating clustering criteria
- on the other hand, to devising efficient solution methods for the corresponding optimization problems

Outline

Clustering criteria / community definition

Two main approaches to evaluate a partition:

(i) Choose a criterion function, to be maximized or minimized

$\Longrightarrow$ define an **optimization problem**

(ii) Specify conditions to be satisfied by a community

Clustering criteria / community definition

Two main approaches to evaluate a partition:

(i) Choose a criterion function, to be maximized or minimized

Example: Modularity (Newman & Girvan, 2004)

$\Longrightarrow$ define an **optimization problem**

(ii) Specify conditions to be satisfied by a community

Clustering criteria / community definition

Two main approaches to evaluate a partition:

(i) Choose a criterion function, to be maximized or minimized

Example: Modularity (Newman & Girvan, 2004)

$\Longrightarrow$ define an **optimization problem**

(ii) Specify conditions to be satisfied by a community

Example: Strong and Weak conditions (Radicchi et al., 2004)

Clustering criteria / community definition

Two main approaches to evaluate a partition:

(i) Choose a criterion function, to be maximized or minimized

Example: Modularity (Newman & Girvan, 2004)

$\Longrightarrow$ define an **optimization problem**

(ii) Specify conditions to be satisfied by a community

Example: Strong and Weak conditions (Radicchi et al., 2004)

Further possibility/recent direction:
combining a criterion for community evaluation with constraints on each community

Clustering criteria

- Minimum cut:

$$\min_{C_1,\ldots,C_k} \sum_{s=1}^{k} links(C_s, V \backslash C_s)$$

- Ratio cut (Cheng and Wei, 1991):

$$\min_{C_1,\ldots,C_k} \sum_{s=1}^{k} \frac{links(C_s, V \backslash C_s)}{|C_s|}$$

- Normalized cut (Shi and Malik, 2000):

$$\min_{C_1,\ldots,C_k} \sum_{s=1}^{k} \frac{links(C_s, V \backslash C_s)}{degree(C_s)}$$

- Min-max cut (Ding et al., 2001):

$$\min_{C_1,\ldots,C_k} \sum_{s=1}^{k} \frac{links(C_s, V \backslash C_s)}{links(C_s, C_s)}$$

Modularity

Newman and Girvan, 2004:

compare the fraction of edges falling within communities to the expected fraction of such edges

Modularity:

$$Q = \sum_s [a_s - e_s]$$

a_s = fraction of all edges in module s

e_s = expected value of the same quantity in a graph with same vertex degree and edges placed at random

Modularity

Newman and Girvan, 2004:

compare the fraction of edges falling within communities to the expected fraction of such edges

Modularity:

$$Q = \sum_s [a_s - e_s]$$

a_s = fraction of all edges in module s

e_s = expected value of the same quantity in a graph with same vertex degree and edges placed at random

- $Q \approx 0$: the network is equivalent to a random network (barring fluctuations)
- $Q \approx 1$: the network has a strong community structure
- in practice, max Q often between 0.3 and 0.7

Maximizing modularity gives an optimal partition with the optimal number of clusters

Criticism to modularity

- Resolution limit:

  in the presence of large clusters, some clusters smaller than a certain size can be undetectable $\Rightarrow$ modular structures like small cliques can be hidden in larger clusters.

- Degeneracy of Q:

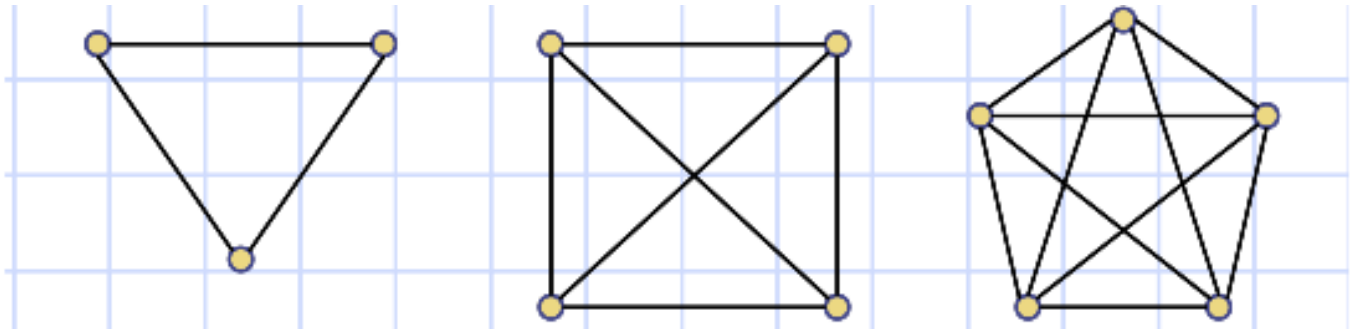
  there can be a large number of partitions, even very different from each other, having high modularity values $\Rightarrow$ easy to find high-scoring partitions but difficult to identify the global optimum.

Community definition

- Clique S:

all pairs of vertices of S are joined by an edge

$$k_i = |S| - 1 \qquad \forall v_i \in S$$



Community definition

- Community S in the *weak sense* (Radicchi et al., 2004)

the sum of all *degrees within S* is larger than
the sum of all *degrees joining S to the rest* of the network:

$$\sum_{i \in S} k_i^{in}(S) > \sum_{i \in S} k_i^{out}(S)$$

(*the number of edges within S is at least half the number of edges in the cut of S*)

Community definition

- Community S in the *weak sense* (Radicchi et al., 2004)

the sum of all *degrees within S* is larger than
the sum of all *degrees joining S to the rest* of the network:

$$\sum_{i \in S} k_i^{in}(S) > \sum_{i \in S} k_i^{out}(S)$$

(*the number of edges within S is at least half the number of edges in the cut of S*)

- Community S in the *strong sense* (Radicchi et al., 2004)

every one of its vertices has more neighbors within the community than outside:

$$k_i^{in}(S) > k_i^{out}(S) \qquad \forall v_i \in S$$

Outline

Is partitioning hard?

The simplest graph partitioning problem is the graph bisection

- Aim: Dividing the vertices into two groups such that the *cut size*, i.e., the number of edges between the groups, is minimized.

Is partitioning hard?

The simplest graph partitioning problem is the graph bisection

- Aim: Dividing the vertices into two groups such that the *cut size*, i.e., the number of edges between the groups, is minimized.
- Naive approach: *exhaustive search*
  simply look through all possible bisections of the required size and choose the one with the smallest cut size.
- What is the computational complexity?

Is partitioning hard?

The simplest graph partitioning problem is the graph bisection

- Aim: Dividing the vertices into two groups such that the *cut size*, i.e., the number of edges between the groups, is minimized.
- The number of ways of dividing $G = (V, E)$ into 2 groups of n_1 and n_2 vertices is $n!/(n_1!\, n_2!)$.

  Stirling's formula: $n! \simeq \sqrt{2\pi n}(n/e)^n$ and $n = n_1 + n_2 \Rightarrow$

$$\frac{n!}{n_1!\, n_2!} \simeq \frac{\sqrt{2\pi n}(n/e)^n}{\sqrt{2\pi n_1}(n_1/e)_1^n \sqrt{2\pi n_2}(n_2/e)_2^n} = \frac{n^{n+\frac{1}{2}}}{n_1^{n_1+\frac{1}{2}} n_2^{n_2+\frac{1}{2}}}$$

Is partitioning hard?

The simplest graph partitioning problem is the graph bisection

- Aim: Dividing the vertices into two groups such that the *cut size*, i.e., the number of edges between the groups, is minimized.
- The number of ways of dividing $G = (V, E)$ into 2 groups of n_1 and n_2 vertices is $n!/(n_1!\, n_2!)$.

  Stirling's formula: $n! \simeq \sqrt{2\pi n}(n/e)^n$ and $n = n_1 + n_2 \Rightarrow$

$$\frac{n!}{n_1!\, n_2!} \simeq \frac{\sqrt{2\pi n}(n/e)^n}{\sqrt{2\pi n_1}(n_1/e)_1^n \sqrt{2\pi n_2}(n_2/e)_2^n} = \frac{n^{n+\frac{1}{2}}}{n_1^{n_1+\frac{1}{2}} n_2^{n_2+\frac{1}{2}}}$$

- $\Rightarrow$ the number of ways to divide a network into 2 parts of equal size $n/2$ is:

$$\frac{n+\frac{1}{2}}{(\frac{n}{2})^{n+1}} = \frac{2^{n+1}}{\sqrt{n}}$$

- the amount of time to look through all the possible divisions goes up exponentially with the size of the network

Partitioning into more than 2 groups

- In general, more than 2 communities
- We do not want to have to specify the number of communities
- We could proceed by doing successive bisections into smaller and smaller networks.

  Need to pay attention to the fact that maximizing a criterion function on separate communities may not maximize it for the network as a whole
- Modularity maximization allows us to handle the problem maximizing over divisions into any number of groups

Clustering algorithms

- Exact algorithms
  - can only solve small instances (with about a hundred entities) in reasonable time
  - provide an optimal solution together with the proof of its optimality
- Heuristics
  - widely used
  - can solve approximately very large instances with up to hundred or thousand entities
  - do not have either an a priori performance guarantee
    (finding always a solution with a value which is at least a given percentage of the optimal one),
    nor an a posteriori performance guarantee
    (that the obtained solution is at least a computable percentage of the optimal one)

Clustering algorithms

Heuristics based on:

Partitioning schemes
aim at finding the best partition into a given number of clusters

- greedy,
- simulated annealing,
- genetic search,
- a variety of other approaches

Hierarchical clustering
lead to a set of nested partitions

- agglomerative schemes
- divisive schemes

Modularity maximization: greedy

(Blondel et al., 2008)

- Greedy algorithm:
- Start with the singleton partition and an ordering of the nodes $\alpha_1, .., \alpha_n$
- Move each vertex from its current class to a neighbour's class and compute the change in modularity
- Select the move with maximal increase in modularity
- Repeat until no further increases in modularity are possible
- The result depends on the order of vertices

Modularity maximization: simulated annealing

Simulated Annealing (SA):

stochastic optimization method (meta-heuristic)

- Based on a temperature T: when T is high the system can explore configurations of high cost whereas at low T the system only explores low-cost regions.
- By starting at high T and slowly decreasing T, the system descends gradually towards minima.

Modularity maximization: simulated annealing

Simulated Annealing (SA):

stochastic optimization method (meta-heuristic)

- Based on a temperature T: when T is high the system can explore configurations of high cost whereas at low T the system only explores low-cost regions.
- By starting at high T and slowly decreasing T, the system descends gradually towards minima.

SA for modularity maximization (Guimerà and Amaral, 2005)

- Using modularity, the cost is $-Q$.

Modularity maximization: simulated annealing

- At each temperature, perform a number of random updates and accept them with probability:

$$p = \begin{cases} 1 & \text{if increase in modularity} \\ exp(\frac{\Delta Q}{T}) & \text{otherwise} \end{cases} \qquad (\Delta Q = \text{old modularity minus new modularity})$$

- Updates include:
  - $f\, n^2$ *individual steps*:
    choosing a node and a community to move it to randomly
  - $f\, n$ *collective steps*:
    randomly choosing two communities to merge, or one community to split

    typically, $f = 1$
- After the movements are evaluated at a certain T, cool down the system to $T = cT$, with $c = 0.995$

Modularity maximization: genetic search

Genetic Algorithm (GA):
stochastic optimization method (meta-heuristic)

A set of candidate solutions is encoded as a kind of numerical *chromosome*.

- GA starts generating a random population of candidate solutions.
- Candidates are evaluated by using the *fitness* (cost) function.
- A new population of solutions is generated by applying biologically inspired manipulations: *mutation*, *crossover*.
- Solutions with higher fitness values are kept in the next generation.

Modularity maximization: genetic search

Genetic Algorithm (GA):
stochastic optimization method (meta-heuristic)

A set of candidate solutions is encoded as a kind of numerical *chromosome*.

- GA starts generating a random population of candidate solutions.
- Candidates are evaluated by using the *fitness* (cost) function.
- A new population of solutions is generated by applying biologically inspired manipulations: *mutation*, *crossover*.
- Solutions with higher fitness values are kept in the next generation.

GA for modularity maximization: (Tasgin et al., 2008)

- Candidate solutions are partitions
- κ_i partition,
  $P = \{\kappa_1, ..., \kappa_p\}$ population, set of chromosomes.

Modularity maximization: genetic search

- Start with a random chromosome, for example each vertex forms a cluster
- Repeat the main loop a given number of times:
  - Apply the fitness function to chromosomes
  - Sort the chromosomes with respect to the fitness value and take the top p
  - Pair the sorted chromosomes and apply crossover to the pairs. Apply mutation.
  - Combine newly obtained p chromosomes and the previously saved p
- **Crossover** on a pair *source-destination chromosome*:

TABLE I: One-way crossing over when v_4 is selected

v		κ_{src}		κ_{dest} (before)		κ_{dest} (after)
1		⑦	→	2	→	⑦
2		⑦	→	2	→	⑦
3		2		5		5
4	→	⑦	→	8	→	⑦
5		⑦	→	3	→	⑦
6		3		3		3
7		9		7		7
8		9		4		4

(Tasgin et al.)

- **Mutation**: randomly pick a chromosome to be mutated; pick two vertices v_i and v_j randomly. Set the cluster of v_j to the cluster of v_i.

Hierarchical heuristics

Hierarchical heuristics are in principle devised for
finding a hierarchy of partitions implicit in the given network
when it corresponds to some situation where hierarchy is observed or postulated.
They aim at finding a set of nested partitions.

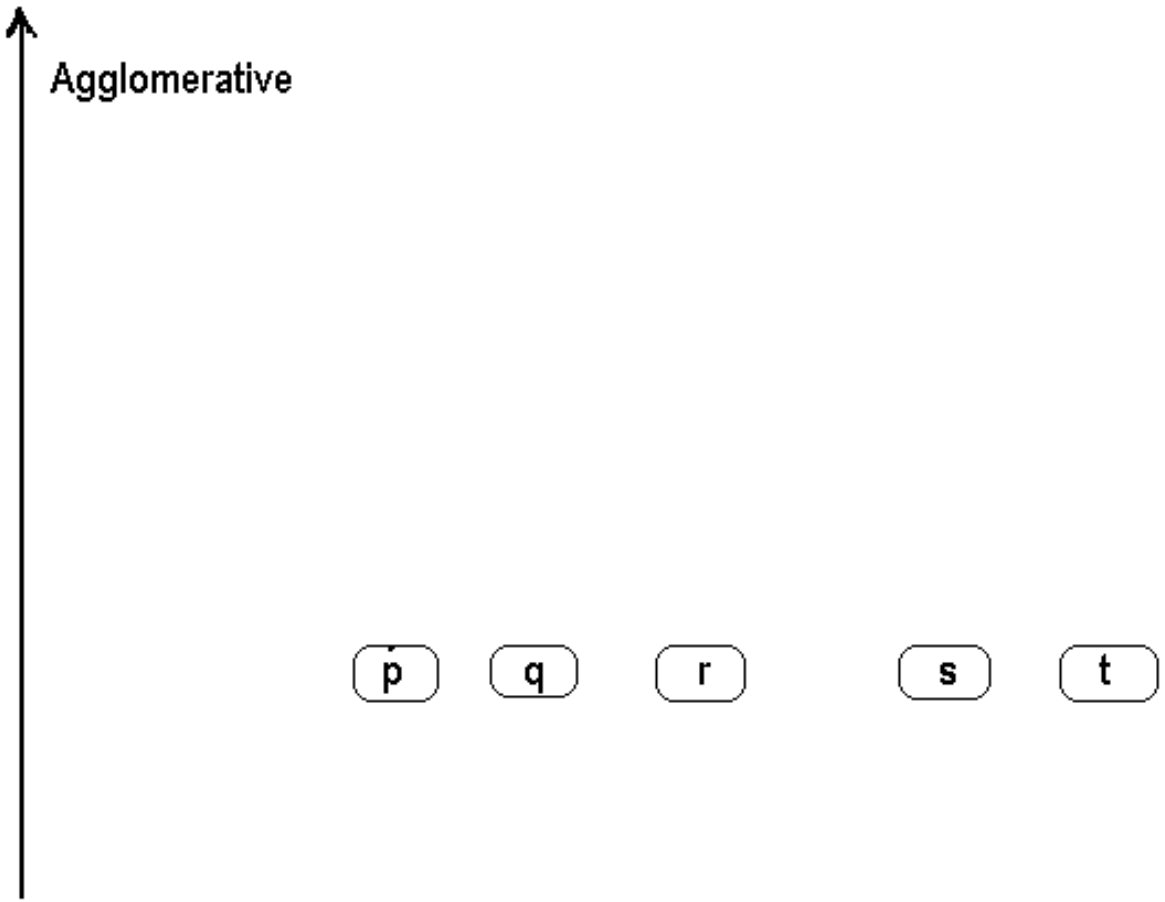
- Agglomerative heuristics
- Divisive heuristics

Hierarchical agglomerative clustering

- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)

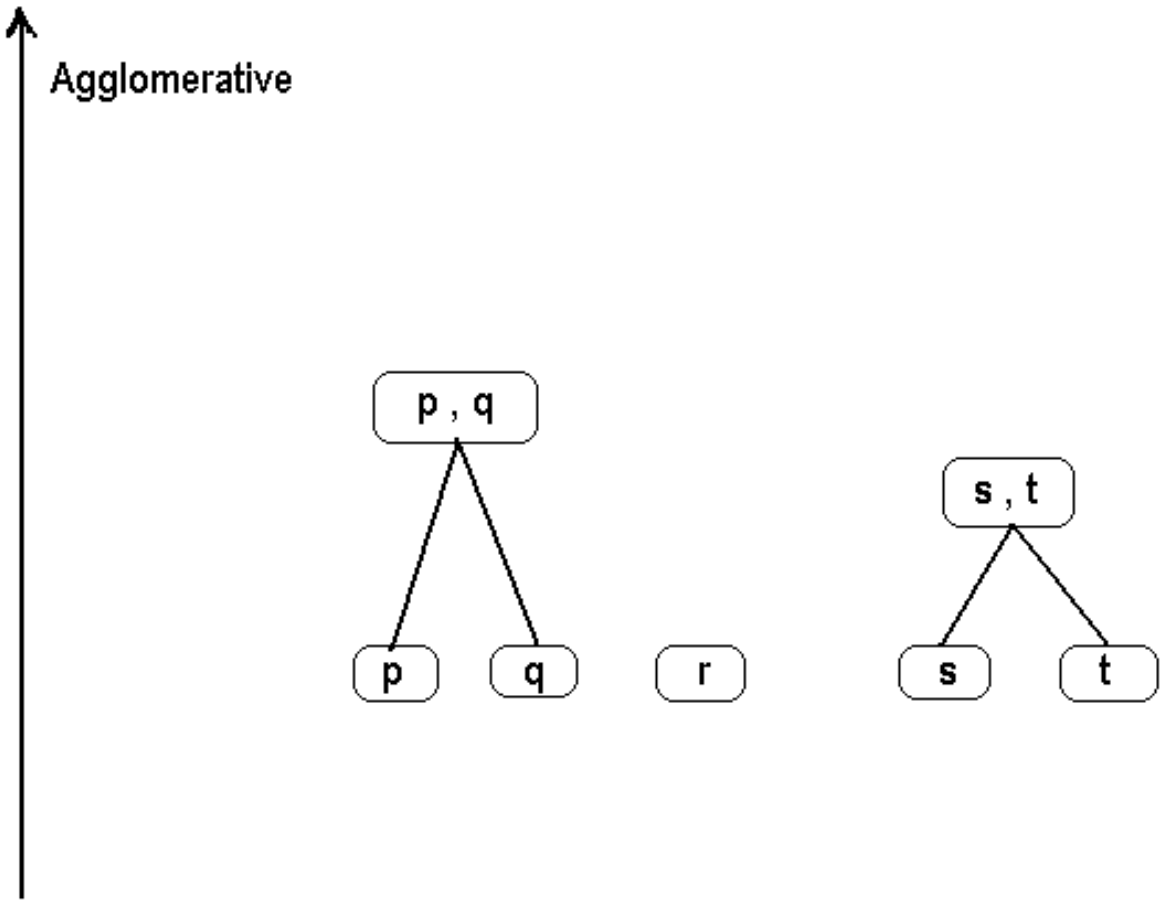
Hierarchical agglomerative clustering

- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)



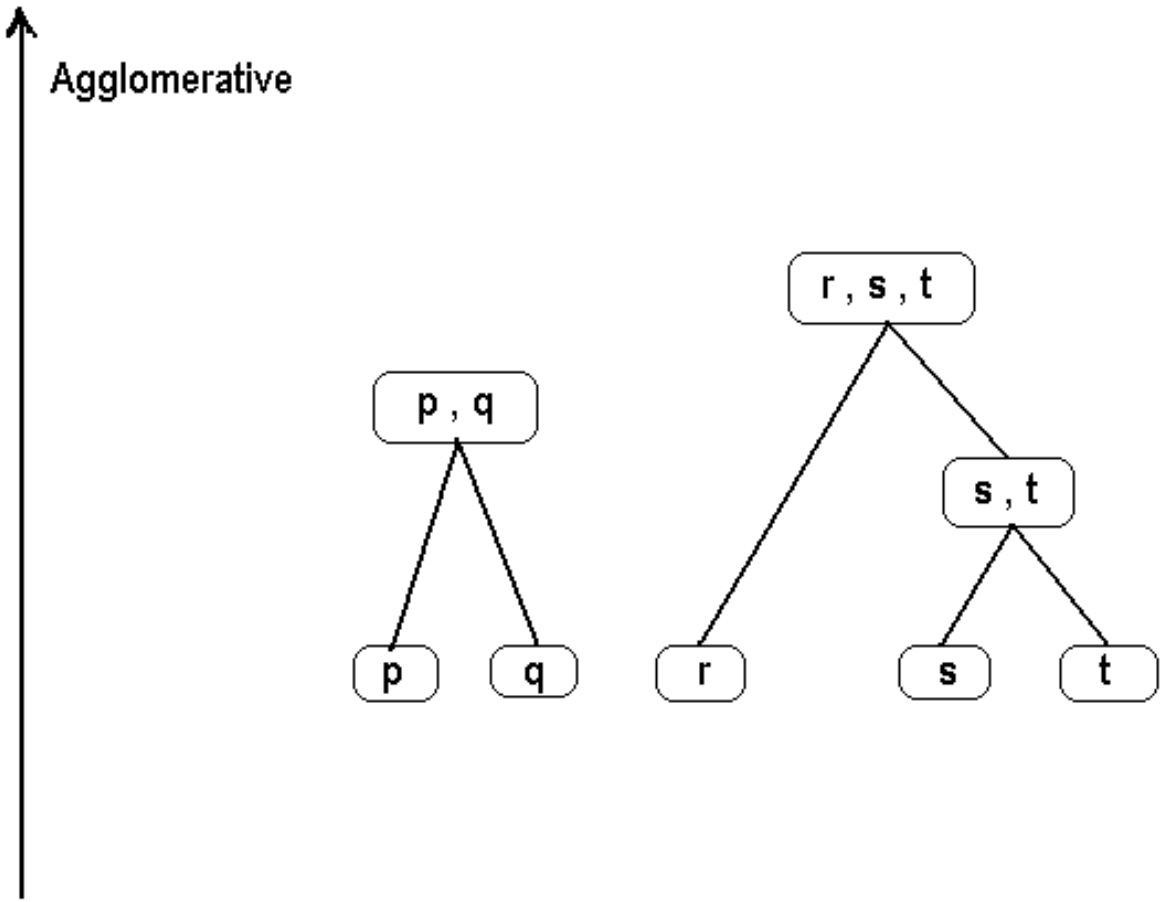
Hierarchical agglomerative clustering

- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)



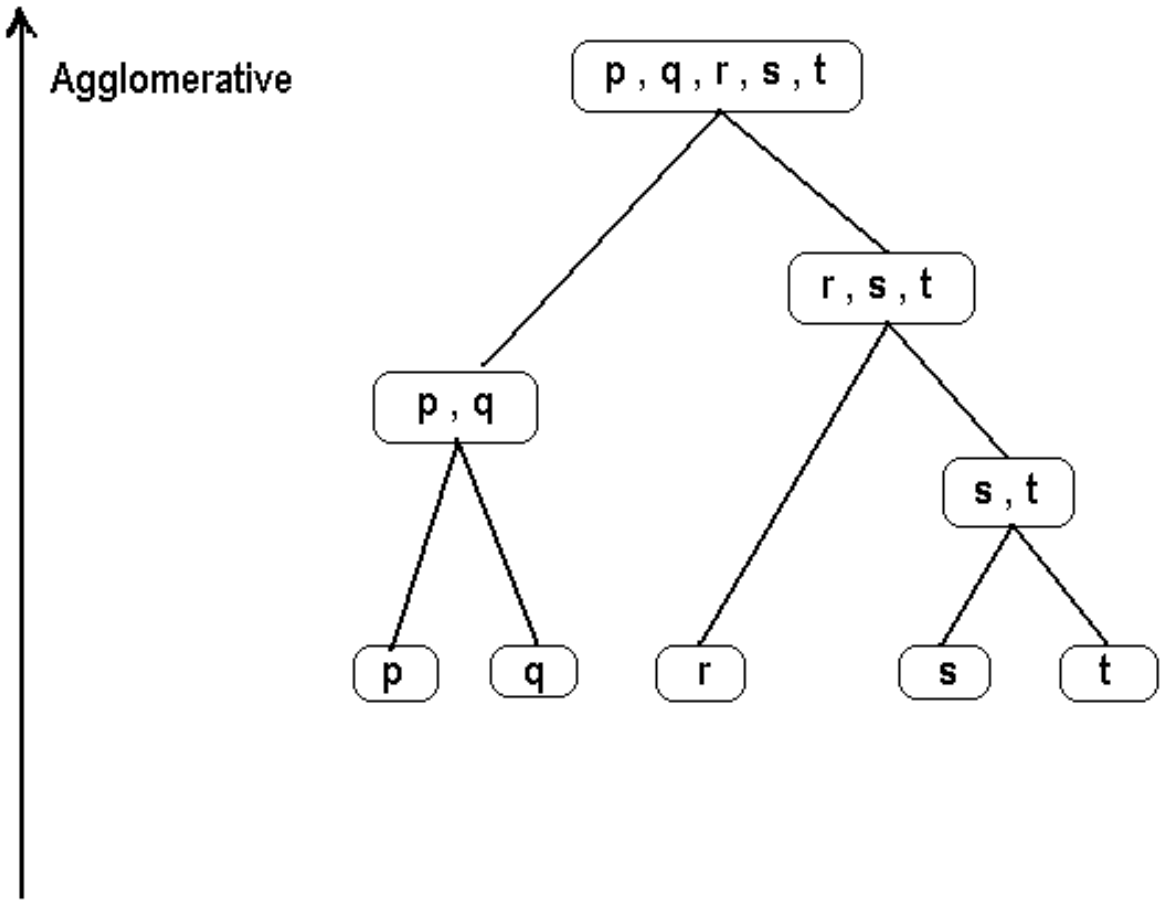
Hierarchical agglomerative clustering

- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)



Hierarchical agglomerative clustering

- Proceed from an initial partition with n communities each containing 1 entity
- Iteratively merge the pair of entities for which merging increases most the objective function (e.g., modularity)

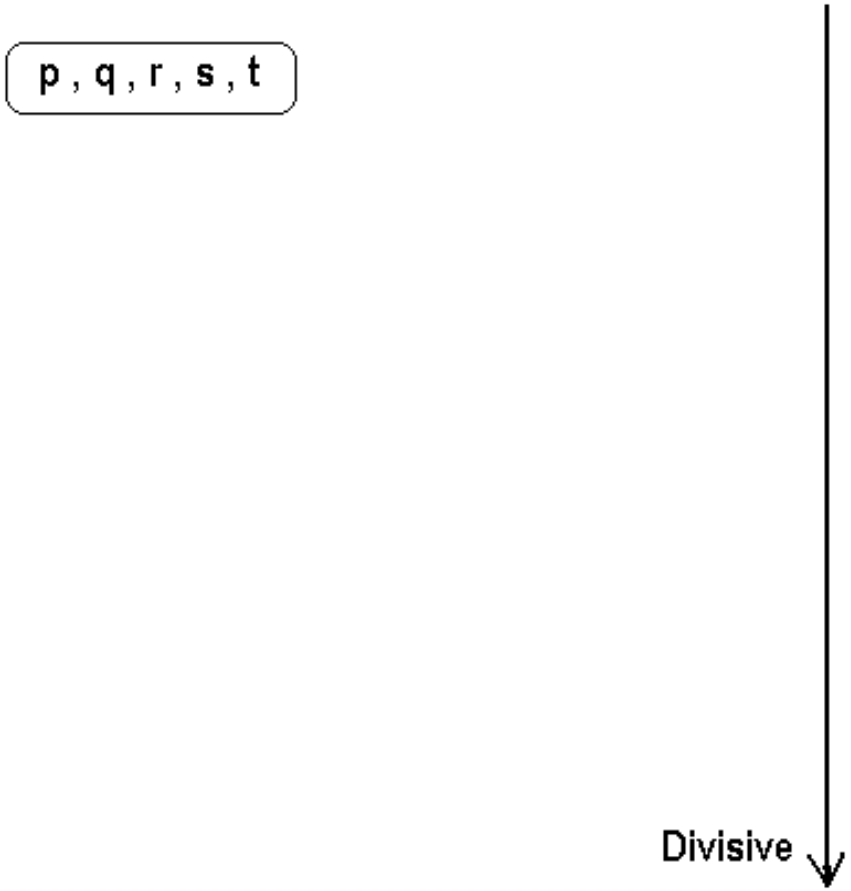


Hierarchical divisive clustering

- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)

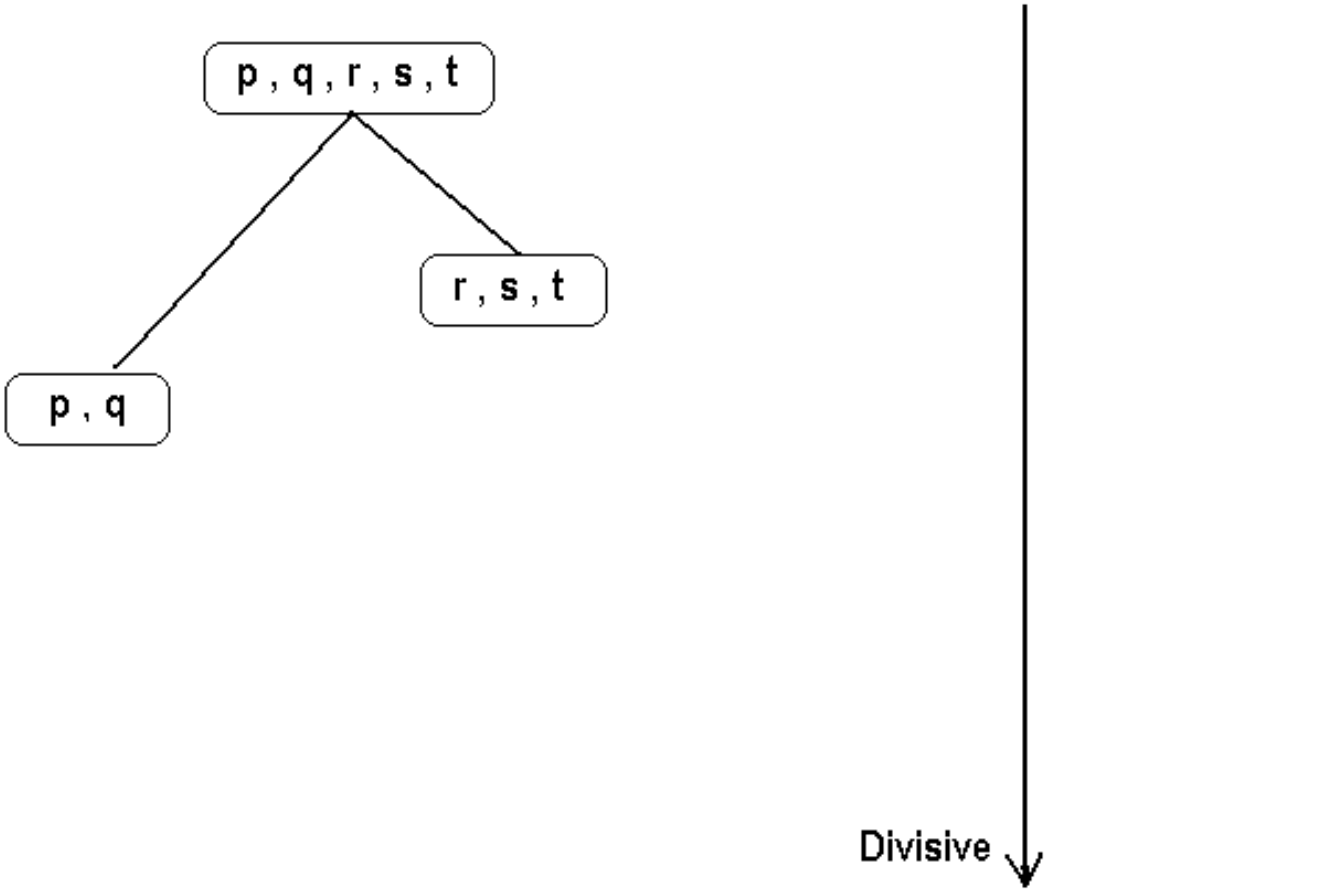
Hierarchical divisive clustering

- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)



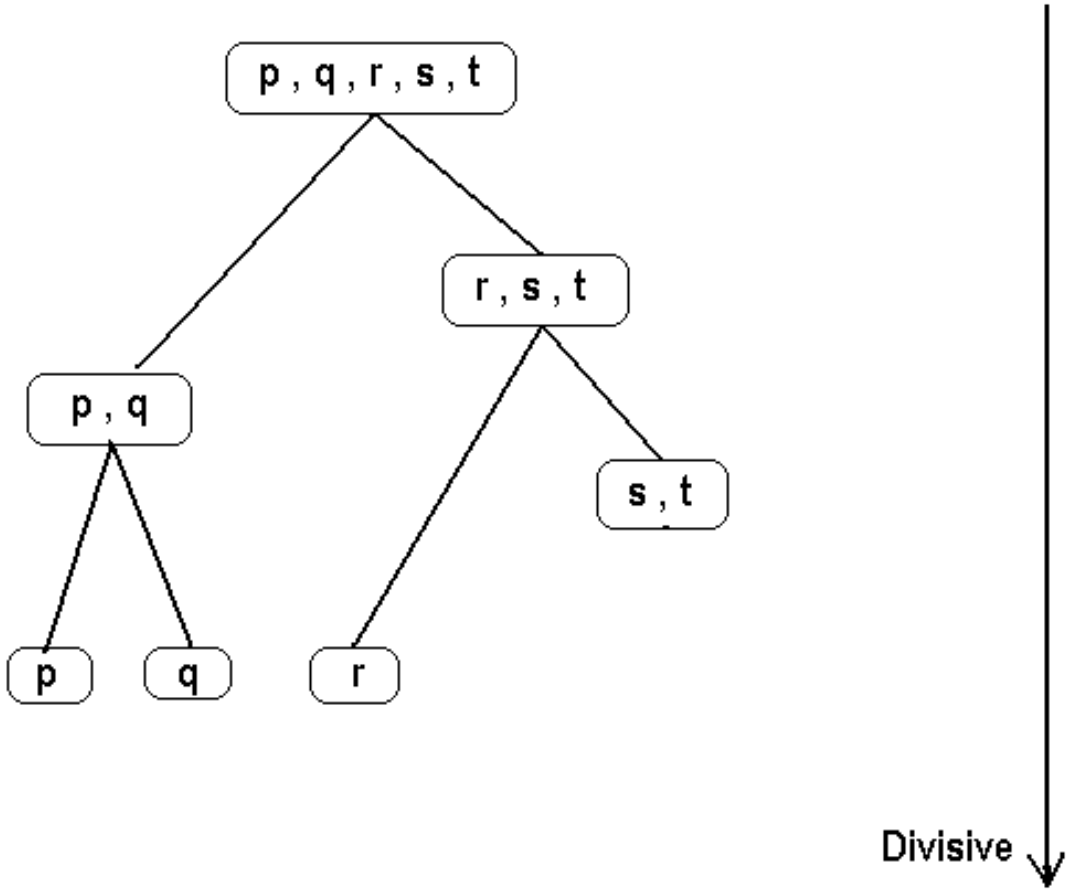
Hierarchical divisive clustering

- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)



Hierarchical divisive clustering

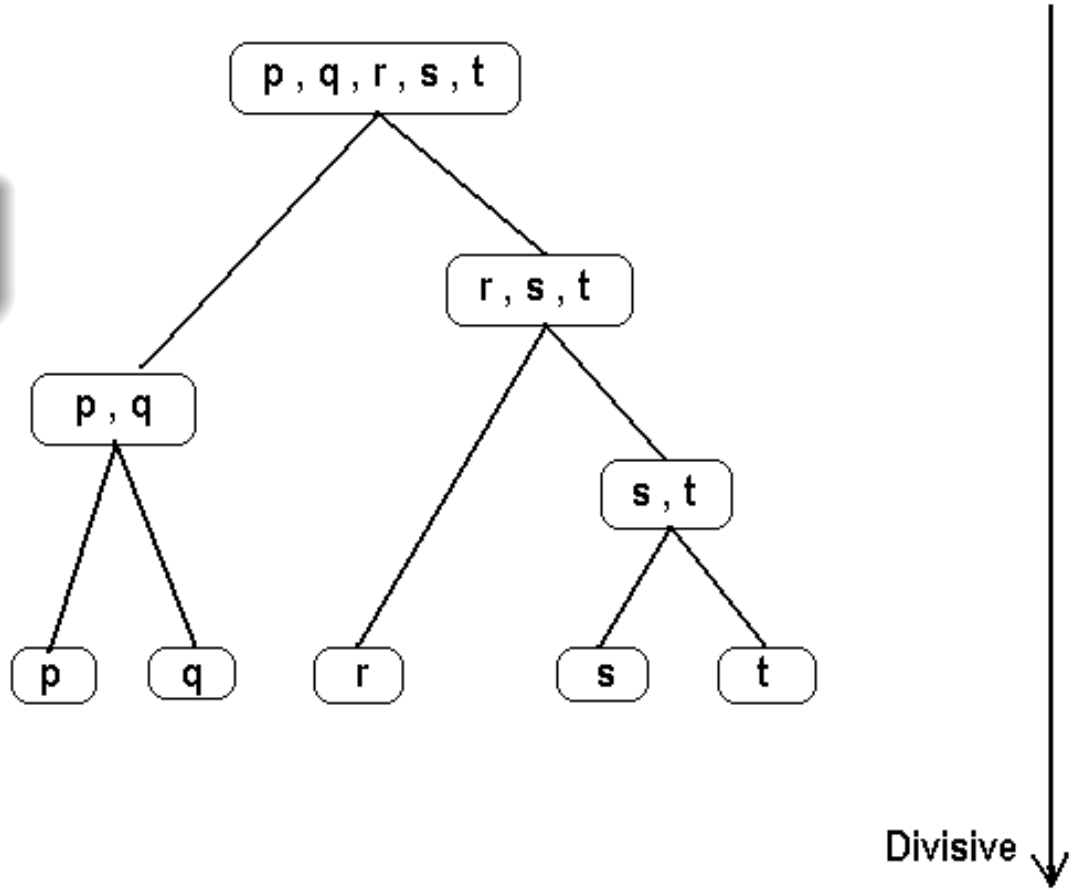
- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)



Hierarchical divisive clustering

- Proceed from an initial partition containing all entities
- Iteratively divide a cluster into two in such a way to increase most the objective function (or the decrease in the objective value is the smallest possible)

critical point:
bipartitioning a cluster



Heuristics in modularity literature: successful approaches

- Clauset, Newman and Moore, 2004:
  agglomerative hierarchical greedy, for sparse networks has a very low complexity and is considerably faster than previously proposed methods.
- Newman, 2006:
  divisive hierarchical heuristic based on spectral graph theory,
  splitting is done according to the sign of the components of the first eigenvector of the modularity matrix.
- Noack and Rotta, 2009:
  heuristic based on a single-step coarsening with a multi-level refinement,
  competitive with other methods in the literature.
- Liu and Murata, 2010:
  heuristic based on label propagation, gives better results than previous heuristics

Exact algorithms for modularity maximization

Why using exact algorithms?

- having an exact solution solves the problem of separating possible inadequacies of the model from eventual errors resulting from the use of heuristics
  $\Rightarrow$ communities may be interpreted with more confidence
- an exact algorithm may be stopped and the best solution found considered as a heuristic one
  (it is not uncommon that the optimal solution is found at an early stage of the resolution)
- an exact algorithm can provide a benchmark of exactly solved instances which can be used to compare heuristics and fine tune them

Exact algorithms for modularity maximization

- Modularity maximization is NP-hard (Brandes et al., 2008)
- Exact algorithms are rare. Based on mathematical programming formulations.

Exact algorithms for modularity maximization

- Modularity maximization is NP-hard (Brandes et al., 2008)
- Exact algorithms are rare. Based on mathematical programming formulations.

Two approaches:

★ reduction of modularity maximization to clique partitioning

$\Rightarrow$ linear optimization problem (LP) in 0-1 variables

★ direct formulation

$\Rightarrow$ mixed 0-1 quadratic optimization problem

Four algorithms:

□ row generation for clique partitioning (Grötschel and Wakabayashi, 1990)

□ column generation for clique partitioning (Cafieri et al., 2010)

□ mixed-integer convex quadratic programming approach (Xu et al., 2007)

□ column generation for the mixed-integer quadratic approach (Cafieri et al., 2010)

Using mathematical programming

$$\left.\begin{array}{l} \min f(x) \\ g(x) \leq b \\ x \in X \end{array}\right\} \Rightarrow$$

Mathematical programming formulation =

a set of entities:

- parameters
- decision variables ($x \in X$)
- objective function ($f(x)$)
- constraints ($g(x) \leq b$)

that define an optimization problem

How to use mathematical programming in network clustering?

Modularity: another expression

Modularity as a sum of values over all edges of the complete graph K_n:

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left(a_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

where:

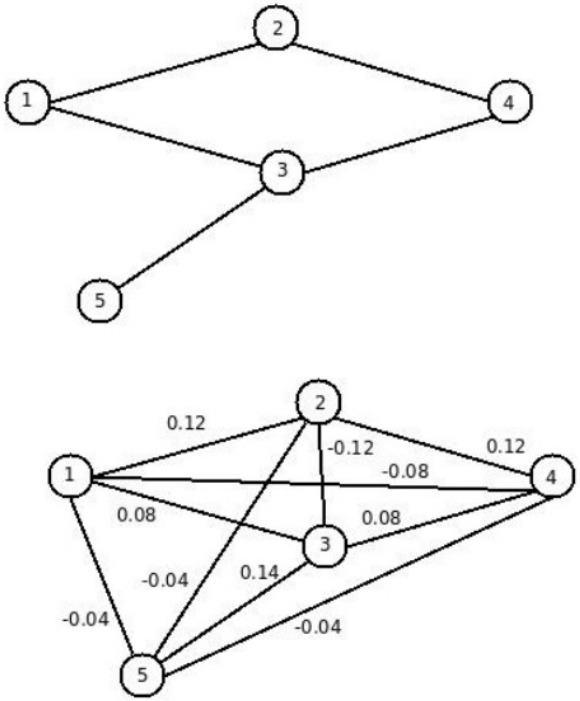
- $m = |E|$
- k_i, k_j = degrees of vertices i and j
- a_{ij} = ij component of the adjacency matrix of G
- $\delta(c_i, c_j)$ = 1 if the communities to which i and j belong are the same, 0 otherwise (Kronecker symbol)
- $k_i k_j / 2m$ = expected number of edges between vertices i and j in a null model where edges are placed at random and the distribution of degrees remains the same.

Modularity maximization as clique partitioning

Introducing

$$w_{ij} = \frac{1}{m}\left(a_{ij} - \frac{k_i k_j}{2m}\right)$$

modularity maximization can be reformulated as a clique partitioning problem:



- $m = 5$
- $k_1 = 2$
- $k_2 = 2$
- $k_3 = 3$
- $k_4 = 2$
- $k_5 = 1$
- $a_{12} = 1, m = 5, k_1 = 2, k_2 = 2$
- $\Rightarrow w_{12} = \frac{1}{5}\left(1 - \frac{2\times 2}{10}\right) = 0.12$

Modularity maximization as clique partitioning

Modularity maximization on K_n:

K_n complete graph $\Rightarrow$ it is a clique and any of its induced subgraphs are cliques.
partitioning G is thus equivalent to partitioning K_n into cliques.

The resulting partition is an equivalence relation:

- reflexivity: each entity is in the same module as itself: $\forall i \quad x_{ii} = 1$
- symmetry: if i is in the same module as j, j is in the same as i: $\forall i, j \quad x_{ij} = x_{ji}$
- transitivity: if i and j are in the same module and j and k are in the same module, then i and k must be in the same module

Modularity maximization as clique partitioning

Introducing binary variables

$$\begin{cases} x_{ij} = 1 & \text{if vertices } i,j \text{ belong to the same community} \\ \quad = 0 & \text{otherwise} \end{cases}$$

$$\begin{cases} \max & \displaystyle\sum_{i<j\in V} w_{ij}x_{ij} - C & -C = -\sum_{i\in V} \frac{k_i k_i}{2m} \\ \text{s.t.} & x_{ij} + x_{jk} - x_{ik} \leq 1 & \forall 1 \leq i < j < k \leq n \\ & x_{ij} - x_{jk} + x_{ik} \leq 1 & \forall 1 \leq i < j < k \leq n \\ & -x_{ij} + x_{jk} + x_{ik} \leq 1 & \forall 1 \leq i < j < k \leq n \\ & x_{ij} \in \{0,1\} & \forall 1 \leq i < j \leq n \end{cases}$$

(Grötschel and Wakabayashi, 1990)

Linear 0-1 program

$$\frac{n(n-1)}{2} = O(n^2) \text{ variables}$$

$$3\binom{n}{3} = \frac{n(n-1)(n-2)}{2} = O(n^3) \text{ constraints}$$

Row generation

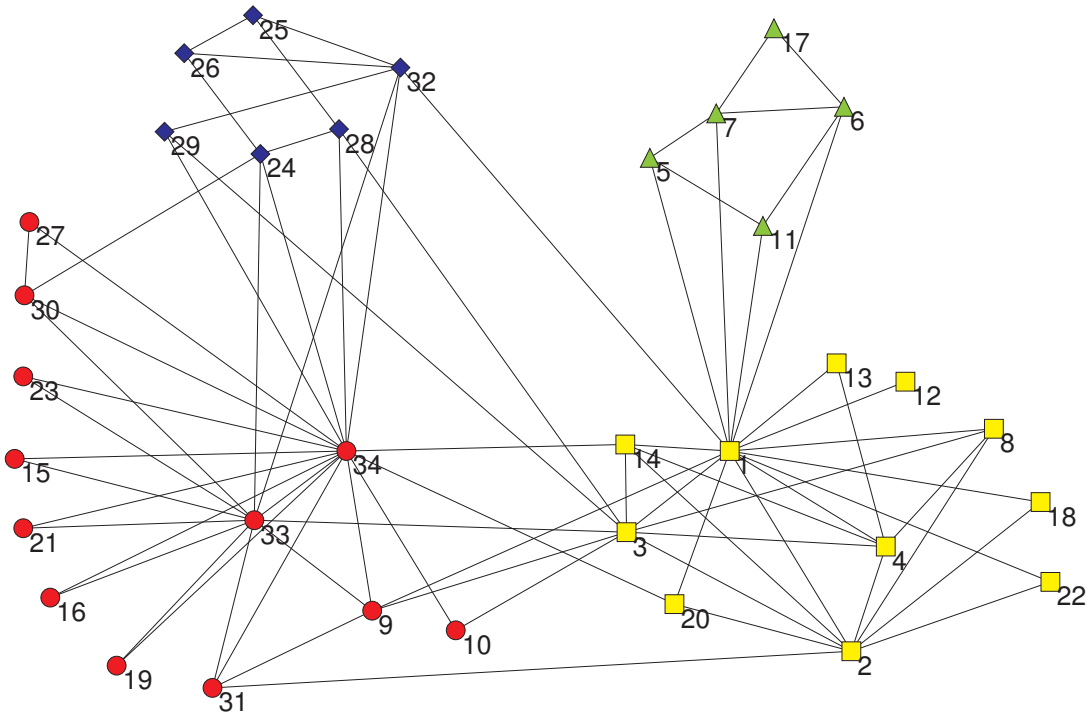
Typically used in combinatorial applications

1. the linear continuous relaxation is first solved
2. if the solution of this relaxation is in integers, it is optimal
   (often the case for modularity maximization)
3. if the solution of the continuous relaxation is fractional, then
   add valid constraints violated by the fractional solution: *cutting planes*
4. the number of constraints grows rapidly with n: they can be added by
   batches of unsatisfied ones.

Example: a social network

Zachary's karate club network

it describes friendship relations between 34 members of a karate club observed over two years by Zachary. In that period the club split into two groups after a dispute between the club owner and the karate instructor.



Column generation

Solving exactly linear programs with a number of variables (columns) exponential in the size of the input

1. select a small number of columns and solve the linear program using only these
2. find an unused column which, if included, would most improve the objective value (with favorable *reduced cost*) or determine that there is none
3. include the column in the linear program, re-solve it, and go to step 2.

The original problem is split into:

- Master problem:
  original problem with only a subset of variables being considered
- Subproblem:
  new problem created to identify a new variable

Column generation

Subproblem (auxiliary problem):

- its precise form depends on the type of problem under study
- it can be solved heuristically as long as a column with a reduced cost of the required sign can be found
- when this is no more the case, an exact algorithm must be applied either to find a column with the adequate reduced cost sign undetected by the heuristic or to prove that there is no such column

Column generation – clique partitioning

Idea: the columns correspond to all subsets of V (all nonempty modules)

implicitly take into account all possible communities

$$\begin{aligned} a_{it} &= 1 \quad \text{if vertex } i \text{ belongs to module } t \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Master problem:

$$\begin{cases} \max \sum_{t \in T} c_t z_t - C & \\ \sum_{t \in T} a_{it} z_t = 1 & \forall i = 1, \ldots, n \\ z_t \in \{0, 1\} & \forall t \in T \end{cases}$$

- sum of modularities of all selected modules minus the constant corresponding to the diagonal terms
- each entity must belong to only one module
- modules must be selected entirely or not at all

$$c_t = \sum_i \sum_{j>i} w_{ij} a_{it} a_{jt}$$

i.e., the value of the module indexed by t, $t = 1 \ldots 2^n - 1$

Column generation – clique partitioning

Improving columns are added progressively to relaxation of the master problem.

Reduced cost associated with column t: $c_t - \sum_i \lambda_i a_{it}$.

To find a column with positive red. cost, replace the coefficients a_{it} by variables y_i.

Auxiliary problem:

$$\max \sum_i \sum_{j>i} w_{ij} y_i y_j - \sum_i \lambda_i y_i$$

Quadratic program in 0-1 variables with a 100% dense matrix of coefficients

MIQP formulation (Xu, Tsoka and Papageorgiou, 2007)

Variables used to identify to which module each vertex and each edge belongs:

$$X_{rs} = \begin{cases} 1 & \text{if edge } r \text{ belongs to module } s \\ 0 & \text{otherwise} \end{cases} \qquad \forall r = 1, 2, \ldots m,\ s = 1, 2, \ldots M$$

$$Y_{is} = \begin{cases} 1 & \text{if vertex } i \text{ belongs to module } s \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i = 1, 2, \ldots n,\ s = 1, 2, \ldots M$$

$$\max Q = \sum_s [a_s - e_s] = \sum_s \left[\frac{m_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right]$$

m_s = number of edges in module s
d_S = sum of degrees k_i of vertices in s

- $m_s = \sum_r X_{rs}$ and $d_S = \sum_i k_i Y_{is}$
- $\sum_s Y_{is} = 1 \quad \forall i = 1, 2, \ldots n$ — each vertex belongs to one module
- $X_{rs} \le Y_{is} \quad \forall r = \{v_i, v_j\} \in E$
  $X_{rs} \le Y_{js} \quad \forall r = \{v_i, v_j\} \in E$
  — any edge $r = \{v_i, v_j\}$ can belong to module s $\Leftrightarrow$ both of its end vertices i,j belong to s
- $u_s \le u_{s-1}$ — module s nonempty $\Leftrightarrow$ $s-1$ is so ($u_s = 1$ if module s nonempty, 0 otherwise)
- symmetry-breaking constraints

MIQP formulation (Xu, Tsoka and Papageorgiou, 2007)

Variables used to identify to which module each vertex and each edge belongs:

$$X_{rs} = \begin{cases} 1 & \text{if edge } r \text{ belongs to module } s \\ 0 & \text{otherwise} \end{cases} \qquad \forall r = 1, 2, \ldots m,\ s = 1, 2, \ldots M$$

$$Y_{is} = \begin{cases} 1 & \text{if vertex } i \text{ belongs to module } s \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i = 1, 2, \ldots n,\ s = 1, 2, \ldots M$$

$$\max Q = \sum_s [a_s - e_s] = \sum_s \left[\frac{m_s}{m} - \left(\frac{d_s}{2m} \right)^2 \right]$$

m_s = number of edges in module s
d_S = sum of degrees k_i of vertices in s

- $m_s = \sum_r X_{rs}$ and $d_S = \sum_i k_i Y_{is}$
- $\sum_s Y_{is} = 1 \quad \forall i = 1, 2, \ldots n$
- $X_{rs} \leq Y_{is} \quad \forall r = \{v_i, v_j\} \in E$
  $X_{rs} \leq Y_{js} \quad \forall r = \{v_i, v_j\} \in E$
- $u_s \leq u_{s-1}$
- symmetry-breaking constraints

$\Downarrow$

Mixed-Integer Quadratic Program

with a convex continuous relaxation

Can be solved directly by a MIQP solver

Column generation for modularity maximization (MIQP)

Idea: the columns correspond to all subsets of V (all nonempty modules)

implicitly take into account all possible communities

Master problem:

$$\begin{cases} \max \sum_{t \in T} c_t z_t - C & \\ \sum_{t \in T} a_{it} z_t = 1 & \forall i = 1, \ldots, n \\ z_t \in \{0, 1\} & \forall t \in T \end{cases}$$

$c_t = \sum_i \sum_{j>i} \frac{1}{m} \left(a_{ij} - \frac{k_i k_j}{2m} \right) a_{it} a_{jt} \qquad t = 1 \ldots 2^n - 1$

- sum of modularities of all selected modules

- each entity must belong to only one module

- modules must be selected entirely or not at all

Column generation for modularity maximization (MIQP)

Idea: the columns correspond to all subsets of V (all nonempty modules)

implicitly take into account all possible communities

Master problem:

$$\begin{cases} \max \sum_{t \in T} c_t z_t - C \\ \sum_{t \in T} a_{it} z_t = 1 \qquad \forall i = 1, \ldots, n \\ z_t \in \{0,1\} \qquad \forall t \in T \end{cases}$$

Auxiliary problem:

$$\begin{cases} \max \quad \sum_r \frac{x_r}{m} - \left(\frac{d}{2m}\right)^2 - \sum_i \lambda_i y_i \\ s.t. \\ d = \sum_i k_i y_i \\ x_r \leq y_i \quad \forall r = \{v_i, v_j\} \in E \\ x_r \leq y_j \quad \forall r = \{v_i, v_j\} \in E \end{cases}$$

$$\begin{cases} x_r = 1 & \text{if edge } r \text{ belongs to the module} \\ & \text{which maximizes the obj.funct.} \\ = 0 & \text{otherwise} \end{cases}$$

$$\begin{cases} y_i = 1 & \text{if vertex } i \text{ belongs to the module} \\ & \text{which maximizes the obj.funct.} \\ = 0 & \text{otherwise} \end{cases}$$

Column generation for modularity maximization (MIQP)

Idea: the columns correspond to all subsets of V (all nonempty modules)

implicitly take into account all possible communities

Master problem:

$$\begin{cases} \max \sum_{t \in T} c_t z_t - C \\ \sum_{t \in T} a_{it} z_t = 1 \qquad \forall i = 1, \ldots, n \\ z_t \in \{0,1\} \qquad \forall t \in T \end{cases}$$

Auxiliary problem:

$$\begin{cases} \max \quad \sum_r \frac{x_r}{m} - \left(\frac{d}{2m}\right)^2 - \sum_i \lambda_i y_i \\ s.t. \\ d = \sum_i k_i y_i \\ x_r \leq y_i \quad \forall r = \{v_i, v_j\} \in E \\ x_r \leq y_j \quad \forall r = \{v_i, v_j\} \in E \end{cases}$$

approach similar to that of Xu *et al.*

$\Downarrow$

mixed 0-1 quadratic program

- $n + m$ binary variables + 1 continuous variable
- $2m + 1$ linear constraints
- a single nonlinear term which is concave, in the obj.funct.

Column generation for modularity maximization (MIQP)

Idea: the columns correspond to all subsets of V (all nonempty modules)

implicitly take into account all possible communities

Master problem:

$$\begin{cases} \max \sum_{t \in T} c_t z_t - C \\ \sum_{t \in T} a_{it} z_t = 1 \qquad \forall i = 1, \ldots, n \\ z_t \in \{0,1\} \qquad \forall t \in T \end{cases}$$

Auxiliary problem:

$$\begin{cases} \max \quad \sum_r \frac{x_r}{m} - \left(\frac{d}{2m}\right)^2 - \sum_i \lambda_i y_i \\ s.t. \\ d = \sum_i k_i y_i \\ x_r \leq y_i \quad \forall r = \{v_i, v_j\} \in E \\ x_r \leq y_j \quad \forall r = \{v_i, v_j\} \in E \end{cases}$$

approach similar to that of Xu *et al.*

$\Downarrow$

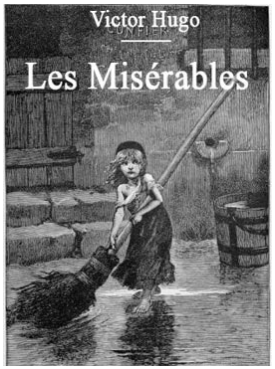
mixed 0-1 quadratic program

Can be solved using:

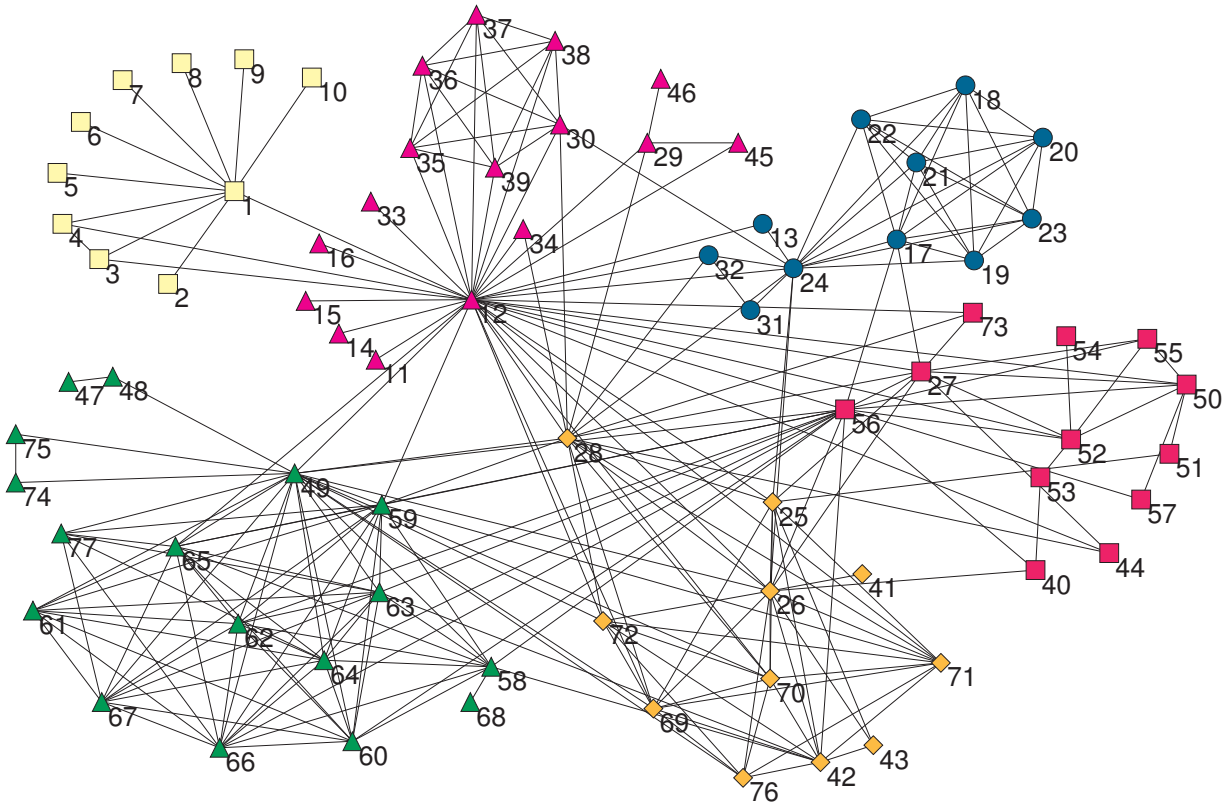
- a heuristic, as long as a column with positive reduced cost can be found
- an exact method, when this is no more the case

Example: a social network

Victor Hugo's *Les Misérables* network:



- ★ 77 vertices associated to characters which interact
- ★ 257 edges associated with pairs of characters appearing jointly in at least one chapter



Bibliography


M. E. J. Newman,
Networks: an introduction,
Oxford University Press, 2010.


S. Fortunato,
Community detection in graphs,
Physics Reports, vol.486(3), pp. 75-174, 2010.


D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, L. Liberti, S. Perron,
Column generation algorithms for exact modularity maximization in networks,
Physical Review E, vol.82(4), 046112, 2010.