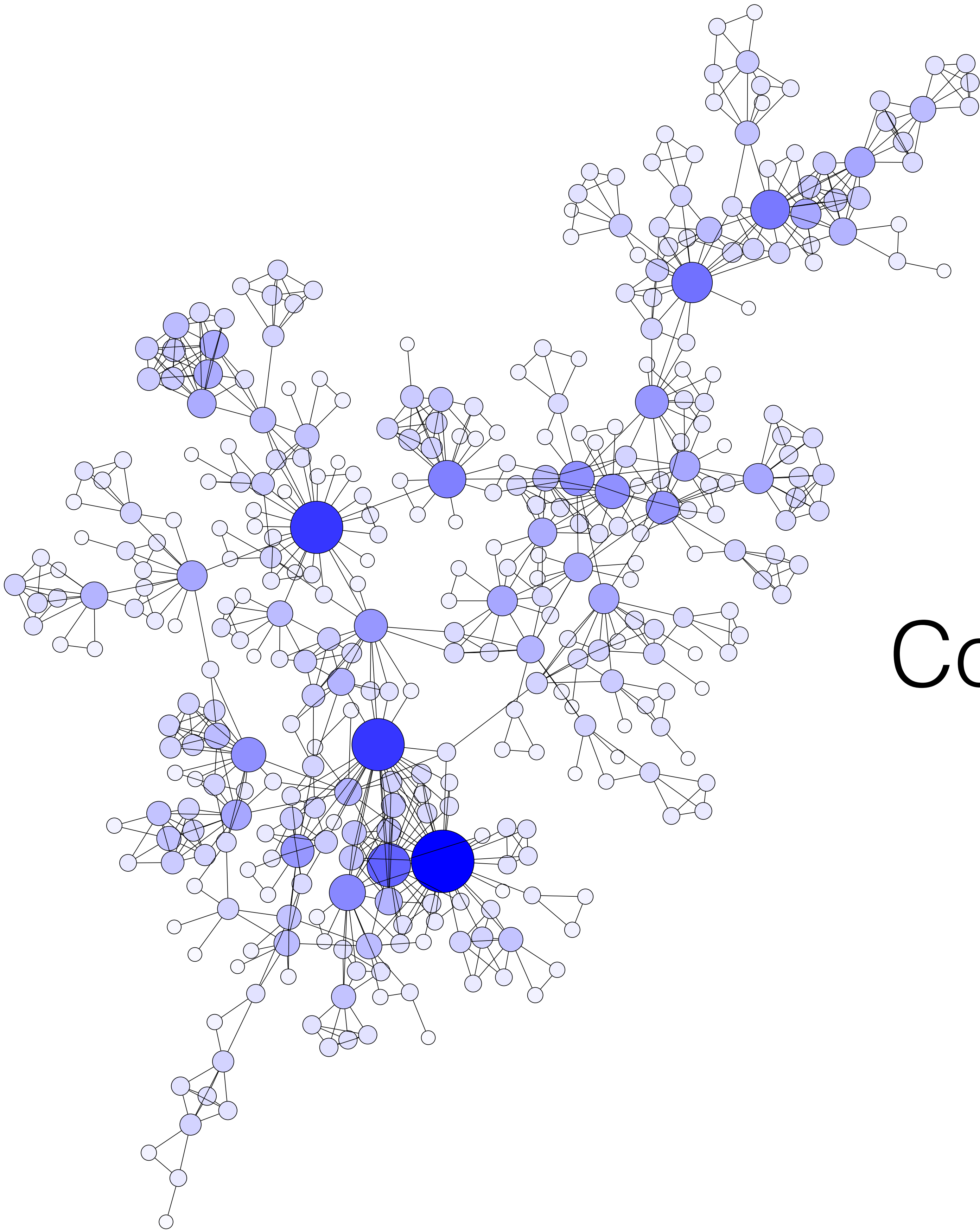


Networks and Community Structure

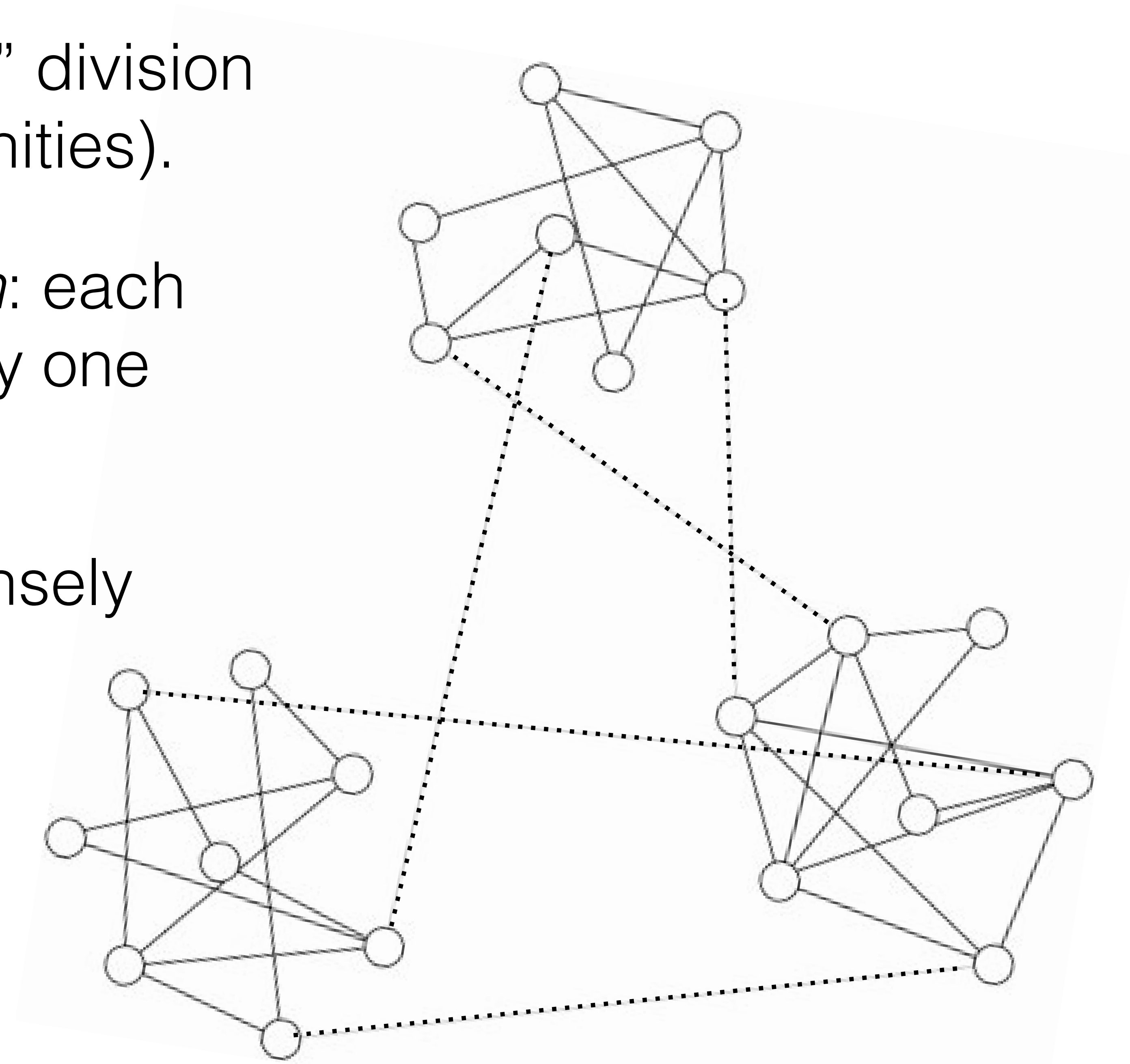
Community Structure

Community structure is a “natural” division of a network into groups (communities).

Community structure is a *partition*: each node is a member of one and only one group

Within a group, the nodes are densely connected, with only sparse connections between groups

This community structure is sometimes known to the people in the network. Sometimes not.



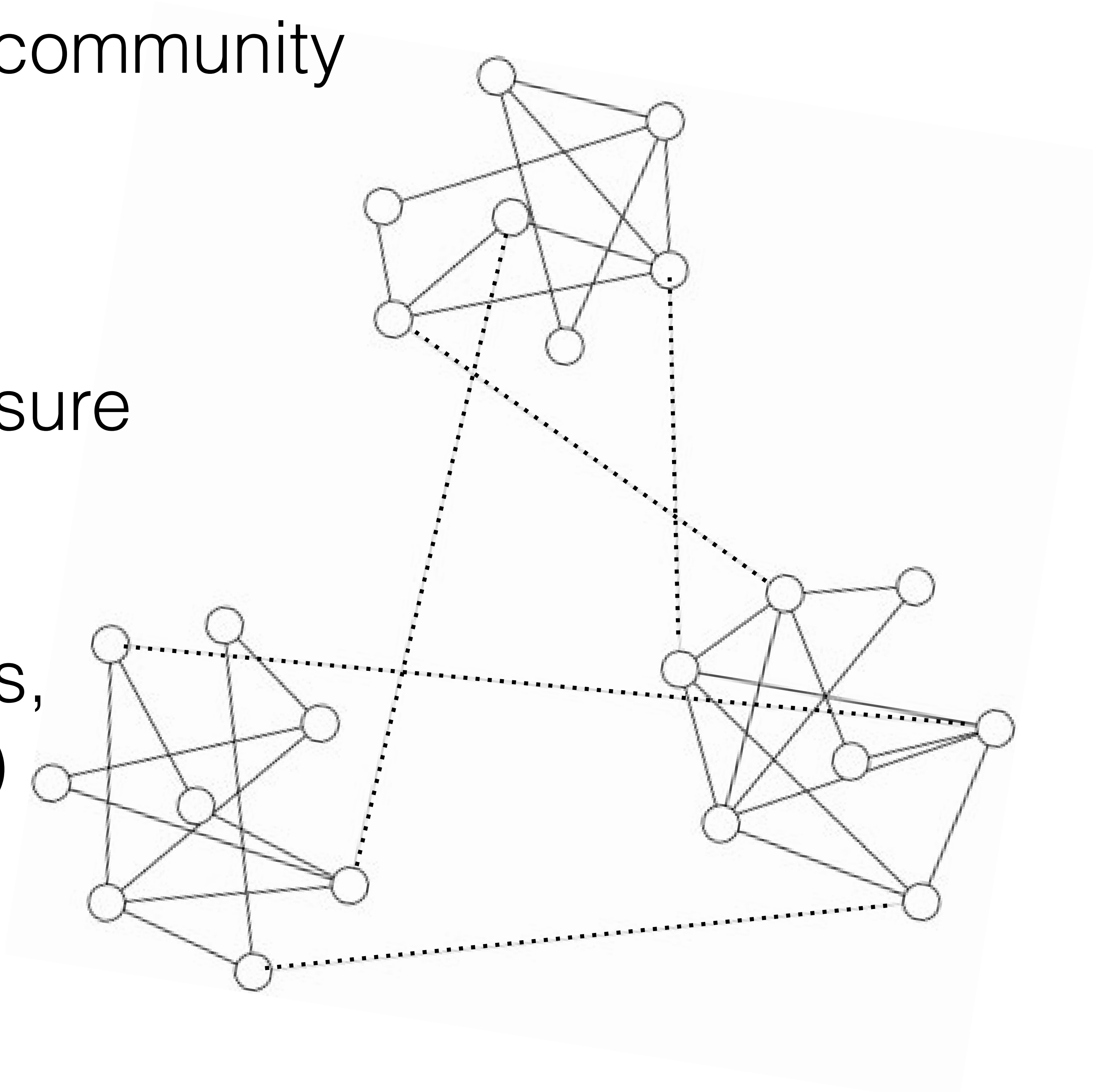
What the heck?

NB: community structure is (confusingly) sometimes referred to as “clustering”

Community Structure

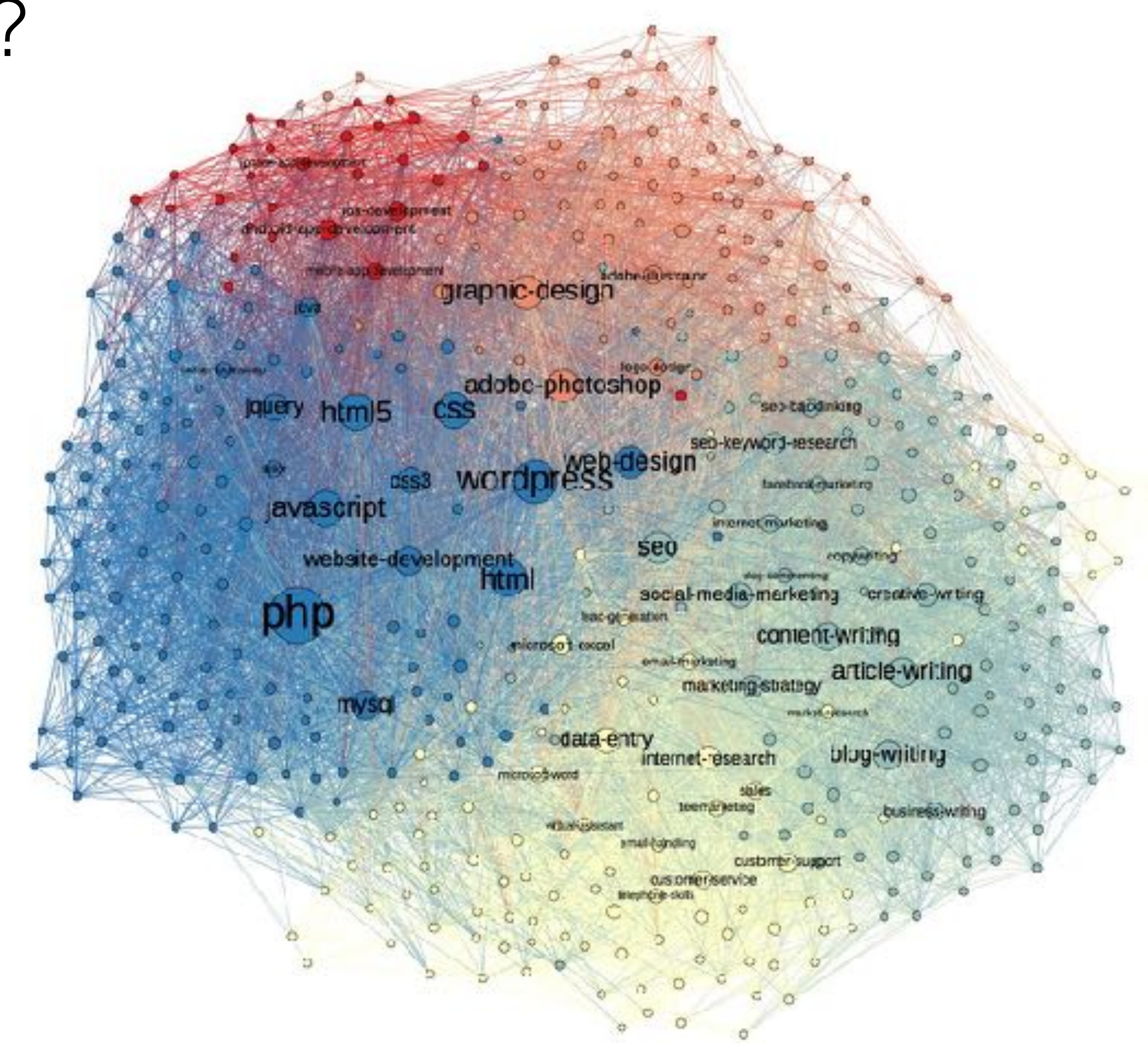
What are some sources of community structure?

- family groups
- homophily and triadic closure
- geography
- organizations (e.g.schools, clubs and teams, firms…)



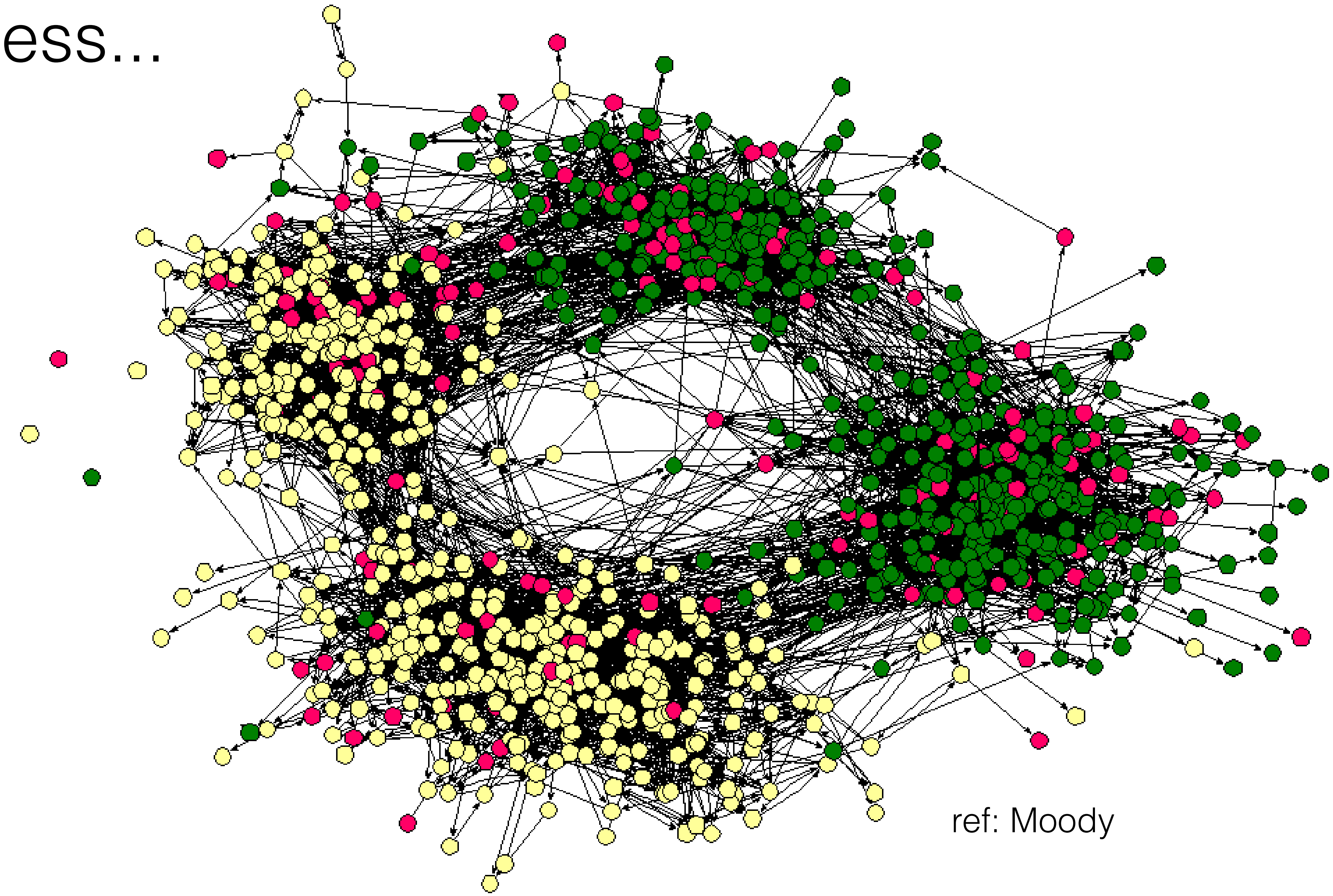
Community Structure

But why would we care about community structure in a network?



Community Structure

Sometimes it reveals a deeper underlying social process...

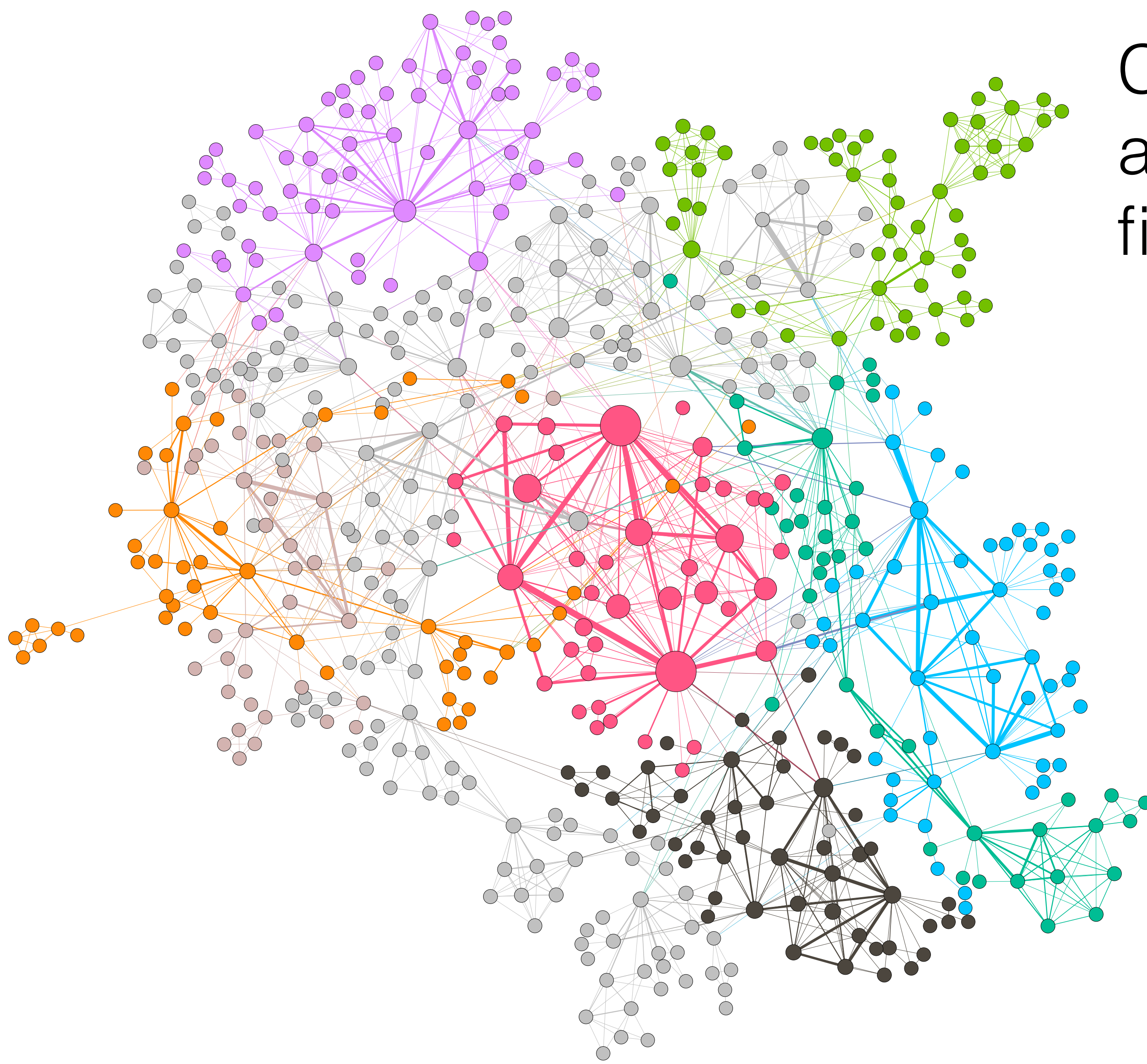


ref: Moody

…for example, self-segregation

Community Structure

Another example: a coauthorship network for a small scientific field



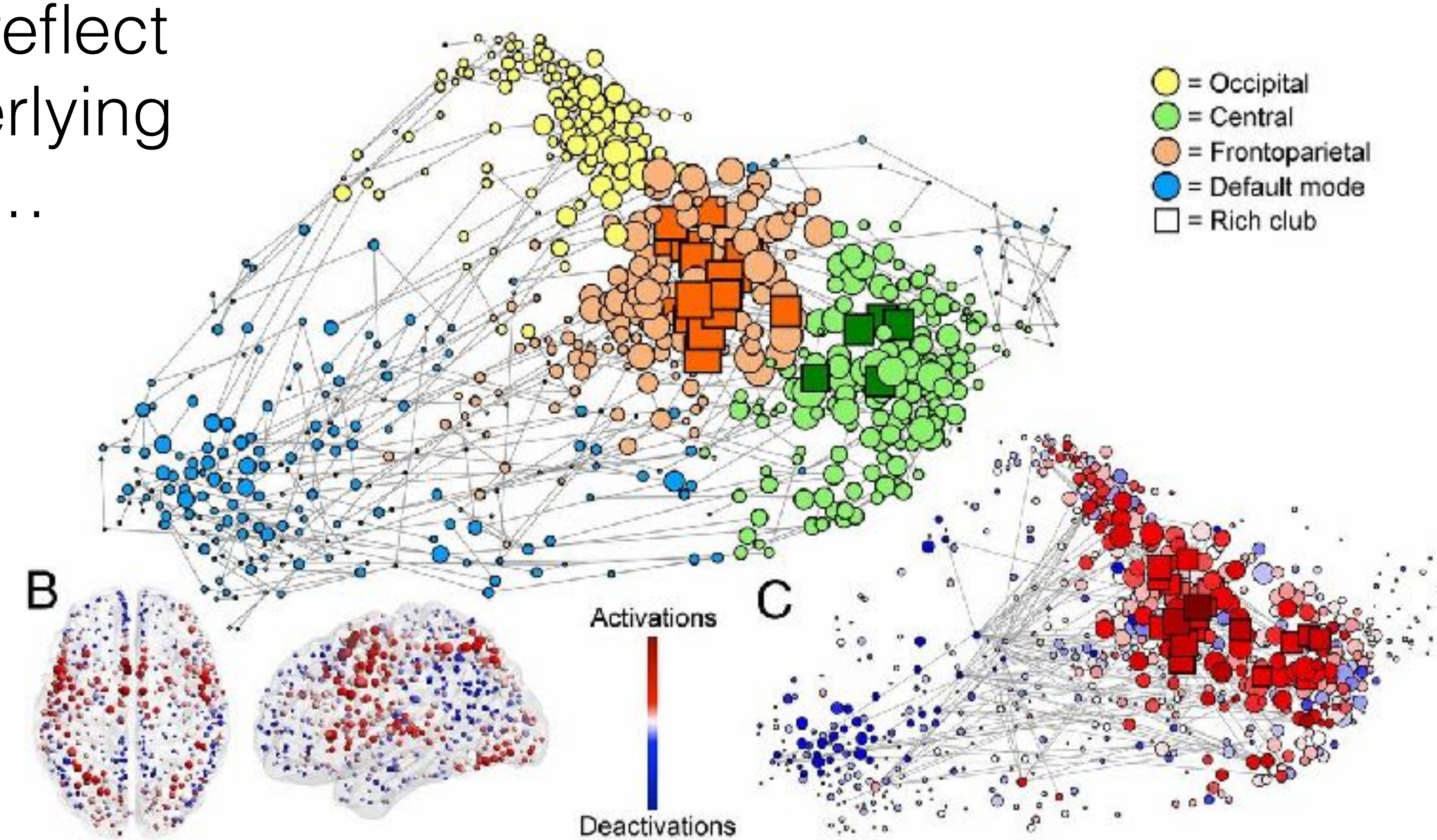
Communities are built around the founders of the field

Suggests that scholars are more likely to work with other people who studied with their PhD advisor

Why might that be?

Community Structure

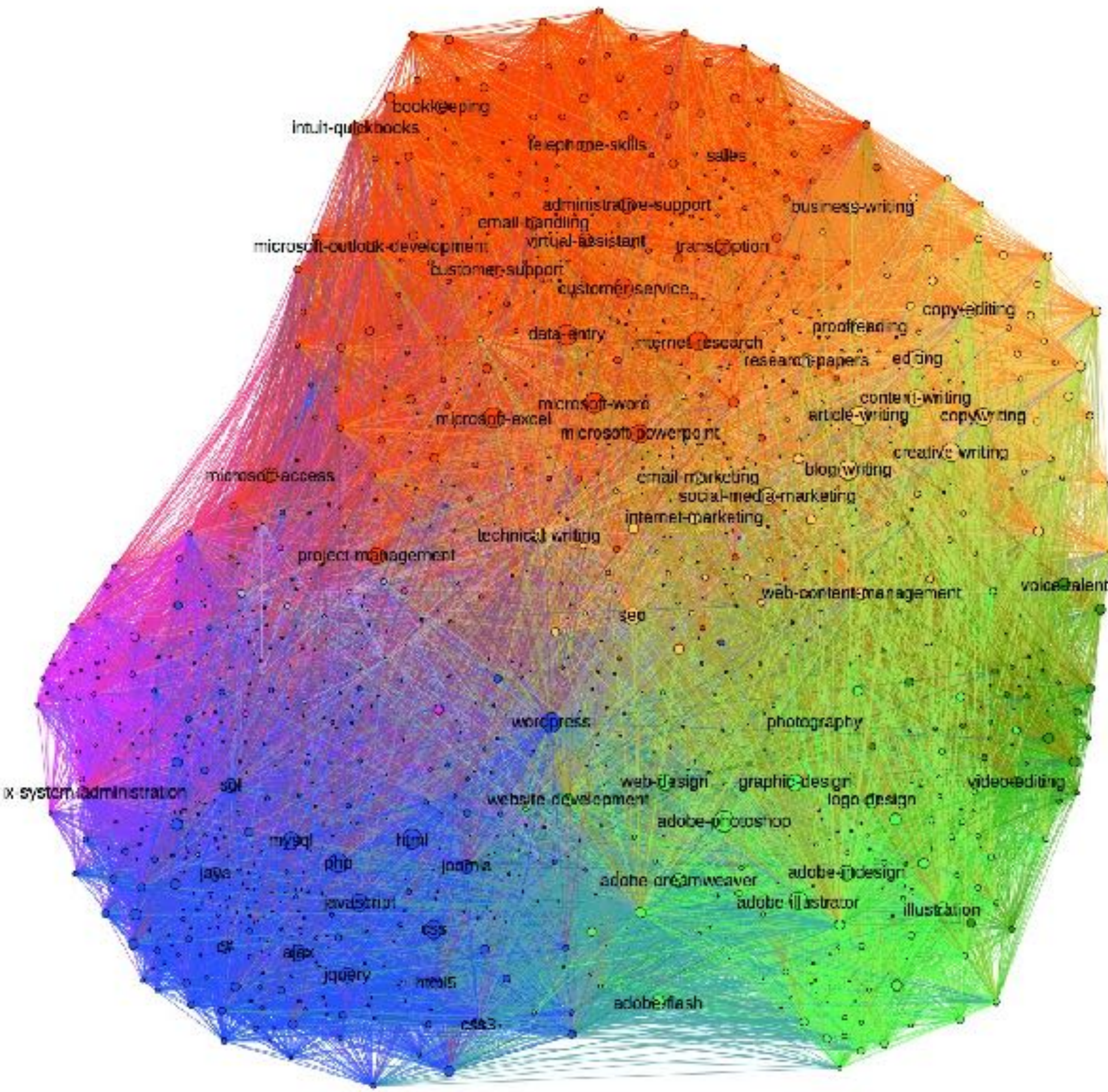
Alternatively,
it might reflect
the underlying
function…



…for example, in neural networks

ref: Crossley et al. (PNAS 2013)

Community Structure



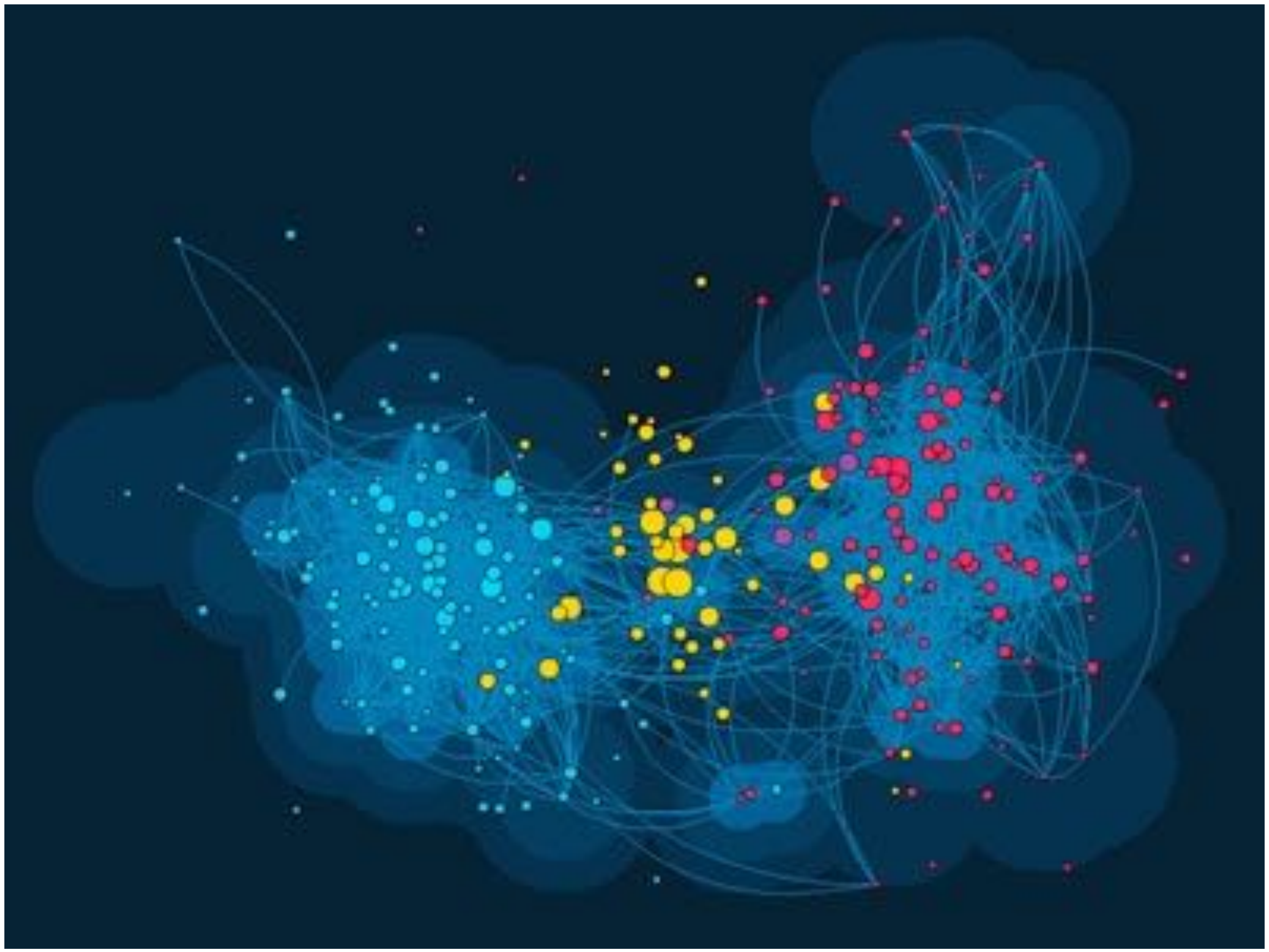
Skill network:

- nodes = skills
- A↔B if a worker has both skills

Communities represent groups of skills that frequently co-occur in the job market.

May reflect skill synergies (skills frequently used together), or may be related in how they are acquired

Community Structure

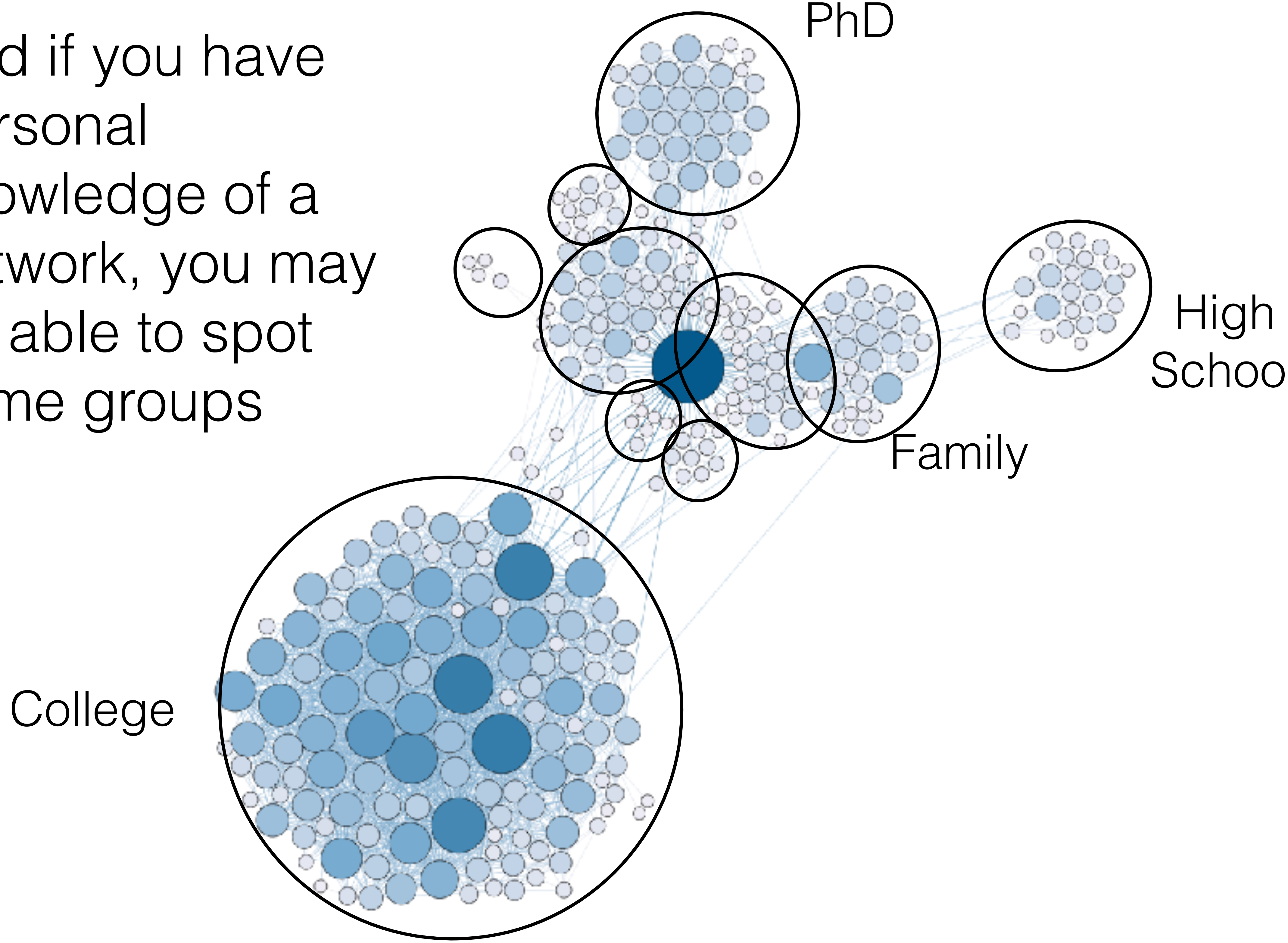


In some cases, community structure is easy to detect by eye...

ref: Lada Adamic

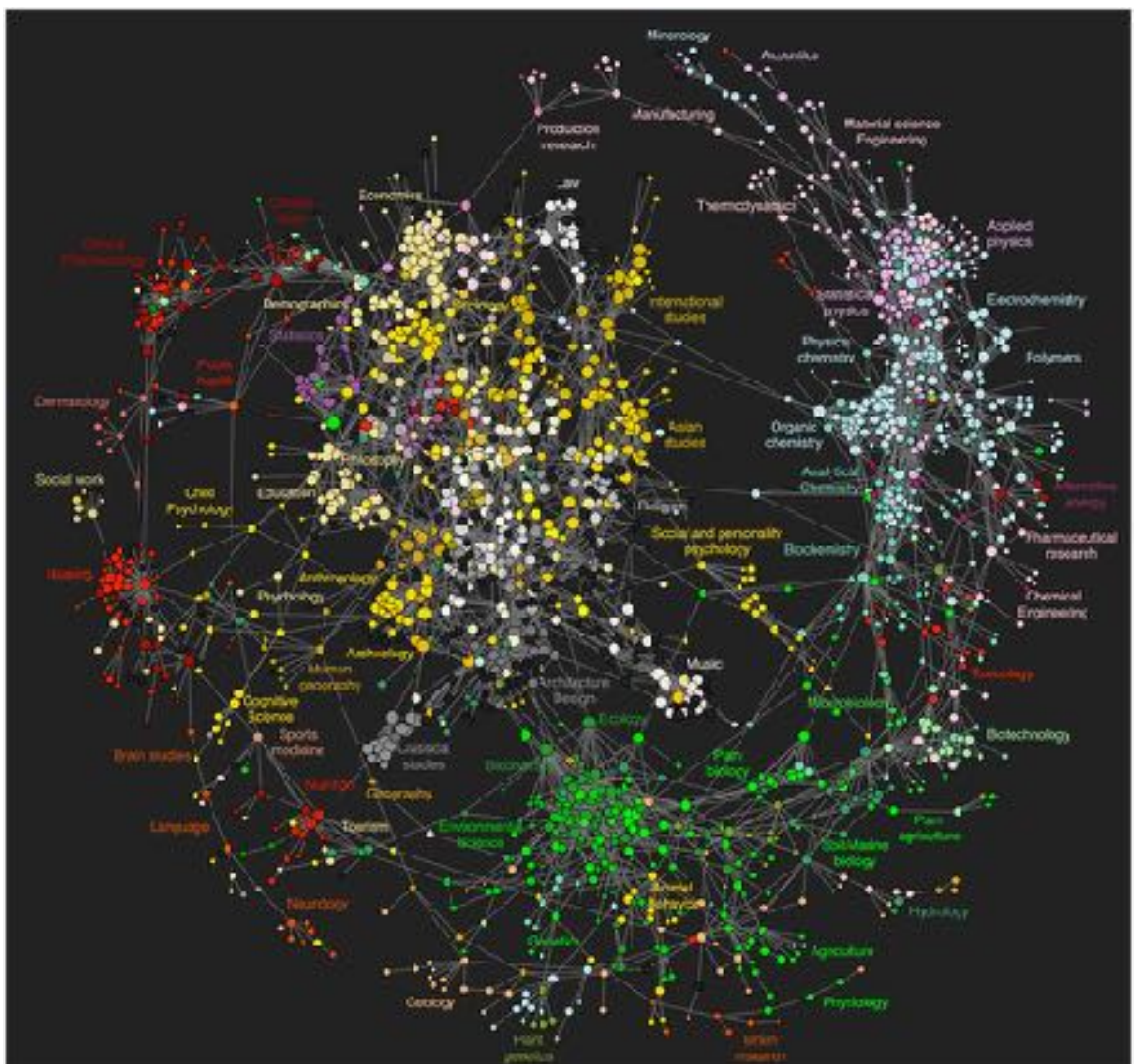
Community Structure

And if you have personal knowledge of a network, you may be able to spot some groups



Community Structure

But in many cases, communities are much harder to pick out by eye (or your eyes lie to your)



ref: network of science, Bollen et al (2009)

It can also be difficult to categorize individual nodes.

So we would like to have a more scientific way of dividing the network up…

Community Detection Algorithms

General idea: create a partition of the nodes, based on where the network “naturally” wants to split

Math fact: a *partition* is a division of a set into smaller, non-overlapping sets.

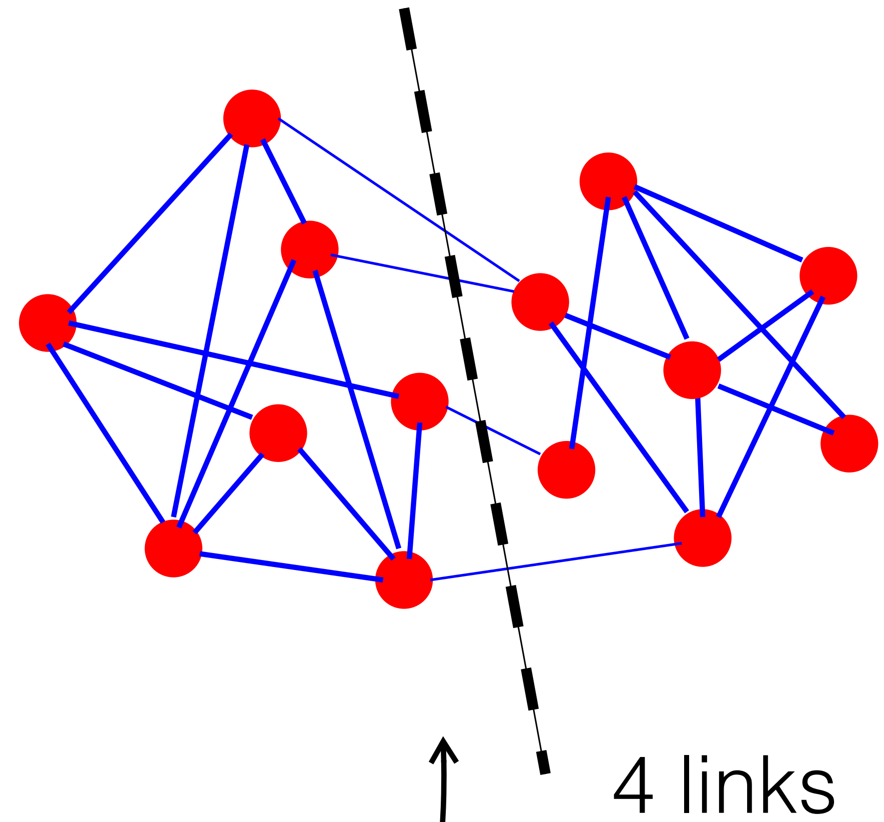
There are lots of ways to do this (we’ll look at three):

- Graph Partitioning
- Hierarchical Clustering
- Girvan-Newman

Method 1:
Graph Partitioning

Graph Partitioning: divide the network into a pre-defined number of chunks of a pre-defined size

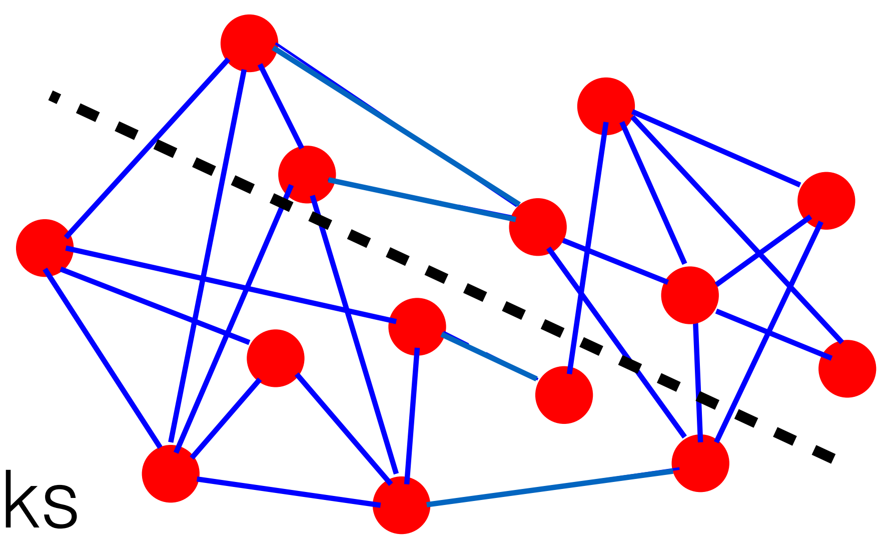
Make the cut in a place that severs the fewest links



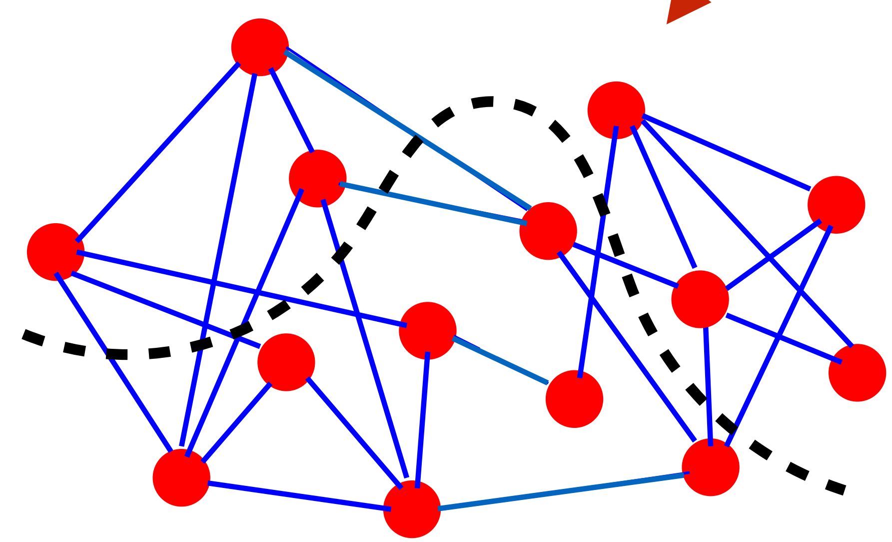
Example: a 14 node network, divided in half

Method 1:
Graph Partitioning

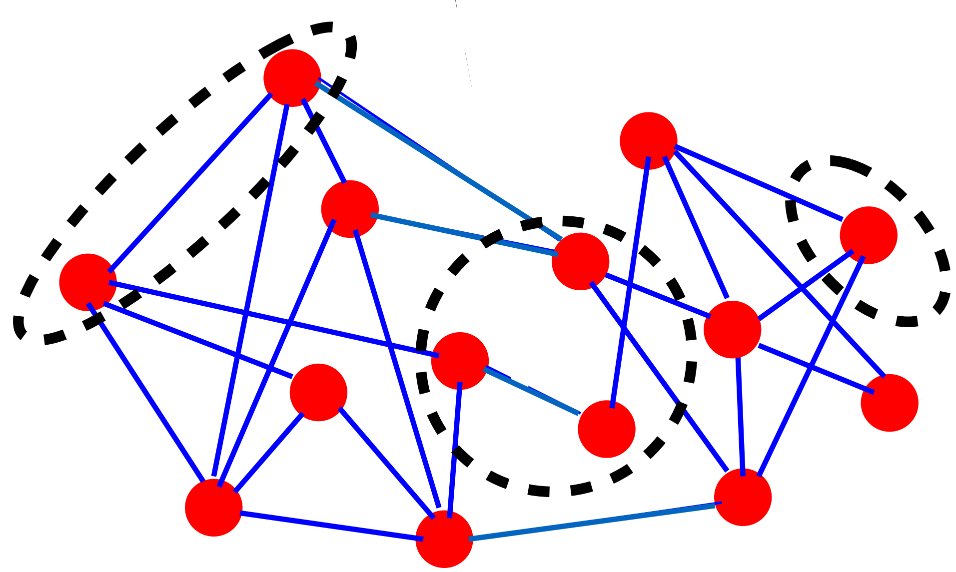
Other divisions require cutting more links



8 links



12 links

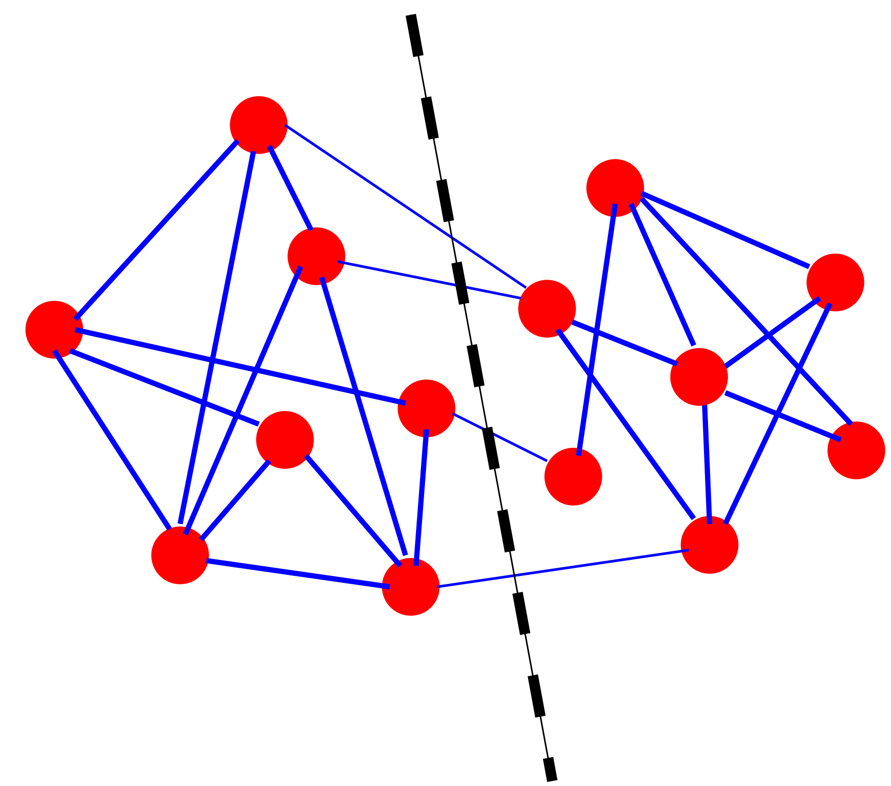


16 links

Community Structure
Graph Partitioning

Graph partitioning is a very straightforward way to divide the network into communities.

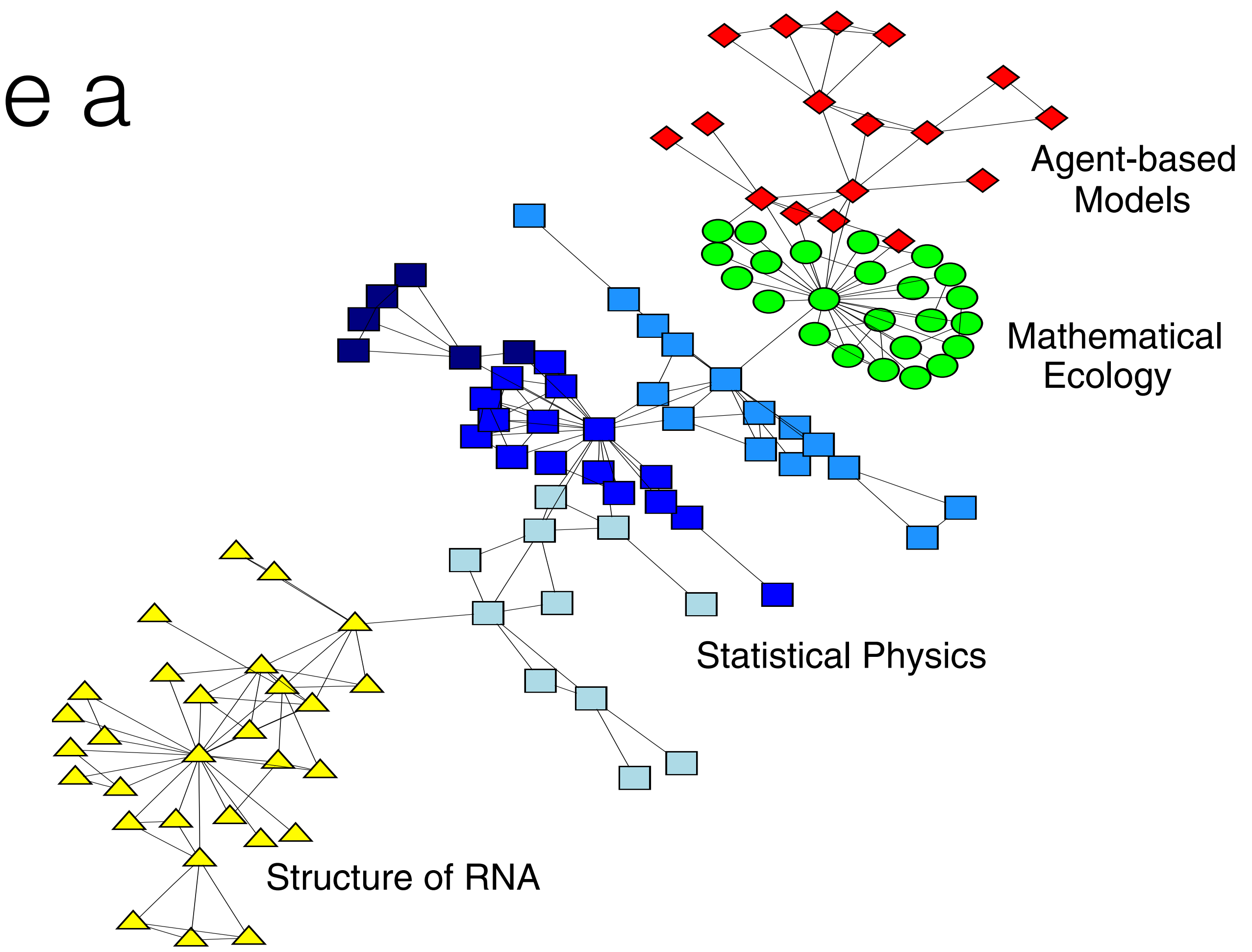
However, the problem is that we need to know how many partitions we want, and how big we want them to be!



Community Structure
Graph Partitioning

In some cases, this may be a reasonable thing to do

But in many cases, the number and size of the communities is exactly what we want to find out…



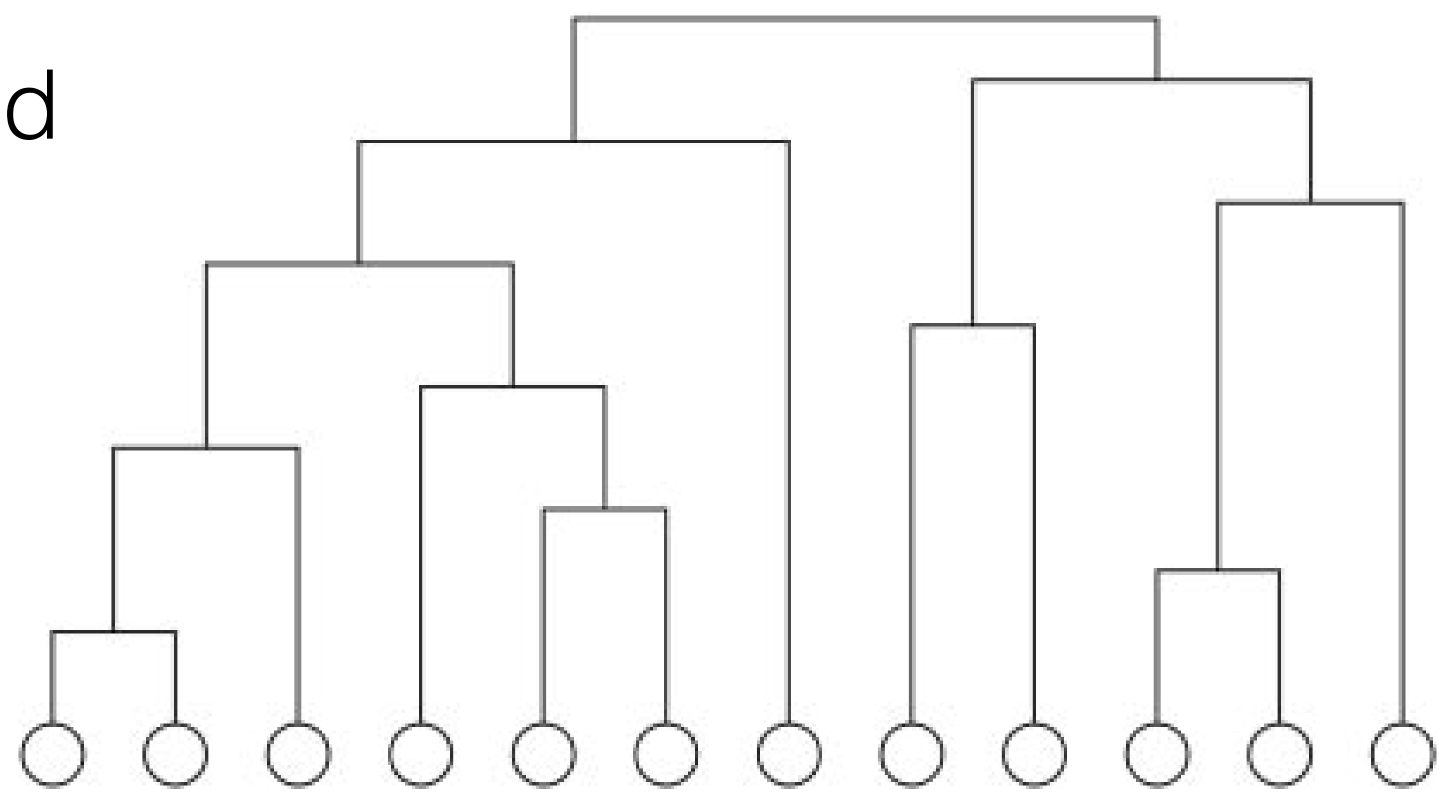
Community Structure
Hierarchical Clustering

Hierarchical Clustering is a method for dividing the network into clusters of sizes determined by the network itself.

- Assign a weight, w_{ij}, to each pair of nodes in the network, representing how closely related they are
- Remove all of the edges in the network.
- Reconnect the nodes, starting with the edge that has the highest weight
- As edges are added, the network is connected back together (it may not be the same way it was before, but that's fine)

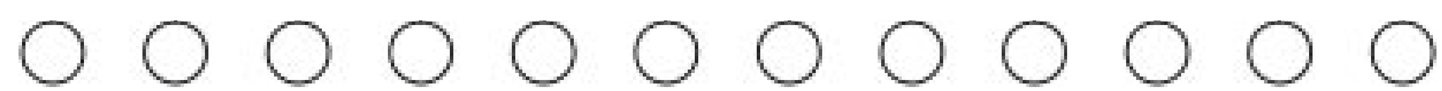
Community Structure
Hierarchical Clustering

The result of this process is summarized by a *dendrogram*



Community Structure
Hierarchical Clustering

nodes

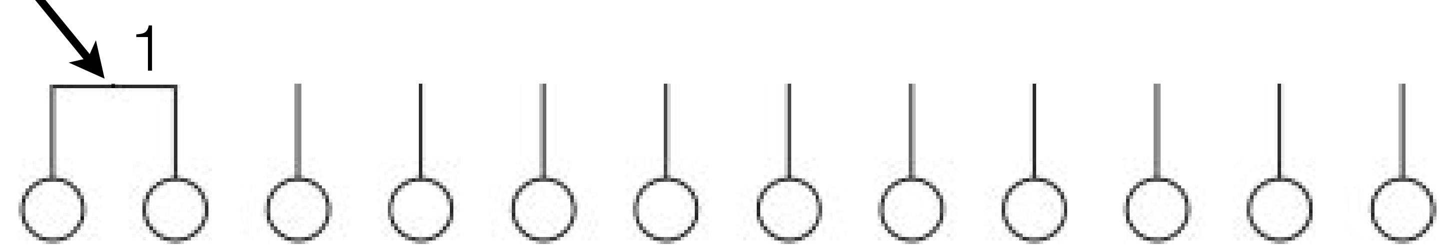


Community Structure

Hierarchical Clustering

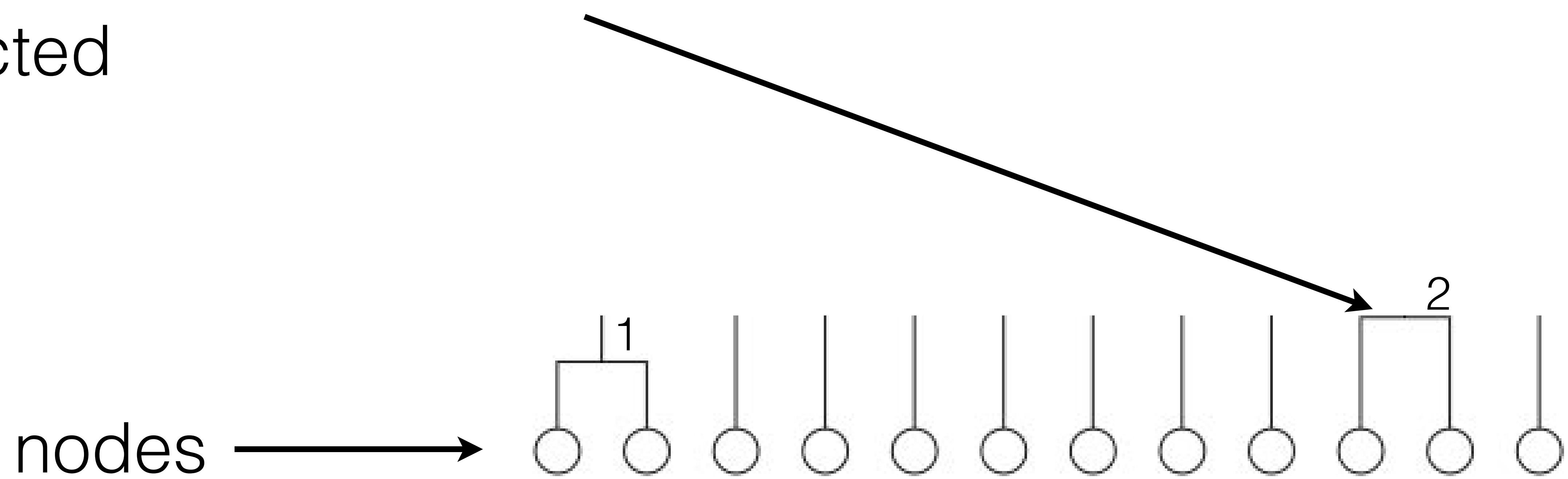
Each edge that is added is represented by a connection in the dendrogram

nodes



Community Structure
Hierarchical Clustering

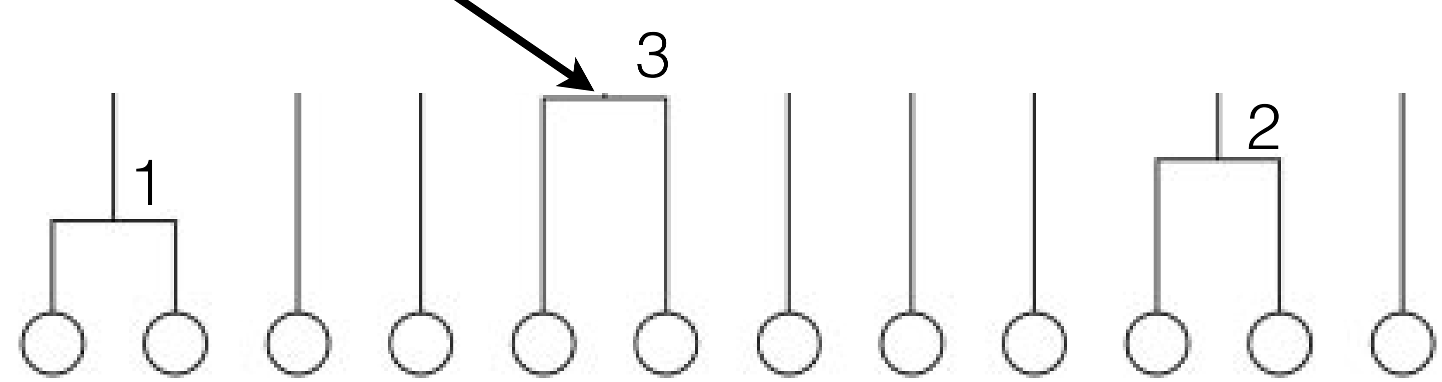
As more edges are added, the network becomes more connected



Community Structure
Hierarchical Clustering

As more edges are added, the network becomes more connected

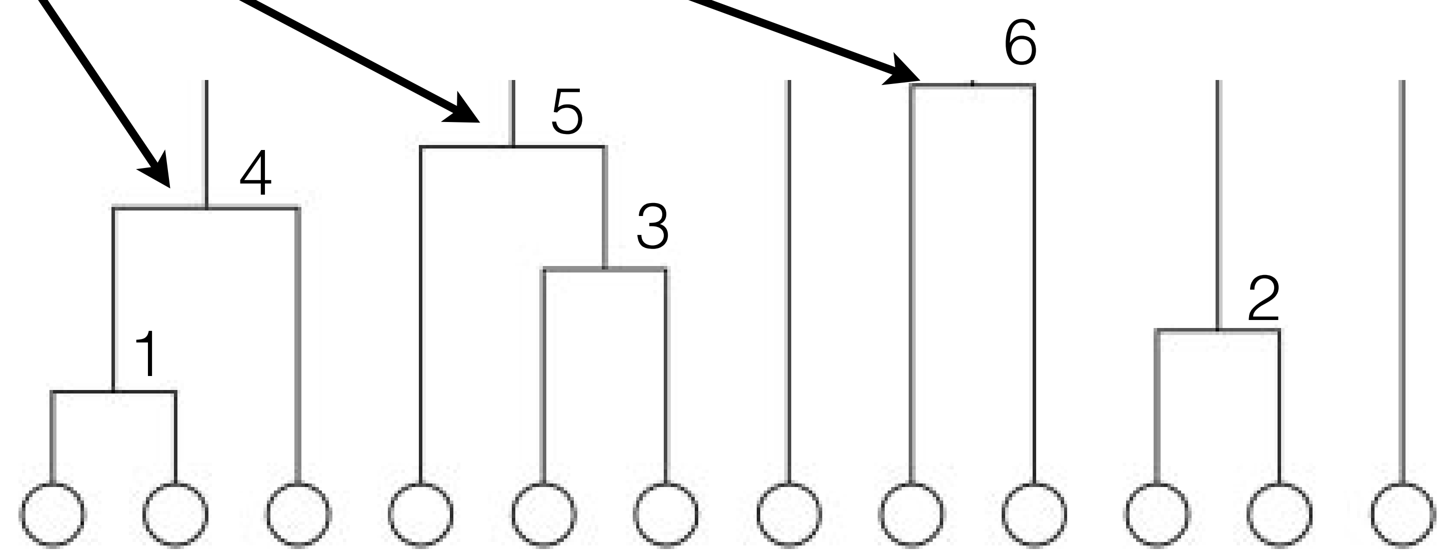
nodes



Community Structure
Hierarchical Clustering

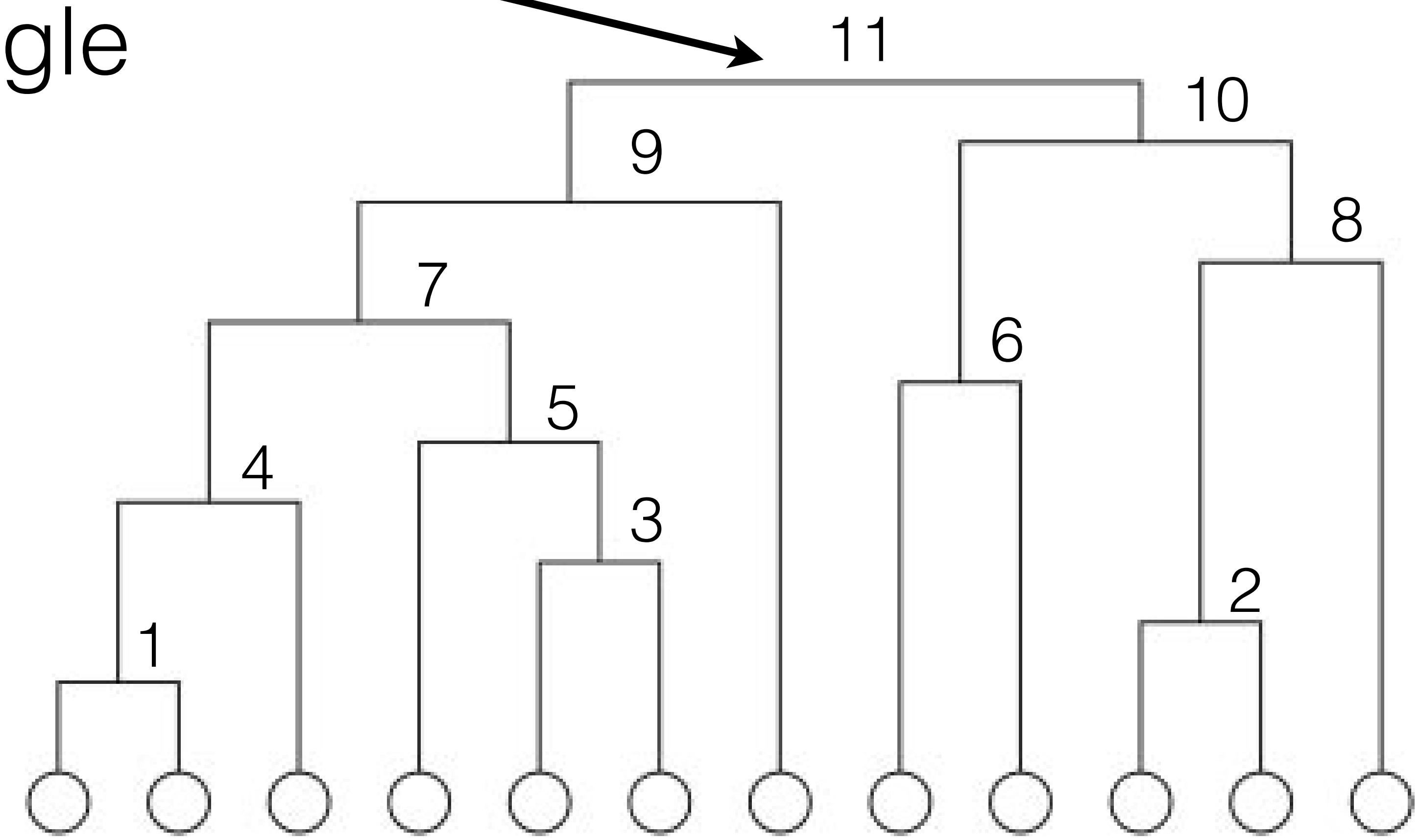
As more edges are added, the network becomes more connected

nodes



Community Structure
Hierarchical Clustering

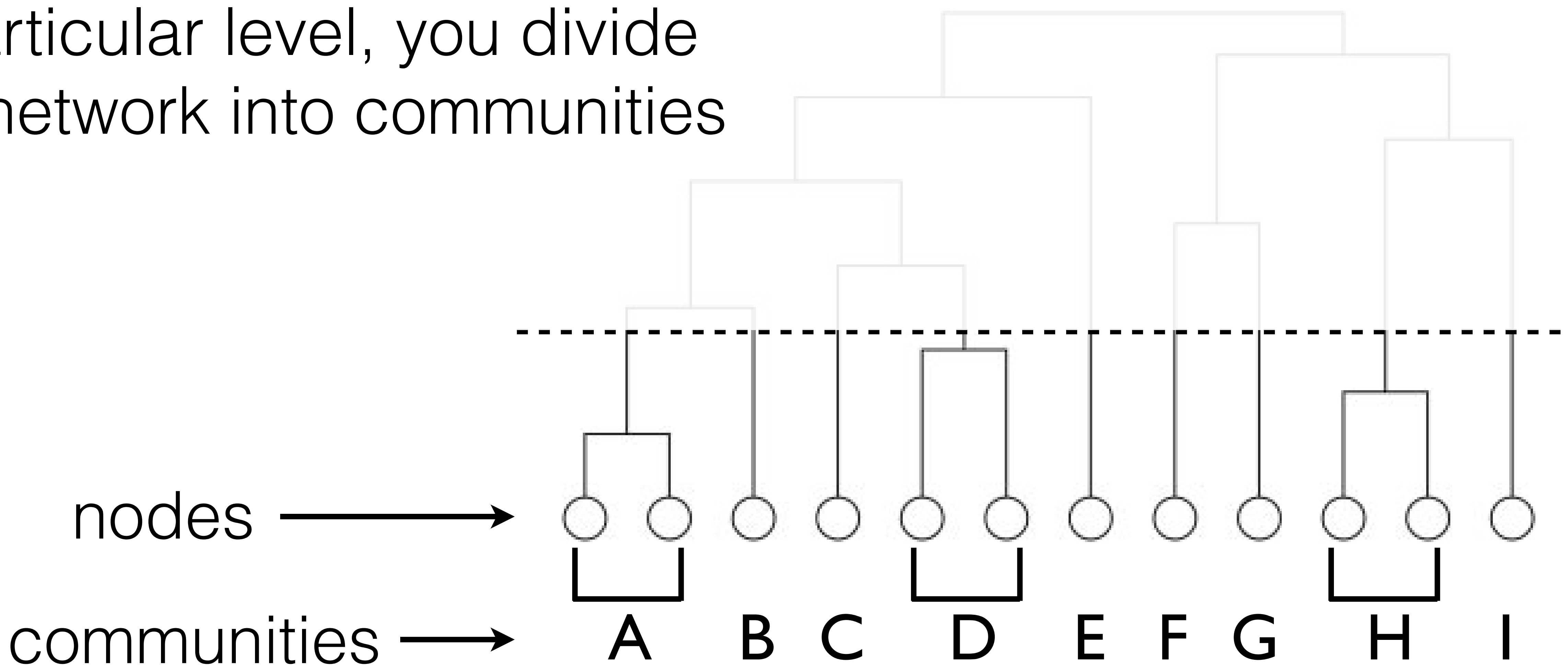
We stop when the nodes are all connected into a single component



nodes

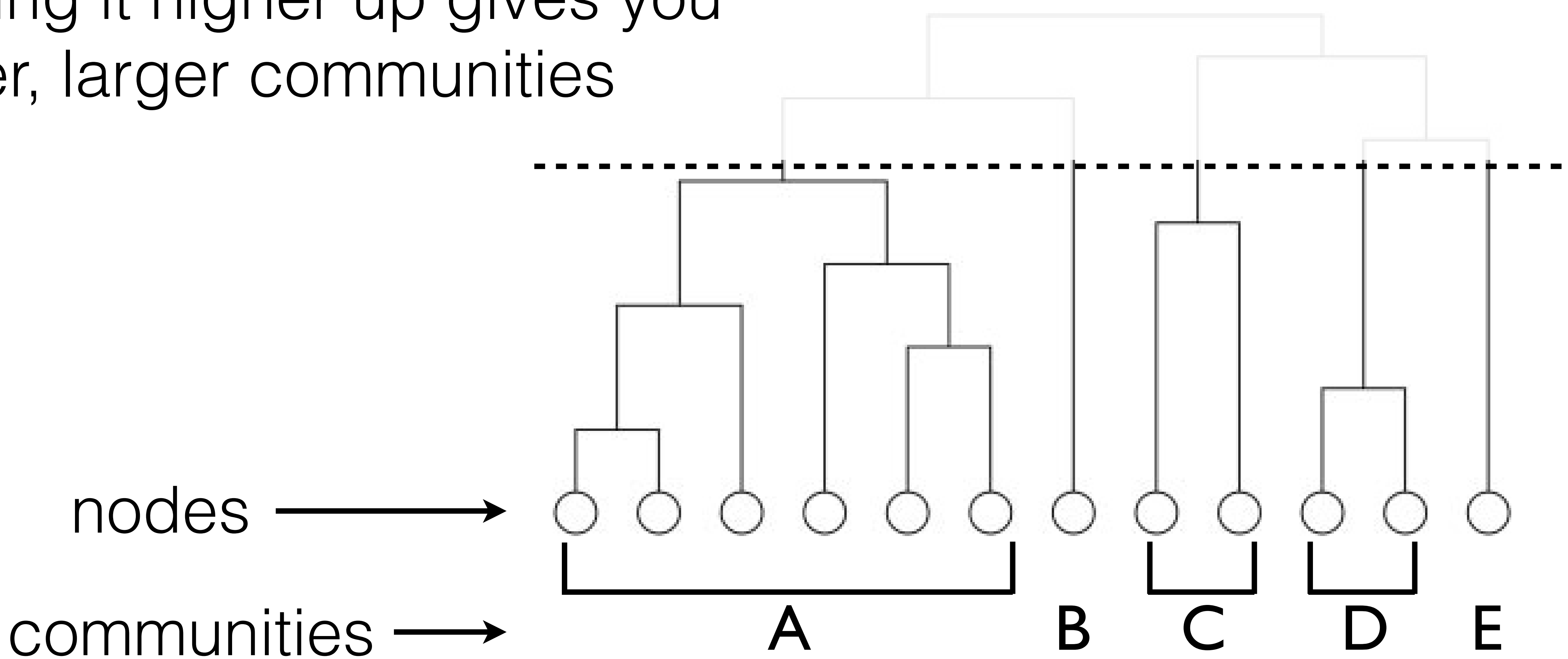
Community Structure
Hierarchical Clustering

By cutting the dendrogram at a particular level, you divide the network into communities



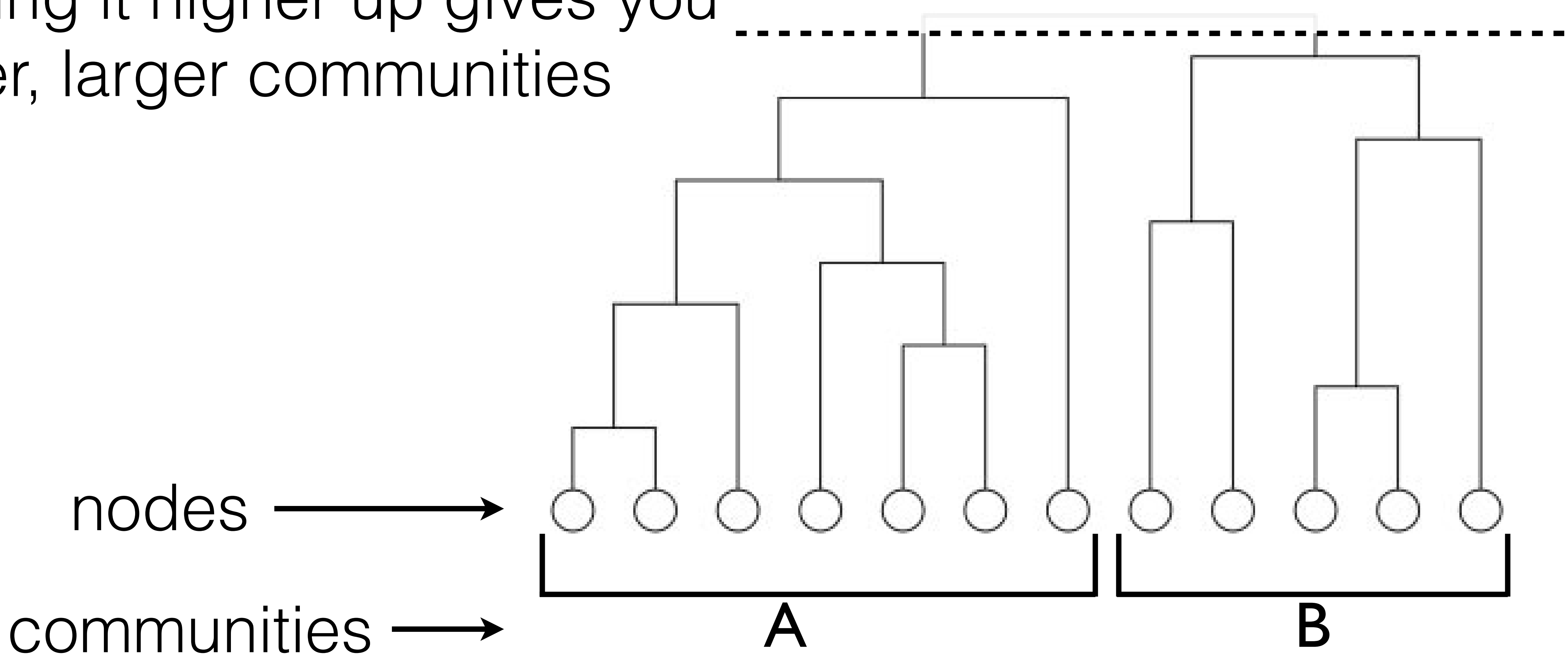
Community Structure
Hierarchical Clustering

Cutting it higher up gives you fewer, larger communities



Community Structure
Hierarchical Clustering

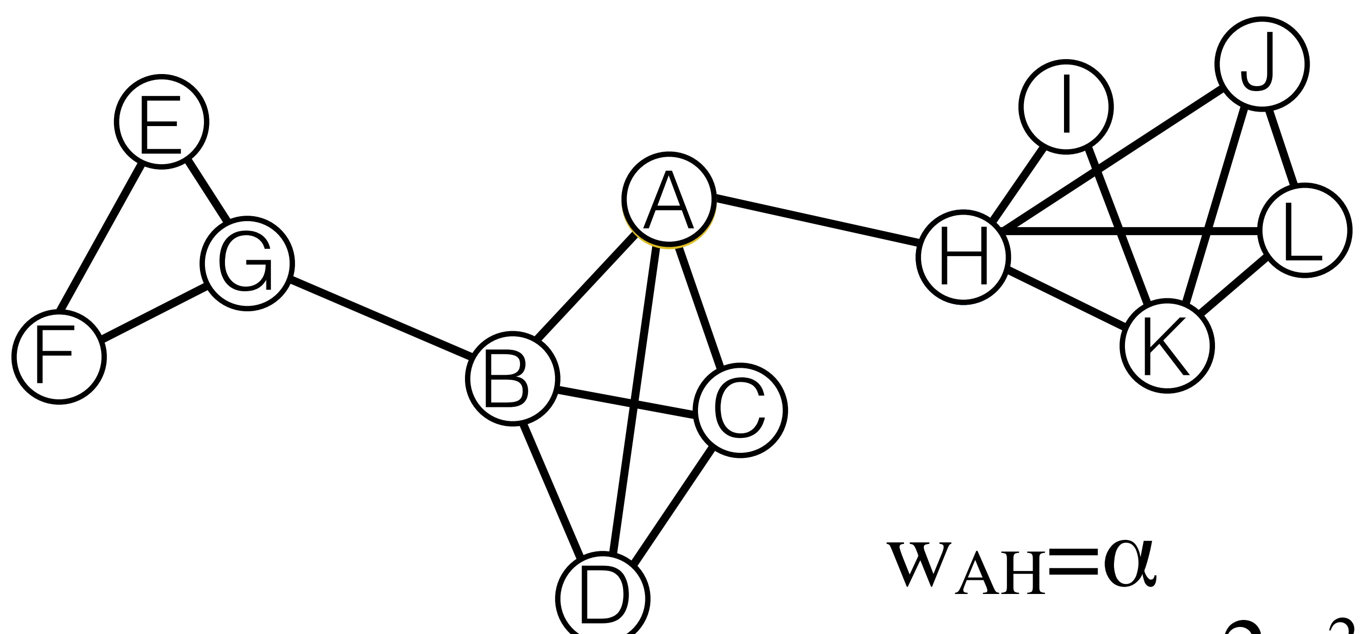
Cutting it higher up gives you fewer, larger communities



Community Structure
Hierarchical Clustering

There are lots of weights that you could use (including the weights you've already assigned

One example: the number of paths connecting the two nodes, weighted by the length of the path



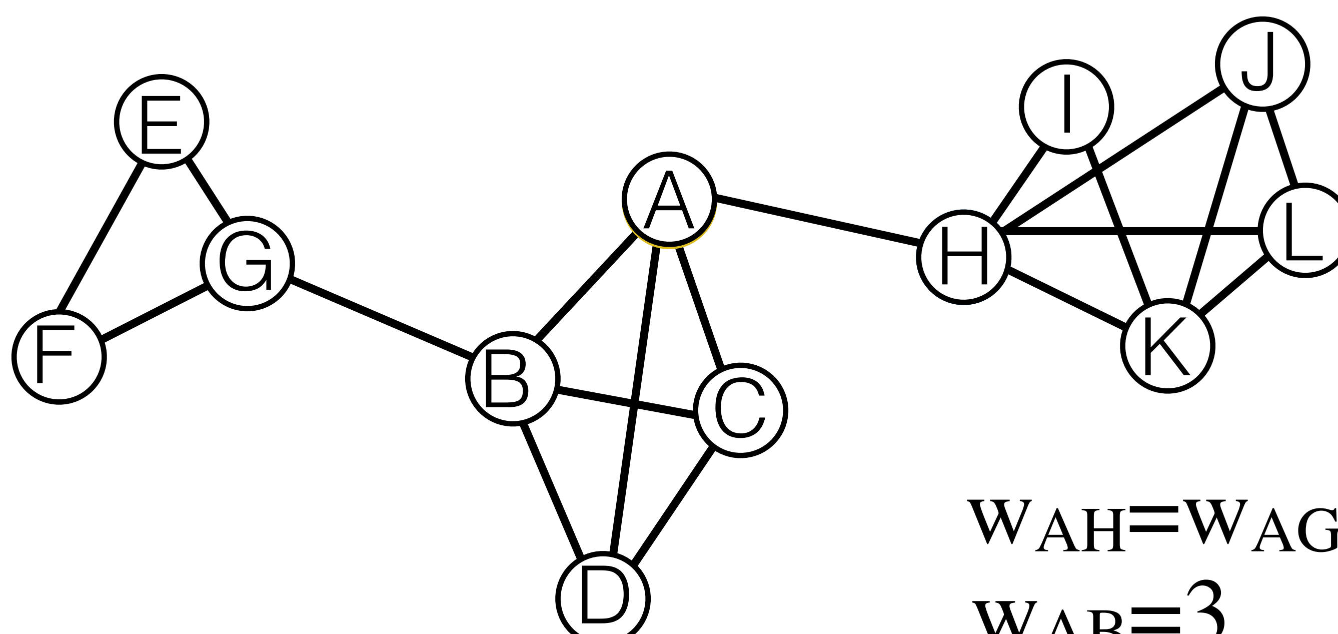
$w_{AH}=\alpha$

$w_{AB}=\alpha+2\alpha^2$

$w_{AG}=\alpha^2+2\alpha^3+\alpha^4$

Community Structure
Hierarchical Clustering

Another example: the number of paths between the nodes that don't have any of the same nodes



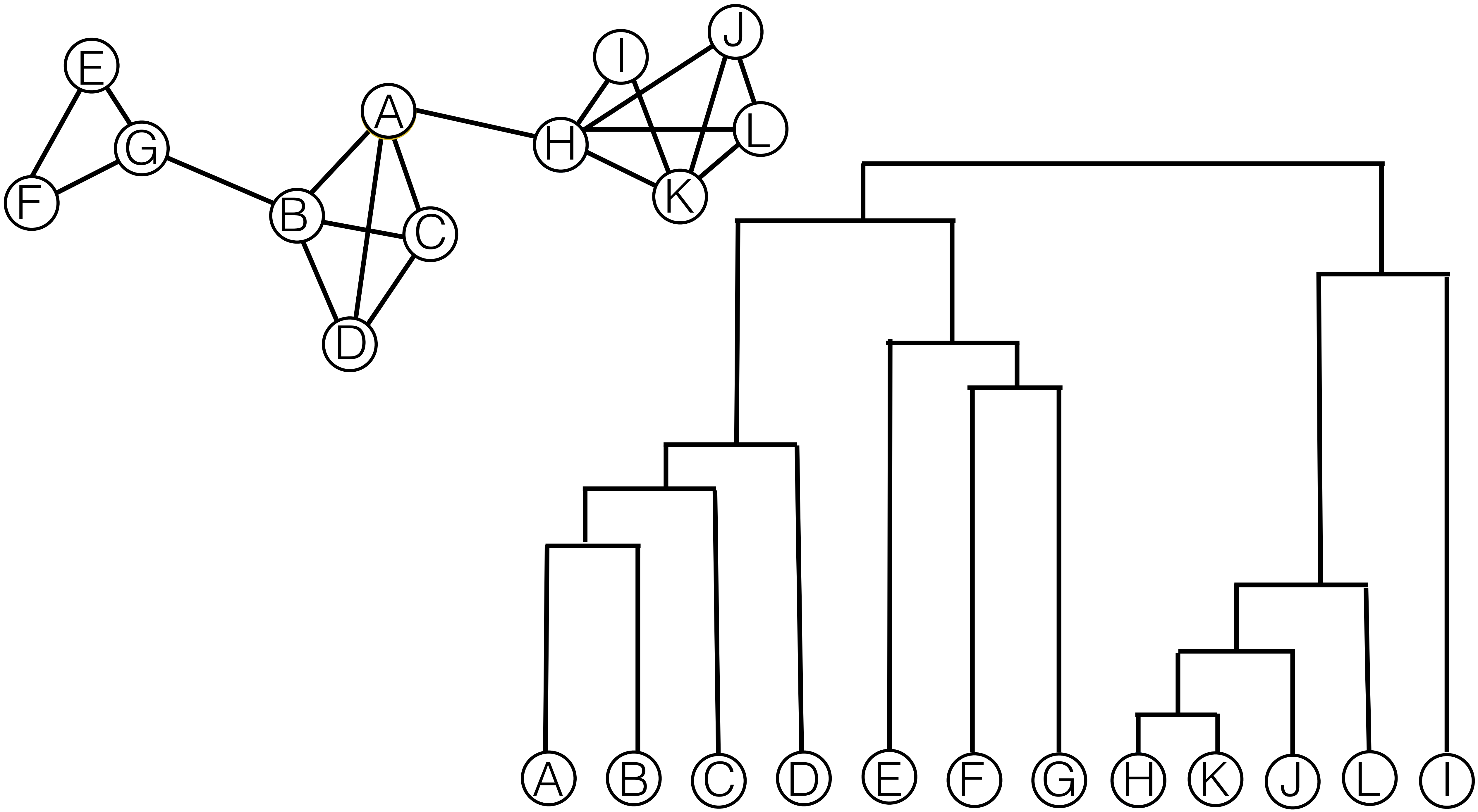
$w_{AH}=w_{AG}=1$

$w_{AB}=3$

$w_{HK}=4$

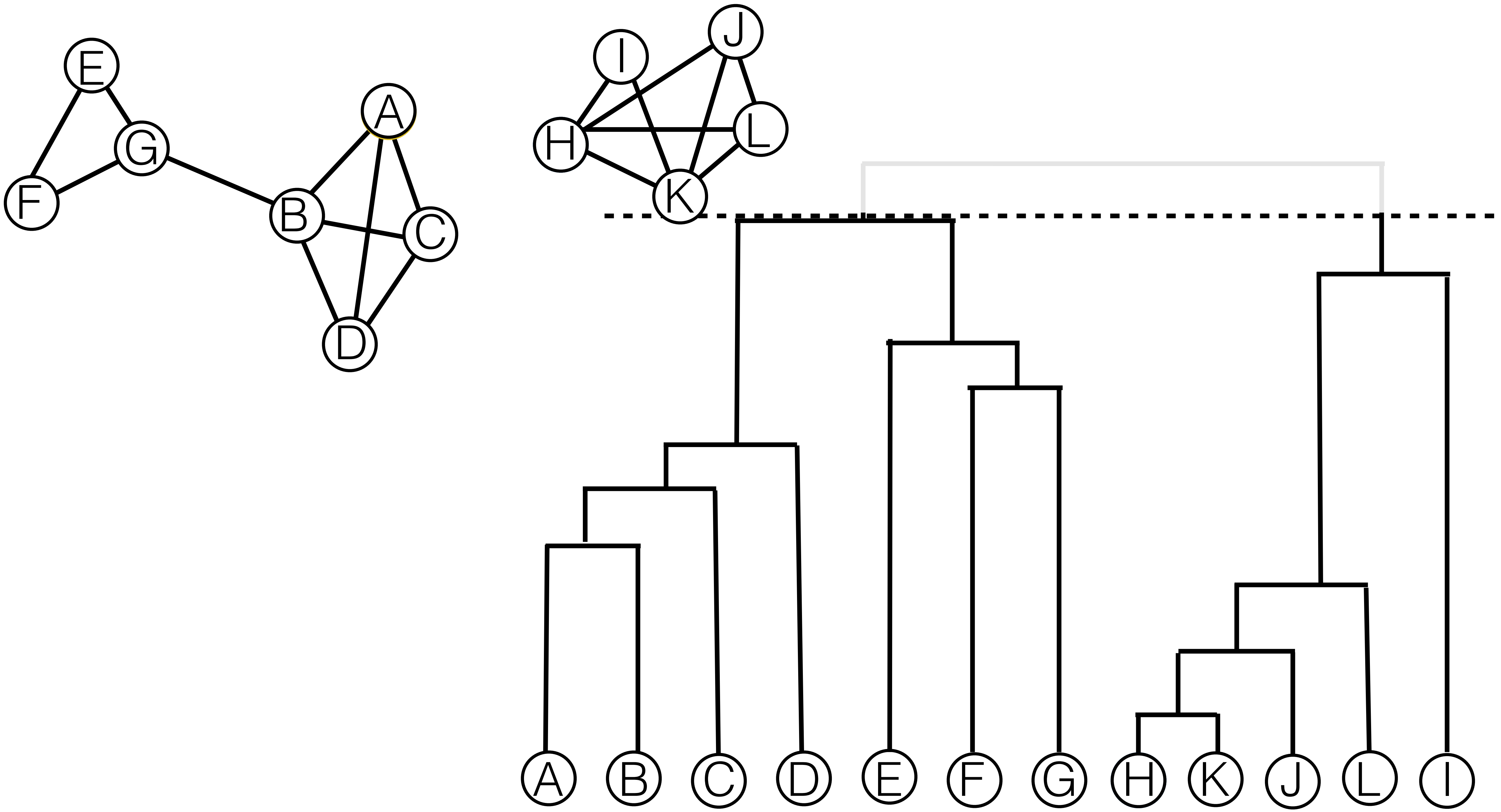
Community Structure

Hierarchical Clustering



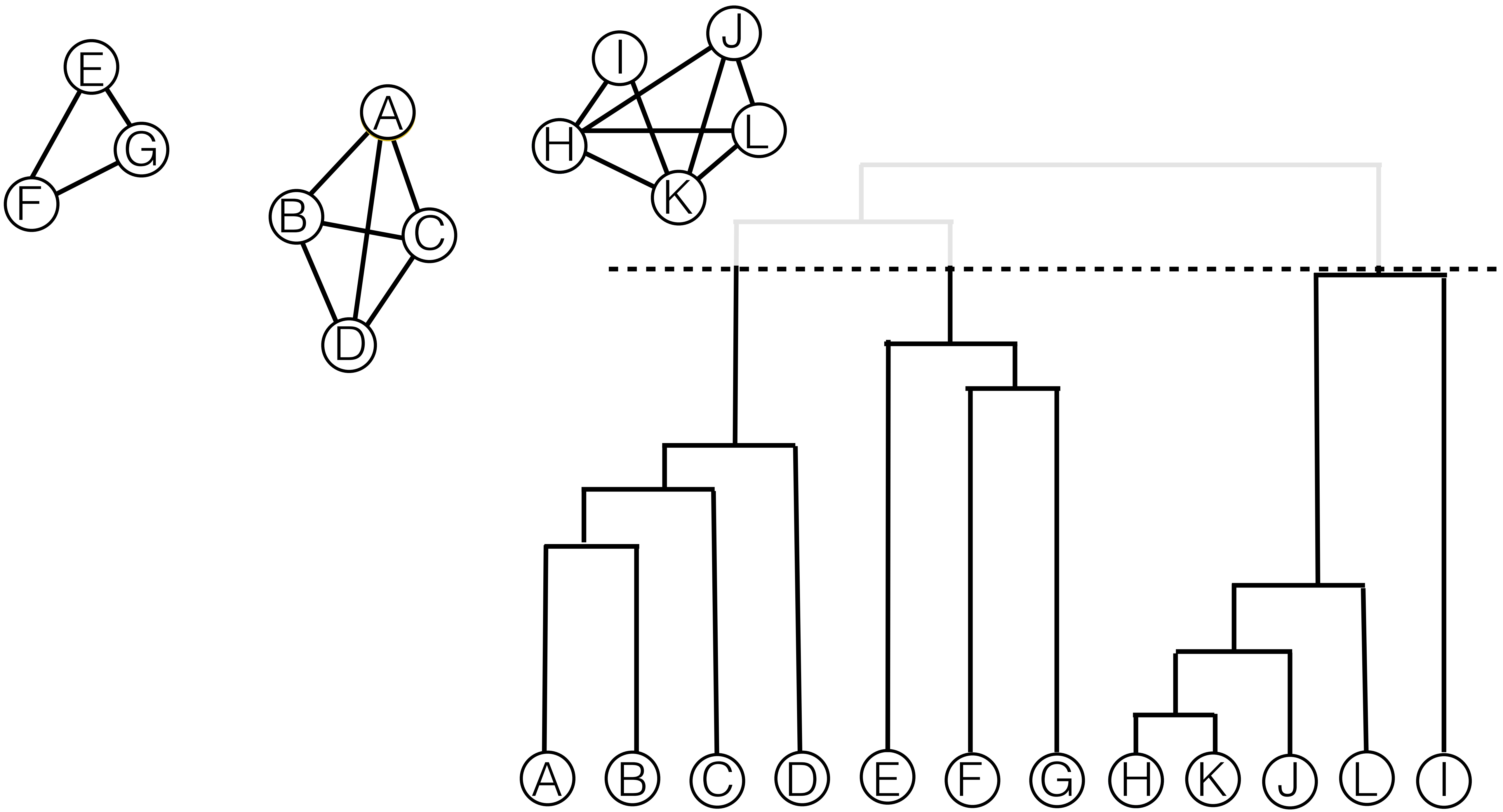
Community Structure

Hierarchical Clustering



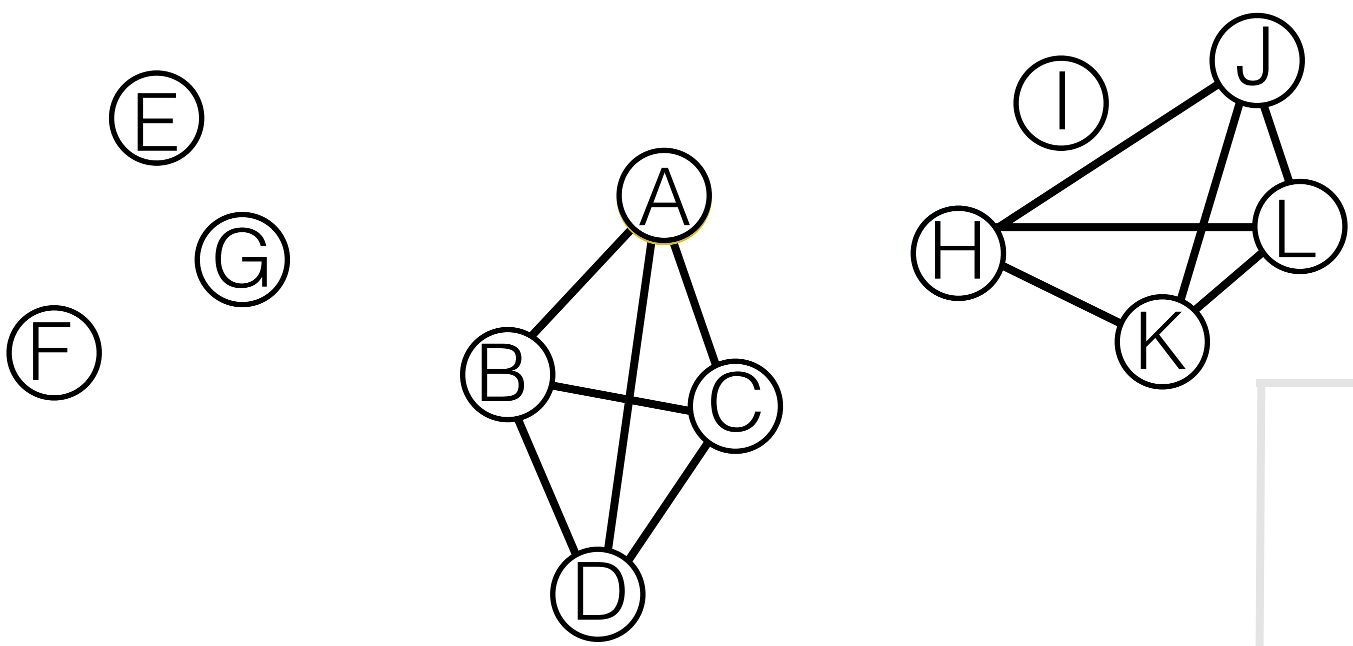
Community Structure

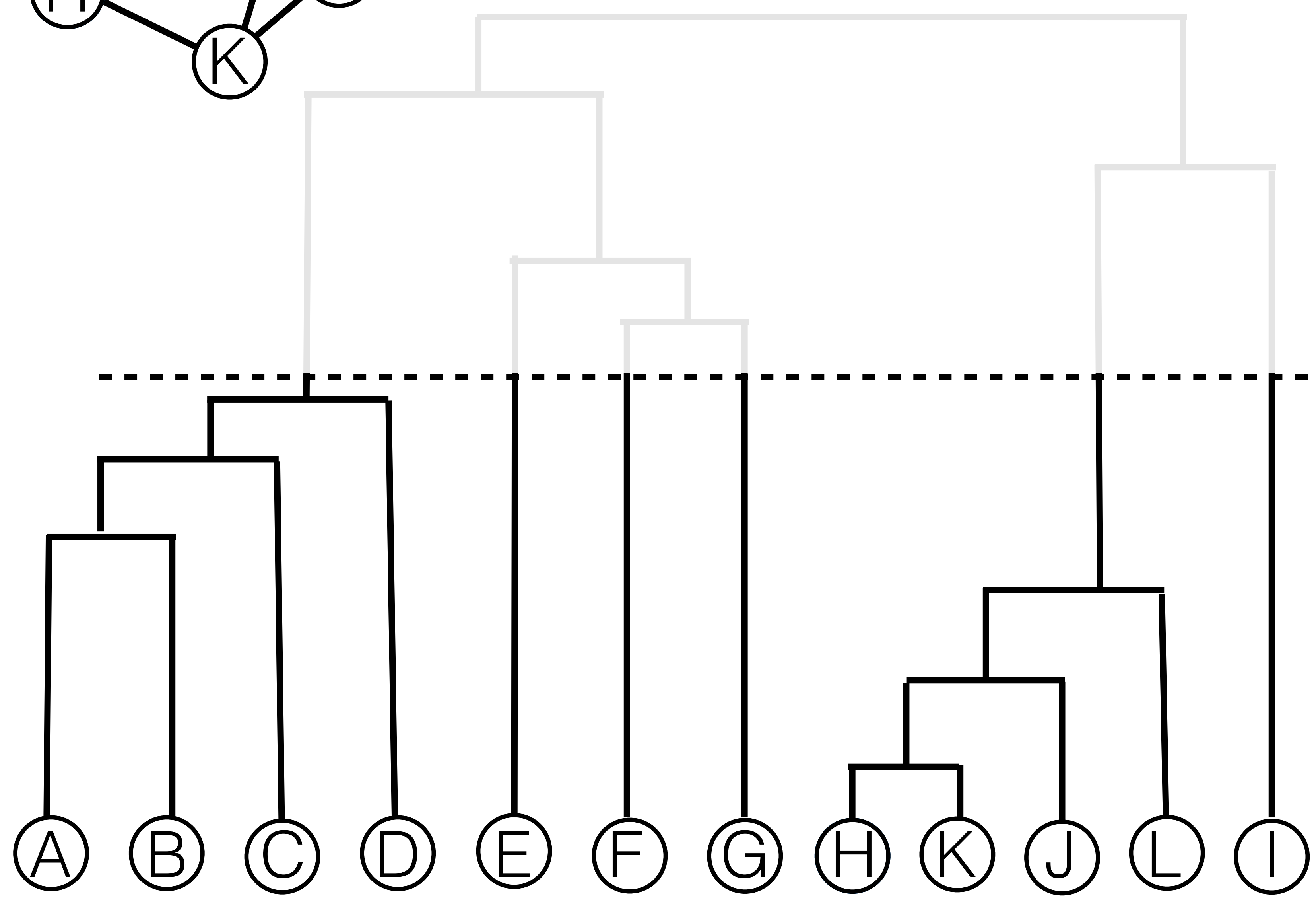
Hierarchical Clustering



Community Structure

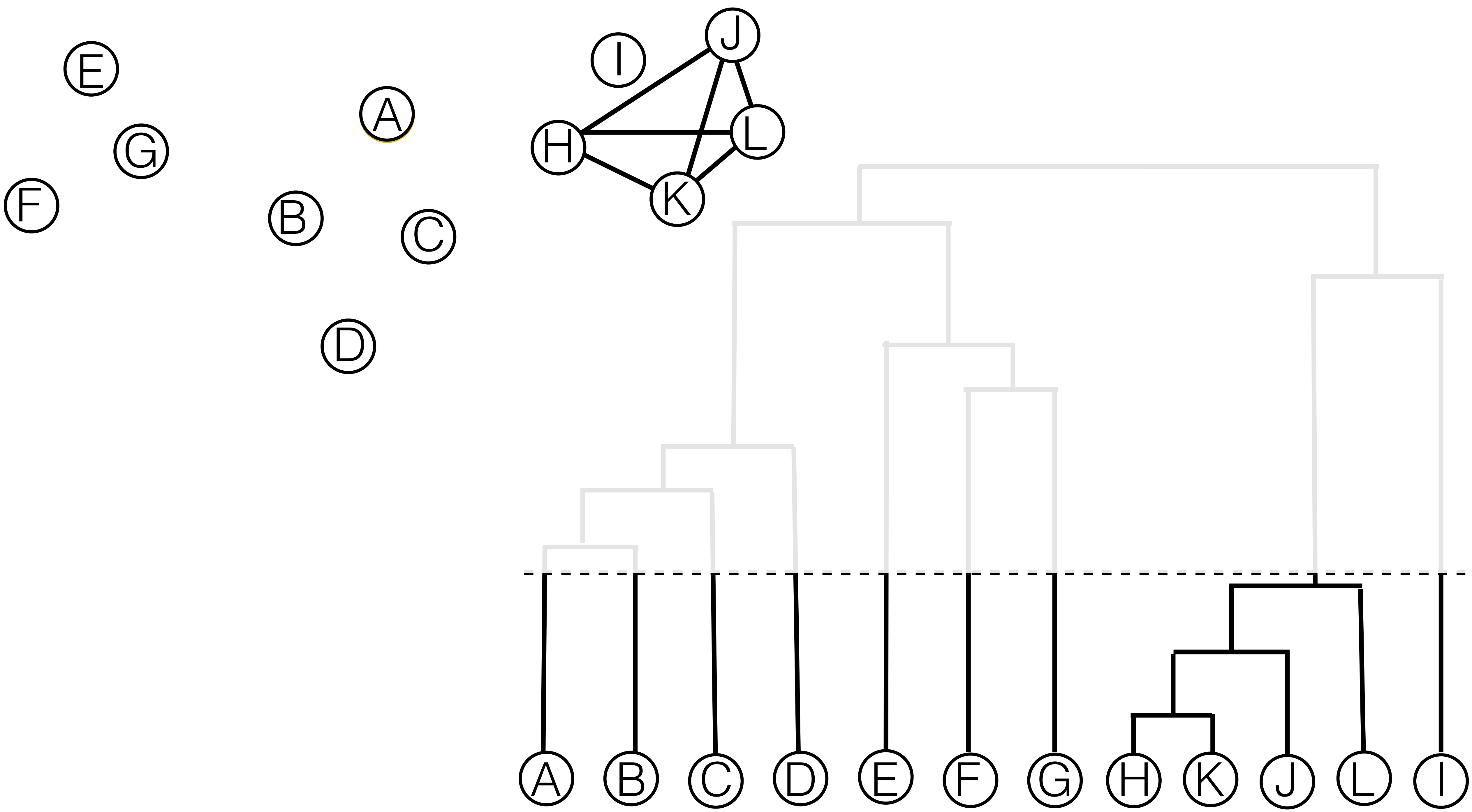
Hierarchical Clustering





Community Structure

Hierarchical Clustering

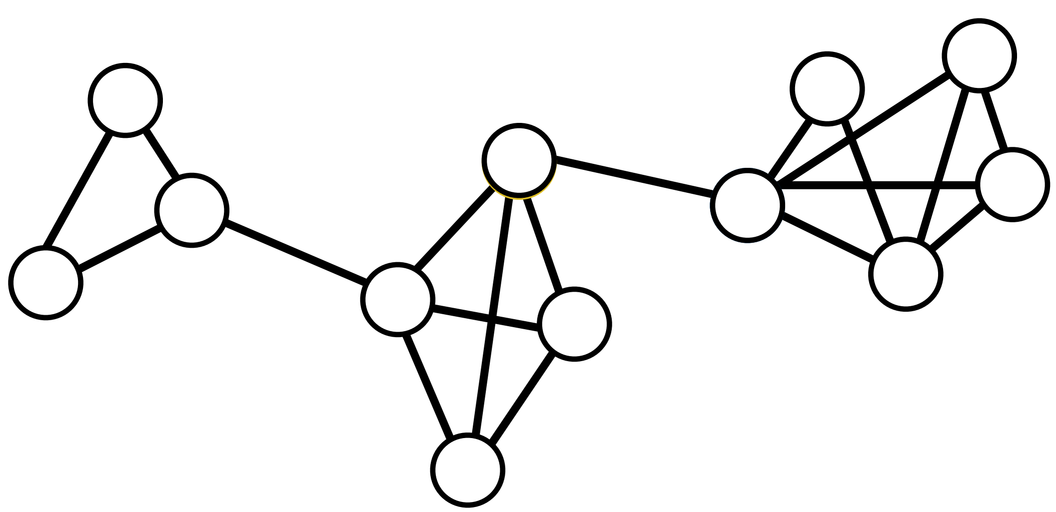


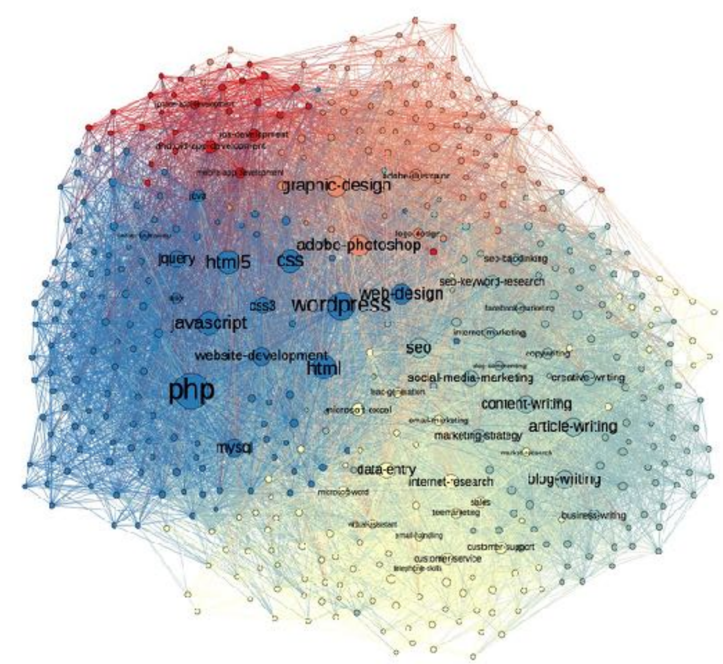
Community Structure
Hierarchical Clustering

There are two disadvantages to hierarchical clustering:

1. It tends to chop off "leaf" nodes that are peripheral to a community

2. It works best on networks that have a naturally hierarchical (nested) structure (which is not all networks)

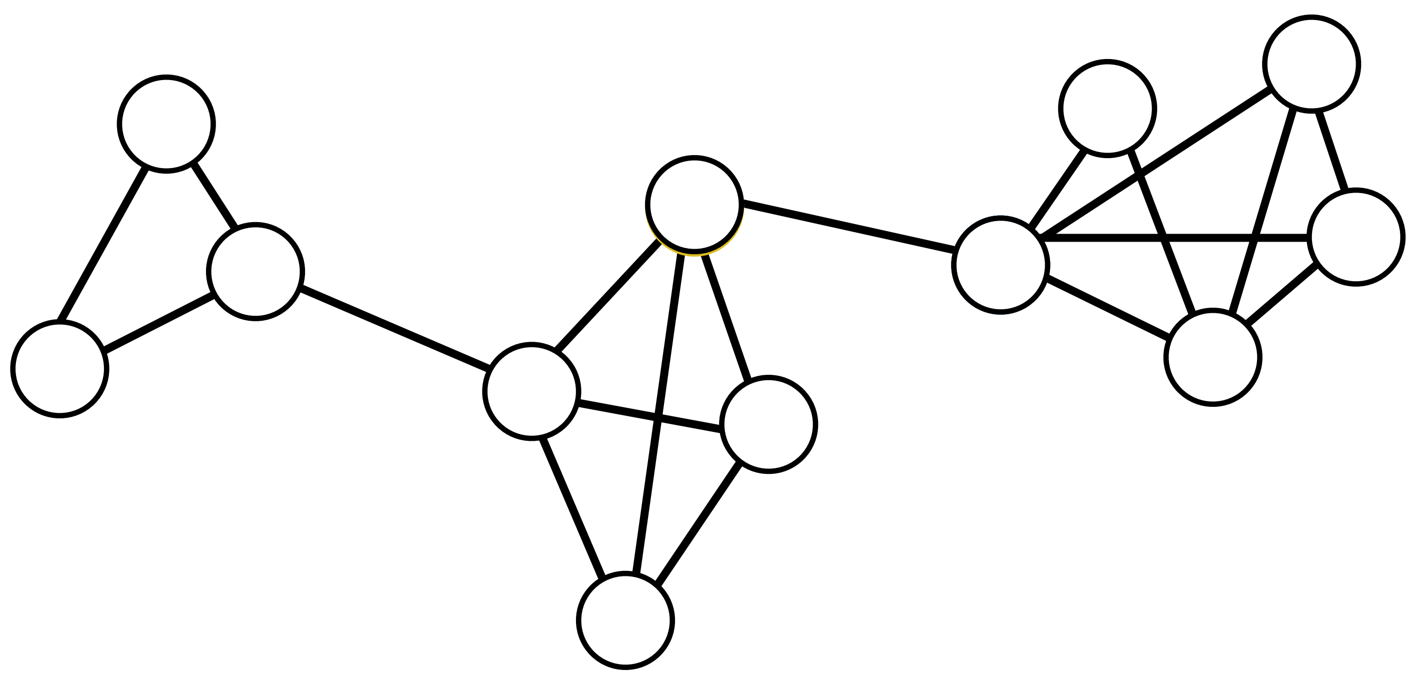




Community Structure
Girvan-Newman

The Girvan-Newman Algorithm tweaks the hierarchical clustering algorithm: it sequentially removes edges with the highest *edge betweenness*

Edge betweenness: The number of shortest paths that go along a particular edge



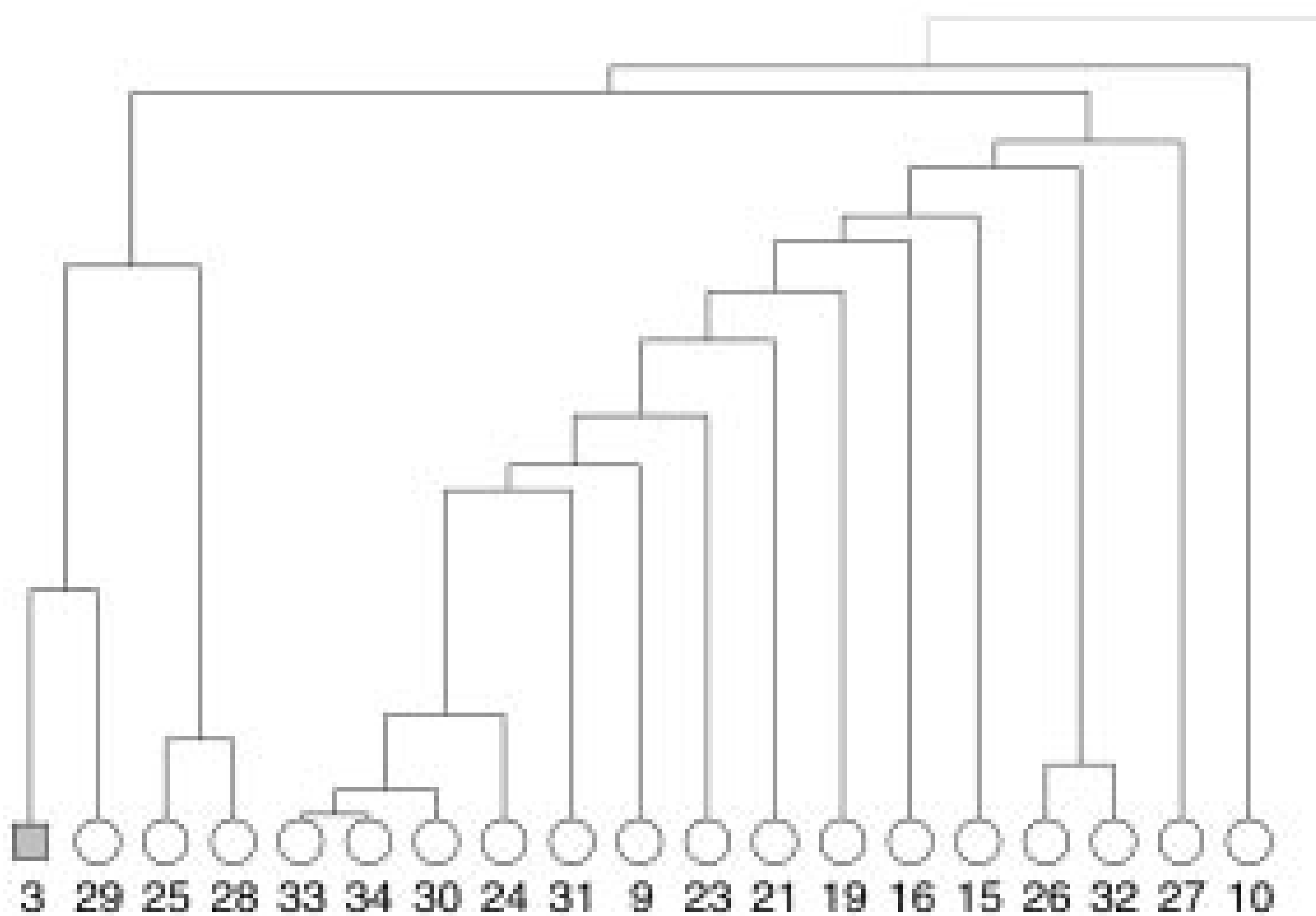
Community Structure
Girvan-Newman

Unlike hierarchical clustering, the Girvan Newman Algorithm recalculates the edge betweenness on each step

1. Calculate betweenness for all edges
2. Remove the edge with the highest betweenness
3. Recalculate the betweenness of all remaining edges
4. Repeat until no edges remain

Community Structure
Girvan-Newman

The result is, again, a dendrogram, which we can cut at different levels to produce different partitions of the network

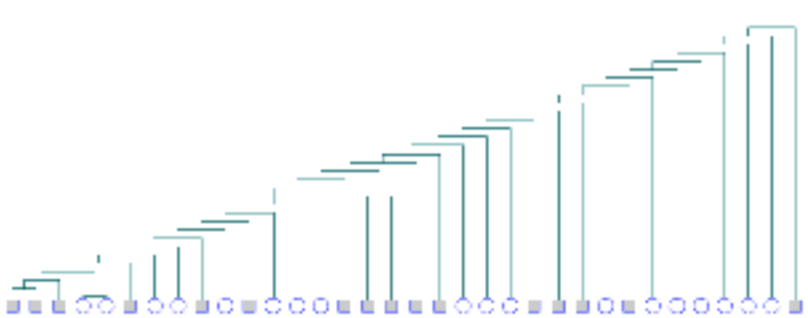


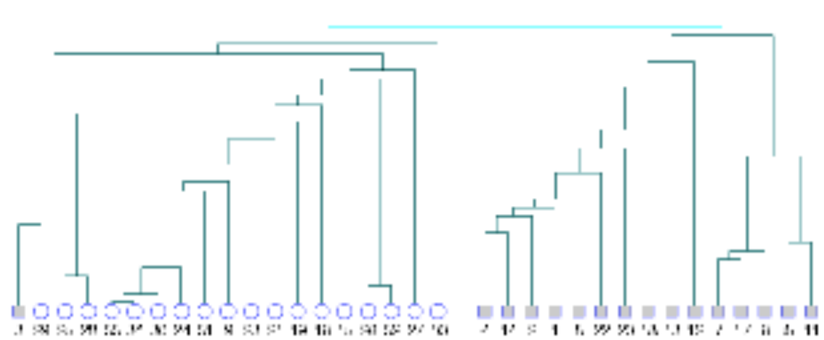
Community Structure
Evaluation

Now we have two algorithms, producing two different community structures. How do we tell which algorithm is best?

→Answer: there is no definitive answer!

However, there are some tests we can perform that give some insight...

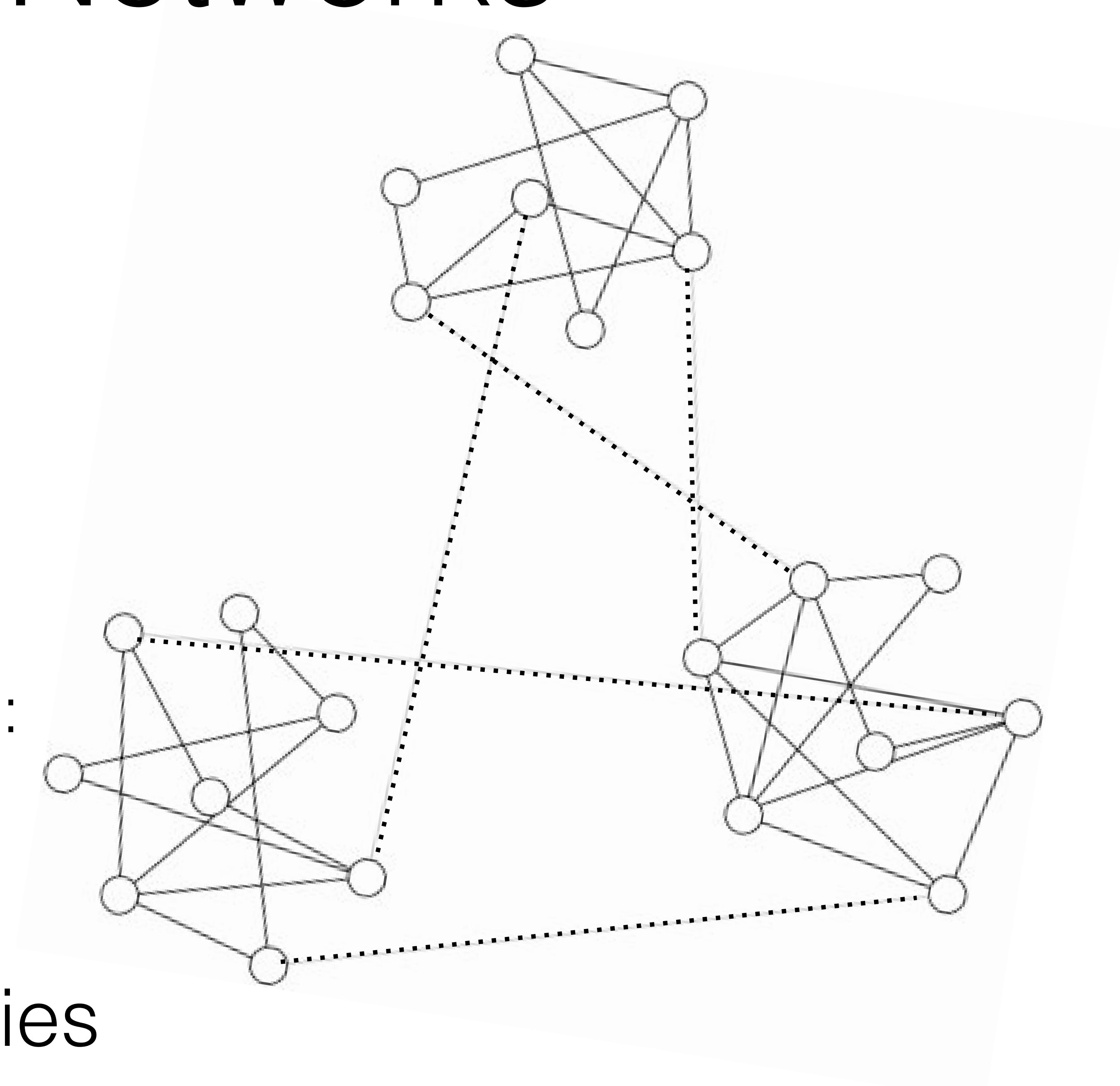




Evaluating Algorithms
Random Networks

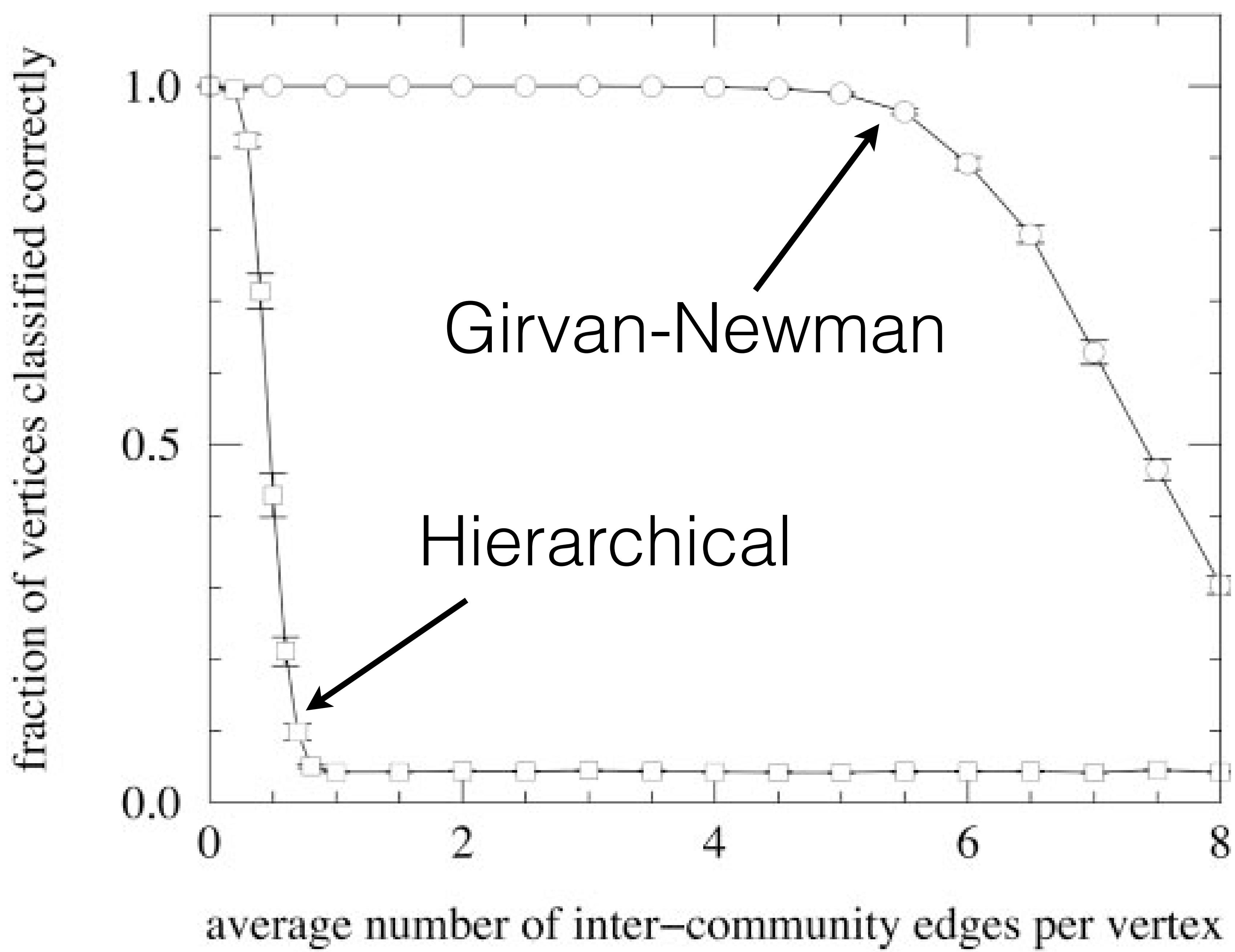
Test 1:
Generate random networks with known communities

- Divide nodes into communities
- Link each node to each other node with a set probability
- Probability of linking within your community greater than outside: $p_{out} < p_{in}$



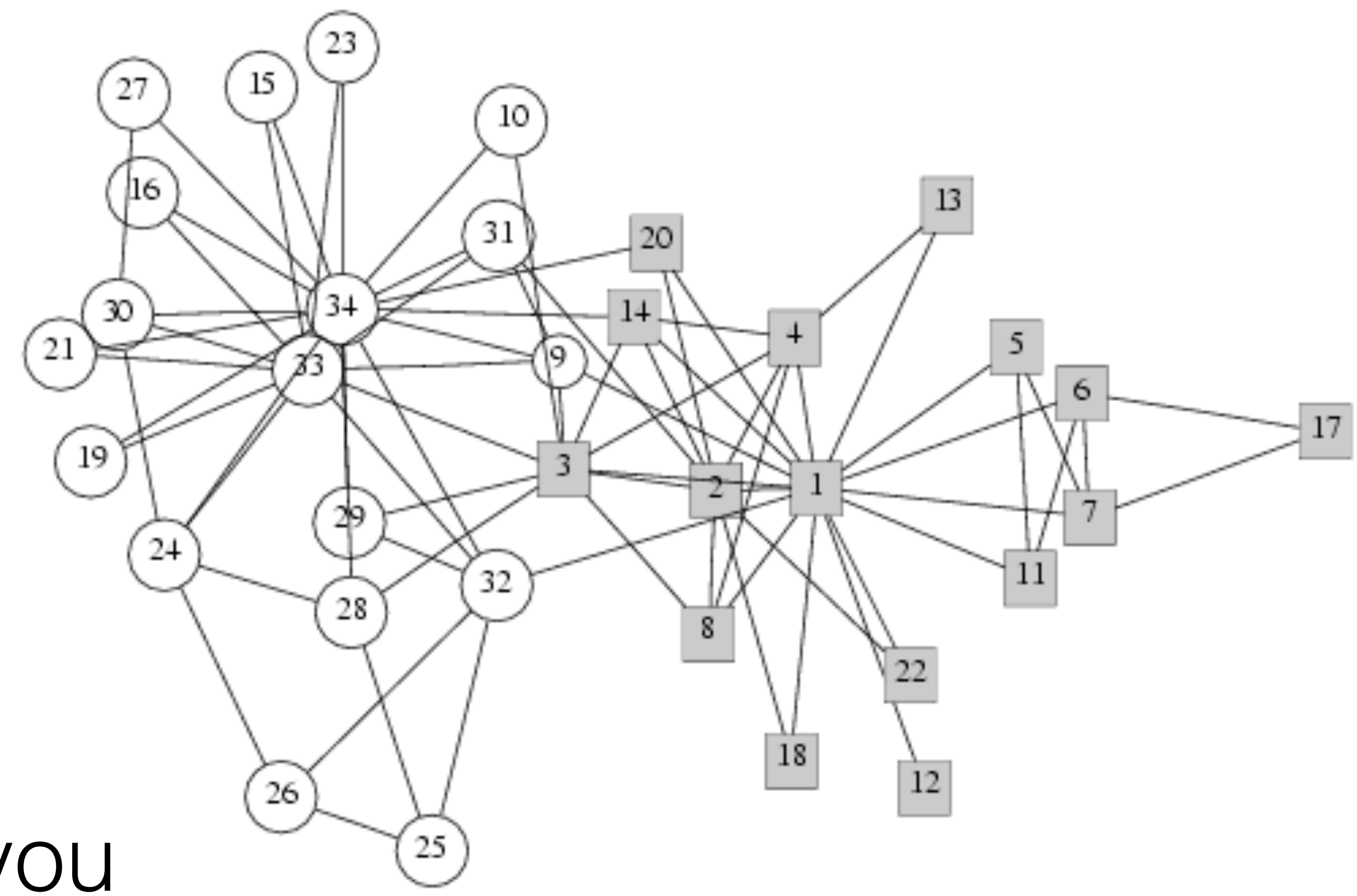
Run your algorithm: do you get out the same communities you put in?

Evaluating Algorithms
Random Networks



Evaluating Algorithms
Known community structure

Test 2:
Use a real social network with known community structure



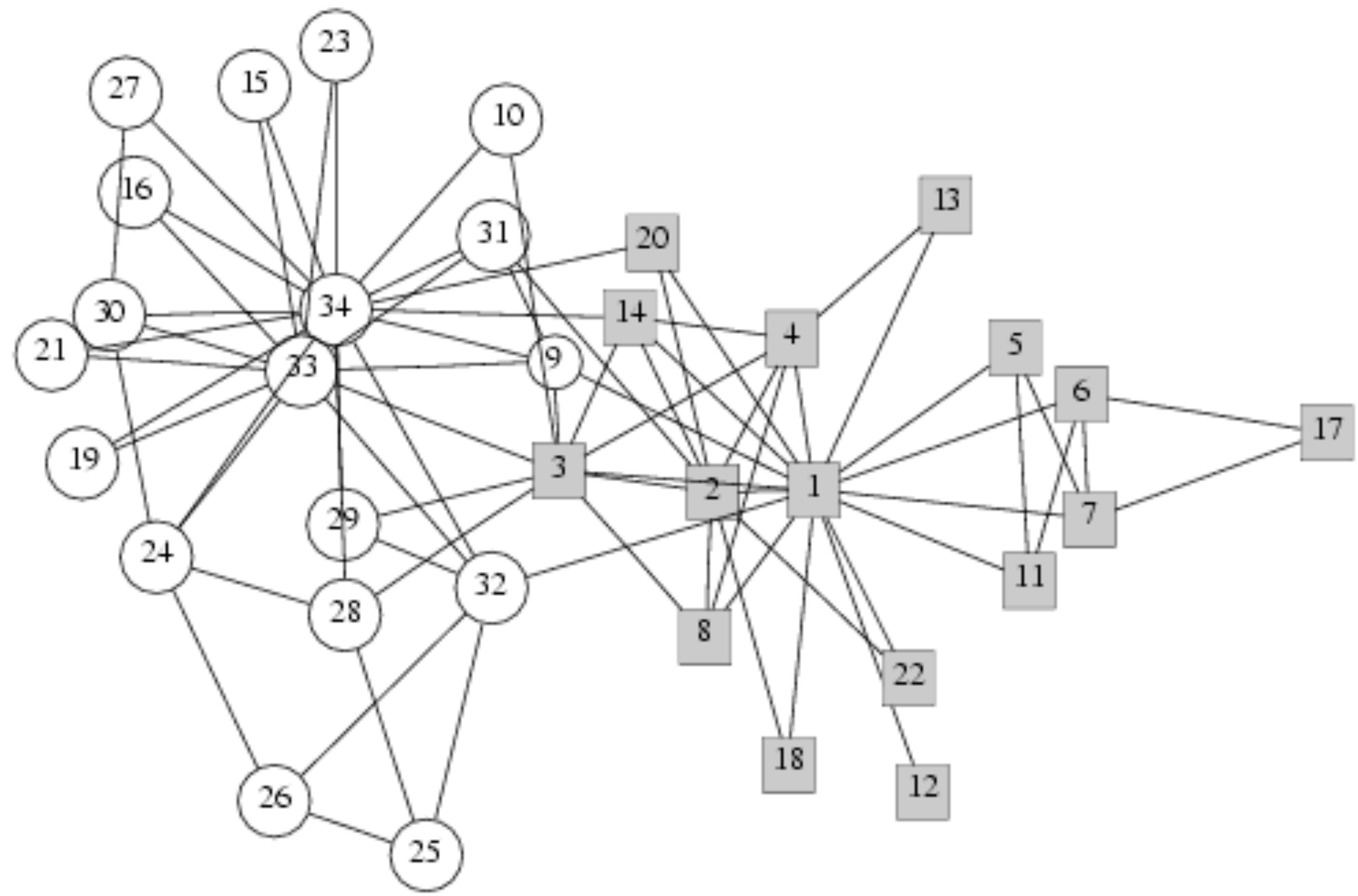
Run your algorithm: do you get out the communities you know exist in the network?

Evaluating Algorithms
Zachary's Karate Club

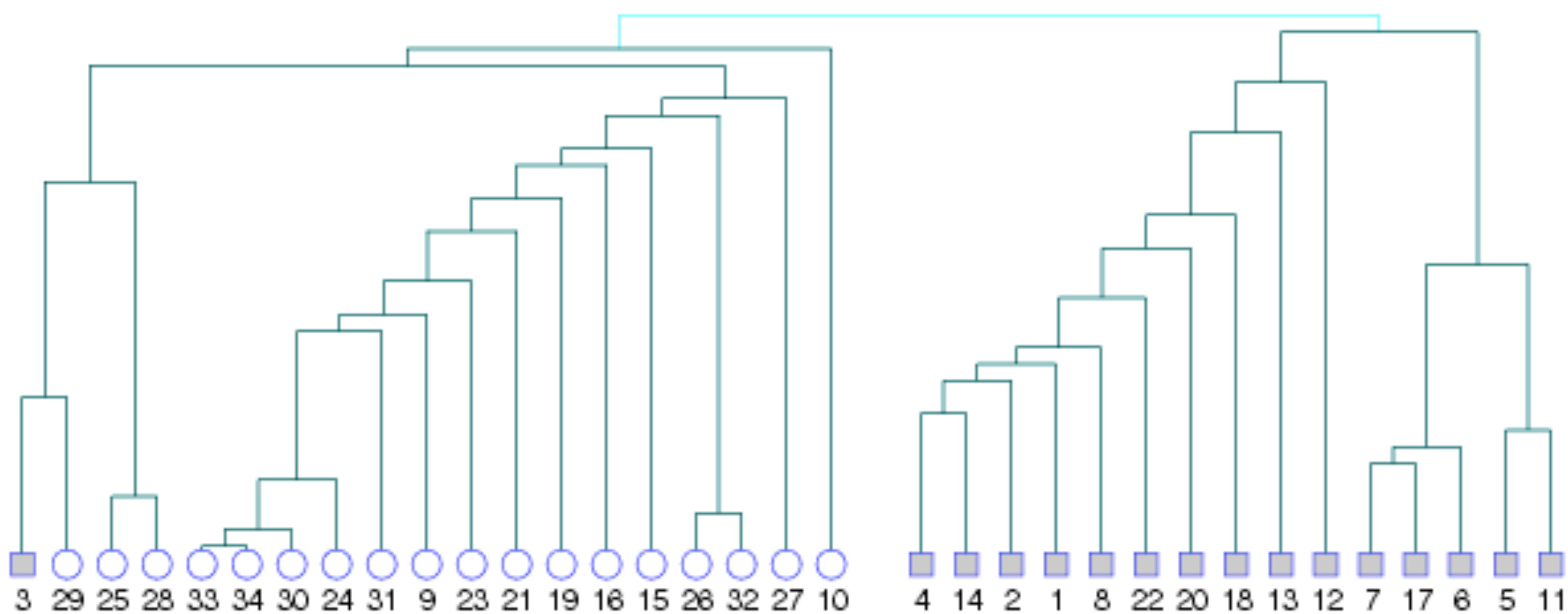
Karate Club with 34 members

During the study, the club split in half due to a disagreement

Based on the network, can the algorithms predict the actual split?

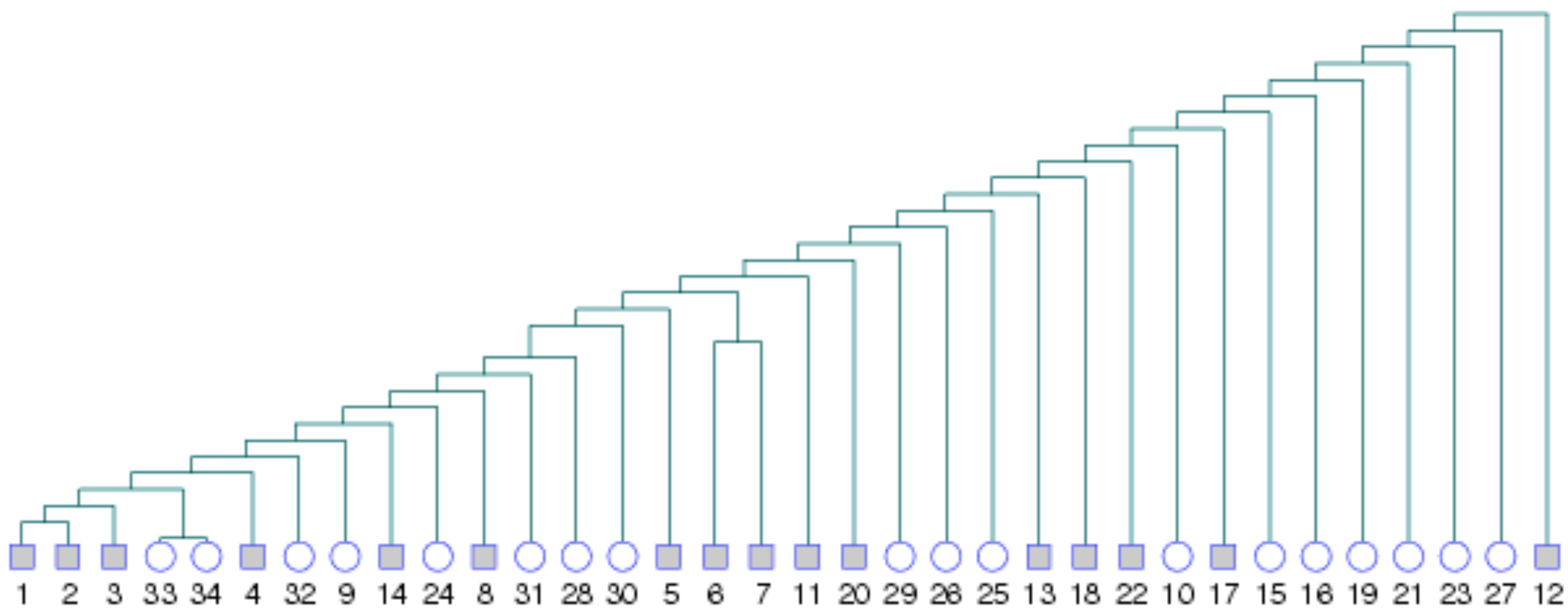


Evaluating Algorithms
Zachary's Karate Club



Girvan-Newman does quite well...

Evaluating Algorithms
Zachary's Karate Club



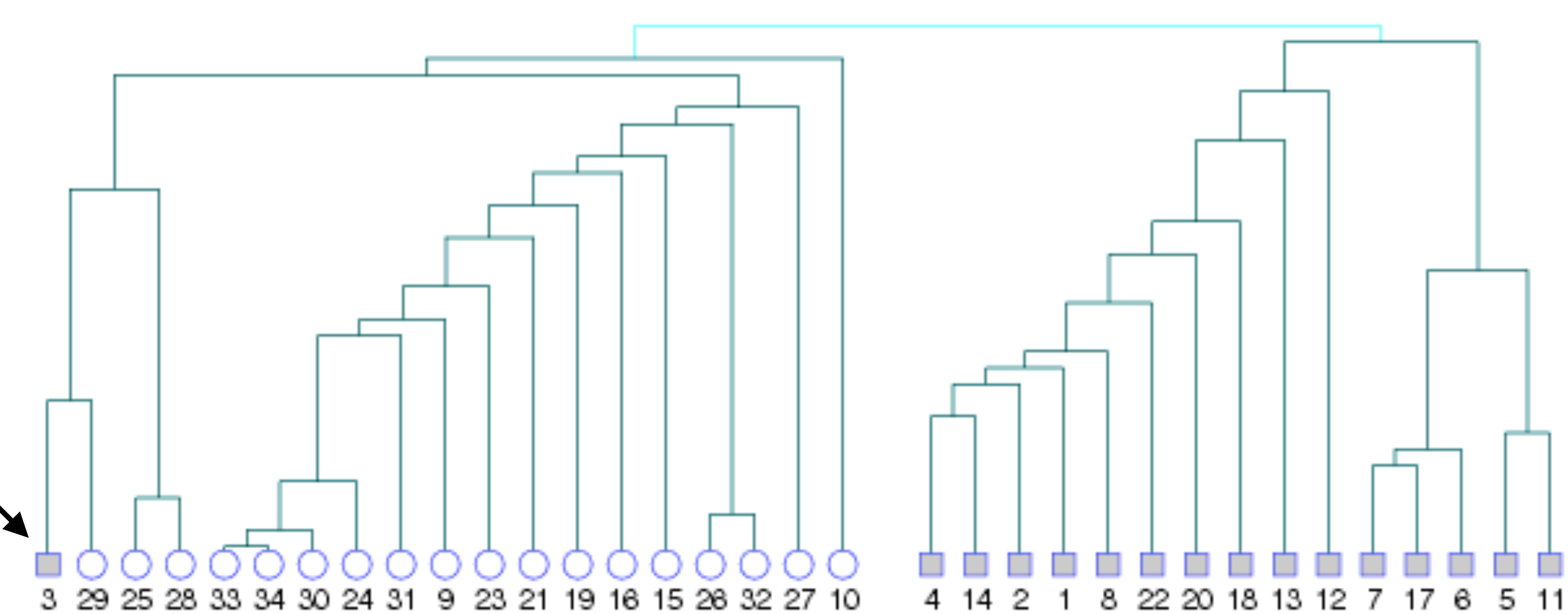
The hierarchical algorithm does quite poorly...

Evaluating Algorithms

Zachary's Karate Club

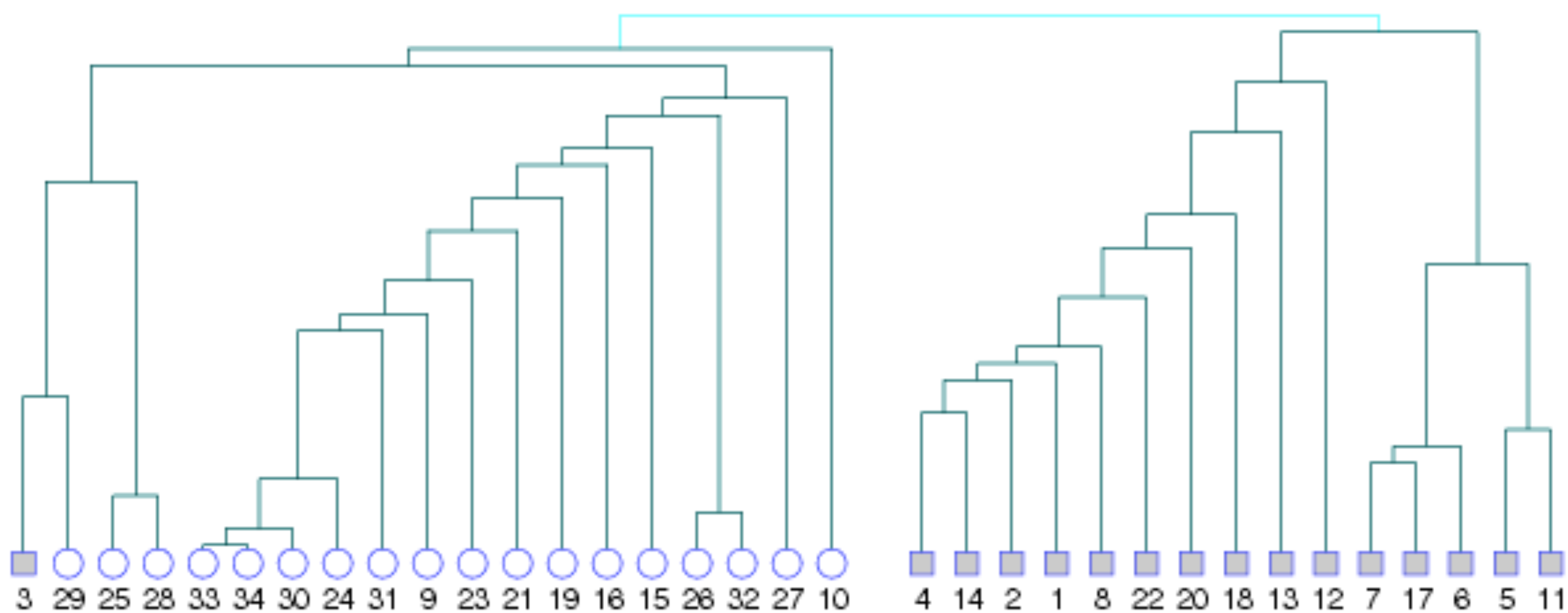
Note: the algorithms tell us about structure, not behavior. They can miss idiosyncrasies…

Node 1's brother

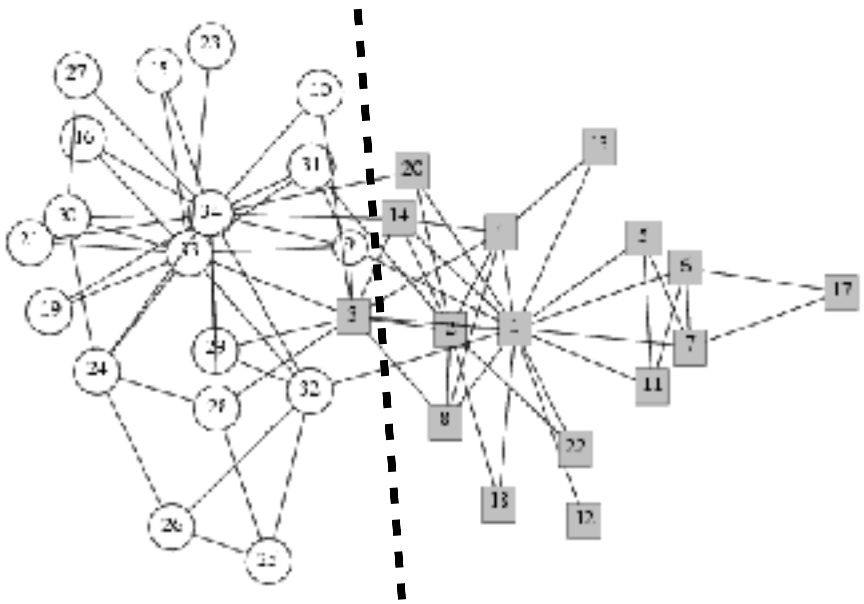


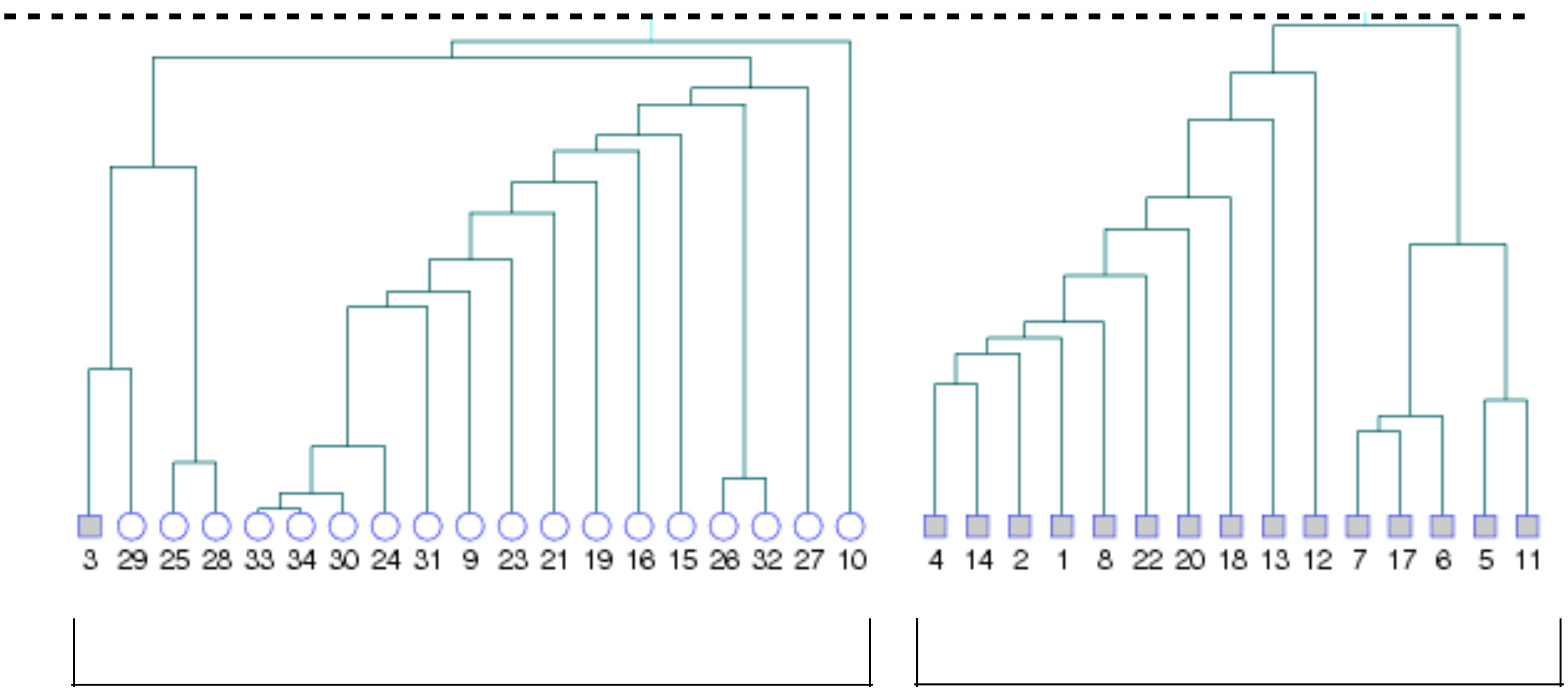
Community Structure

But one issue: these algorithms let us cut the network apart again and again…but when do we stop?

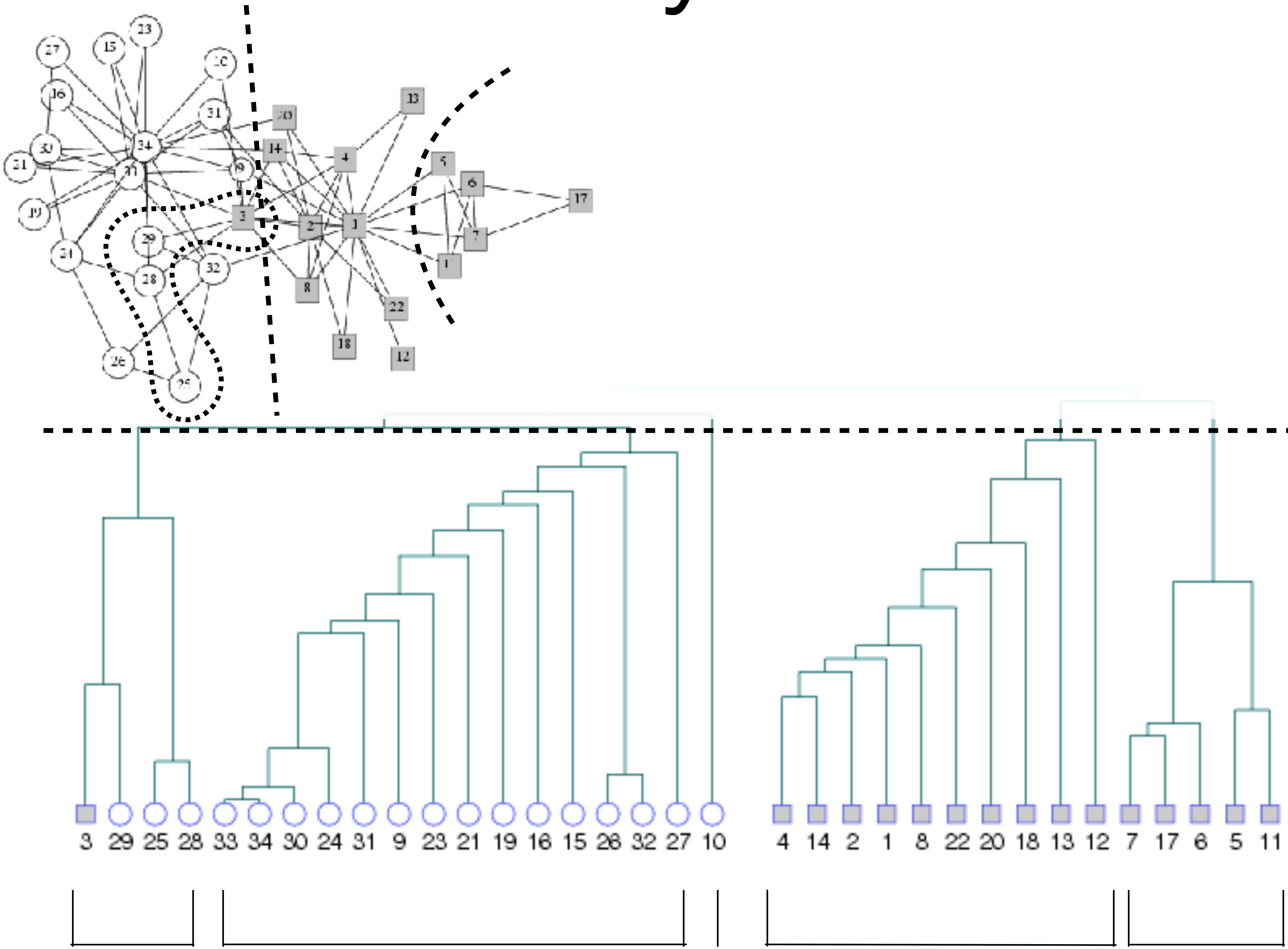


Community Structure

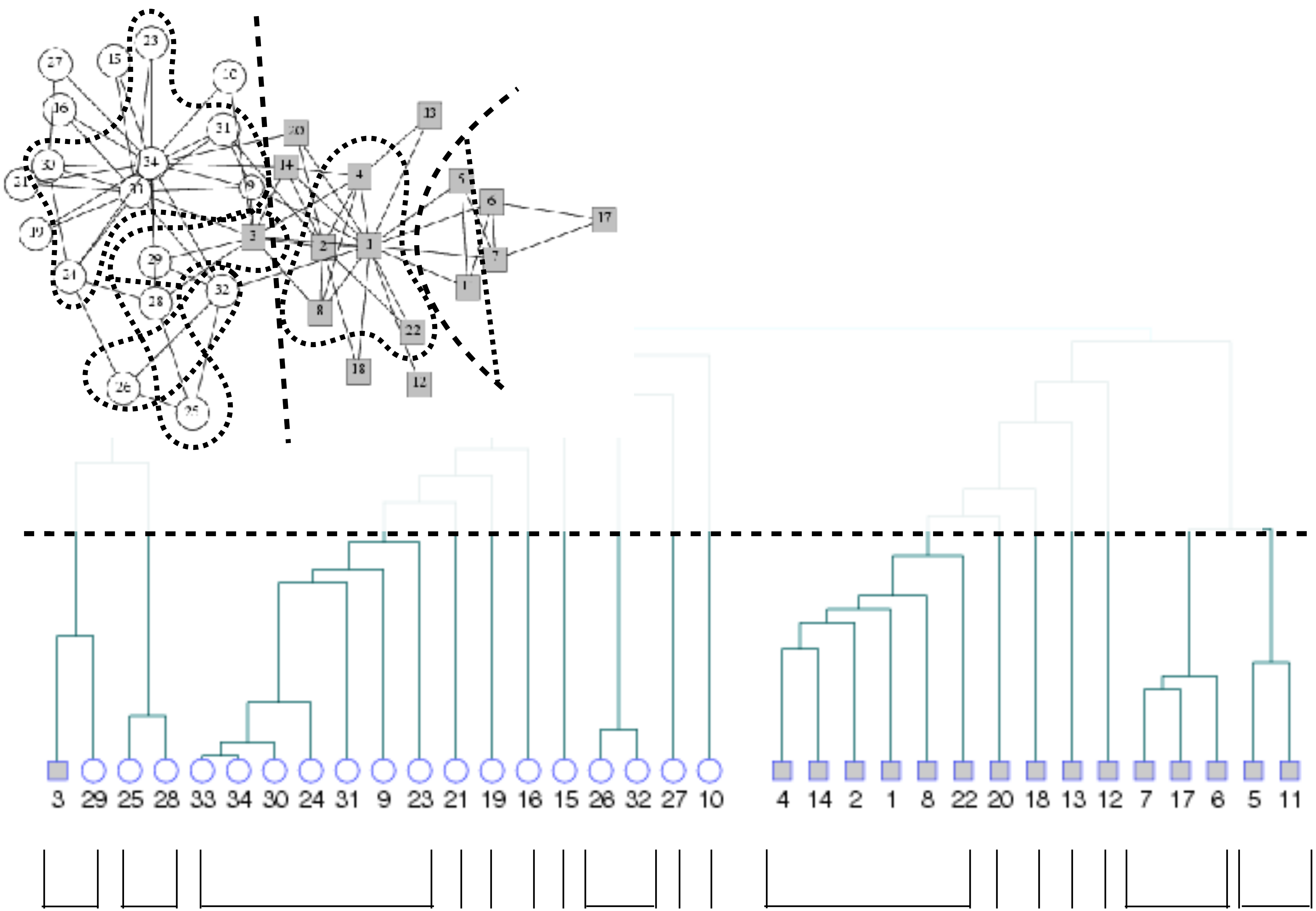




Community Structure

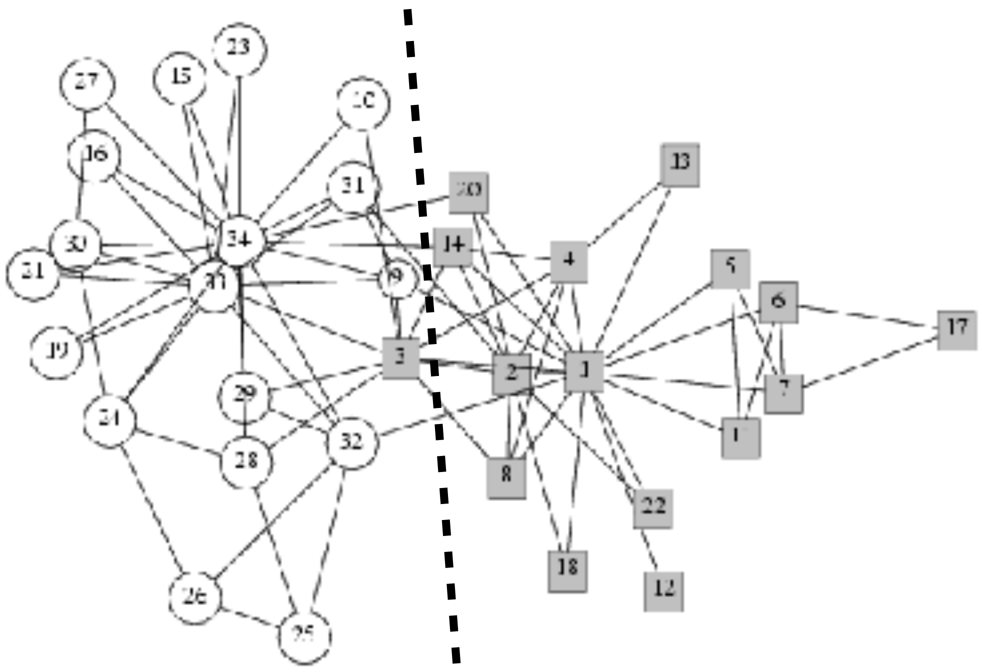


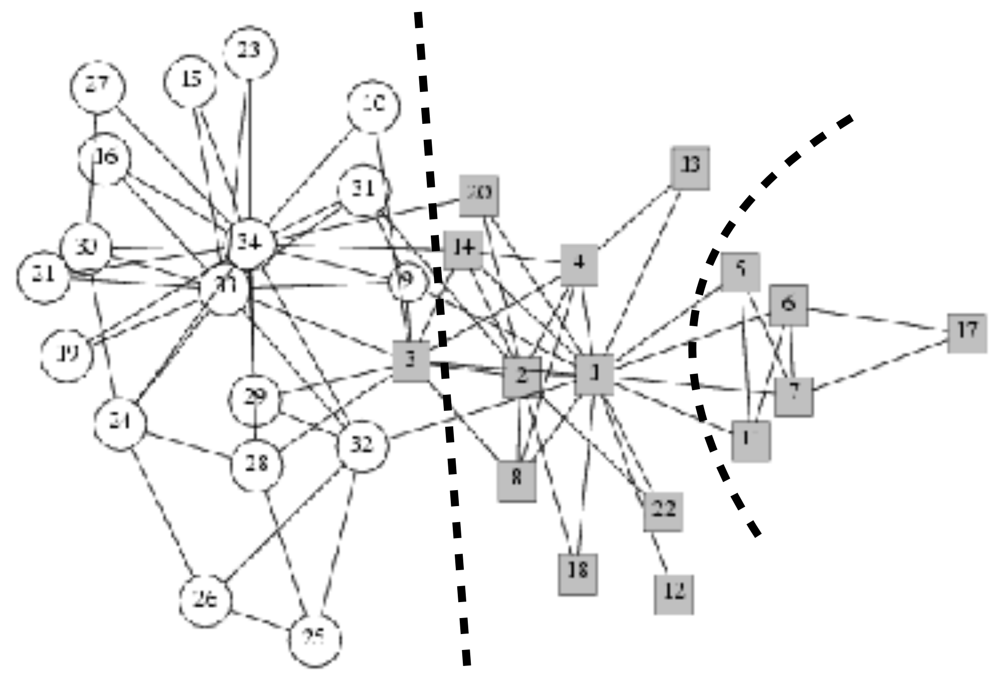
Community Structure

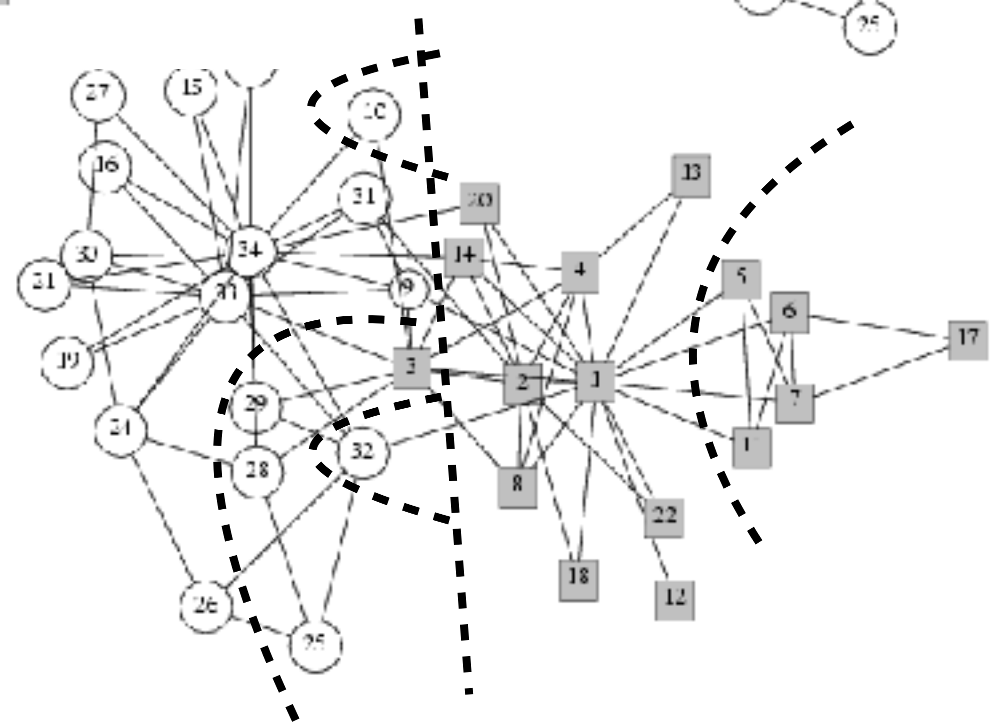


Community Structure

That is a problem if you don't have some exogenous information about community structure



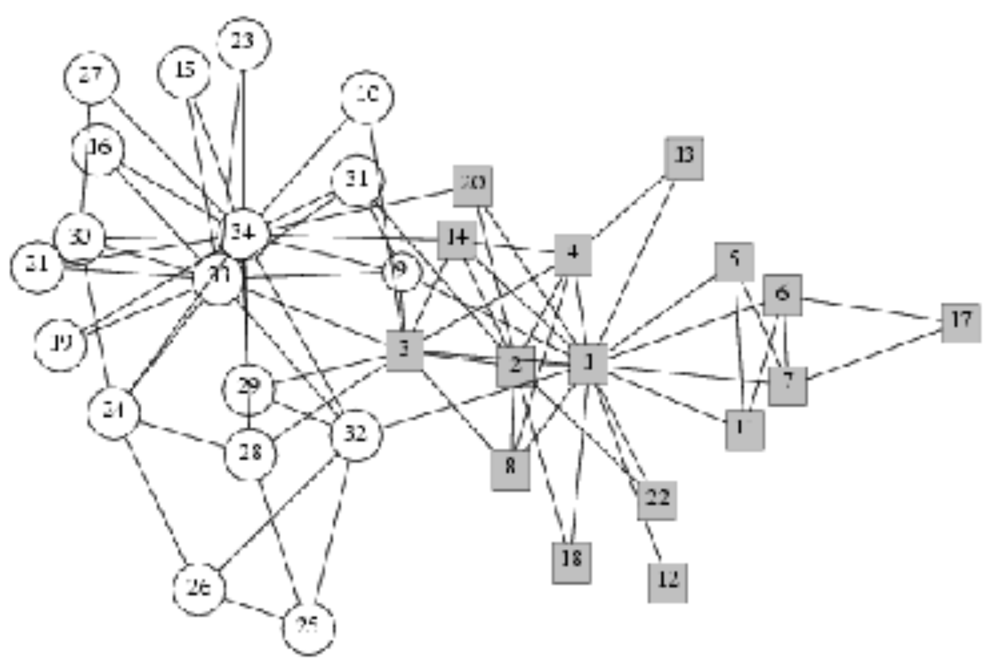




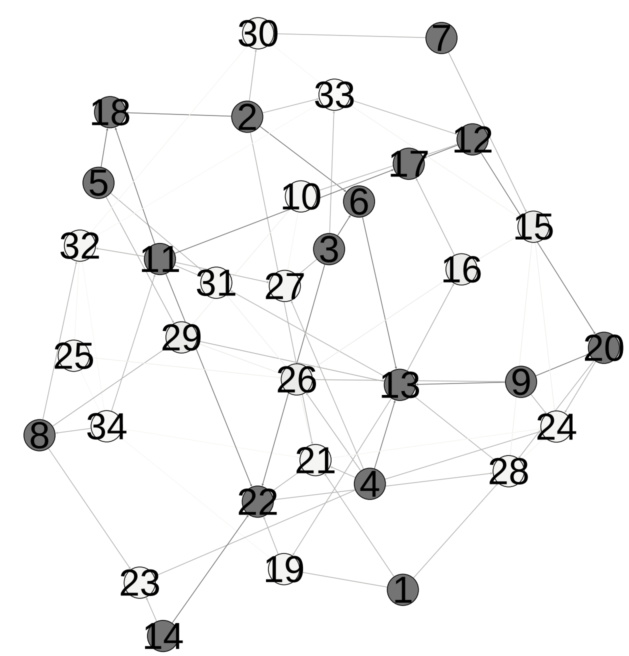
Community Structure

What we want is to not find communities where they don't exist, but pull them out when they are unusual

One method: compare the partition you make on the actual network with the partition you would get on a similar random network:



Karate Club

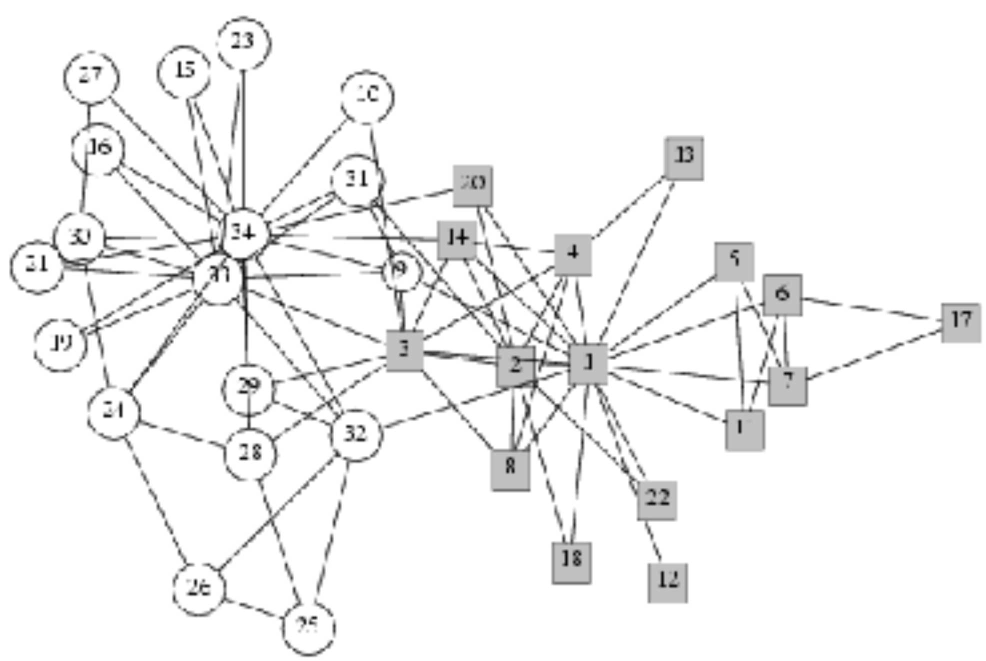


random network with the same number of nodes and same number of links

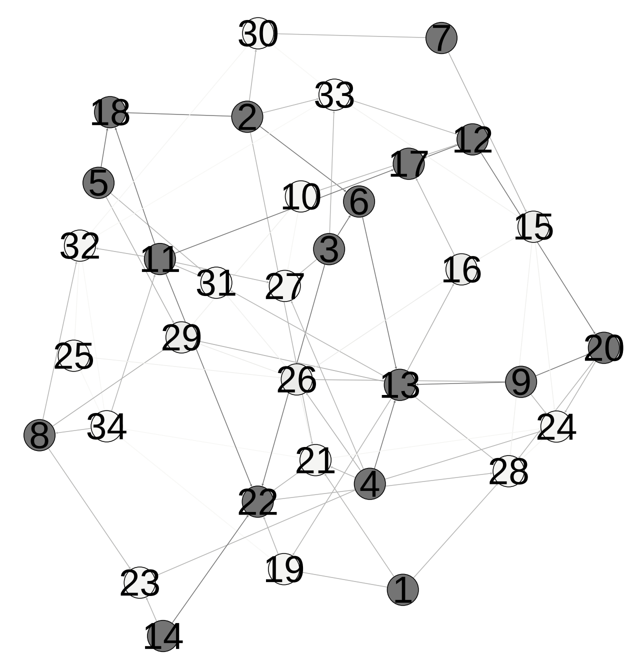
Community Structure

When you partition your network, what fraction of the links are between communities?

When you use the same partition on a random network, what fraction are between communities?



Karate Club



random network with the same number of nodes and same number of links

Modularity

Given a partition of the network into groups, *modularity* is a measure of how cohesive those groups are, relative to a random network

$$Q = \sum_i (e_{ii} - \bar{e_i})$$

Sum over all groups

fraction of edges in the network that fall between nodes in group i

fraction we would expect in a random network

In a random network: $Q = \sum_i (\bar{e_i} - \bar{e_i}) = 0$

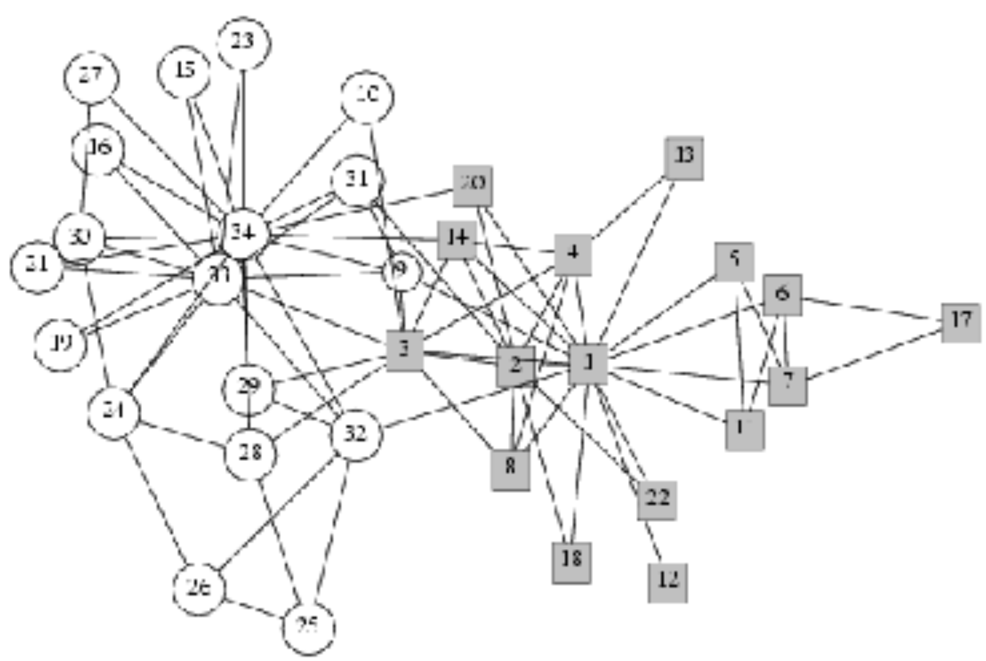
$0.3 < Q < 0.7$ indicates significant community structure

Community Structure

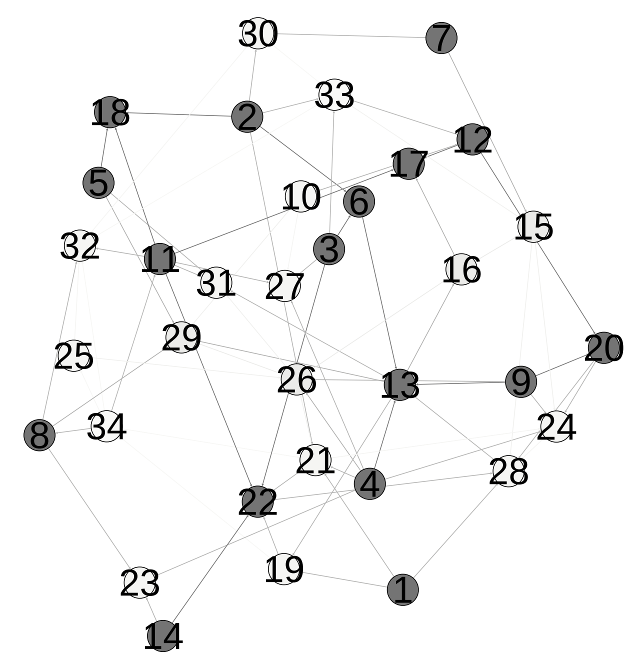
For this partition of the Karate Club Graph, there are 8 links between communities: ~11% of the links

In the random version, approximately 50% are

Modularity: 0.5 - 0.11 = .39



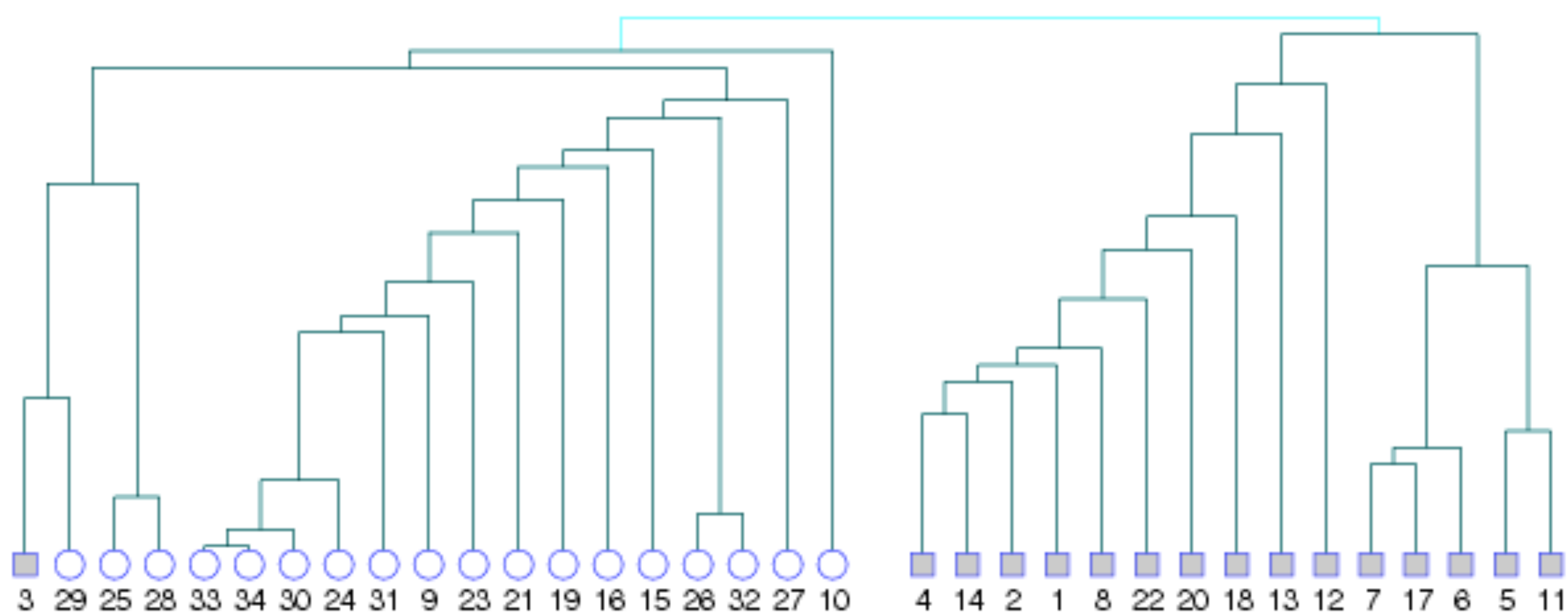
Karate Club



random network with the same number of nodes and same number of links

Modularity

One idea for choosing when to stop dividing the network: we could just choose a division into communities that maximizes modularity (Q)



But for larger networks, that would be very costly, computationally...

Community Structure

Newman’s Fast Algorithm

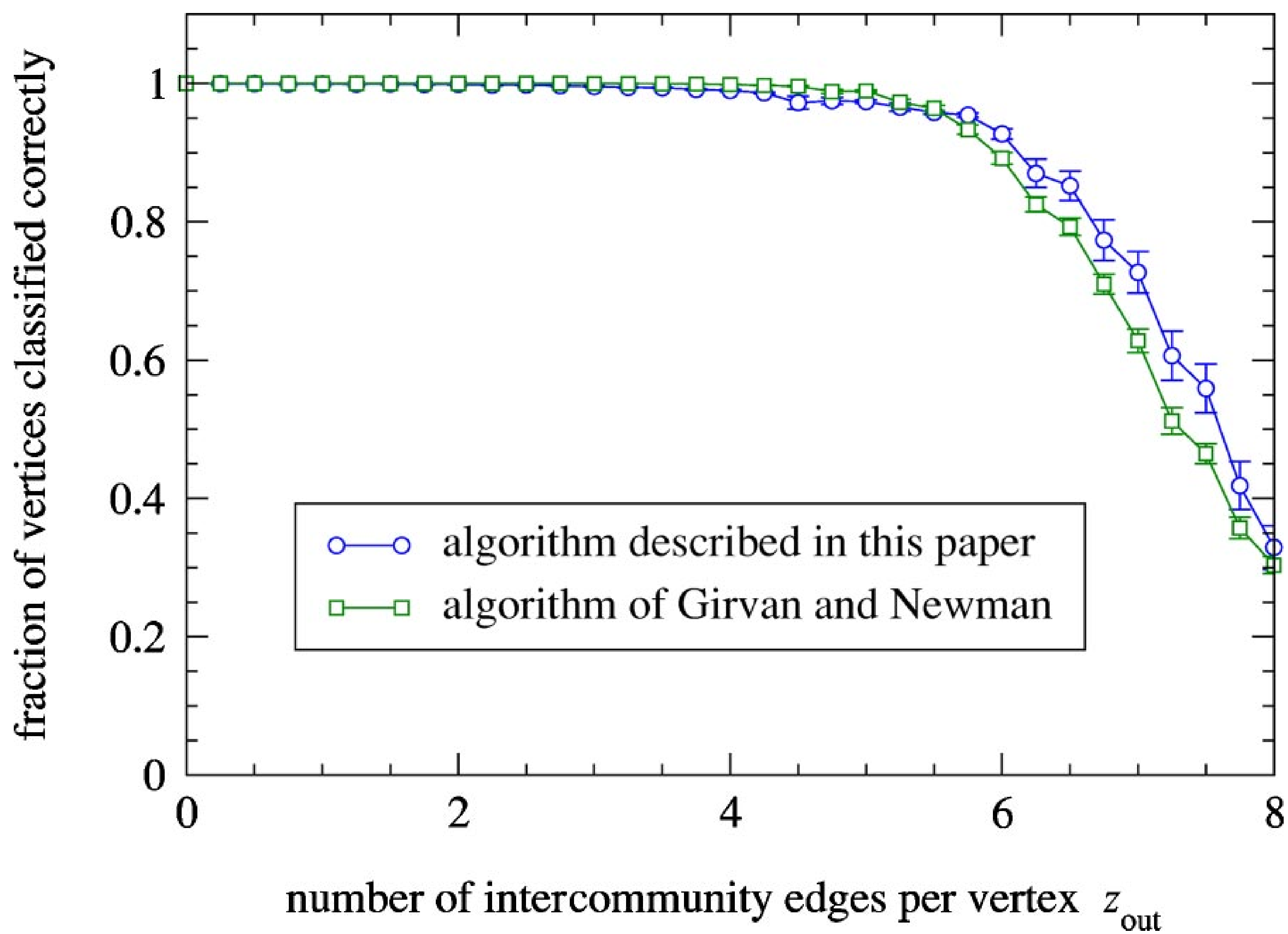
Newman’s “Fast Algorithm” uses a hierarchical algorithm, using modularity as a weight

1. Start with all nodes separated
2. At each step, connect the two communities that maximize modularity
3. Cut the dendrogram at the point in the process where modularity was maximized

Community Structure

Newman’s Fast Algorithm

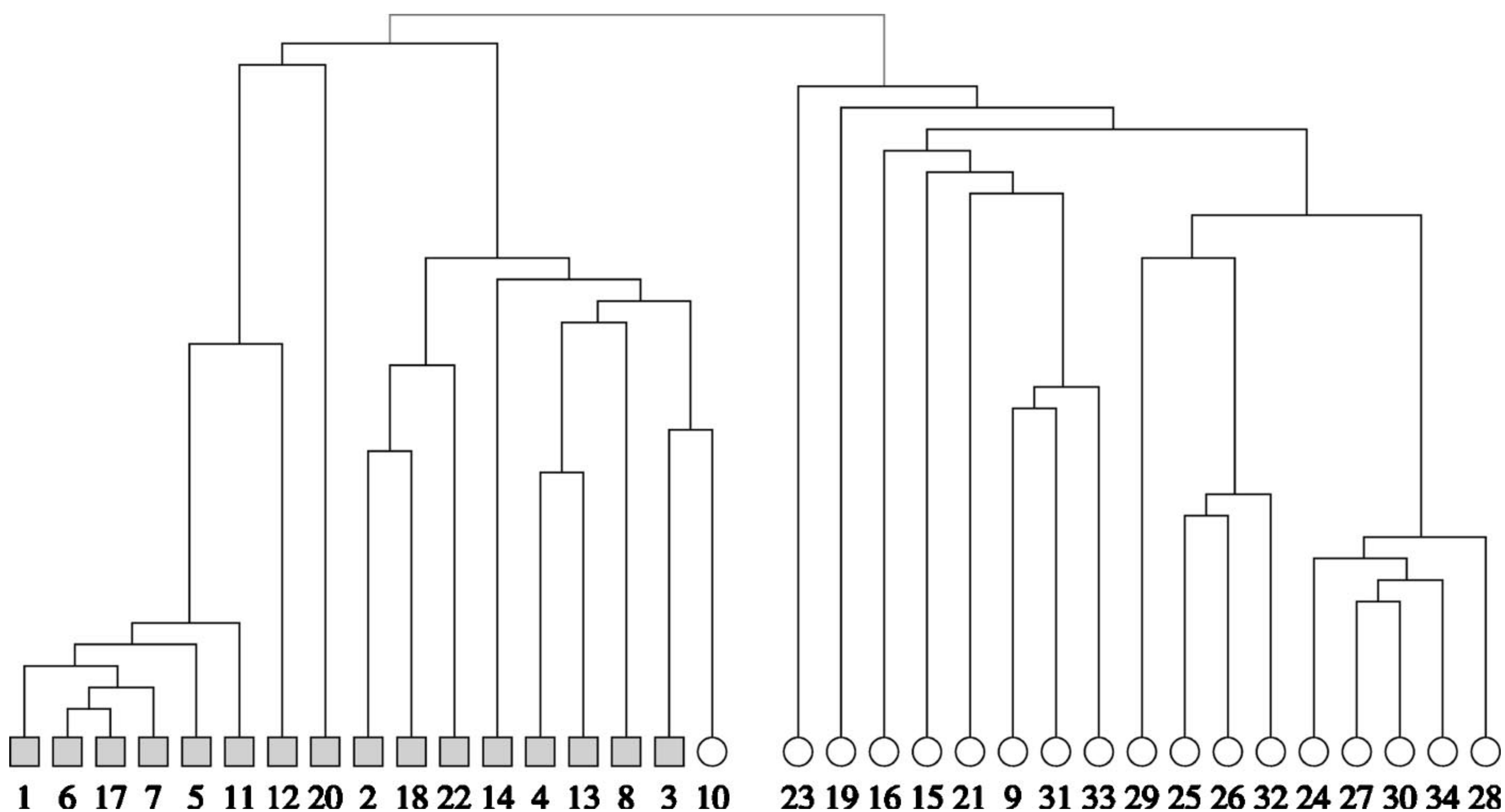
It works as well or better than the G-N algorithm on random networks...



Community Structure

Newman’s Fast Algorithm

It also splits the karate club network reasonably well (though not as well as the G-N algorithm)



Community Finding

Big Picture

- Community structure is an interesting global property of networks
- There are many algorithms that one can use to distinguish communities
- The algorithms play off of different elements in the network, and produce different results
- When you stop dividing is important, and not obvious