Community structure: Clusters and partitions

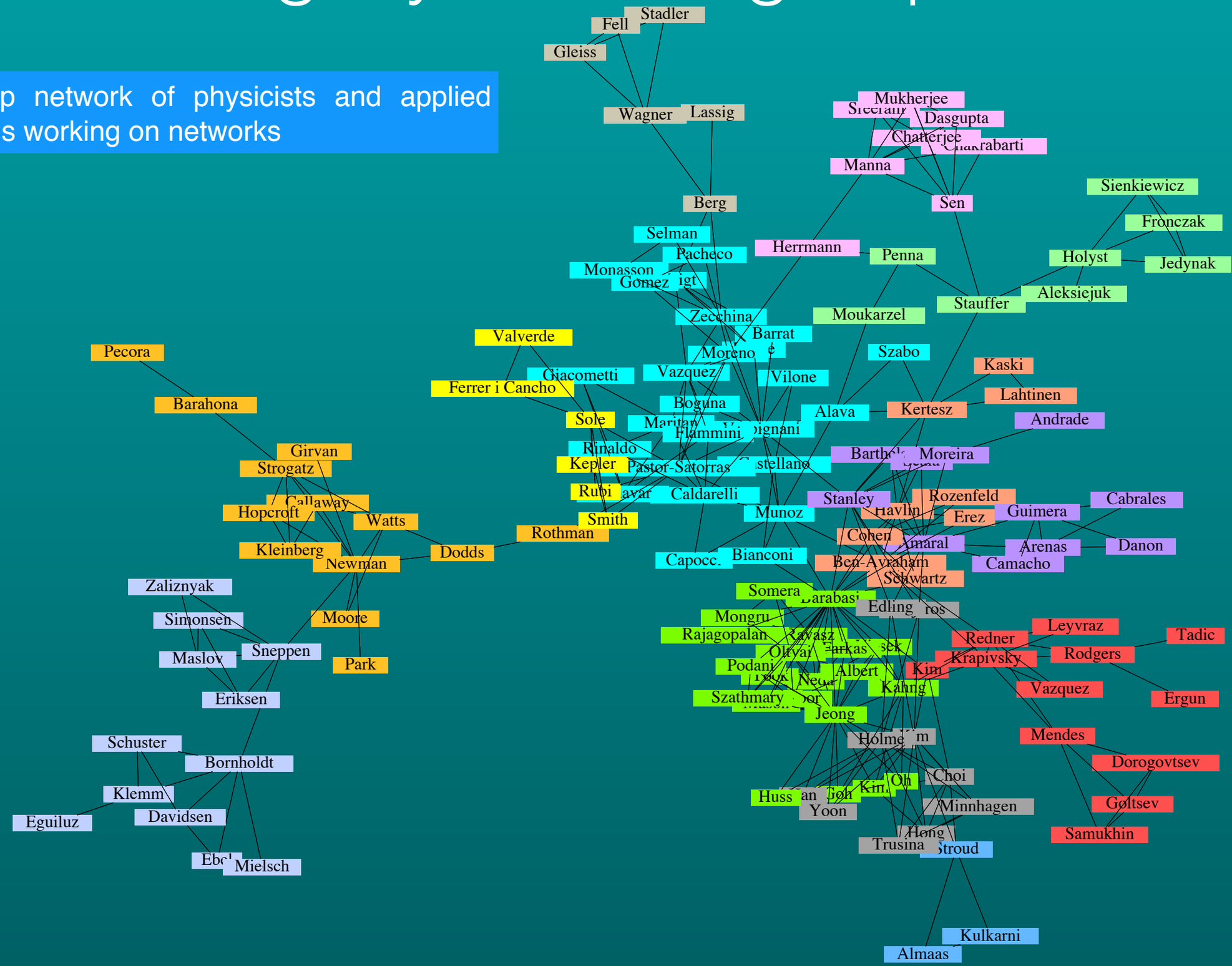
Seminars in Social Networks and Markets

Tightly-knit subgroups

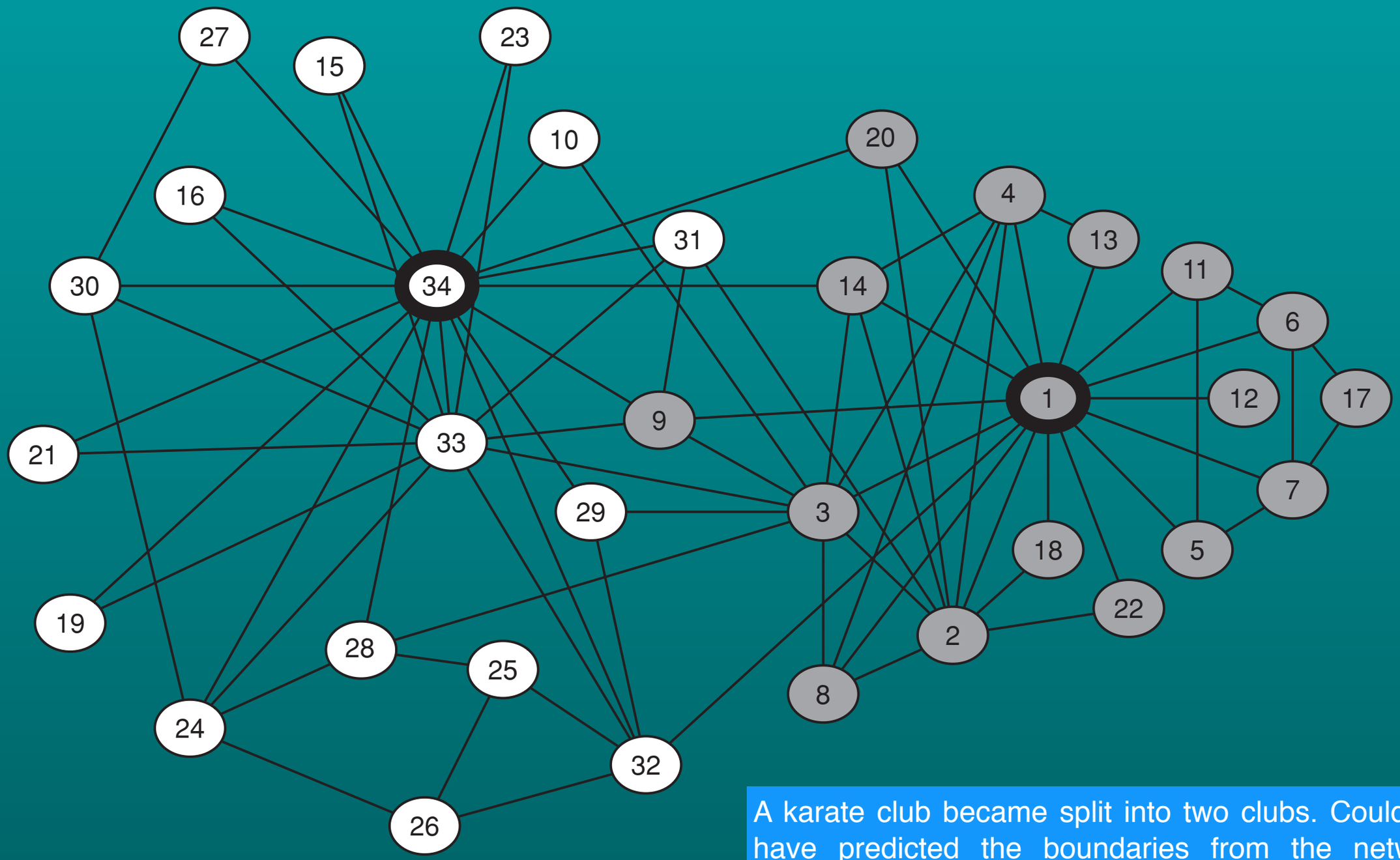
Network of interactions between major characters in the novel *Les Misérables* by Victor Hugo

Tightly-knit subgroups

A coauthorship network of physicists and applied mathematicians working on networks



Tightly-knit subgroups



A karate club became split into two clubs. Could we have predicted the boundaries from the network structure?
1 = club administrator, 34 = instructor

What is a “cluster”?

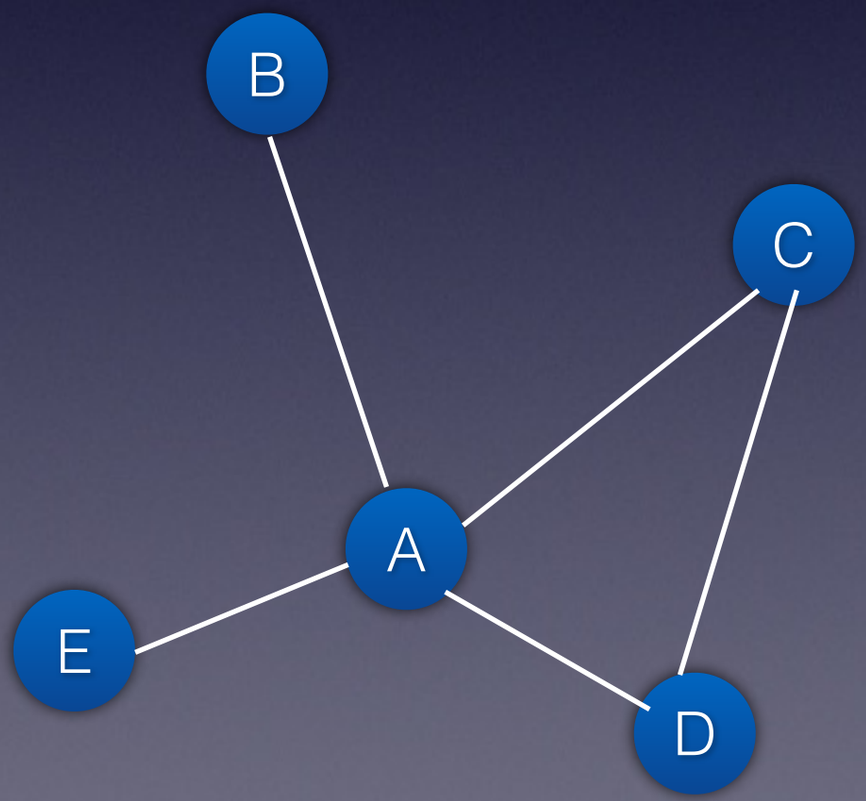
- The exact notion of “tightly-knit subgroup” may depend on the domain

- But a formal definition is crucial, if we want to identify such **regions** or **clusters**

- Let’s discuss some of the possibilities

Cliques

- A ***k*-clique** is a subset of *k* nodes forming a complete subgraph



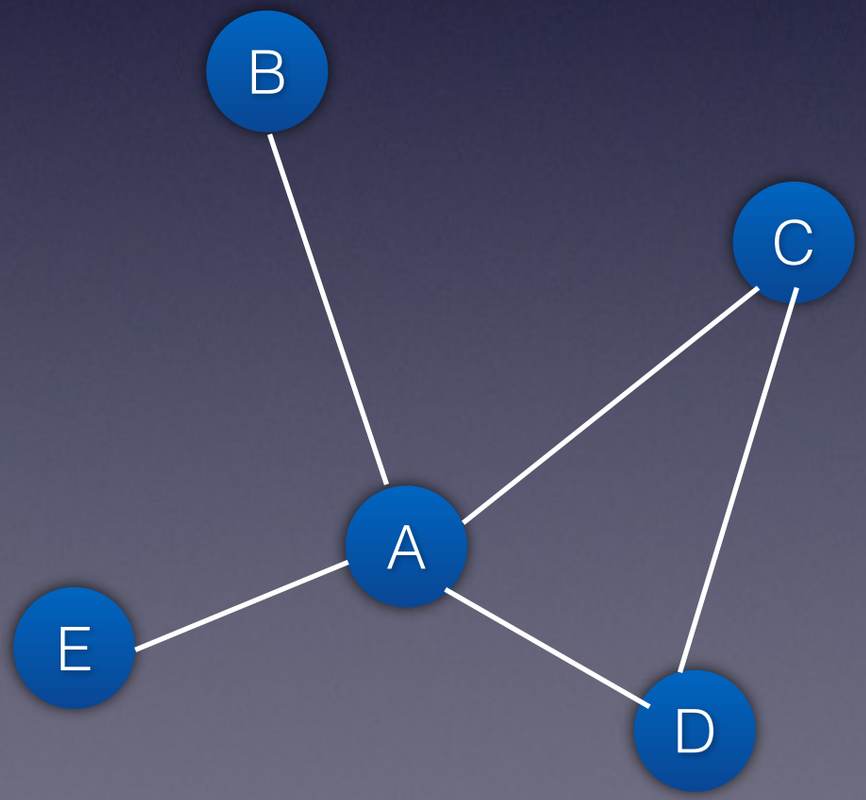
For example,

{A, C, D} is a 3-clique

{A, D} is a 2-clique

Maximal vs maximum cliques

- A *k*-clique is **maximal** if it is not contained in any *q*-clique with $q > k$
- A *k*-clique is **maximum** if there is no *q*-clique in the graph with $q > k$



{A, C, D} is a maximal and maximum 3-clique

{A, D} is a 2-clique
but it is not maximal

{A, B} is a maximal 2-clique
but it is not maximum,
because of {A, C, D}

Finding cliques

- We can look for *k*-cliques in time $O(n^k \cdot k^2)$ by enumerating all subsets of *k* nodes
- Practical only if *k* is a *very* small constant
- In fact, finding a maximum clique is NP-hard, even if we allow approximations!
- Finding a *maximal* clique is easy:
  - Start from any node (a 1-clique!) and greedily try to extend the clique one node at a time

Density of a set of nodes

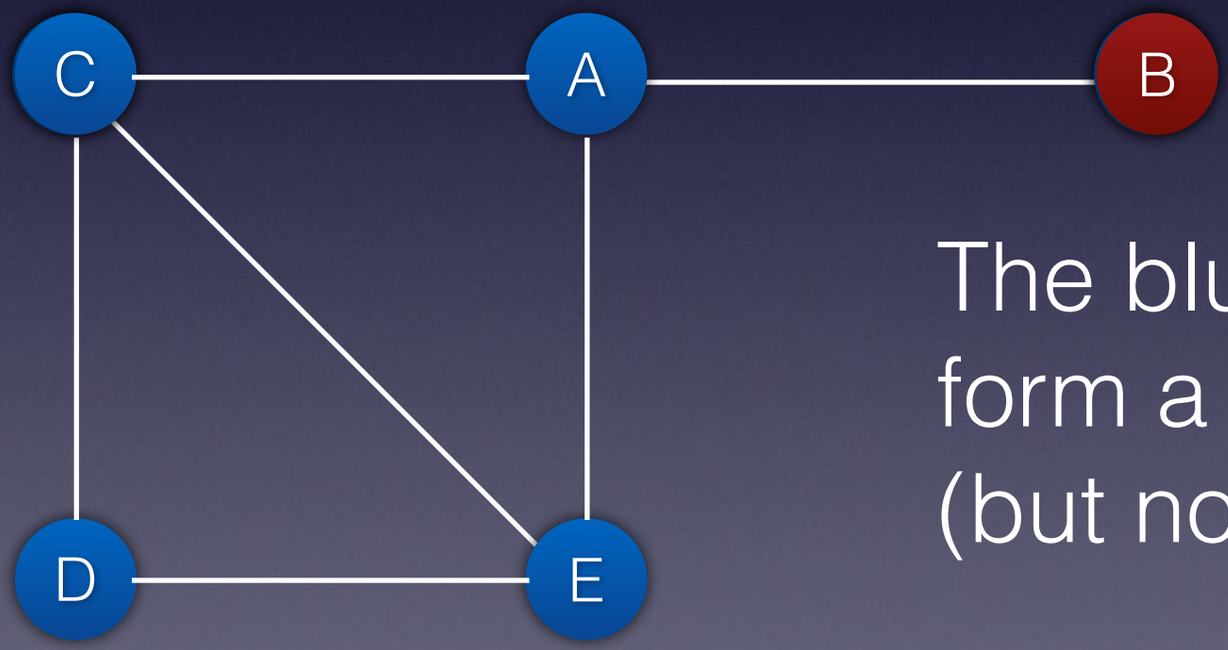
- The clique definition can be relaxed
- The **density** of a set S of at least 2 nodes is

$$\text{density}(S) = \frac{\frac{1}{2}\sum_{v \in S} \deg_S(v)}{\binom{|S|}{2}}$$

- density(S) is always between 0 and 1:
  - 0 if the subgraph induced by S consists of isolated nodes
  - 1 for a clique
  - In fact, density(S) is exactly the *probability* that two distinct random nodes of S are linked by an edge
- However, finding large, high-density subsets of nodes remains NP-hard…

k-cores

- A ***k*-core** is a set of nodes S such that each node in S has at least *k* neighbors in S
- Any *k*-clique is a (*k*-1)-core, but the reverse is not true



The blue nodes form a 2-core (but not a 3-clique)

Finding the *k*-cores

- Iterative approach:

  1. Consider the input graph *G*
  2. Remove from *G* all nodes with degree < *k*
  3. If no node is removed, stop;
     else go back to 2

- Polynomial time, even if *k* is large

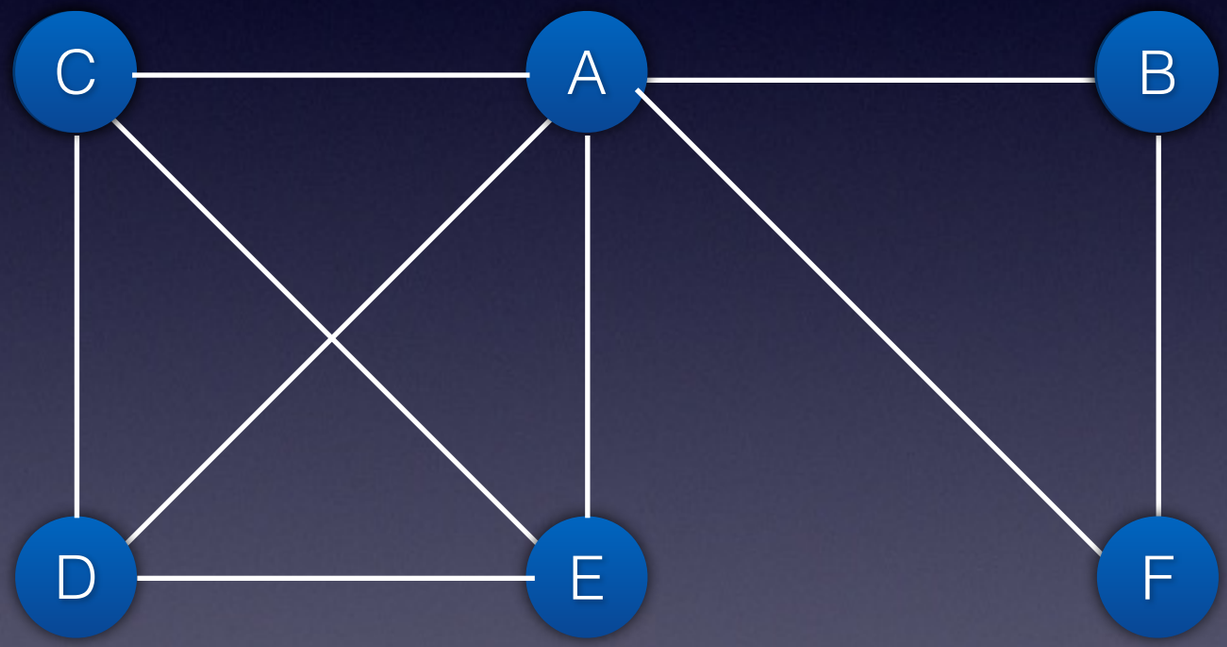
k-connected components

- We already defined the **connected components** of a graph:
  - A connected component is a maximal set of nodes *S* such that there is a path between every pair of nodes in *S*
- To generalize this to *k-connected components*, we need a notion of *independence* between paths
  - Two paths between *u* and *v* are **node-independent** if they do not share any node, except *u* and *v*
  - Two paths between *u* and *v* are **edge-independent** if they do not share any edge

k-connected components

- A *k*-**(node)-connected component** is a maximal set of nodes *S*, of at least *k* nodes, such that there are *k* node-independent paths between every pair of nodes in *S*

- So, to disconnect *S* we have to remove *at least k* nodes

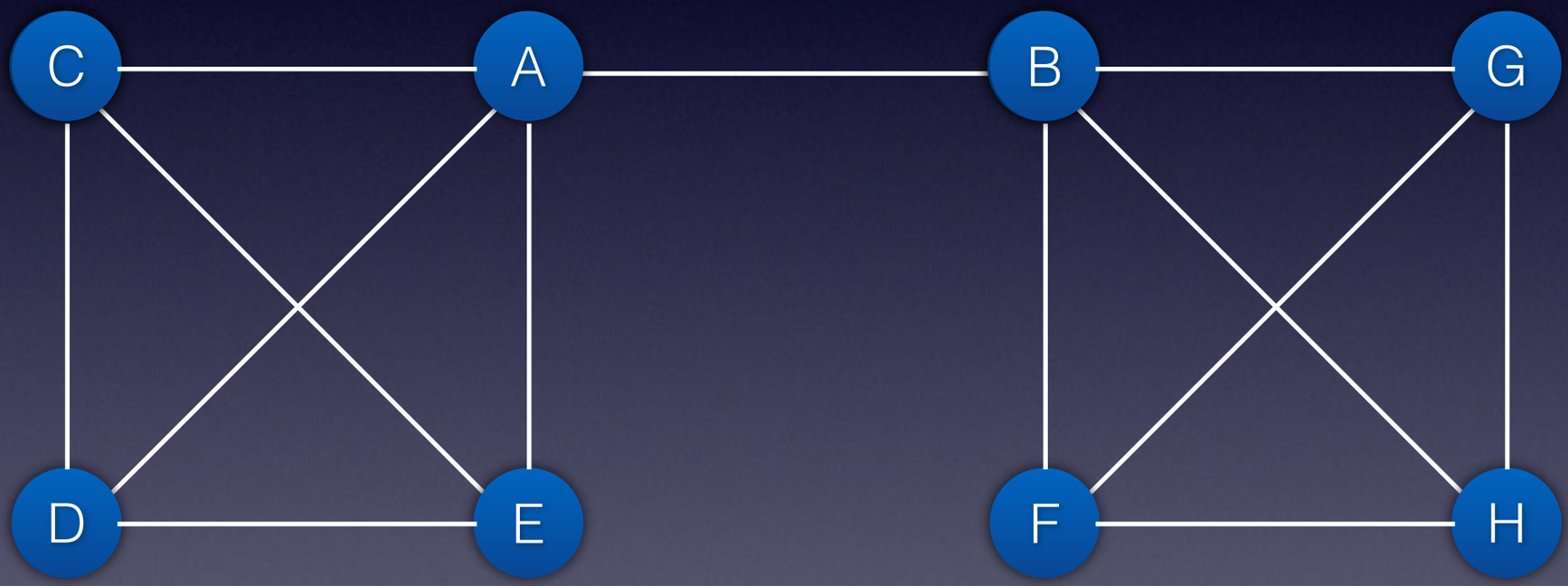


- The above graph has two 2-connected components: {A, C, D, E} and {A, B, F}

- The graph itself is connected but not 2-connected

k-edge-connected components

- A ***k*-edge-connected component** is a maximal set of nodes *S* such that there are *k* edge-independent paths between every pair of nodes in *S*
- So, to disconnect *S* we have to remove *at least k* edges



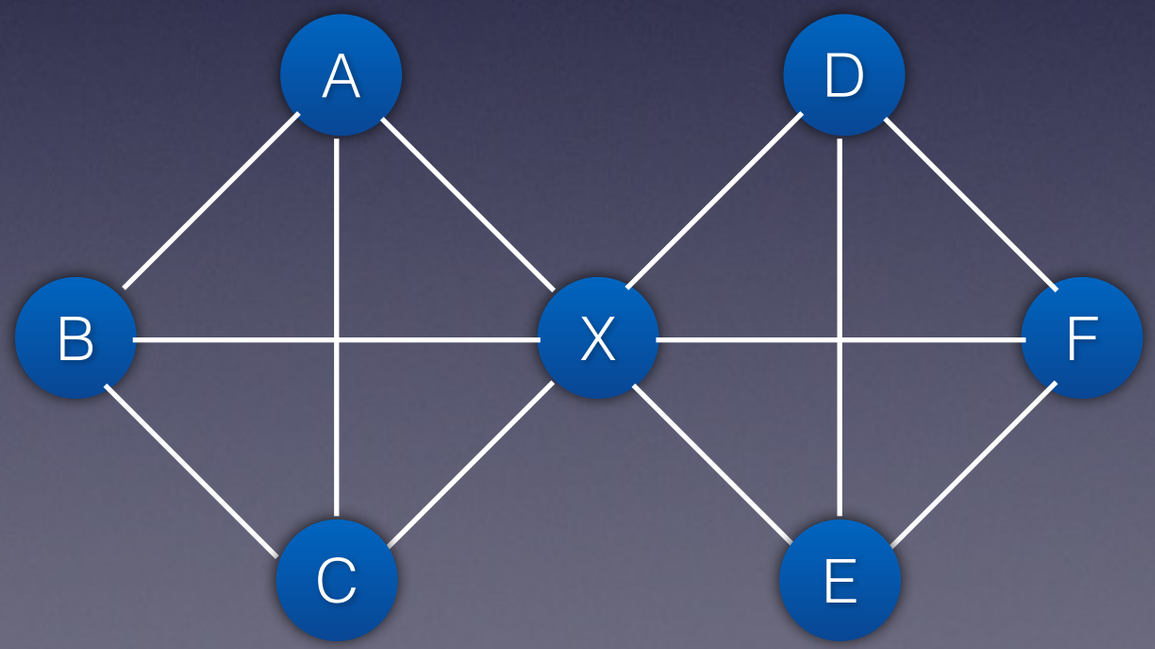
- The above graph has two 3-edge-connected components: {A, C, D, E} and {B, F, G, H}
- The graph itself is 1-edge-connected but not 2-edge-connected

Menger's Theorem

- Let *s*, *t* be distinct nodes of a graph
- Edge version: *the minimum number of edges whose removal disconnects s and t is equal to the maximum number of edge-independent paths from s to t*
- Node version: *if s and t are not adjacent, the minimum number of nodes whose removal disconnects s and t is equal to the maximum number of node-independent paths from s to t*

Connectivity values of a subgraph

- $\kappa(G) = \max \{\kappa : G \text{ is } \kappa\text{-connected}\}$
- $\lambda(G) = \max \{\lambda : G \text{ is } \lambda\text{-edge-connected}\}$
- $\kappa(G) \leq \lambda(G) \leq$ minimum degree
- $\lambda(G)$ can be arbitrarily larger than $\kappa(G)$:

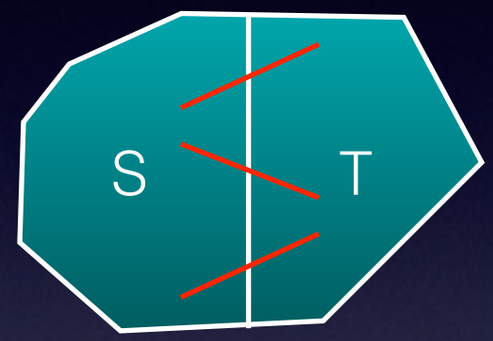


Computing node- and edge-connectivity

- The **max-flow/min-cut theorem** is a generalization of Menger's Theorem (edge version)
- To compute the edge-connectivity between two nodes *s* and *t*, we can use any **maximum-flow algorithm**:
  - assign capacity 1 to every edge
  - invoke the max-flow algorithm, with source *s* and sink *t*
  - the maximum amount of flow that can be sent is the value of the edge-connectivity between *s* and *t*
- To compute λ(G), repeat for all pairs *s*, *t* and return the minimum
  - In fact, the source can be fixed arbitrarily! Iterate only over *t*
- A similar approach works for node-connectivity as well

Graph bipartitioning

- How “well” can a network be split into two parts?
- Partition V(G) into S, T:
  - S ∪ T = V(G)
  - S ∩ T = ∅

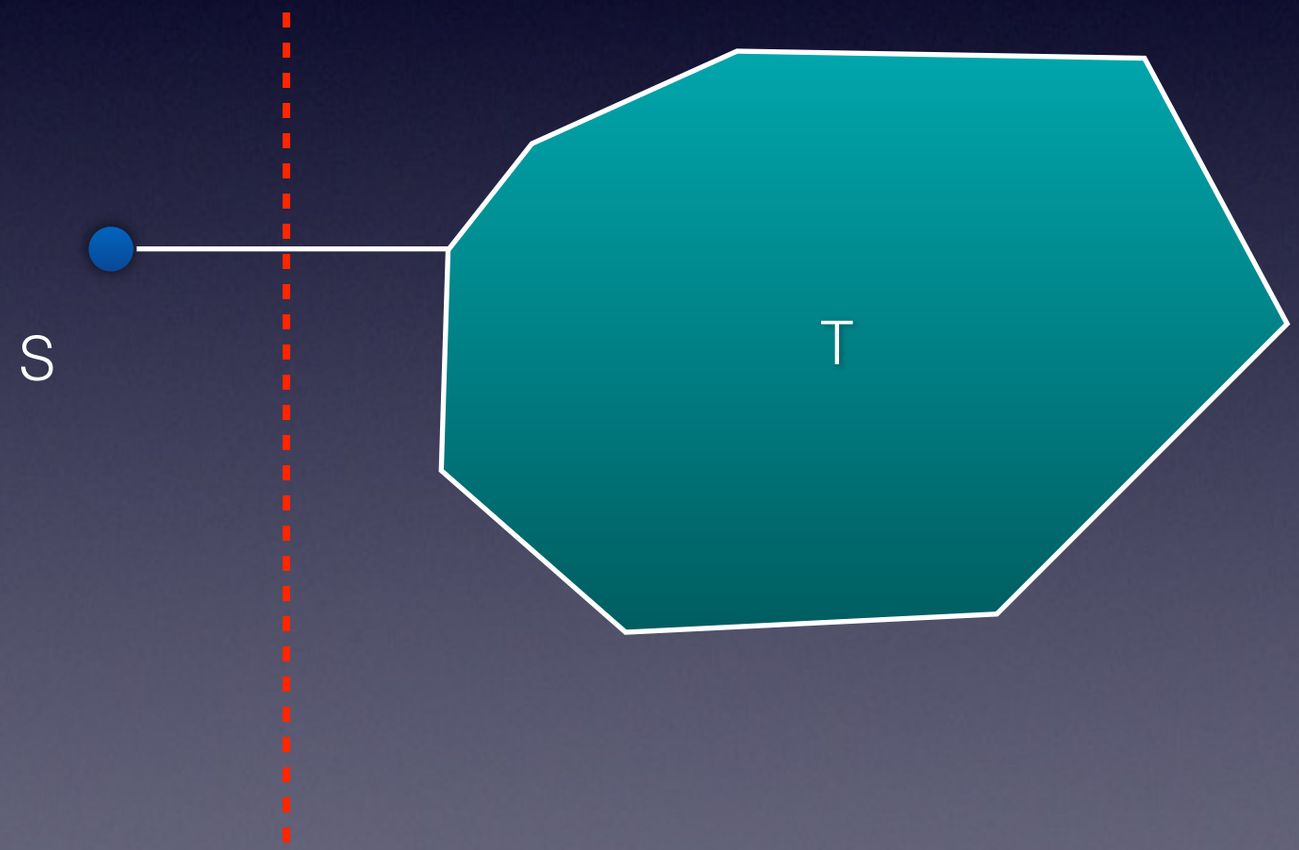


- But: need to formalize the objective function…
- One idea is to minimize the size of the cut:

$$e(S,T) = |\{\{u,v\} \in E(G) : u \in S, v \in T\}|$$

Graph bipartitioning

- Finding a globally minimum cut is easy, but it may yield trivial solutions:



Graph bipartitioning

- Different options to circumvent the problem:

1. Impose that S and T have *prescribed* size
   - $|S| = p$, $|T| = q$ for given p, q
2. Impose that S and T have *similar* size
   - $|S| \leq (1+\varepsilon)n/2$, $|T| \leq (1+\varepsilon)n/2$ for given $\varepsilon>0$
3. Incorporate the sizes in the *objective function*

Graph bipartitioning objective functions

- **Expansion**:

$$\frac{e(S,T)}{\min(|S|,|T|)}$$

- **Cut-ratio** (or **sparsity**):

$$\frac{e(S,T)}{|S| \cdot |T|}$$

- **Conductance**:

$$\frac{e(S,T)}{\min(\mathrm{vol}(S),\mathrm{vol}(T))}$$

$$(\mathrm{vol}(X) = \sum_{i \in X} \deg(i))$$

Graph bipartitioning algorithms we'll discuss

- With prescribed sizes:
  - Local search: Kernighan-Lin algorithm
  - Spectral bipartitioning
- Conductance minimization:
  - Variant of spectral bipartitioning

Homophily, assortativity and modularity

Homophily

- Social networks often exhibit **homophily**: people tend to select friends with similar characteristics

- Immutable characteristics (like ethnicity, language) influence the formation of links (**selection**)

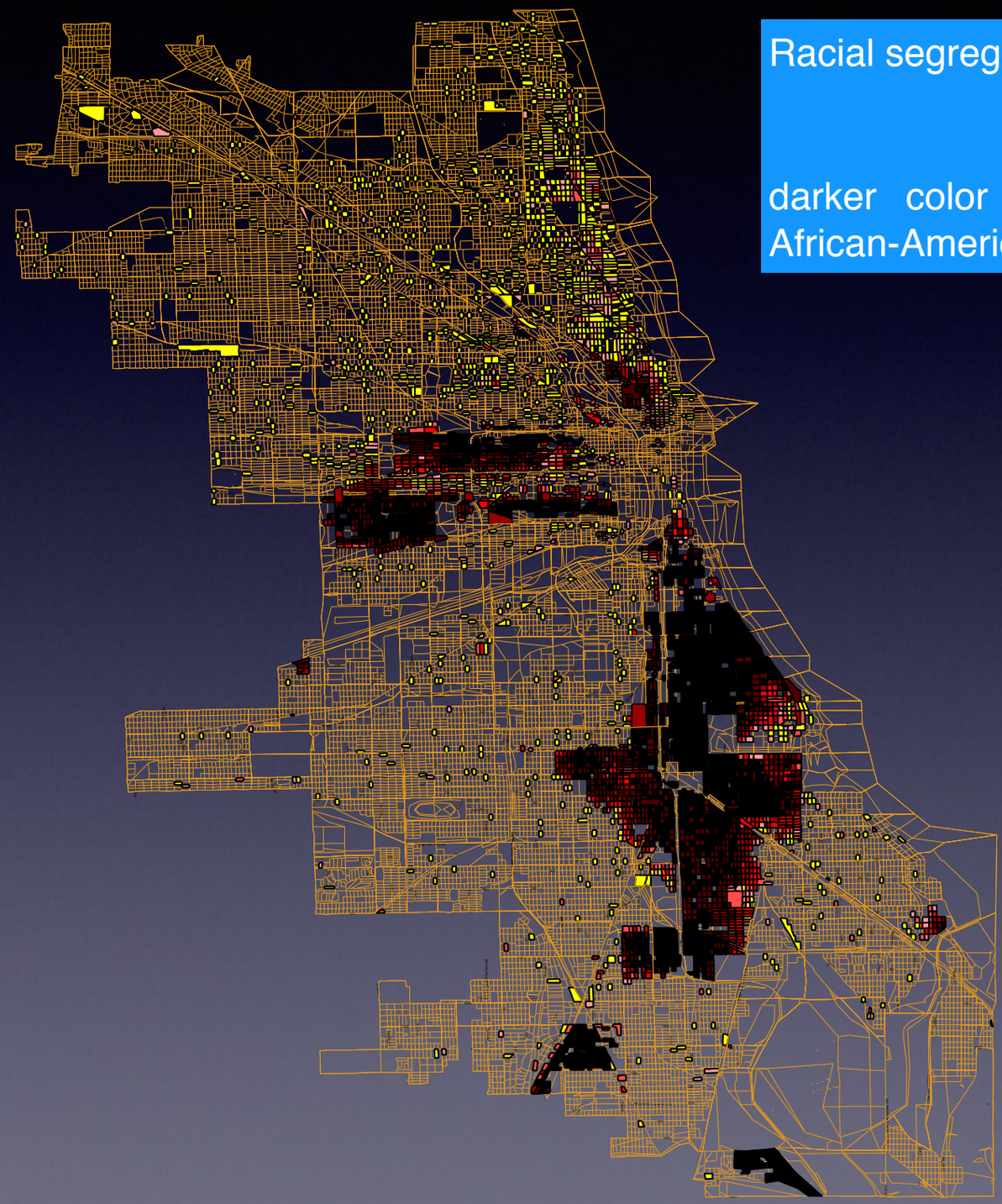
- In turn, existing links influence people's behavior and mutable characteristics (like living place) (**social influence**)

Spatial segregation

Racial segregation in 1940 Chicago

darker color = higher density of African-American people

Spatial segregation



Racial segregation in 1960 Chicago

darker color = higher density of African-American people

Assortative and disassortative mixing

- Homophily is also known as **assortative mixing**
- The reverse phenomenon is **disassortative mixing**, where people tend to form links with others who are *unlike* them
- More rare: main example is the sexual contact network

Quantifying assortativity: notation

- Let c_i be the class (category), or type, of node i
- The number of edges running between nodes of the *same* type is

$$\sum_{(i,j)\in E} \delta(c_i, c_j) = \frac{1}{2} \sum_{i,j\in V} A_{ij}\delta(c_i, c_j)$$

where $\delta(c_i, c_j) = 1$ if $c_i = c_j$ and 0 otherwise

Quantifying assortativity: notation

- We compare this with the *expected* number of edges between nodes when edges are placed at random:

$$\frac{1}{2}\sum_{i,j\in V}\frac{k_i k_j}{2m}\delta(c_i,c_j)$$

where k_i is the degree of i

The modularity score

- The normalized difference is called the **modularity** of the clustered network:

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

- Modularity is between -1 and +1
- Positive modularity = assortativity
- Negative modularity = disassortativity

General approaches to clustering

- **Divisive methods**
  - Start with 1 global cluster, and recursively subdivide it into smaller clusters
- **Agglomerative methods**
  - Start with *n* individual node clusters, and iteratively aggregate them into larger clusters
- **Other methods**
  - Local search, algebraic, …

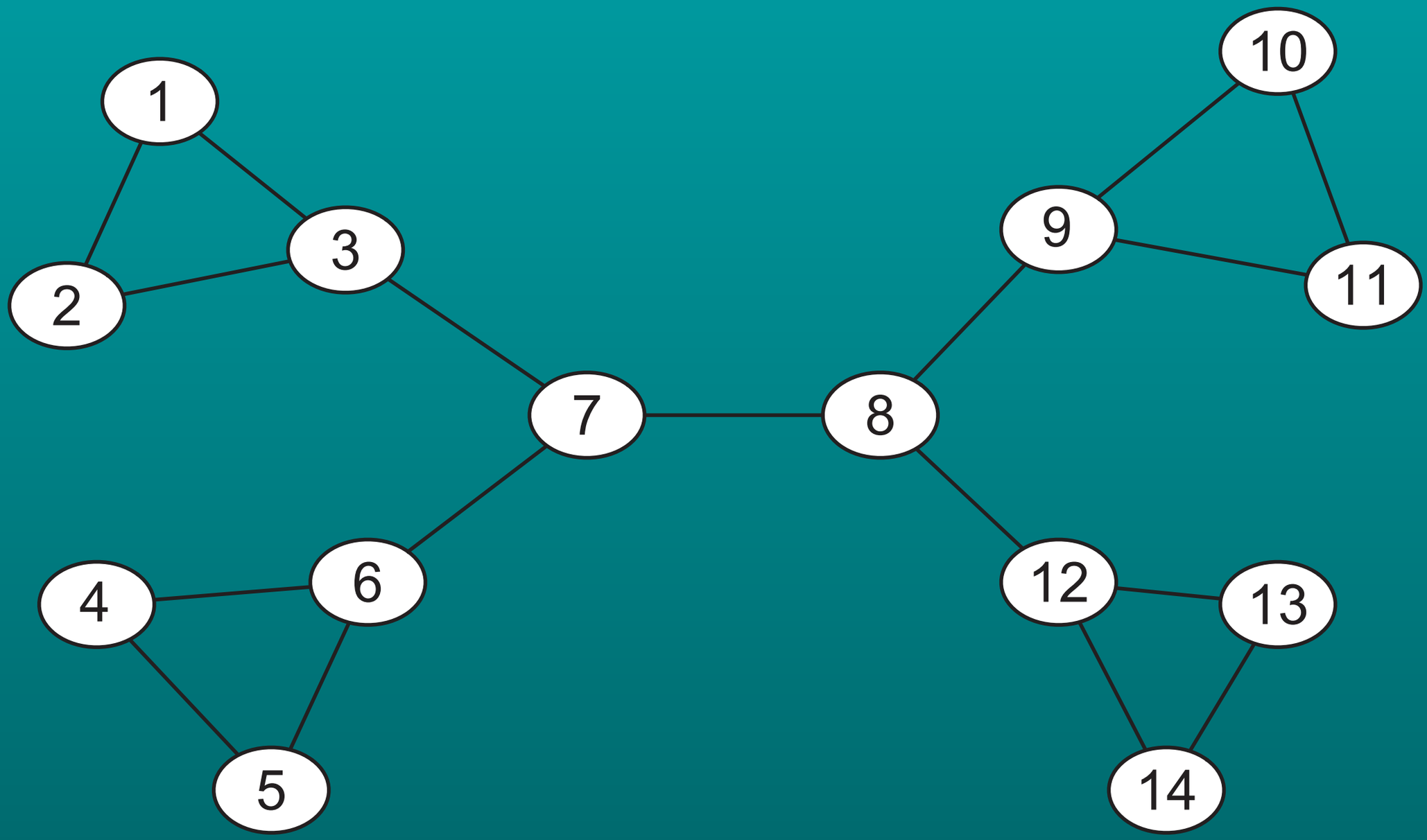
Girvan-Newman method

- Example of a divisive hierarchical method
- Produces a sequence of partitions:
  each is a **refinement** of the previous one
- At the end, return the partition with highest modularity score

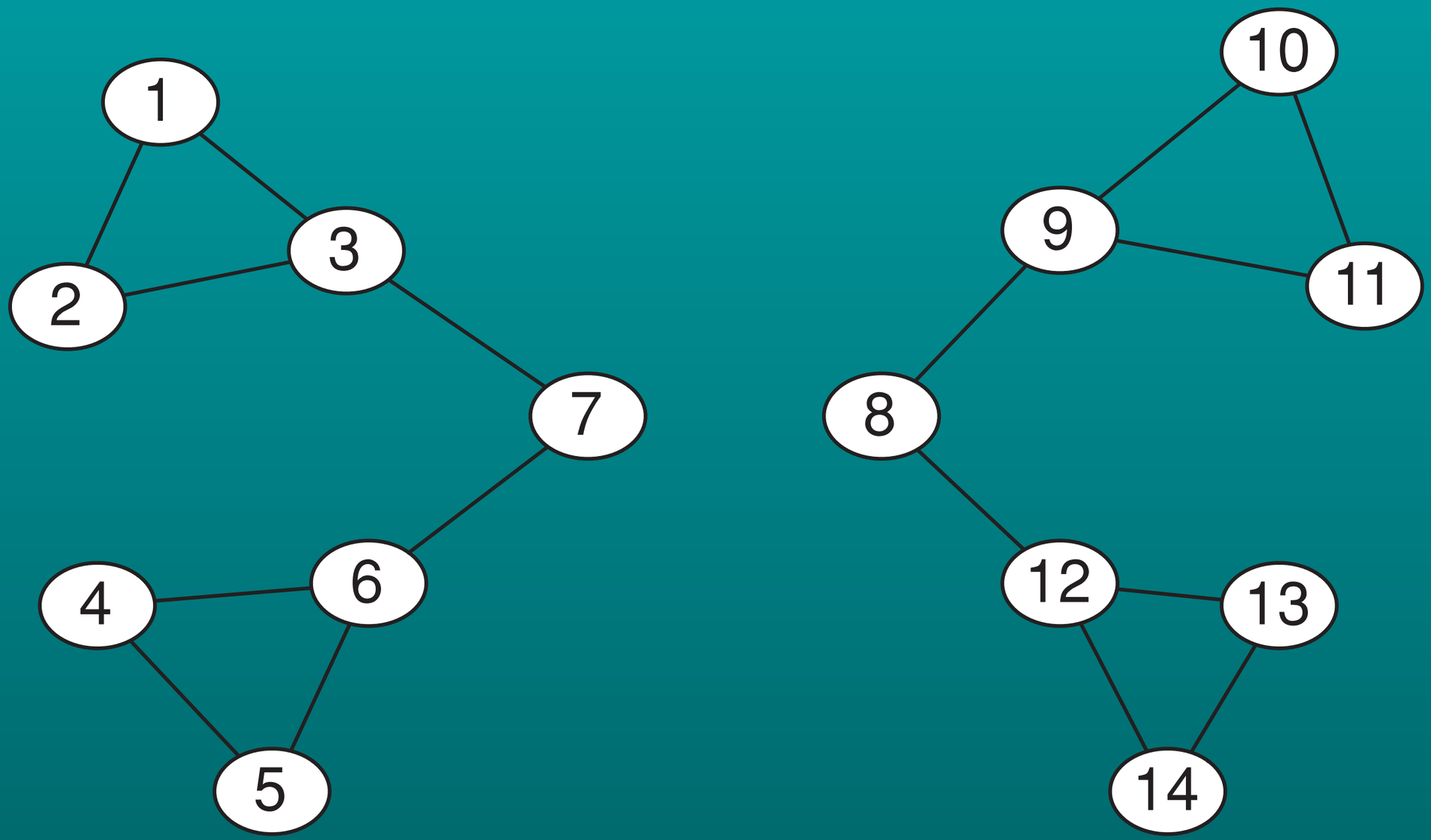
Girvan-Newman method

1. Initialization: all nodes have the *same* type
2. Find the edge(s) with **highest betweenness**
3. Remove that edge(s) from the graph
    - If the graph splits, assign a different type to each component
4. If there are edges, go back to point 2
5. Return the partition with highest modularity

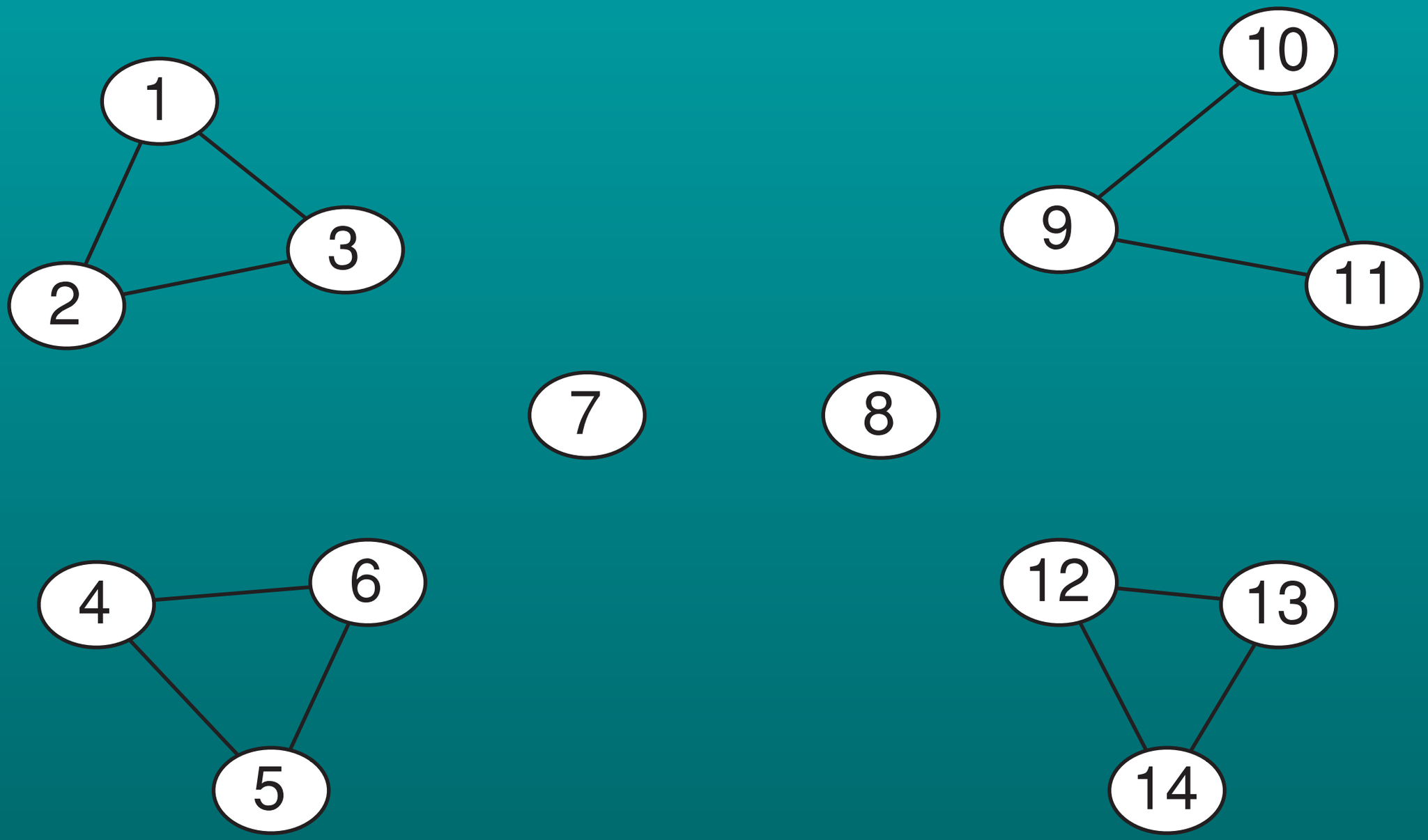
Girvan-Newman example



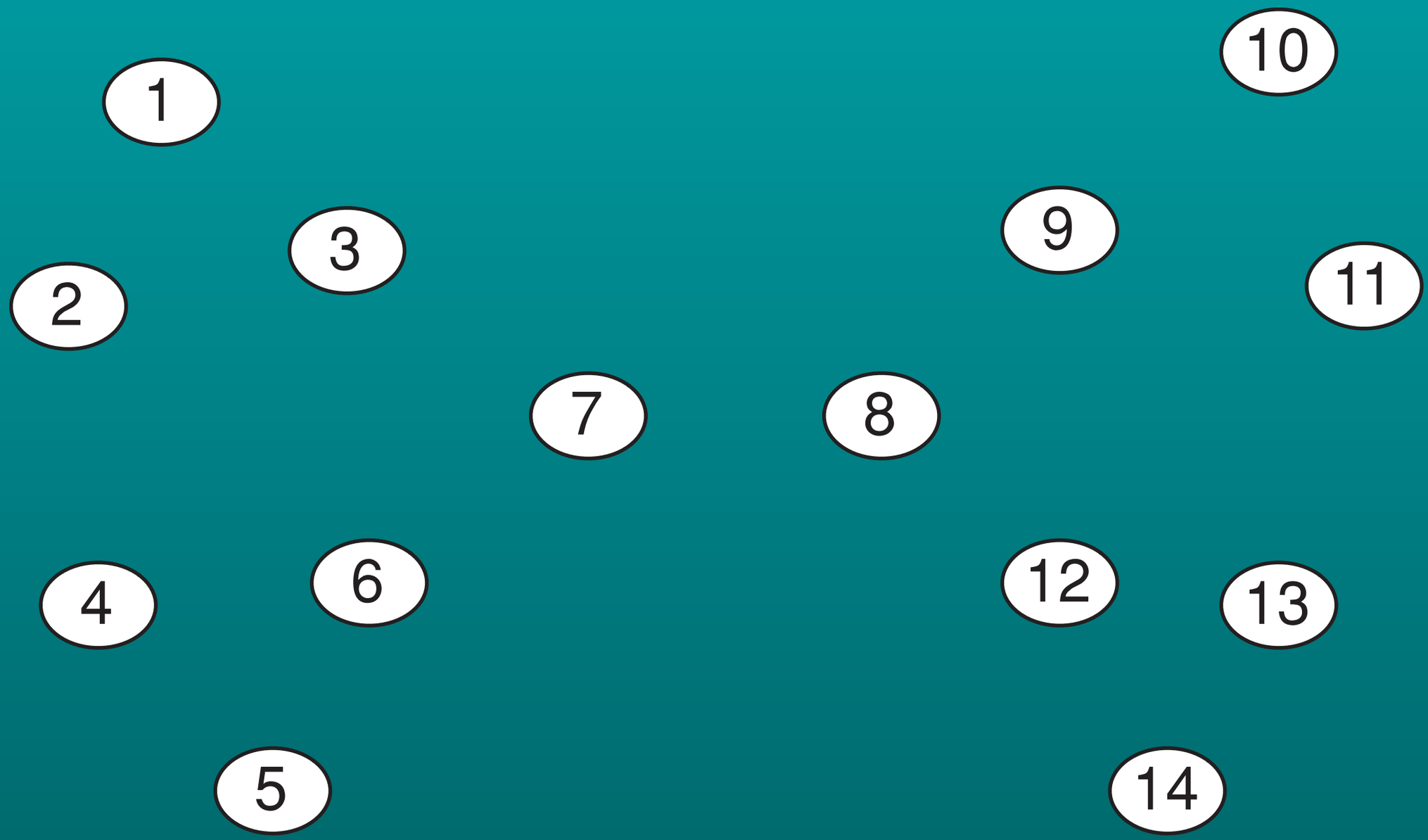
Girvan-Newman example



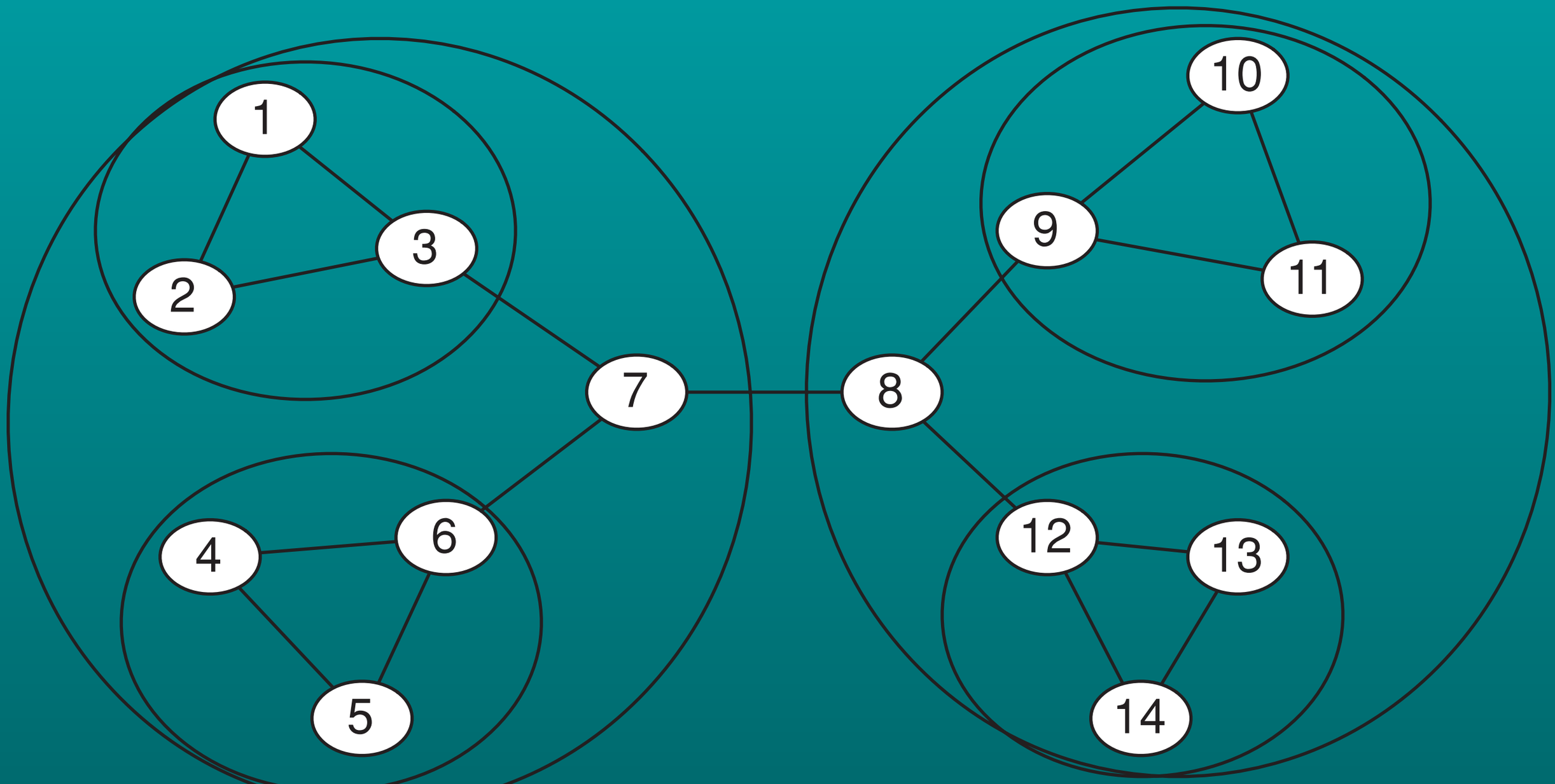
Girvan-Newman example



Girvan-Newman example



Girvan-Newman example



Girvan-Newman in practice

- Works well for moderate size networks (up to a few thousand nodes)
- Recomputing the betweenness values at every step is the computational bottleneck
- Option: use *approximate* betweenness (Riondato & Kornaropoulos 2014)

Newman's greedy method

- Example of an agglomerative hierarchical method

- Produces a sequence of partitions:
  each is a **coarsening** of the previous one

- At the end, return the partition with highest modularity score

Newman's greedy method

1. Initialization: each node has a *distinct* type
2. Join the pair of communities that results in the **larger increase in modularity** (may be negative!)
3. Merge their types
4. If there is more than one type, go back to 2
5. Return the partition with highest modularity

Spectral modularity maximization

1. Consider the **modularity matrix**, B:

$$B_{ij} = A_{ij} - \frac{\deg(i)\deg(j)}{2m}$$

2. Compute the eigenvector x of B associated to the **largest** (= most positive) eigenvalue

3. Each node *i* goes in community 1 if $x_i \geq 0$, and in community 2 if $x_i < 0$

- Good quality solutions, but limited to 2 clusters